Skill-Adaptive Noise Scheduling for Diffusion Policies

Woo Kyung Kim[1], Gwangpyo Yoo[1], Eunyoung Park[2], Honguk Woo[1]*
[1] Department of Computer Science and Engineering, Sungkyunkwan University
[2] Department of Artificial Intelligence, Sungkyunkwan University
{kwk2696,necrocathy,eunyoung927,hwoo}@skku.edu

Abstract

Recent advances in diffusion policies have demonstrated strong effectiveness when integrated into hierarchical skill-based learning frameworks, leveraging offline data to capture multimodal and temporally abstracted behaviors. However, prior approaches have rarely considered the diffusion process as a learnable component, adopting fixed noise schedules applied uniformly across skills, regardless of their distinct behavioral contexts. Such a design fundamentally limits the achievable performance, particularly for contact-rich and dexterous manipulation tasks where different skills exhibit distinct couplings between action dimensions. In this paper, we present a skill-adaptive noise diffusion framework (SaND) which introduces a learnable, skill-conditioned noise schedule that modulates the diffusion process per action dimension, adapting the noising dynamics to each skill. To prevent the learned noise schedule from collapsing into trivial solutions, we employ an auxiliary inverse objective that reconstructs the skill embedding from the Legendre coefficients of the derivative of the noise schedule, encouraging the schedules to remain skill-discriminative. This formulation naturally enables adaptive control over the denoising steps for each skill, where noise schedules are clustered in the Legendre coefficient space, and the appropriate number of denoising steps is determined at the point where reconstruction error abruptly increases. At inference, we identify each skill embedding's cluster from its noise schedule and apply the corresponding number of denoising steps, dedicating more compute to skills requiring fine-grained control and less to simpler ones. Through experiments in simulation and on real robot hardware, we demonstrate that SaND outperforms existing skill learning and planning baselines on contact-rich and dexterous manipulation tasks, achieving higher success rates while requiring fewer denoising steps.

1 Introduction

Hierarchical skill-based learning has emerged as a promising paradigm for robotic manipulation by abstracting temporal behaviors into latent skill representations [12, 33]. Meanwhile, diffusion policies have demonstrated strong capability in modeling sequential decision-making tasks, owing to their ability to capture multimodal action distributions [1, 31, 47] and generate coherent trajectory-level plans [5, 15, 25]. Building on these successes, recent works have augmented skill-based frameworks with diffusion models to further enhance the expressiveness of skill decoders [7, 19, 22, 49, 52].

Despite their effectiveness, prior methods share key limitations due to their reliance on a fixed noise schedule across all skills and action dimensions. Such uniform denoising prevents the model from leveraging mutual information among actions, i.e., using easily inferred dimensions to guide the harder ones. This is particularly important in contact-rich and dexterous manipulation tasks, where kinematic contact constraints and dynamic equilibrium tightly couple action dimensions. Moreover, these coupling patterns vary across skills, as different behaviors yield distinct contact and force

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

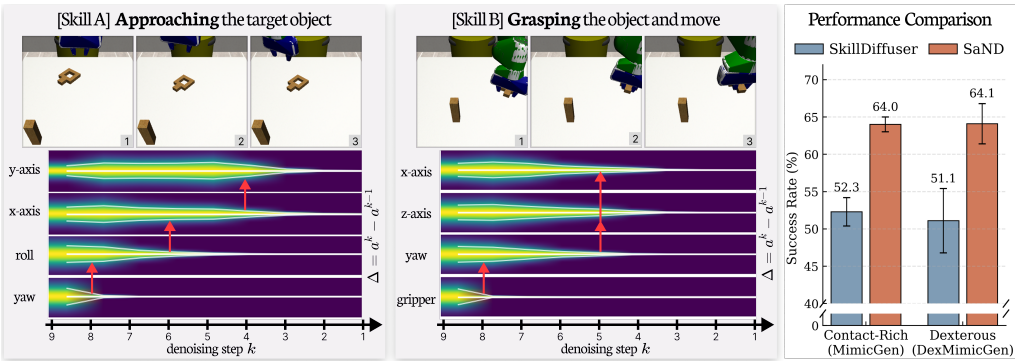


Figure 1: Concept of SaND: The first two panels visualize per-step denoising residuals along each action dimension for two different skills (approaching and grasping), illustrating that SaND learns distinct per-dimension denoising priorities for each skill. Red arrows mark the convergence point of each action dimension, showing that dimensions are resolved at distinct stages of the denoising process. The rightmost panel compares success rates of SaND with those of SkillDiffuser, which uses a fixed noise schedule, on contact-rich and dexterous manipulation tasks, showing consistent gains.

structures. Theoretically, a fixed noise schedule yields a loose bound on the diffusion ELBO [37]. Under the deterministic denoising process of diffusion policies [1, 10, 35], the resulting inaccuracy propagates irreversibly and cannot be mitigated by additional denoising steps (see Appendix C.2).

To address this, inspired by adaptive noise schedules in image diffusion models [20, 37], we propose to learn a skill-conditioned noise schedule that adapts the denoising rate per action dimension for each skill. Such adaptive scheduling enables prioritized denoising across action dimensions, first resolving dimensions that are easier to predict and subsequently refining more contact-sensitive ones, yielding a tighter ELBO on the action distributions. This is analogous to the Knothe–Rosenblatt map, which guarantees exact likelihood evaluation by decomposing a joint distribution into a chain of conditionals [2]. In Figure 1, we visualize the per-step denoising residuals along each action dimension for two different skills learned with a skill-conditioned noise schedule. When approaching the target object, rotational dimensions (yaw, roll) resolve first to align the end-effector orientation, while positional dimensions (x, y) converge later. In contrast, during grasping, the gripper actuation dimension commits early to secure a firm grasp on the object, while positional dimensions are refined later. This skill-aware prioritization yields average success rate gains of 11.7 percentage points (pp) on contact-rich tasks and 13.0 pp on dexterous manipulation tasks over the most competitive baseline, SkillDiffuser [22], as shown on the rightmost side of the figure. Beyond improving generation quality, the geometry of the learned schedule naturally reveals the denoising budget required for each skill, thereby enabling adaptive allocation of denoising steps at inference time.

To this end, we propose the **Skill-Adaptive Noise Diffusion (SaND)** framework, designed to learn robust skills for contact-rich and dexterous manipulation tasks that demand fine-grained coordination across tightly coupled action dimensions. Building on skill diffusion, a hierarchical skill-learning paradigm with a skill encoder and a diffusion-based decoder, we introduce a skill-conditioned noise schedule that modulates the noise injection across action dimensions based on the skill embedding. To prevent the schedule from collapsing into a skill-agnostic (fixed) one, we employ an auxiliary inverse schedule that reconstructs the skill embedding from the Legendre coefficients of the noise schedule's derivative. Furthermore, the learned noise schedule naturally encodes the denoising demand of each skill, which allows us to adaptively determine the number of denoising steps without additional training. We introduce a schedule-driven adaptive denoising mechanism, which allocates denoising steps per skill based on the geometry of the learned schedule. Specifically, we cluster skills by the Legendre coefficients of their corresponding noise schedules and identify, for each cluster, the minimal denoising budget that maintains reconstruction quality. At inference, we select the total number of denoising steps based on the inferred skill, yielding improved inference efficiency with negligible impact on task performance. The contributions of our work are summarized as follows:

- We propose the SaND framework, skill diffusion augmented with a skill-conditioned noise schedule that modulates noise injection across action dimensions based on the skill embedding, paired with an auxiliary inverse schedule to prevent the schedule from collapsing.

- We introduce a schedule-driven adaptive denoising mechanism that assigns a distinct number of denoising steps for each skill based on the geometry of the learned noise schedule, enabling efficient inference without sacrificing task performance.
- Through extensive experiments on contact-rich and bimanual dexterous tasks, we show that SaND consistently outperforms existing diffusion-based skill learning and planning baselines while accelerating inference.

2 Related Work

Diffusion Policies. Recently, diffusion models have been widely adopted for sequential decision-making problems. A straightforward approach is to directly model the policy as a diffusion process over actions [9, 31, 47], with several extensions to offline reinforcement learning settings [17, 24, 47]. Another line of work formulates diffusion as a trajectory optimization problem, performing planning by iteratively denoising state-action sequences [5, 15, 25]. SkillDiffuser [22] and LDuS [18] incorporate diffusion into hierarchical skill-based frameworks to model long-horizon manipulation via latent skill representations. Despite these advances, none of these approaches considers learning a skill-conditioned noise schedule. Inspired by adaptive noise schedules in image diffusion models [20, 37], we extend this idea to skill generation, enabling the diffusion process to adaptively modulate noise injection across action dimensions based on the skill embedding.

Meanwhile, several studies have explored accelerating the inference of diffusion policies. CP [34] and OneDP [48] distill diffusion policies into fewer-step models, while D3P [50] learns to dynamically adjust the number of denoising steps to the current context via online adaptation. Our work achieves a similar effect by leveraging the geometry of our skill-conditioned noise schedule to adaptively determine per-skill denoising steps, thereby addressing the varying denoising complexity across skills. This emerges as a favorable side-effect rather than an explicit design objective.

Robotic Manipulation. Robot learning has achieved remarkable progress on classical locomotion [6, 11] and manipulation benchmarks [39, 51]. Building on this success, the community has increasingly turned toward more complex scenarios that demand richer physical interaction with the environment. This includes contact-rich manipulation [28, 29], where the space of possible contacts varies with object geometry and pose, and dexterous manipulation [8, 16], where multi-fingered hands must coordinate fine-grained motions to perform intricate manipulations. To address these challenges, FORGE [30] introduces a force-aware manipulation policy for contact-rich tasks, while DexHandDiff [23] proposes a diffusion-based planning framework for adaptive dexterous manipulation tasks via guidance from large language models. Yet, these approaches address such tasks by introducing external signals, while adapting the diffusion process itself to individual skills remains underexplored. In this work, we develop a skill-based framework broadly applicable to various robotic manipulation settings, proposing skill-adaptive noise diffusion for robust skill learning and schedule-driven adaptive denoising for efficient inference.

3 Preliminaries

Notations. Subscript t denotes environment timesteps, and superscript k denotes diffusion steps. To avoid ambiguity between the superscript and an exponent, we write the n-th power of $\sigma^{(k)}$ as $[\sigma^{(k)}]^n$.

