DEXPRIOR: BOOTSTRAPPING ONLINE DEXTEROUS ADAPTATION WITH MULTI-TASK LATENT PRIORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Dexterous manipulation is challenging to learn from scratch due to its contact-rich interactions and long-horizon dynamics. While demonstrations can provide a useful initialization for online adaptation, conventional behavioral policies are typically task-specific and may degrade when trained on heterogeneous multi-task data. We present DexPrior, a framework that learns task-agnostic latent skills from multi-task demonstrations, providing a reusable prior for online adaptation. DexPrior adopts an Encoder–Prior–Decoder architecture to learn reusable skills from shared state representations and privileged information. The pretrained skill prior is then reused for online adaptation, where a diffusion-based action decoder and an additional latent-skill transition enable skill-level exploration within the two-layer MDP formulation. On DexJoCo, we evaluate online adaptation against from-scratch and pretrained-policy RL baselines, demonstrating the effectiveness of multi-task skills for long-horizon dexterous manipulation.

1 INTRODUCTION

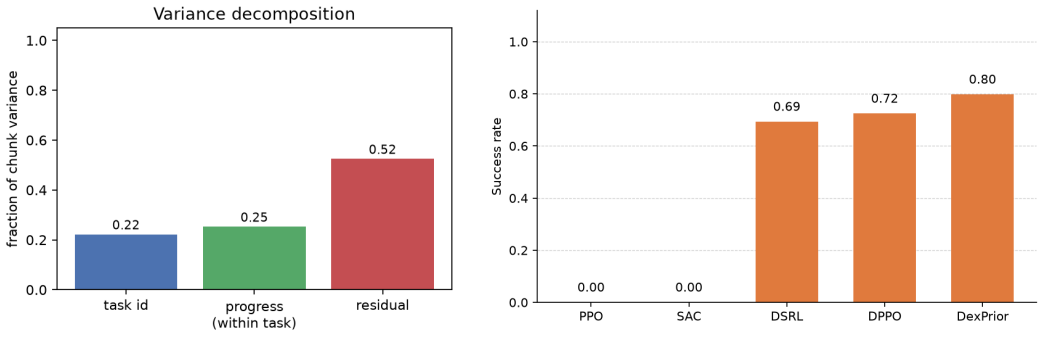
Dexterous manipulation requires a robot to coordinate many degrees of freedom under complex contact interactions and long-horizon dynamics. Compared with conventional robotic manipulation, successful execution often depends on temporally structured behaviors, such as establishing contact, reorienting objects, and coordinating multiple fingers over extended horizons. Learning such behaviors directly from trial and error is particularly challenging for high-dimensional dexterous hands, where the large action space and numerous potential contacts make exploration inefficient (Rajeswaran et al., 2017; Arunachalam et al., 2023; Luo et al., 2025). Demonstrations can therefore provide a valuable initialization for online reinforcement learning by exposing the agent to meaningful behaviors before task-specific exploration.

Recent advances in dexterous imitation learning (An et al., 2025) have further demonstrated the potential of large-scale human demonstrations and generative policies for learning complex manipulation behaviors. However, directly learning a behavioral policy from heterogeneous demonstrations remains challenging. Different tasks may involve different objects, interaction patterns, and behavioral objectives, making the mapping from a shared state representation to actions inherently multimodal. A single behavioral policy must therefore represent substantially different behaviors within the same action space, while the target task may not be known during offline pretraining. As shown in Figure 1a, approximately 22% of the variance in multi-task demonstrations cannot be explained without conditioning on the specific task. This raises a fundamental question:

Can heterogeneous demonstrations be organized into a reusable representation of behaviors rather than directly collapsed into a single multi-task behavioral policy?