Diffusion Models. To capture the dependency across action dimensions, which a squashed diagonal Gaussian cannot express, recent works adopt diffusion models to model expressive action distributions [1, 31]. The *forward process* in diffusion maps an input x into a noise-corrupted variable x^k for $k \in [0, 1]$ by gradually adding Gaussian noise, where $k = 0$ corresponds to the clean input and $k = 1$ corresponds to the fully noised version. By discretizing the diffusion step into K steps of width $1/K$, the forward process factorizes into a Markov chain,

$$q(x^{0:1}) = q(x^0) \prod_{i=1}^{K} q(x^{k(i)}|x^{s(i)}), \quad \text{where } k(i) = i/K \text{ and } s(i) = (i-1)/K. \tag{1}$$

The marginal distribution of x^k is given by $q(x^k|x) = \mathcal{N}(x^k; \alpha^k x, [\sigma^{(k)}]^2 \boldsymbol{I})$ where $\alpha^k, \sigma^k \in \mathbb{R}_+$ are the noise schedule functions of k. Then, the diffusion model ϵ_θ is trained to predict the noise

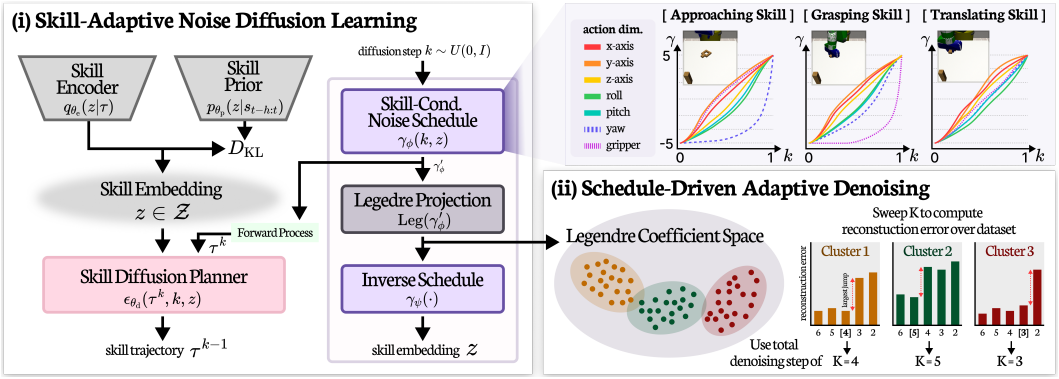


Figure 2: SaND framework: In (i), skills are learned via a diffusion-based hierarchical structure with a skill-conditioned noise schedule and an inverse schedule. The skill-conditioned noise schedule assigns action dimension-wise noise rates for each skill, capturing mutual dependencies among action dimensions. In (ii), skills are clustered according to the Legendre coefficients of their noise schedule derivatives, and denoising budgets are assigned per cluster based on the smallest K that maintains reconstruction quality. This enables skill-adaptive inference with reduced computational cost.

from a noise-corrupted input via the denoising score matching (DSM) loss [13]:

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{k\sim U[0,1],\epsilon\sim\mathcal{N}(0,\boldsymbol{I})} \left[\lambda^k \|\epsilon_\theta(\alpha^k x + \sigma^k \epsilon, k) - \epsilon\|_2^2\right], \tag{2}$$

where λ^k is a reweighting term. In practice, we adopt v-parameterization, where the model predicts $v^k = \alpha^k \epsilon - \sigma^k \tau$. For readability, we write the remainder of this section in terms of noise ϵ prediction.

Skill Diffusion. Given a dataset $\mathcal{D} = \{d_i\}_{i\leq N}$, where each demonstration $d = \{(s_t, a_t)\}_{t\leq H}$ is a sequence of state-action pairs, hierarchical skill-based policy learning aims to extract latent skill representations $z \in \mathbb{R}^{d_z}$ that capture temporal behaviors (i.e., skills) in an unsupervised manner [12, 33]. Here, a skill is defined as a sequence of h consecutive actions $\boldsymbol{a} = \{a_t, ..., a_{t+h-1}\}$, and $\tau = (s_{t:t+h-1}, a_{t:t+h-1})$ denotes the corresponding skill trajectory. Recent works incorporate diffusion models for skill-based policy learning [18, 22], i.e., *skill diffusion*, comprising a skill encoder $q_{\theta_e}(z|\tau)$ and a diffusion-based skill decoder $\epsilon_{\theta_d}(\boldsymbol{a}^k, k, z)$, optimized via

$$\mathcal{L}(\theta_e, \theta_d) = \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}, z\sim q_{\theta_e}(z|\tau)} [\underbrace{\|\epsilon_{\theta_d}(\boldsymbol{a}^k, k, z) - \epsilon\|_2^2}_{\text{DSM loss}} + \beta \underbrace{D_{\text{KL}}(q_{\theta_e}(z|\tau), p(z))}_{\text{regularization loss}}] \tag{3}$$

where $\boldsymbol{a}^k$ denotes the noise-corrupted action sequence at diffusion step k, D_{KL} is the Kullback–Leibler (KL) divergence, $p(z) = \mathcal{N}(0, \boldsymbol{I})$ is a standard Gaussian prior, and β is a regularization coefficient. Depending on the implementation, the skill decoder reconstructs either the skill or the full skill trajectory, or alternatively predicts a state sequence with actions inferred via inverse dynamics.

4 SaND Framework

4.1 Overall framework

To reduce the variational gap by implicitly embedding mutual dependencies among action dimensions into the noise schedule, we present SaND, a framework that consists of two phases: (i) skill-adaptive noise diffusion learning and (ii) schedule-driven adaptive denoising at inference, as illustrated in Figure 2. In (i), we extend the skill diffusion framework with a skill-conditioned noise schedule that adaptively adjusts the noise injection rates across action dimensions based on the skill embedding. To ensure that the learned noise schedule captures skill-specific structure, we introduce an inverse schedule as an auxiliary module, which provides a supervisory signal by recovering the skill embedding from the Legendre coefficients of the learned schedule's derivative. In (ii), prior to deployment, we cluster skills by their learned noise schedules and sweep over the number of denoising steps to identify the minimal value that maintains reconstruction quality for each cluster. At deployment, the inferred skill embedding is mapped to its nearest cluster, from which the corresponding number of denoising steps is retrieved. This enables skill-aware compute allocation, where contact-rich skills that demand precision receive more denoising steps, while routine skills use fewer.

Algorithm 1 Skill-Adaptive Noise Diffusion

Inputs: dataset $\mathcal{D}$, skill encoder q_{θ_e}, skill prior p_{θ_p}, skill diffusion planner ϵ_{θ_d}, skill-conditioned noise schedule γ_ϕ, inverse schedule γ_ψ
while not converged **do**
  $(\tau, s_{t-h:t}) \sim \mathcal{D}$; $k \sim \mathcal{U}[0,1]$; $\epsilon \sim \mathcal{N}(0, \boldsymbol{I})$
  $z \sim q_{\theta_e}(\cdot|\tau)$; $\gamma^k \leftarrow \gamma_\phi(k, z)$
  $\alpha^k \leftarrow \sqrt{\text{sigmoid}(-\gamma^k)}$; $\sigma^k \leftarrow \sqrt{\text{sigmoid}(\gamma^k)}$; $\tau^k \leftarrow \alpha^k \tau + \sigma^k \epsilon$ ▷ Forward diffusion
  Update $q_{\theta_e}, \epsilon_{\theta_d}, \gamma_\phi, \gamma_\psi$ using $\mathcal{L}_{\text{du}} + \eta \mathcal{L}_{\text{inv}}$ in (6) and (8) ▷ $\eta > 0$ is a weight for $\mathcal{L}_{\text{inv}}$
  Update p_{θ_p} using $\mathcal{L}_{\text{pri}}$ in (7)
end while

4.2 Skill-Adaptive Noise Diffusion

We employ skill diffusion, which encodes an h-length skill trajectory $\tau = (s_{t:t+h-1}, a_{t:t+h-1})$ into a skill embedding z. The skill diffusion framework consists of three components: a skill encoder q_{θ_e} that maps the skill trajectory to the skill embedding, a skill prior p_{θ_p} that predicts the skill embedding from the past states, and a skill diffusion planner ϵ_{θ_d} that reconstructs the skill trajectory, i.e.,

$$q_{\theta_e} : \tau \mapsto z \in \mathbb{R}^{d_z}, \quad p_{\theta_p} : s_{t-h:t} \mapsto z, \quad \epsilon_{\theta_d} : (\tau^k, k, z) \mapsto \epsilon. \tag{4}$$

Skill-Conditioned Noise Schedule. Due to the limited capability of a fixed noise schedule to capture skill-specific dependencies among action dimensions, as discussed in Section 1, we extend the skill diffusion framework into a skill-adaptive noise diffusion framework that adaptively adjusts the noise injection rates across action dimensions based on the skill embedding z. To this end, we introduce a trainable noise schedule γ_ϕ that maps a diffusion step k and the skill embedding z to an N-dimensional vector, where N is the action dimensionality. We implement the noise schedule as a monotonic fifth-degree polynomial following MuLAN [37], with fixed boundary values $\gamma_\phi(0, z) = \gamma_{\text{min}}$ and $\gamma_\phi(1, z) = \gamma_{\text{max}}$. Concretely, it is factorized as $\gamma_\phi(k, z) = f(k; g_\phi(z))$, where g_ϕ predicts polynomial coefficients and f evaluates the corresponding polynomial at k. Then, $\gamma_\phi(k, z)$ is passed through a sigmoid function to yield the variance schedule parameters α^k and σ^k as follows:

$$\begin{gathered} g_\phi : \mathbb{R}^{d_z} \to \mathbb{R}^{6\times N}, \quad f(k, C) = \sum_{n=0}^{5} C_n k^n, \quad \gamma_\phi(k, z) = f(k; g_\phi(z)) \in \mathbb{R}^N, \\ \alpha^k = \sqrt{\text{sigmoid}(-\gamma_\phi(k, z))}, \quad \sigma^k = \sqrt{\text{sigmoid}(\gamma_\phi(k, z))}, \end{gathered} \tag{5}$$

where $C_n \in \mathbb{R}^N$ is an n-degree coefficient vector across all action dimensions. Note that we learn a noise schedule only over the action dimensions, rather than the full state-action space. State dimensions use a fixed noise schedule. Then, the training objective of the skill-adaptive noise diffusion is defined as

$$\mathcal{L}_{\text{du}}(\theta_e, \theta_d, \phi) = \mathop{\mathbb{E}}_{k\sim U[0,1], \epsilon\sim\mathcal{N}(0,\boldsymbol{I})} \left[\gamma'_\phi(k, z)\|\epsilon_{\theta_d}(\tau^k, k, \text{sg}(z)) - \epsilon\|_2^2\right] + \beta D_{\text{KL}}\left(q_{\theta_e}(z|\tau), p(z)\right), \tag{6}$$

where $\text{sg}(\cdot)$ is the stop-gradient operator and $\gamma'_\phi = \partial_k \gamma_\phi$ serves as the reweighting term. We apply a stop-gradient to z in the DSM loss to block the direct gradient path to the skill encoder, ensuring that gradients propagate through γ_ϕ to learn a skill-adaptive noise schedule.

At inference time, the skill encoder $q_{\theta_e}(z|\tau)$ is unavailable since the skill trajectory τ is not observed. We therefore learn a skill prior p_{θ_p} that predicts the skill embedding from the past state sequence $s_{t-h:t}$, trained to match the skill encoder's posterior as

$$\mathcal{L}_{\text{pri}}(\theta_p) = D_{\text{KL}}\left(q_{\theta_e}(z|\tau), p_{\theta_p}(z|s_{t-h:t})\right). \tag{7}$$

Inverse Schedule. When transitioning between skills (e.g., from moving to grasping), the state observations barely change, causing representation collapse during learning. This in turn causes the noise schedule to degenerate into a fixed one. To prevent this, we introduce an inverse schedule γ_ψ as an auxiliary module that reconstructs the skill embedding from the learned noise schedule. Since the noise schedule cannot be fed into a model directly, we instead exploit the structural properties of its polynomial form. As a monotone function with fixed boundary values, the noise schedule is fully

characterized by its derivative. We therefore project γ'_ϕ onto a Legendre polynomial basis, whose orthogonality over the derivative space provides a natural discriminative representation of schedule shapes. Then, the loss for the inverse schedule is defined as

$$\mathcal{L}_{\text{inv}}(\phi, \psi) = \mathbb{E}\left[\|\gamma_\psi(\text{Leg}(\gamma'_\phi(k, z))) - z\|_2^2\right], \tag{8}$$

where $\text{Leg}(\cdot)$ denotes the projection coefficients onto the Legendre basis (Legendre coefficients). In practice, rather than regressing z directly, we regress to a prototype skill embedding from a codebook, similar to VQ-VAE [44]. Details and pseudocode are provided in Appendix B.1 and Algorithm 1.

4.3 Schedule-Driven Adaptive Denoising

In sequential decision-making tasks, not all contexts require the same number of denoising steps at inference [50]. Some skills demand precise control, while others can be resolved with fewer denoising steps. As illustrated in Figure 1, our skill-conditioned noise schedule naturally captures this variation: different skills exhibit distinct convergence behavior across action dimensions, suggesting that the learned schedule encodes the denoising demand of each skill. We therefore use the geometry of the learned noise schedule as a proxy for determining the per-skill denoising budget.

Specifically, we iterate over the dataset to obtain the Legendre coefficients $\text{Leg}(\gamma'_\phi(k, z))$ of the learned noise schedule's derivative across action dimensions, and apply K-means clustering to group skills by their noise schedule geometry. We sweep the total number of denoising steps K in decreasing order, computing the reconstruction MSE over the skills within each cluster at each K. The optimal K is determined as the smallest value before a sharp increase in MSE, identified via the Kneedle algorithm [38]. At inference, given a skill embedding z inferred from the skill prior p_{θ_p}, we compute $\text{Leg}(\gamma'_\phi(k, z))$, identify its nearest cluster, and retrieve the corresponding number of denoising steps. Since γ_ϕ is implemented as a 4-layer MLP, this lookup introduces negligible overhead relative to the cost of additional denoising steps. See Algorithm 2 in Appendix B.1 for the full procedure.

5 Experiments

5.1 Experiment Settings

Benchmarks. We evaluate our method on two robot simulation platforms: Robosuite [53] and Isaac Gym [26]. For Robosuite, we consider two benchmark datasets targeting *contact-rich* and *bimanual dexterous* manipulation tasks. Specifically, **MimicGen** [28] focuses on single-arm manipulation with a Panda robot, while **DexMimicGen** [16] targets dual-arm manipulation using either Panda robots equipped with dexterous hands or a GR-1 humanoid. We train on the publicly available 500 demonstrations per task from each benchmark, and report the task success rate (%). For Isaac Gym, we use benchmark environments from **Bi-DexHands** [8], which features *bimanual dexterous* manipulation tasks. We train on 100 demonstrations per task, and report the accumulated reward.

Baselines. For comparison, we consider several diffusion-based planning and hierarchical skill learning baselines. (i) **BC-RNN** [27] serves as a basic baseline, employing a recurrent neural network. (ii) **DiffBC** [31] models the policy directly as a diffusion process over actions. (iii) **DP** [9] predicts a chunk of future actions as a single diffusion process. (iv) **OneDP** [48] distills a pretrained diffusion policy into a one-step action generator via score distillation. (v) **Diffuser** [15] performs planning by iteratively denoising full state-action trajectories via a diffusion model. (vi) **DV** [25] is a diffusion-based planning framework that consolidates key design principles from modern diffusion planners. (vii) **SkillDiffuser** [22] is a hierarchical skill framework with a diffusion decoder that generates state trajectories, with actions inferred via an inverse dynamics model. (viii) **LDuS** [18] is a diffusion-based hierarchical skill framework that generates full state-action trajectories. All diffusion-based baselines use a fixed cosine schedule, whereas SaND employs a learnable skill-conditioned noise schedule. For the diffusion-based baselines, we report the number of denoising steps K required for each skill or action generation at inference.

5.2 Main Results

Performance on contact-rich manipulation tasks. In the upper part of Table 1, we evaluate performance on six *contact-rich* manipulation tasks in MimicGen using a single Panda robot equipped

Table 1: Performance on Robosuite benchmarks: All baselines and SaND are evaluated on 5 different random seeds, and success rate (%) is reported in 95% confidence interval.

Benchmark: Contact-Rich Manipulation (MimicGen [28]) – Low Dimensional Observation								
Methods	K	**Square**	**Threading**	**Mug**	**Hammer**	**Coffee**	**Kitchen**	**Average**
BC-RNN	-	53.5±5.5	42.0±11.4	55.2±5.6	48.2±3.6	52.2±7.5	52.2±12.0	50.5±2.8
DiffBC	10	44.3±1.6	57.6±1.4	43.1±5.9	37.5±2.8	68.2±10.9	28.6±4.1	46.5±1.1
DP	10	50.6±2.6	48.8±3.5	**62.1±3.6**	50.1±1.1	47.8±9.2	49.2±2.9	51.4±1.4
OneDP	1	47.1±2.8	45.6±3.0	59.8±6.6	47.1±2.3	44.4±10.6	41.4±4.0	47.6±2.1
Diffuser	10	45.6±1.5	31.5±6.9	54.2±6.2	47.0±0.7	45.4±7.7	47.0±4.0	45.1±3.0
DV	10	58.8±2.3	44.4±3.7	57.3±4.1	50.0±4.1	63.2±13.8	41.0±3.1	52.5±3.6
SkillDiffuser	10	53.6±1.6	40.9±6.5	58.3±3.4	47.5±2.7	69.0±9.5	44.2±3.4	52.3±1.9
LDuS	10	44.7±2.5	42.7±0.7	57.6±3.6	47.4±2.8	58.7±3.6	46.4±2.7	49.6±0.5
SaND	10	60.9±1.9	66.9±5.5	60.0±1.6	58.4±1.9	**82.4±2.2**	**55.3±4.4**	**64.0±1.0**
SaND+AD	3.7	**61.7±3.2**	**66.9±3.9**	55.9±5.0	**59.6±2.2**	81.0±2.8	54.6±3.6	63.3±0.8

Benchmark: Bimanual Dexterous Manipulation (DexMimicGen [16]) – Multimodal Observation								
Methods	K	**Box**	**Drawer**	**Tray**	**Pouring**	**Coffee**	**Can**	**Average**
BC-RNN	-	42.6±7.3	61.8±7.4	14.2±4.4	65.2±8.6	27.7±2.7	62.4±5.0	45.6±3.4
DiffBC	10	47.6±10.7	79.8±4.2	12.0±7.9	27.0±10.0	10.0±6.0	9.2±7.8	30.9±3.4
DP	10	35.2±8.2	49.6±5.1	25.4±4.6	12.0±7.8	19.4±9.4	40.0±17.5	30.3±3.9
OneDP	1	23.8±6.3	39.6±3.4	20.6±8.6	0.0±0.0	19.0±7.2	22.2±9.4	20.9±1.0
Diffuser	10	59.6±7.3	70.4±6.9	26.4±9.2	50.4±4.4	14.6±4.1	47.4±6.2	44.8±2.2
DV	10	71.6±9.6	56.4±9.1	14.8±4.7	68.8±10.2	26.6±5.7	38.0±16.5	46.0±5.6
SkillDiffuser	10	68.6±8.0	66.2±5.7	38.6±7.8	57.0±11.9	24.0±4.2	52.4±14.9	51.1±4.3
LDuS	10	57.6±9.4	76.6±8.6	35.2±4.7	35.4±3.7	25.8±4.3	52.0±10.4	47.1±3.4
SaND	10	79.4±5.5	**81.0±1.4**	40.6±6.0	**71.0±13.3**	**32.0±4.6**	80.8±8.7	**64.1±2.7**
SaND+AD	5.4	**83.6±4.9**	80.6±4.8	**42.0±7.8**	63.6±10.8	30.8±5.3	**82.2±8.9**	63.8±1.9

with a parallel-jaw gripper [28]. We consider a *low-dimensional* observation space, which consists of the object state and the proprioceptive state of the robot. As shown, SaND yields the best performance on all tasks except Mug, outperforming the most competitive baseline SkillDiffuser by 11.7 pp in average success rate. Furthermore, SaND with schedule-driven adaptive denoising, denoted as SaND+AD, reduces the total number of denoising steps K by 63.2% while exhibiting only a negligible degradation of 0.7 pp in average performance. In some cases, SaND+AD even achieves better performance than SaND, as it automatically skips under-optimized denoising regions and finds solutions with more plausible actions. Note that K varies across tasks and skills for SaND+AD; we report the average here, with per-task K provided in Table 7 in Appendix C.6.

Performance on bimanual dexterous manipulation tasks. In the lower part of Table 1, we evaluate performance on six *bimanual dexterous* manipulation tasks in DexMimicGen using dual Panda robots equipped with dexterous hands (first three tasks) and the GR-1 humanoid (last three tasks). Here, we consider a *multimodal* observation space, which consists of three RGB images from an agent-view camera and wrist cameras on both hands, along with the proprioceptive state of the robot. As shown, SaND yields the best performance, outperforming the strongest baseline, SkillDiffuser, by 13.0 pp in average success rate. Furthermore, SaND+AD reduces K by 45.6% while maintaining comparable performance to SaND. Additionally, Table 2 reports performance on the Bi-DexHands benchmark in Isaac Gym using two Shadow dexterous hands with floating bases, under a *low-dimensional* observation space. Similarly, SaND consistently achieves the best performance across all tasks.

In these experiments, DiffBC exhibits the lowest performance, as it lacks temporal modeling. Planning-based (Diffuser, DV) and skill-based (SkillDiffuser, LDuS) baselines improve performance by incorporating temporal dependencies or skill abstractions, respectively. Meanwhile, OneDP reduces the denoising steps of DP from 10 to 1 through distillation, yet loses considerable performance relative to DP, particularly on more complex dexterous manipulation tasks. Yet, all baselines plateau beyond a certain performance level, suggesting that their fixed noise schedule fundamentally limits their capability to handle the complexity of contact-rich and dexterous manipulation tasks. SaND outperforms the baselines by employing a skill-conditioned noise schedule along with an inverse schedule for auxiliary supervision. Such a design ensures that different skills are diffused at distinct noise levels across action dimensions, enabling more robust skill trajectory generation.

Table 2: Performance on Isaac Gym benchmarks: All baselines and SaND are evaluated on 3 different random seeds, and accumulated reward is reported in 95% confidence interval.

Benchmark: Bimanual Dexterous Manipulation (Bi-DexHands [8]) – Low Dimensional Observation								
Methods	K	**Catch**	**Door**	**Lift**	**Scissors**	**Pen**	**Cup**	**Average**
BC-RNN	-	21.9±0.6	329.8±49.1	142.3±97.3	293.6±91.6	79.3±25.4	400.5±130.3	211.2±17.0
DiffBC	5	18.8±2.2	362.0±23.4	235.8±26.9	535.6±40.3	171.5±3.7	514.7±19.5	306.4±6.8
SkillDiffuser	5	14.7±0.6	394.1±18.4	95.1±30.2	570.1±11.5	147.4±24.3	564.0±26.0	297.6±8.7
DV	5	19.9±0.7	349.9±17.2	241.8±10.1	565.3±25.0	172.9±1.0	529.9±17.4	313.3±3.3
SaND	5	**24.0**±0.7	**437.9**±21.1	**242.8**±8.8	**620.7**±10.7	**175.3**±3.5	**595.0**±16.6	**349.3**±5.5

Furthermore, our schedule-driven adaptive denoising reduces the number of denoising steps K by assigning skill-specific denoising budgets according to the geometry of the noise schedule, with minimal performance degradation or even marginal gains in some cases.

5.3 Ablation Study

Ablation on skill-adaptive noise diffusion. Table 3 shows an ablation study on the key components of our skill-adaptive noise diffusion. **SaND-Legendre** removes the Legendre projection $\mathrm{Leg}(\cdot)$ applied to γ'_ϕ prior to the inverse schedule; **SaND-Inverse** further removes the entire inverse schedule γ_ψ; and **SaND-Schedule** removes the skill-conditioned noise schedule γ_ϕ entirely, falling back to a fixed noise schedule. As shown, removing each component leads to a consistent drop in average performance, validating the design choices behind our skill-adaptive noise diffusion. Specifically, on MimicGen, SaND-Legendre, SaND-Inverse, and SaND-Schedule incur average success rate degradations of 2.7 pp, 3.6 pp, and 6.4 pp, respectively. A similar trend is observed on DexMimicGen, with performance drops of 3.2 pp, 3.4 pp, and 7.1 pp, confirming the necessity of all three components. We further ablate the skill conditioning and the per-dimension design in Appendix C.5.

Table 3: Ablation on the key components of skill-adaptive noise diffusion

Benchmark: Contact-Rich Manipulation (MimicGen [28]) – Low Dimensional Observation							
Methods	**Square**	**Threading**	**Mug**	**Hammer**	**Coffee**	**Kitchen**	**Average**
SaND	**60.9**±1.9	**66.9**±5.5	**60.0**±1.6	**58.4**±1.9	**82.4**±2.2	**55.3**±4.4	**64.0**±1.0
SaND-Legendre	60.5±2.3	61.3±9.3	56.3±6.5	56.2±2.4	81.6±7.0	52.1±8.9	61.3±1.5
SaND-Inverse	56.9±1.5	61.6±7.7	58.4±5.5	55.3±2.3	77.2±8.5	53.2±4.7	60.4±1.7
SaND-Schedule	49.7±7.0	61.5±10.5	55.8±5.2	55.4±2.7	73.6±9.5	49.6±5.9	57.6±3.5
Benchmark: Bimanual Dexterous Manipulation (DexMimicGen [16]) – Multimodal Observation							
Methods	**Box**	**Drawer**	**Tray**	**Pouring**	**Coffee**	**Can**	**Average**
SaND	**79.4**±5.5	**81.0**±1.4	**40.6**±6.0	**71.0**±13.3	**32.0**±4.6	**80.8**±8.7	**64.1**±2.7
SaND-Legendre	74.0±6.2	79.6±4.5	36.4±14.6	63.0±24.6	31.4±1.8	**80.8**±12.0	60.9±5.1
SaND-Inverse	73.4±6.3	76.8±6.4	38.8±7.1	67.2±14.9	31.2±7.5	76.6±14.4	60.7±3.8
SaND-Schedule	68.0±9.9	78.6±4.8	38.2±7.3	54.6±19.4	24.2±6.7	78.6±10.9	57.0±4.0

Ablation on schedule-driven adaptive denoising. Figure 3 depicts an ablation study on the schedule-driven adaptive denoising mechanism, aimed at addressing two questions. First, we examine whether the performance of SaND+AD could have been achieved by simply using a fixed K without adaptive denoising (e.g., $K = 3, 4$ for MimicGen, $K = 5, 6$ for DexMimicGen). Fixed K fails to match SaND+AD for both benchmarks, while $K = 10$ is required to achieve comparable performance, indicating that adaptive denoising reduces K without performance loss. Second, we assess whether schedule-based clustering yields better performance than skill embedding-based clustering for adaptive denoising, showing up to 3.7% (2.6% on average) improvement in success rate for MimicGen, and up to 8.2% (4.3% on average) improvement for DexMimicGen. Results on individual tasks are provided in Appendix C.6. This gain stems from skill embeddings capturing behavioral similarity rather than denoising complexity, whereas schedule-based clustering directly groups skills by their noise schedule geometry. Figure 11 in Appendix C.1 further supports this, showing that similar skill embeddings can be assigned to different noise schedule clusters due to the temporal continuity of skills, while remaining clearly distinguishable in the Legendre coefficient space.

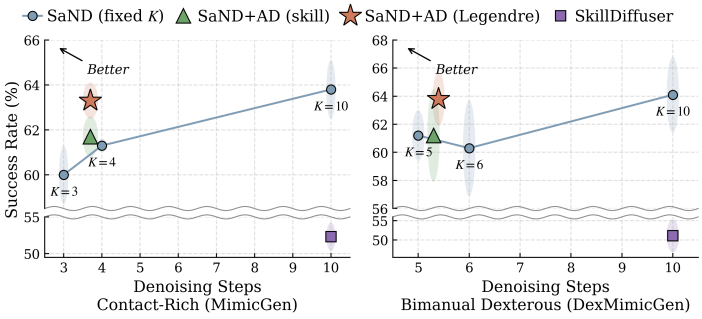


Figure 3: Ablation on schedule-driven adaptive denoising

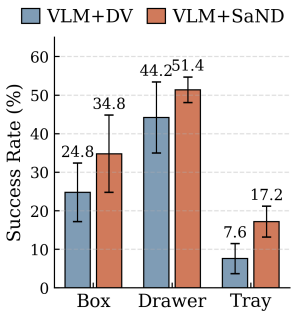


Figure 4: Extension to VLA

5.4 Case Study

Visualization of schedule-driven adaptive denoising. Figure 5 illustrates how the adaptive denoising budget varies across skills on the Drawer task. The left panel plots the number of denoising steps used at each timestep, with colored dots marking representative skill snapshots shown below. Colors correspond to clusters in the Legendre coefficient space (right panel), showing that skills with distinct denoising steps are well-separated into different clusters. Coarse motions such as arm reaching or drawer opening require as few as 4 denoising steps, whereas contact-rich skills such as grasping, preshaping, and orienting demand up to 6-7 steps. This indicates that our schedule-driven adaptive denoising effectively allocates denoising budgets according to the complexity of each skill.

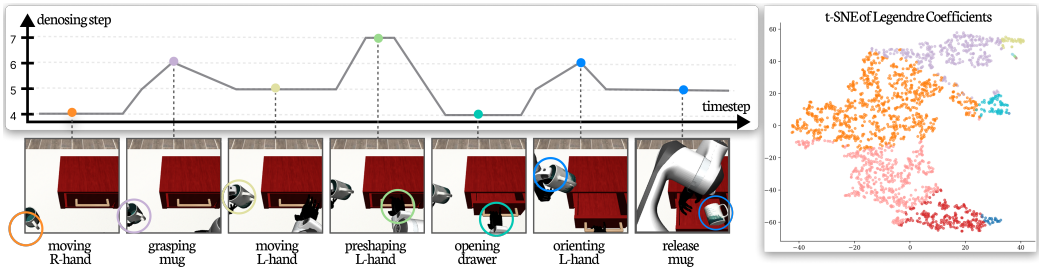


Figure 5: Visualization of schedule-driven adaptive denoising on the Drawer task

Extension to VLA. With diffusion models increasingly serving as action experts in vision-language models (VLMs) to form vision-language-action models (VLAs) [3, 4], we extend our framework to this setting. Following GR00T [3], we adopt the SigLIP-2 vision-language backbone [43] as an encoder, with SaND (VLM+SaND) or DV (VLM+DV) as the action expert. We evaluate on three bimanual dexterous tasks from DexMimicGen (Box, Drawer, Tray), which share the same action space. As shown in Figure 4, VLM+SaND achieves a 9.0 pp higher success rate on average than VLM+DV, demonstrating that a skill-conditioned noise schedule offers a clear advantage over a fixed one in the VLA setting. We note that only the last four layers of the VLM backbone are finetuned; consequently, the absolute success rates here are not directly comparable to those in Table 1, where the visual encoder is trained end-to-end on the imitation learning data.

Real-world experiment. In Figure 6, we validate SaND on a real UR7e arm with a Robotiq 2F-85 gripper. We evaluate on two tasks, Cube Stacking and Nut Insertion, both requiring precise rotational alignment and contact-rich control under randomized object poses (see Appendix A.3 for the detailed setup). We collect 100 trajectories per task and evaluate each policy over 20 trials with randomized initial configurations, reporting success rate. We compare against DP [9] rather than SkillDiffuser, the strongest skill-based baseline in simulation, because DP has been extensively validated on physical hardware and serves as the standard reference for real-world baseline. As shown in the right panel, SaND outperforms DP on both tasks, by 40 pp on Cube Stacking and 70 pp on Nut Insertion. Inspecting failures in the left panel, we find that DP most often drifts laterally or misaligns the yaw during insertion and stacking, whereas SaND establishes the grasp orientation more consistently and maintains alignment throughout the approach. This is consistent with these

tasks coupling dimensions of differing tolerance, which a per-dimension schedule can accommodate by assigning different denoising rates.

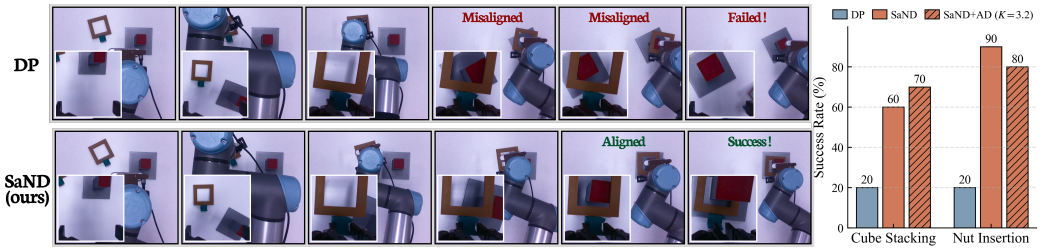


Figure 6: Real-world experiment on a UR7e arm with a Robotiq 2F-85 gripper

6 Conclusion

In this work, we presented the SaND framework to enable robust and efficient skill-based robotic manipulation, particularly tailored for contact-rich and dexterous tasks. The framework employs a diffusion-based hierarchical skill learning structure, augmented with a skill-conditioned noise schedule that adaptively adjusts the noise injection rates across action dimensions for each skill. The noise schedule is further regularized through an inverse schedule module, which reconstructs skill embeddings from the Legendre projection of the learned schedule's derivative, encouraging skill-discriminative schedule shapes. At inference, we exploit this geometry of the learned noise schedules to assign a skill-specific number of denoising steps via schedule-driven adaptive denoising, substantially reducing inference cost with negligible performance drop. Experiments on contact-rich and dexterous manipulation tasks demonstrate that SaND consistently outperforms competitive baselines in both success rate and efficiency.

Limitations and future work. Our framework learns a noise schedule conditioned on skill embeddings, which is specifically tailored to hierarchical skill-based policy learning. This is because the skill embedding naturally provides meaningful conditional signals that capture temporal behaviors, making it a natural fit for context-aware schedule learning. Extending this principle to non-skill-based diffusion would require identifying equivalent context signals, which is a promising future direction.

Acknowledgements

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2025-02218768, Accelerated Insight Reasoning via Continual Learning (50%); RS-2025-25442569, AI Star Fellowship Support Program (Sungkyunkwan Univ.)), by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2026-25474409), National IT Industry Promotion Agency (NIPA) grant funded by the Korea government (MSIT) (Research and Development of SDF-Specialized Physical AI Foundation Models for Heterogeneous Device Collaboration, No. RS-2026-25621569) and by Samsung Electronics Co., Ltd. (1%).

References

[1] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

[2] Julio Backhoff, Mathias Beiglbock, Yiqing Lin, and Anastasiia Zalashko. Causal transport in discrete time and applications. *SIAM Journal on Optimization*, 27(4):2528–2562, 2017.

[3] 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.

[4] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y Galliker, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of the 9th Annual Conference on Robot Learning*, 2025.

[5] Chang Chen, Fei Deng, Kenji Kawaguchi, Caglar Gulcehre, and Sungjin Ahn. Simple hierarchical planning with diffusion. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

[6] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Proceedings of the 35th Annual Conference on Neural Information Processing Systems*, 2021.

[7] Lili Chen, Shikhar Bahl, and Deepak Pathak. Playfusion: Skill acquisition via diffusion from language-annotated play. In *Proceedings of the 7th Annual Conference on Robot Learning*, pages 2012–2029, 2023.

[8] Yuanpei Chen, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuan Jiang, Zongqing Lu, Stephen McAleer, Hao Dong, Song-Chun Zhu, and Yaodong Yang. Towards human-level bimanual dexterous manipulation with reinforcement learning. In *Proceedings of the 36th Annual Conference on Neural Information Processing Systems*, 2022.

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

[10] Mateo Clemente, Leo Brunswic, Rui Heng Yang, Xuan Zhao, Yasser Khalil, Haoyu Lei, Amir Rasouli, and Yinchuan Li. Two-steps diffusion policy for robotic manipulation via genetic denoising. In *Proceedings of the 39th Annual Conference on Neural Information Processing Systems*, 2025.

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

[12] Kourosh Hakhamaneshi, Albert Zhan Ruihan Zhao, Pieter Abbeel, and Michael Laskin. Hierarchical few-shot imitation with skill transition models. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th Annual Conference on Neural Information Processing Systems*, 2020.

[14] Chin-Wei Huang, Jae Hyun Lim, and Aaron C Courville. A variational perspective on diffusion-based generative models and score matching. *Advances in Neural Information Processing Systems*, 34:22863–22876, 2021.

[15] Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

[16] Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Jim Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2025.

[17] Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. In *Proceedings of the 37th Annual Conference on Neural Information Processing Systems*, 2023.

[18] Woo Kyung Kim, Youngseok Lee, Jooyoung Kim, and Honguk Woo. Llm-based skill diffusion for zero-shot policy adaptation. In *Proceedings of the 38th Annual Conference on Neural Information Processing Systems*, 2024.

[19] Woo Kyung Kim, Minjong Yoo, and Honguk Woo. Robust policy learning via offline skill diffusion. In *Proceedings of the 38th Annual Conference on Artificial Intelligence*, 2024.

[20] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Proceedings of the 35th Annual Conference on Neural Information Processing Systems*, 2021.

[21] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[22] 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 35th IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[23] Zhixuan Liang, Yao Mu, Yixiao Wang, Tianxing Chen, Wenqi Shao, Wei Zhan, Masayoshi Tomizuka, Ping Luo, and Mingyu Ding. Dexhanddiff: Interaction-aware diffusion planning for adaptive dexterous manipulation. In *Proceedings of the 36th IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[24] Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[25] Haofei Lu, Dongqi Han, Yifei Shen, and Dongsheng Li. What makes a good diffusion planner for decision making? In *Proceedings of the 13th International Conference on Learning Representations*, 2025.

[26] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[27] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Proceedings of the 5th Annual Conference on Robot Learning*, 2021.

[28] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *Proceedings of the 7th Annual Conference on Robot Learning*, 2023.

[29] Yashraj Narang, Kier Storey, Iretiayo Akinola, Miles Macklin, Philipp Reist, Lukasz Wawrzyniak, Yunrong Guo, Adam Moravanszky, Gavriel State, Michelle Lu, et al. Factory: Fast contact for robotic assembly. *arXiv preprint arXiv:2205.03532*, 2022.

[30] Michael Noseworthy, Bingjie Tang, Bowen Wen, Ankur Handa, Chad Kessens, Nicholas Roy, Dieter Fox, Fabio Ramos, Yashraj Narang, and Iretiayo Akinola. Forge: Force-guided exploration for robust contact-rich manipulation under uncertainty. *arXiv preprint arXiv:2408.04587*, 2025.

[31] Tim Pearce, Tabish Rashid, Anssi Kanervisto, David Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, and Sam Devlin. Imitating human behaviour with diffusion models. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

[32] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[33] Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Proceedings of the 4th Annual Conference on Robot Learning*, pages 188–204. PMLR, 2021.

[34] Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. In *Proceedings of the Robotics Science and Systems*, 2024.

[35] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. In *Proceedings of the Robotics Science and Systems*, 2023.

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. *arXiv preprint arXiv:1505.04597*, 2015.

[37] Subham S. Sahoo, Aaron Gokaslan, Christopher De Sa, and Volodymyr Kuleshov. Diffusion models with learned adaptive noise. In *Proceedings of the 38th Annual Conference on Neural Information Processing Systems*, 2024.

[38] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *Proceedings of the 31st International Conference on Distributed Computing Systems Workshops*, pages 166–171. IEEE, 2011.

[39] Thomas Schmied, Markus Hofmarcher, Fabian Paischer, Razvan Pascanu, and Sepp Hochreiter. Learning to modulate pre-trained models in rl. In *Proceedings of the 37th Annual Conference on Neural Information Processing Systems*, 2023.

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

[41] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

[42] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[43] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

[44] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[45] Aaron Voelker, Ivana Kajić, and Chris Eliasmith. Legendre memory units: Continuous-time representation in recurrent neural networks. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, 2019.

[46] Haiyong Wang and Shuhuang Xiang. On the convergence rates of legendre approximation. *Mathematics of computation*, 81(278):861–877, 2012.

[47] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *Proceedings of the 11th International Conference on Learning Representations*, 2023.

[48] Zhendong Wang, Max Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, Ming-Yu Liu, and Yu Zeng. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 63399–63416. PMLR, 2025.

[49] Minjong Yoo, Woo Kyung Kim, and Honguk Woo. In-context policy adaptation via cross-domain skill diffusion. In *Proceedings of the 39th Annual Conference on Artificial Intelligence*, 2025.

[50] Shu-Ang Yu, Feng Gao, Yi Wu, Chao Yu, and Yu Wang. D3p: Dynamic denoising diffusion policy via reinforcement learning. *arXiv preprint arXiv:2508.06804*, 2025.

[51] 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 *Proceedings of the 4th Annual Conference on Robot Learning*, 2020.

[52] Edwin Zhang, Yujie Lu, William Wang, and Amy Zhang. Language control diffusion: Efficiently scaling through space, time, and tasks. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

[53] Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

A Experimental Environments

A.1 Robosuite

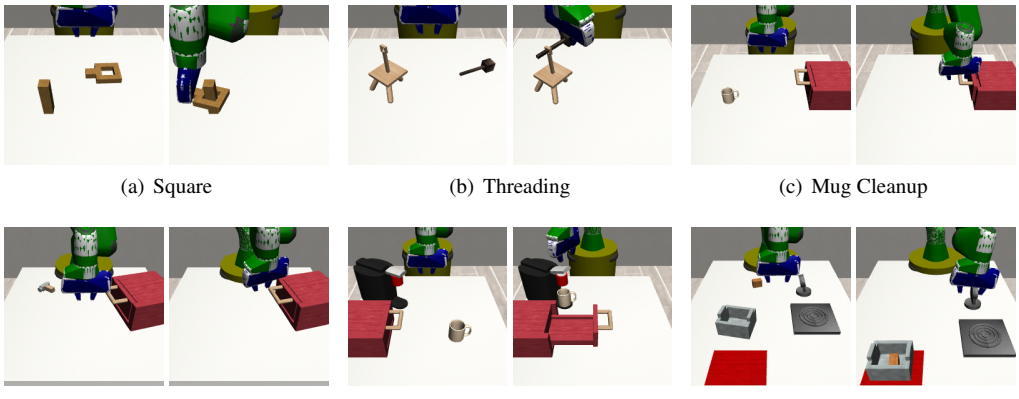
Robosuite [53] is a simulation framework for robot learning built on the MuJoCo physics engine [42]. It provides a diverse set of robotic environments, ranging from gripper-based manipulation setups to dexterous hand systems. In particular, we adopt two publicly available benchmark datasets from **MimicGen** [28] and **DexMimicGen** [16], which focus on *contact-rich* and *bimanual dexterous* manipulation tasks, respectively. Both datasets are automatically synthesized from a small number of human demonstrations (e.g., 10 demonstrations).

MimicGen [28] consists of single-arm manipulation tasks using a Panda robot equipped with a parallel-jaw gripper. We consider a *low-dimensional* observation space consisting of object state and the robot's proprioceptive state. The action space is 7-dimensional, consisting of Cartesian end-effector position increments ($\Delta x, \Delta y, \Delta z$), 3D rotational increments (Δyaw, $\Delta pitch$, $\Delta roll$), and a gripper open/close command. We evaluate on the following six tasks:

- **Square** : Pick a square nut and place it on a peg.
- **Threading** : Pick a needle and thread it through a hole on a tripod.
- **Mug Cleanup** : Open a drawer, pick a mug, and place it inside the drawer.
- **Hammer Cleanup** : Open a drawer, pick up a hammer, and place it inside the drawer.
- **Coffee Preparation** : Load a mug onto the machine, open the machine, retrieve a coffee pod from the drawer, and insert it into the machine.
- **Kitchen** : Turn on the stove, place a pot onto the stove, place a piece of bread into the pot, place the pot in front of the serving region and push it there, and turn off the stove.

For simplicity, each task name in the experiment tables is abbreviated using the first word (e.g., Mug Cleanup as Mug). Object positions and orientations are randomized across episodes to ensure contact-rich evaluation. For each task, we use 500 demonstrations for training. Figure 7 visualizes the start and end states of task environments in MimicGen.

In the original MimicGen paper, performance is reported by selecting the best checkpoint based on evaluation over 50 episodes. However, due to environmental stochasticity, we observe substantial performance variance across runs. Notably, the success rate of the selected best checkpoint fluctuates significantly when re-evaluated, suggesting that checkpoint selection based on a small number of episodes is insufficient for providing a statistically reliable estimate of performance. To ensure a more statistically reliable evaluation, we instead report the success rate of the final trained model and increase the number of evaluation episodes from 50 to 500.



(a) Square (b) Threading (c) Mug Cleanup

(d) Hammer Cleanup (e) Coffee Preparation (f) Kitchen

Figure 7: Visualization of start and end states for each task in MimicGen

DexMimicGen [16] consists of bimanual manipulation tasks with two variants: (i) dual Panda robots equipped with dexterous hands, and (ii) a GR-1 humanoid with dexterous hands. We consider a

multimodal observation space consisting of three camera views—an agent (or front) view and two robot wrist-mounted views (one for the left hand and one for the right hand)—together with the robot proprioceptive state. The action space is 24-dimensional for both variants, comprising 3D Cartesian end-effector position commands, 3D rotational commands, and 6D dexterous hand joint commands for each of the two arms. For the dual-Panda setup, these actions are represented as relative increments, whereas for the GR-1 humanoid, the actions are represented as absolute values. We evaluate on the following six tasks, the first three for dual Panda robots with dexterous hands and the last three for the GR-1 humanoid, each requiring coordinated manipulation using both arms:

- **Box Cleanup** : Lift the lid of a box and then close it.
- **Drawer Cleanup** : Open a drawer and place a mug inside it.
- **Tray Lift** : Pick up a cube with each arm, place them into a tray, and lift the tray together.
- **Pouring** : Pour a ball from a cup into a bowl, then place the bowl in the serving region.
- **Coffee** : Insert a coffee pod into the machine and close the lid.
- **Can Sort** : Pick up a can with one hand, hand it over to the other hand, and place it on a target spot.

For simplicity, each task name in the experiment tables is abbreviated using the first word (e.g., Tray Lift as Tray). For each task, we use 500 demonstrations for training, and report the success rate of the final trained model evaluated over 100 episodes. Figure 8 visualizes the start and end states of task environments in DexMimicGen.



(a) Box Cleanup (b) Drawer Cleanup (c) Tray Lift

(d) Pouring (e) Coffee (f) Can Sort

Figure 8: Visualization of start and end states for each task in DexMimicGen

A.2 Isaac Gym

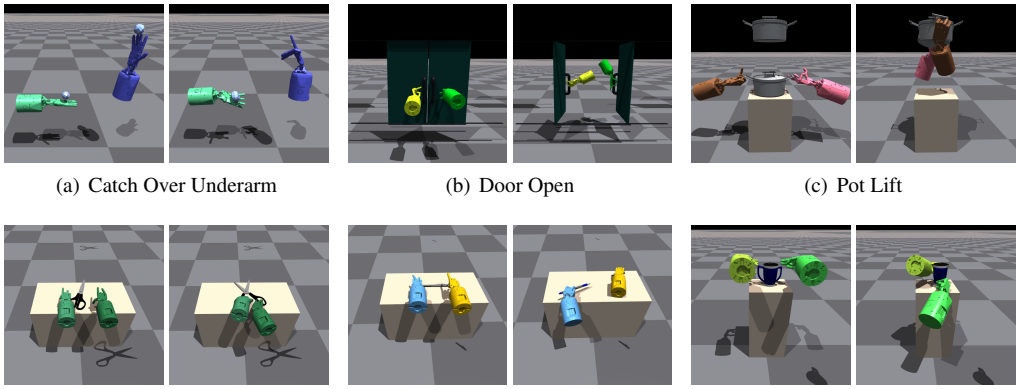
Isaac Gym [26] is a GPU-accelerated robotics simulation framework built on the NVIDIA Omniverse platform with the PhysX physics engine. We evaluate on the **Bi-DexHands** [8] benchmark environments, which feature *bimanual dexterous* manipulation tasks. Since no publicly available demonstration dataset exists for these tasks, we train an agent using the Proximal Policy Optimization (PPO) algorithm [40] and collect successful trajectories from the trained policy.

Bi-DexHands [8] consists of dexterous manipulation tasks using two Shadow dexterous hands with floating bases. We consider a *low-dimensional* observation space comprising object poses and robot proprioceptive state. The action space is 52-dimensional, comprising 20 actuated joint position targets per hand and 6 DoF for the floating base of each hand. We evaluate on the following six tasks, each requiring coordinated manipulation by both hands:

- **Catch Over Underarm**: Throw an object from a vertically oriented hand and catch it with a palm-up hand.
- **Door Open**: Push open a door that can only be opened in the outward direction.

- **Pot Lift**: Grasp the handles of a pot with both hands and lift it to a target position.
- **Scissors Open**: Open the scissors using both hands.
- **Pen Open**: Remove the cap from a pen using both hands.
- **Cup Swing**: Hold the cup handle with both hands and rotate it 90 degrees.

For each task, we use 100 demonstrations for training, and report the accumulated reward of the final trained model evaluated over 200 episodes. Figure 9 visualizes start and end states of task environments in Bi-DexHands.



(a) Catch Over Underarm (b) Door Open (c) Pot Lift

(d) Scissors Open (e) Pen Open (f) Cup Swing

Figure 9: Visualization of start and end states for each task in Bi-DexHands

A.3 Real-World Experiment Setup

We conduct real-robot experiments using a UR7e robot arm equipped with a Robotiq 2F-85 parallel gripper. The observation space consists of RGB images captured by two Intel RealSense D435 cameras (one top-mounted and one wrist-mounted) at 256×256 resolution, along with the robot's joint states. The action space is 7-dimensional, comprising Cartesian end-effector position increments, rotational increments, and a binary gripper command.

For evaluation, we configure two tasks: (i) In the **Cube Stacking** task, two cubes are initialized at random positions and orientations, requiring the policy to align the gripper yaw to each cube before grasping and placing it into a bowl. (ii) In the **Nut Insertion** task, mirroring the Square task in MimicGen, a square nut must be inserted onto a peg. The peg cross-section closely matches the nut's hole, leaving little tolerance at insertion, and the nut is initialized at a randomized orientation. Figure 10 shows the top and wrist camera views for each task.



(a) Cube Stacking (b) Nut Insertion

Figure 10: Top and wrist camera views for each real-world task

B Implementation Details

In this section, we provide the implementation details of SaND and the baselines. All experiments are conducted on a system equipped with an AMD Ryzen Threadripper PRO 5975WX 32-core CPU and an NVIDIA RTX 4090 GPU.

B.1 SaND

B.1.1 Model Architecture

Skill encoder and skill prior. $q_{\theta_e}(z|\tau)$ and $p_{\theta_p}(z|s_{t-h:t})$ are each implemented by a 2-layer LSTM network with a hidden dimension of 512. The skill horizon h is set to 8, and the skill embedding dimension is 128 for MimicGen and 256 for both DexMimicGen and Bi-DexHands.

Skill diffusion planner. $\epsilon_{\theta_d}(\tau^k, k, z)$ is implemented by a U-Net [36] architecture with 6 residual blocks, and diffusion step k is encoded through a 2-layer MLP with a sinusoidal activation. The skill diffusion planner predicts the h-horizon skill trajectory following prior works [15, 18], and we adopt the v-parameterization [37].

Skill-conditioned noise schedule. $\gamma_\phi(k, z)$ is factorized as $f(k; g_\phi(z))$, where $g_\phi(z)$ is implemented by a 4-layer MLP that produces the coefficients of a monotonic fifth-degree polynomial and $f(\cdot)$ evaluates the corresponding polynomial at diffusion step k. The noise schedule is bounded at its endpoints, with $\gamma_\phi(0, z) = \gamma_{\text{min}} = -5$ and $\gamma_\phi(1, z) = \gamma_{\text{max}} = 5$. Further details follow the implementation of Sahoo et al. [37]; we refer the reader to Appendix E.2 of the original paper.

Inverse schedule. $\gamma_\psi(\text{Leg}(\gamma'_\phi(k, z)))$ is implemented by a Set Transformer [21] over action dimensions, each represented by its Legendre projection coefficients. Treating the action dimensions as an unordered set prevents the inverse schedule from minimizing the loss via positional shortcuts, and instead forces it to rely on the Legendre coefficients. Moreover, rather than directly reconstructing the skill embedding, we maintain a codebook of prototype skill embeddings and train the inverse network to predict the nearest prototype, which we found stabilizes training. This is because skills tend to recur across trajectories and form clusters in the embedding space, making prototype-based prediction more natural than continuous regression. We find that this design yields improvements of 2.8 pp on MimicGen and 3.9 pp on DexMimicGen over directly regressing z. The number of prototype skill embeddings is set to either 8 or 16 depending on the task.

B.1.2 Legendre Polynomial Basis and Projection

In this section, we provide detailed background on Legendre polynomials and describe how we use them to project the schedule's derivative $\gamma'_\phi(k, z)$ onto a finite-dimensional representation suitable for the inverse schedule module γ_ψ.

Legendre polynomials [45] $\{P_n(x)\}_{n=0}^{\infty}$ form a complete orthogonal basis for square-integrable functions on the interval $[0, 1]$[2], i.e., the $L^2[0, 1]$ space. The Legendre polynomials can be calculated from Rodrigues' formula,

$$P_n(x) = \frac{1}{n!}\frac{d^n}{dx^n}\left[(x^2 - x)^n\right], \tag{9}$$

and satisfy the orthogonality relation,

$$\int_0^1 P_m(x)P_n(x)\,dx = \frac{1}{2n+1}\delta_{mn}, \tag{10}$$

where δ_{mn} is the Kronecker delta; i.e., $\delta_{mn} = 1$ if $m = n$, otherwise, $\delta_{mn} = 0$. Then, the first five Legendre polynomials are as follows:

$$\begin{aligned} P_0(x) &= 1, \\ P_1(x) &= 2x - 1, \\ P_2(x) &= 6x^2 - 6x + 1, \\ P_3(x) &= 20x^3 - 30x^2 + 12x - 1, \\ P_4(x) &= 70x^4 - 140x^3 + 90x^2 - 20x + 1. \end{aligned} \tag{11}$$

[2] Obtained by the change of variable $u = 2x - 1$ for the Legendre polynomials defined on $[-1, 1]$

Algorithm 2 Schedule-Driven Adaptive Denoising

Inputs: Dataset $\mathcal{D}$, skill prior p_{θ_p}, noise schedule γ_ϕ, number of clusters M, candidate denoising step set $\mathcal{K} = \{K_{\min}, \ldots, K_{\max}\}$
Initialize $\mathcal{B} \leftarrow \varnothing$
for each state sequence $s_{t-h:t}$ obtained from $\mathcal{D}$ **do**
  Sample skill embedding $z \sim p_{\theta_p}(\cdot \mid s_{t-h:t})$
  Compute Legendre coefficients $\boldsymbol{b} \leftarrow \text{Leg}(\gamma'_\phi(k, z))$
  $\mathcal{B} \leftarrow \mathcal{B} \cup \{(z, \boldsymbol{b})\}$
end for
Cluster $\mathcal{B}$ into M clusters $\{\mathcal{C}_1, \ldots, \mathcal{C}_M\}$ via K-means on Legendre coefficients
for each cluster $\mathcal{C}_m$ **do**
  for each $K \in \mathcal{K}$ in decreasing order **do**
    Compute reconstruction MSE on skills in $\mathcal{C}_m$ with K denoising steps
  end for
  $K^*_m \leftarrow$ smallest K before sharp MSE increase (via Kneedle Algorithm)
end for

Legendre projection of noise schedule. For each skill embedding z and action dimension d, the schedule derivative $\gamma'_\phi(k, z)_d : [0, 1] \to \mathbb{R}$ can be expanded as

$$\gamma'_\phi(k, z)_d = \sum_{n=0}^{\infty} c_n^{(d)}(z)\, P_n(k), \tag{12}$$

where $c_n^{(d)}(z) = (2n + 1) \int_0^1 \gamma'_\phi(k, z)_d\, P_n(k)\, dk$. In practice, we use Legendre coefficients up to order N_{Leg}, excluding the term c_0 as it is constant due to the fixed boundary values of the schedule. The Legendre projection operator $\text{Leg}(\cdot)$ is then defined as

$$\text{Leg}(\gamma'_\phi(\cdot, z)) = \left[c_1^{(d)}(z), c_2^{(d)}(z), \ldots, c_{N_{\text{Leg}}}^{(d)}(z)\right]_{d=1}^{N} \in \mathbb{R}^{N_{\text{Leg}} \times N}, \tag{13}$$

where N is the action dimensionality and N_{Leg} is set to 4 throughout our experiments. Since the schedule is a monotone function whose endpoints $\gamma_\phi(0, z)$ and $\gamma_\phi(1, z)$ are fixed, we project the derivative γ'_ϕ rather than the schedule γ_ϕ itself. Given that our schedule function is parameterized as a polynomial, its Legendre coefficients decay exponentially [46], so a small number of basis functions suffices for accurate approximation.

We adopt Legendre projection, since a small number of Legendre coefficients suffices to capture the polynomial schedule's geometry, providing a suitable input representation to γ_ψ. An alternative would be to feed sampled values $\gamma_\phi(k_i, z)$ at multiple discrete points $\{k_i\}$ directly to γ_ψ, resulting in a larger input space and degraded performance in our ablation study (see SaND-Legendre variant in Table 3). Moreover, the orthogonality of Legendre polynomials ensures that each coefficient c_n captures an independent aspect of the schedule's shape, producing a discriminative representation that distinguishes schedules with different geometric structures. Other alternatives could be B-splines and Chebyshev polynomials. B-splines, however, require $d + c$ coefficients for degree d and c intervals. Matching the expressiveness of our fourth-degree Legendre projection requires $d \geq 4$, so any B-spline representation uses more coefficients than ours. Chebyshev polynomials, an orthogonal basis of the same size, therefore remain a viable alternative. Yet, we empirically found that Legendre polynomials yield better performance, and we therefore adopt them.

B.1.3 Hyperparameters

For all experiments, we use the Adam optimizer with a learning rate of 1×10^{-4}, and exponential decay rates of $\beta_1 = 0.9, \beta_2 = 0.99$. The weight for the skill encoder regularization loss in Equation (6) (β) is set to 1×10^{-3}, and the inverse loss $\mathcal{L}_{\text{inv}}$ in Equation (8) is scaled by a factor of 1×10^{-2} (η in Algorithm 1). We use a batch size of 256 and clip gradients at a maximum norm of 1.0. All models are trained for 200,000 or 400,000 steps, depending on the maximum episode length of the task, with a linear learning rate warm-up over the first 10,000 steps. At inference, we use DDIM [41] sampling with 10 denoising steps by default. For schedule-driven adaptive denoising, we sweep the number of denoising steps $K \in \{3, 4, 5, 6\}$ for MimicGen and $K \in \{4, 5, 6, 7, 8, 9, 10\}$ for DexMimicGen. The number of clusters is set to 8 for MimicGen and 24 for DexMimicGen.

B.2 Baselines

The baselines in our experiments fall into three categories based on their action generation mechanism: **[a]** action-level (i.e., BC-RNN, DiffBC, DP, OneDP), **[t]** trajectory-level (i.e., Diffuser, DV), and **[s]** skill-level (i.e., SkillDiffuser, LDuS). Except for BC-RNN, all baselines employ a diffusion model for the policy.

[a] BC-RNN [27] serves as a non-diffusion baseline, implementing the policy via a recurrent neural network that maintains a history of 10 past observations. For MimicGen, we use a Gaussian mixture model to model the action distribution, while for DexMimicGen we use a single Gaussian distribution.

[a] DiffBC [31] is the most straightforward application of diffusion models to imitation learning, where a diffusion model conditions solely on the current observation and denoises a single action at each timestep. The denoising network is implemented as a 4-layer MLP with 512 hidden units for MimicGen, and as a multi-head attention network with 4 transformer blocks and 16 attention heads for DexMimicGen and Bi-DexHands. We adopt x_0-parameterization with a cosine schedule, training with 300 diffusion steps and accelerating inference via DDIM [41] with 10 steps.

[a] DP [9] extends diffusion-based imitation learning by predicting a chunk of future actions at each timestep, enabling temporally consistent action generation through receding horizon control. The denoising network is implemented as a U-Net [36]. We set the prediction horizon $T_p = 16$, executing $T_a = 8$ actions per replan. All remaining training and inference settings follow DiffBC.

[a] OneDP [48] distills a pretrained diffusion policy into a one-step action generator via score distillation, training the student to match the teacher's score along the diffusion chain. We adopt the stochastic variant (OneDP-S), which alternately updates an auxiliary score network and the actor, both warm-started from the teacher's weights. All architecture and receding-horizon settings are inherited from the DP model described above, with the number of denoising steps set to 1.

[t] Diffuser [15] is a pioneering work that leverages diffusion models as a trajectory planner. It jointly denoises an h-length sequence of state-action pairs while replacing the first state with the current observation. Such joint modeling of states and actions over the full horizon enables the model to enforce only local consistency within each denoising step. This local consistency accumulates into global coherence of the sampled plan after the full denoising chain. The denoising network architecture and all remaining training and inference settings follow DP. We set the horizon $h = 8$.

[t] DV [25] is the state-of-the-art diffusion-based planning model. Unlike Diffuser, DV's diffusion component is only responsible for planning desired trajectories of future states. The actions are derived via a learned inverse dynamics model that recovers actions from consecutive future states. The denoising network is implemented as a DiT [32] with 4 Transformer blocks and 4 attention heads, and all remaining training and inference settings follow DiffBC.

[s] SkillDiffuser [22] adopts a diffusion model within a hierarchical skill-based imitation learning framework [12]. Such a hierarchical structure is especially tailored for long-horizon robotic manipulation tasks, as it decomposes the task into a sequence of skills, reducing the burden of modeling long-range dependencies directly in the action space. It uses a skill encoder and a skill decoder to learn a skill embedding, where the decoder is implemented as a diffusion model. Specifically, the diffusion-based skill decoder reconstructs a state sequence conditioned on the skill embedding, which is then passed through an inverse dynamics model to produce the corresponding action sequence. The denoising network is implemented as a U-Net.

[s] LDuS [18] is another hierarchical skill-based imitation learning framework that employs a diffusion model as the skill decoder. Unlike SkillDiffuser, LDuS directly denoises sequences of state-action pairs, eliminating the need for a separate inverse dynamics model. The denoising network is implemented as a U-Net structure.

For all baselines, we use the same optimizer and training budgets as SaND in Appendix B.1.3.

B.3 Multimodal Encoder

To handle the multimodal observation space for DexMimicGen in Table 1, we pre-train a multimodal encoder and freeze its parameters during policy learning. While end-to-end joint training of the multimodal encoder and policy is possible, it incurs prohibitive computational overhead; we therefore adopt the decoupled pre-training strategy.

We employ a separate ResNet-18 encoder for each image modality and an MLP-based encoder for the proprioceptive sensor inputs [27]. The resulting embeddings are concatenated and passed through an MLP projection layer to obtain a latent representation with a hidden dimension of 128. To train the multimodal encoder Φ, we attach an inverse dynamics head g on top of the latent representation. The encoder and the inverse dynamics head are jointly optimized to predict the action given two consecutive observations, i.e.,

$$\begin{aligned} \hat{a}_t &= g(\Phi(s_t), \Phi(s_{t+1})) \\ \mathcal{L}_{\text{idm}}(g, \Phi) &= \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}} \left[\|\hat{a}_t - a_t\|_2^2 \right]. \end{aligned} \tag{14}$$

We apply the same pre-training strategy to the Bi-DexHands benchmark, despite it using only low-dimensional observations (with no image modality). This is due to the high dimensionality of the state in Bi-DexHands, which exceeds 400 dimensions; the pre-trained encoder serves to compress this into a more manageable representation. As no image modalities are present, we use only an MLP-based encoder to project the input down to a 64-dimensional latent representation, and randomly mask s_{t+1} with a probability of 20% during training. The same encoder is shared across SaND and all baselines.

B.4 VLA Settings

For our VLA experiments in Figure 4 of Section 5.4, we adopt SigLIP-2 [43] as the vision-language foundation model and finetune only its last 4 layers to keep computational overhead manageable. We leverage the same inverse dynamics loss in Equation (14) to finetune the model. We train and evaluate on three DexMimicGen tasks (Box Cleanup, Drawer Cleanup, and Tray Lift) that share the same 24-dimensional action space, enabling joint multi-task learning. As the images in the datasets are 84×84 in size, we resize them to 224×224 to match SigLIP-2's input resolution. Regarding the language modality, each task is paired with a natural language instruction describing the goal, as listed in Appendix A.1.

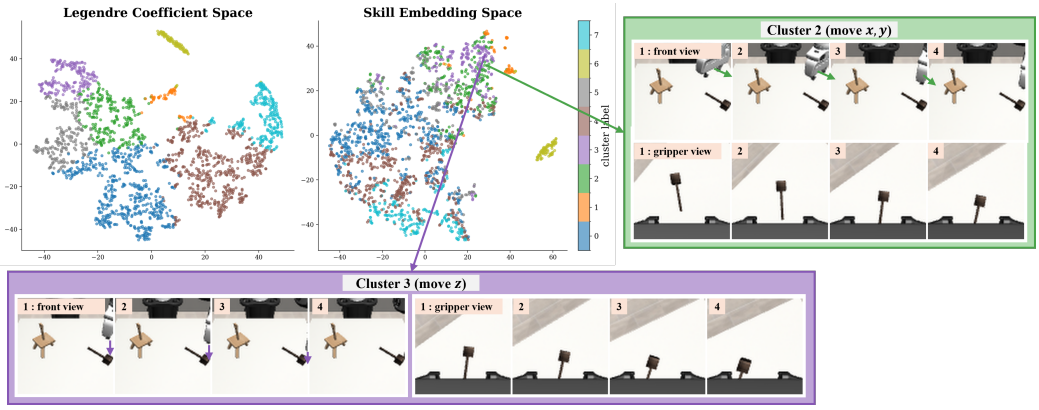
C Additional Experiments

C.1 Visualization of Legendre Coefficients

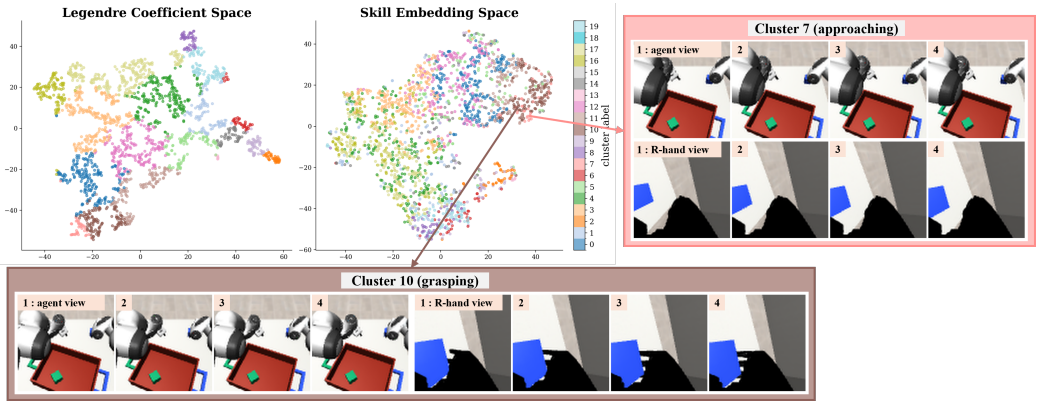
In Figure 11, we visualize the t-SNE embeddings of the Legendre coefficients $\mathrm{Leg}(\gamma'_\phi)$ (left panel) and the skill embeddings z (right panel) for the Threading (MimicGen) and Tray Lift (DexMimicGen) tasks. We perform K-means clustering in the Legendre coefficient space, and the resulting cluster labels are used to color the points in both panels – i.e., points sharing the same color correspond to skills with similar noise schedule geometry.

As shown in both Threading and Tray Lift tasks, the cluster labels are occasionally entangled in the skill embedding space, indicating that neighboring skill embeddings may belong to different noise schedule clusters. For example, in the Threading task (Figure 11(a)), cluster 2 (green) and cluster 3 (violet) appear entangled as their corresponding skill embeddings are temporally consecutive: cluster 2 corresponds to planar movements along the x- and y-axes, while cluster 3 additionally involves downward motion along the z-axis. However, these two clusters are well-separated in the Legendre coefficient space. In the Tray Lift task (Figure 11(b)), cluster 7 (peach) corresponds to the skill approaching the blue cube, while cluster 10 (brown) represents the subsequent skill making contact with the blue cube in preparation for lifting. Similarly, the temporal continuity between these skills leads to their entanglement in the skill embedding space, while they remain clearly disentangled in the Legendre coefficient space.

Such observations offer a direct intuition for why Legendre-based clustering yields better performance than skill embedding-based clustering for adaptive denoising, as reported in Table 7.



(a) Threading Task (MimicGen)

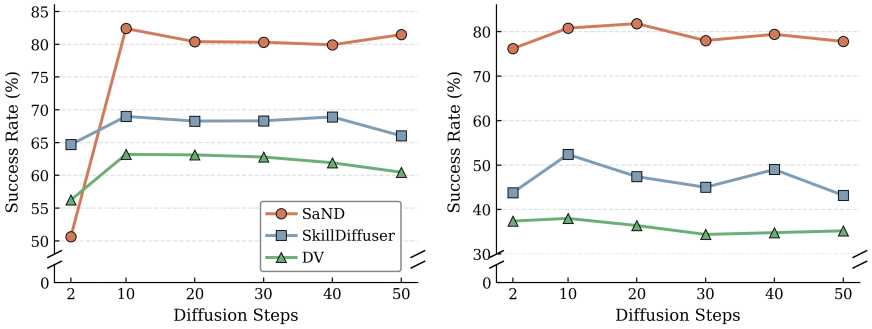


(b) Tray Lift Task (DexMimicGen)

Figure 11: t-SNE visualization of the Legendre coefficients (left panel) and skill embeddings (right panel): Colors indicate K-means cluster labels computed in the Legendre coefficient space and shared across both panels; identical colors indicate skills with similar noise-schedule shapes.

C.2 Performance Ceiling with Increasing Denoising Steps

In Figure 12, we visualize the performance of SaND, SkillDiffuser, and DV with respect to the number of denoising steps $\{2, 10, 20, 30, 40, 50\}$ on the Coffee Preparation (MimicGen) and Can Sort (DexMimicGen) tasks. For both tasks, the success rates of all three methods saturate as the step budget grows from 2 to 10, with no meaningful improvement beyond 10 steps. Yet, SaND attains a markedly higher ceiling than the baselines (SkillDiffuser, DV). These gains stem from jointly learning a skill-conditioned noise schedule with the skill diffusion planner, which tightens the ELBO at its source rather than relying on a fixed noise schedule [37]. This result confirms that the attainable performance ceiling is governed by the tightness of the bound, highlighting the importance of learning the noise schedule.



(a) Coffee Preparation Task (MimicGen) (b) Can Sort Task (DexMimicGen)

Figure 12: Performance with respect to the number of denoising steps

We further analyze why simply increasing the number of denoising steps is insufficient, and motivate the need for learning a noise schedule from a theoretical perspective. From the continuous-time variational inference perspective [14], the Brownian motion driving the forward diffusion can be treated as an infinite-dimensional latent variable. Applying variational inference over this latent variable, analogous to a Variational AutoEncoder (VAE) with an infinite-dimensional code, yields a continuous-time ELBO (CT-ELBO) that lower-bounds the log-likelihood.

The variational gap between the CT-ELBO and the true log-likelihood admits a closed form:

$$\log p(x^1, 1) = \underbrace{\int_0^1 \mathbb{E}\left[\|a(\boldsymbol{x}^k, k) - \sigma^\top \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}^k,\, 1-k)\|^2\right] dk}_{\text{reducible by increasing diffusion steps}} + \underbrace{\mathcal{E}_\infty}_{\substack{\text{irreducible by} \\ \text{increasing diffusion steps}}}, \tag{15}$$

$$\text{where } \mathcal{E}_\infty = \mathbb{E}\left[-\frac{1}{2}\int_0^1 \|a(\boldsymbol{x}^k, k)\|^2 dk + \log p_0(\boldsymbol{x}^1) - \int_0^1 \nabla_{\boldsymbol{x}} \cdot \mu(\boldsymbol{x}^k, k) dk\right].$$

Here a is the drift of the inference SDE, the counterpart of a VAE, i.e., $a \approx \sigma^\top(k)\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})$[3] and $\mu(\boldsymbol{x}^k, k) := \frac{1}{2}(\partial_k \log \alpha^k)\left[\boldsymbol{x}^k + 2\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}^k)\right]$.

That is, the gap is the time integral of the squared error between the inference drift a and the optimal drift $\sigma^\top \nabla_{\boldsymbol{x}} \log p$ (the ground truth score) over the noise schedule, and vanishes if and only if a recovers the true score exactly. Increasing the number of denoising steps only refines the numerical approximation of this integral; it cannot reduce the integrand itself, which is the point-wise score error at each timestep. Fortunately, the scheduler can affect a and μ of $\mathcal{E}_\infty$ in Equation (15), tightening the ELBO even with a smaller step budget.

C.3 Additional Benchmark Experiments

Our method targets contact-rich and dexterous manipulation, where action dimensions are tightly coupled by contact constraints and the coupling patterns differ across skills. This is precisely the

[3] Here, $\sigma^\top$ denotes the transpose of σ.

regime in which a skill-conditioned noise schedule is most advantageous, motivating our selection of MimicGen, DexMimicGen, and Bi-DexHands as primary benchmarks. To verify that SaND also maintains consistent performance on standard diffusion policy benchmarks, we additionally evaluate on Push-T [9]. The observation consists of a 96×96 RGB image with the agent pose, and the action is a 2-dimensional target position. We use the official dataset of 206 expert demonstrations without modification. As shown in Table 4, SaND marginally outperforms DP, though the gap is negligible. This is expected, as the 2-dimensional action space offers limited cross-dimensional structure for a per-dimension schedule to exploit, and performance on this benchmark is already near saturation.

Table 4: Performance on the Push-T benchmark

Methods	K	**Push-T**
DP	10	0.9145$\pm$0.0075
SaND	10	**0.9195**$\pm$**0.0005**

C.4 Analysis on Inference Cost

In Table 5, we measure the inference cost of SaND on a single RTX 4090, reporting FLOPs and wall-clock time per skill trajectory τ generation. We compare three variants: SaND, SaND+AD, which applies adaptive denoising, and SaND-Schedule, which replaces the learned noise schedule with a fixed one (the ablation variant in Table 3). The learned noise schedule adds 0.31% in FLOPs and 1.7% in wall-clock time over SaND-Schedule, supporting the negligible-overhead claim in Section 4.3. Adaptive denoising reduces FLOPs by 56.8% and wall-clock time by 48.7% relative to SaND, confirming that fewer denoising steps translate into lower inference cost.

Table 5: Analysis on inference cost

Methods	**FLOPs**	**Wall-clock Time**
SaND	777.1 M	5.065 ms
SaND+AD	335.4 M	2.599 ms
SaND-Schedule	774.6 M	4.978 ms

C.5 Additional Ablation Results on Skill-Adaptive Noise Diffusion

Table 6 presents an additional ablation study to determine whether the gains of SaND require both skill conditioning and per-dimension scheduling. **SaND-SkillCond** removes skill conditioning and learns a single schedule shared across all skills. **SaND-Global** learns a single noise schedule shared across all action dimensions, removing dimension-wise scheduling. Removing skill conditioning results in an average drop of 20.4 pp, as the schedule can only fit the average coupling pattern rather than skill-specific ones. Similarly, removing dimension-wise scheduling leads to a 23.3 pp drop, as a single schedule must serve all action dimensions simultaneously, causing conflicting per-dimension gradients. These results confirm that both mechanisms are independently essential.

Table 6: Additional ablation on skill-adaptive noise diffusion

Benchmark: Contact-Rich Manipulation (MimicGen [28]) – Low Dimensional Observation							
Methods	**Square**	**Threading**	**Mug**	**Hammer**	**Coffee**	**Kitchen**	**Average**
SaND	**60.9**$\pm$**1.9**	**66.9**$\pm$**5.5**	**60.0**$\pm$**1.6**	**58.4**$\pm$**1.9**	**82.4**$\pm$**2.2**	**55.3**$\pm$**4.4**	**64.0**$\pm$**1.0**
SaND-SkillCond	33.6$\pm$2.9	33.6$\pm$5.0	50.9$\pm$4.5	40.6$\pm$2.2	64.3$\pm$1.9	38.4$\pm$3.9	43.6$\pm$1.8
SaND-Global	30.3$\pm$3.8	36.1$\pm$10.2	45.6$\pm$10.9	42.0$\pm$4.4	54.2$\pm$28.0	35.9$\pm$1.6	40.7$\pm$4.1

C.6 Detailed Ablation Results on Schedule-Driven Adaptive Denoising

Here, we present detailed ablation results on schedule-driven adaptive denoising across individual tasks, complementing Figure 3 of Section 5.3. Specifically, we compare fixed-K variants of SaND and SaND+AD with skill embedding-based clustering. Note that the highest performance is highlighted in **bold**, and the second highest is underlined. As shown, for all cases except the Coffee task in

MimicGen, SaND+AD (Leg) demonstrates improved performance over fixed-K variants, confirming that the performance gain stems from dynamically adapting K per skill according to the corresponding Legendre coefficient cluster. Moreover, the alternative variant with skill embedding-based clustering, SaND+AD (skill), consistently yields degraded performance compared to Legendre-based clustering, further validating our design choice.

Table 7: Detailed ablation on schedule-driven adaptive denoising

Benchmark: Contact-Rich Manipulation (MimicGen [28]) – Low Dimensional Observation								
Methods	**Metric**	**Square**	**Threading**	**Mug**	**Hammer**	**Coffee**	**Kitchen**	**Average**
SaND ($K = 10$)	SR	60.9±1.9	**66.9**±5.5	**60.0**±1.6	58.4±1.9	**82.4**±2.2	**55.3**±4.4	**64.0**±1.0
SaND ($K = 4$)	SR	60.2±1.0	64.0±4.1	51.0±2.7	59.4±2.2	81.9±1.6	53.8±1.8	61.7±0.4
SaND ($K = 3$)	SR	58.0±3.6	60.4±3.9	54.4±6.0	58.0±2.4	78.5±3.8	50.6±3.5	60.0±1.3
SaND+AD (skill)	SR	61.4±3.0	64.5±3.0	55.7±7.2	58.9±2.0	80.8±3.6	54.6±3.6	62.6±1.6
	K	3.7 ±0.2	3.5 ±0.3	3.6 ±0.4	3.9 ±0.2	3.5 ±0.2	3.8 ±0.2	3.7 ±0.2
SaND+AD (Leg)	SR	**61.7**±3.2	**66.9**±3.9	55.9±5.0	**59.6**±2.2	81.0±2.8	54.6±3.6	63.3±0.8
	K	3.7 ±0.1	3.4 ±0.2	3.5 ±0.6	4.0 ±0.6	3.5 ±0.2	3.8 ±0.2	3.7 ±0.2
Benchmark: Bimanual Dexterous Manipulation (DexMimicGen [16]) – Multimodal Observation								
Methods	**Metric**	**Box**	**Drawer**	**Tray**	**Pouring**	**Coffee**	**Can**	**Average**
SaND ($K = 10$)	SR	79.4±5.5	**81.0**±1.4	40.6±6.0	**71.0**±13.3	**32.0**±4.6	80.8±8.7	**64.1**±2.7
SaND ($K = 6$)	SR	78.0±5.3	77.6±4.5	36.4±6.5	60.4±13.3	29.4±5.9	79.8±10.2	60.3±3.4
SaND ($K = 5$)	SR	79.0±5.9	79.6±4.1	38.2±9.2	60.6±13.8	29.4±7.3	80.2±9.5	61.2±1.8
SaND+AD (skill)	SR	80.0±6.6	79.0±6.4	38.8±5.5	60.2±11.8	30.6±7.9	78.6±12.4	61.2±3.3
	K	5.4 ±0.2	5.4 ±0.2	5.2 ±0.1	5.7 ±0.3	4.8 ±0.2	5.6 ±0.6	5.3 ±0.1
SaND+AD (Leg)	SR	**83.6**±4.9	80.6±4.8	**42.0**±7.8	63.6±10.8	30.8±5.3	**82.2**±8.9	63.8±1.9
	K	5.4 ±0.2	5.3 ±0.3	5.1 ±0.1	5.9 ±0.1	5.0 ±0.2	5.8 ±0.9	5.4 ±0.1

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the paper's main contributions.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of our work are discussed in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: Our paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We include the details of benchmark environment setting in Appendix A and experiment settings in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide the source code of our method in the supplemental material.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the benchmark environment configuration in Appendix A and the experimental settings in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report confidence intervals for the experiment results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We include information on the computer resources in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: All authors have reviewed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no societal impact of the work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original papers of the datasets used for training (e.g., MimicGen [28] and DexMimicGen [16]).

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in our paper does not involve LLMs as any core components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.