Based on this motivation, we introduce **DexPrior**, a framework that adopts an Encoder–Prior–Decoder architecture to learn task-agnostic latent skills from heterogeneous multi-task offline demonstrations and reuses them as a prior for online adaptation. During offline training, an encoder uses privileged short-horizon future motion and contact information to infer the latent skill underlying a demonstrated action chunk, while a state-conditioned prior learns to infer the corresponding skill using only deployment-time information. A skill-conditioned diffusion decoder then maps the inferred skill and the current state to a temporally coherent action chunk. The resulting latent space provides a shared interface through which behaviors from different tasks can be repre-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Variance decomposition.

(b) Overall performance comparison.

Figure 1: *Left.* Variance decomposition for multi-task demonstrations; *Right.* Overall performance comparison against from-scratch and pretrained-policy RL baselines.

sented without requiring explicit task labels. During online adaptation, we freeze the learned skill prior and introduce a lightweight latent residual policy to explore the pretrained skill space, while simultaneously adapting the diffusion decoder. This effectively extends the diffusion-policy inner MDP with an additional latent-skill transition before the denoising transitions, while preserving the original two-layer environment–denoising MDP structure (Ren et al., 2025; Ding et al., 2026; Wen et al., 2025). The resulting formulation enables task-level rewards to jointly provide credit assignment for latent skill exploration and action generation.

We evaluate DexPrior on DexJoCo (Wang et al., 2026), a challenging benchmark for long-horizon and fine-grained dexterous manipulation. We compare against both from-scratch and pretrained-policy RL baselines. The overall performance comparison can be seen in Figure 1b.

In summary, our contributions are threefold:

- We introduce **DexPrior**, a task-agnostic latent skill prior learned from heterogeneous multi-task dexterous manipulation demonstrations, enabling reusable behavioral representations without requiring task labels during offline pretraining.
- We develop an Encoder–Prior–Decoder framework that uses privileged future motion and contact information to abstract temporally structured skills, while learning a deployment-time conditional prior and a skill-conditioned diffusion decoder.
- We introduce latent-skill exploration for online reinforcement learning by extending the diffusion denoising MDP with an additional latent-skill transition, enabling joint adaptation of skill selection and diffusion-based action generation.

2 Preliminaries

Problem setting. We focus on dexterous manipulation tasks in which robot arms equipped with dexterous hands interact with multiple objects to achieve desired states. The state $\mathcal{S}$ is formulated as $(\mathcal{S}_p, \mathcal{S}_o)$, where $\mathcal{S}_p$ and $\mathcal{S}_o$ denote the robot states and the object states, respectively. Given a state $\mathcal{S}$, a policy π_θ selects an action $\mathcal{A}$ to drive the environment to a new state. The environment provides a non-zero reward $\mathcal{R}$ only upon reaching a goal state. To overcome the initial exploration dilemma caused by sparse rewards, we learn an initial policy π_θ^{init} from multi-task offline data $\mathcal{D} = \{(\mathcal{S}_p^i, \mathcal{S}_o^i), \hat{\mathcal{A}}^i\}$, where the object states $\mathcal{S}_o^i$ are heterogeneous across tasks due to variations in the number, types, and roles of manipulated objects. This setting is more general, as the target online task may be unknown during offline pretraining, or corresponding offline data may be unavailable in many practical scenarios. To model the complex and multimodal action distributions arising in dexterous manipulation, we use diffusion models (Ho et al., 2020; Song et al., 2020) as the policy backbone.

Diffusion Policy Policy Optimization. Following DPPO (Ren et al., 2025), we formulate diffusion policy as a two-layer MDP consisting of an outer environment MDP and an inner denoising MDP.

In the outer MDP, given the current environment state $\mathcal{S}_t$, the diffusion policy generates an action chunk $\mathcal{A}_t$ executed in the environment, yielding a reward $\mathcal{R}_t$ and the next state $\mathcal{S}_{t+1}$.[1] Formally,

$$\text{The outer MDP:} \quad \mathcal{S}_t \xrightarrow{\pi_\theta} \mathcal{A}_t \xrightarrow{\mathcal{T}} (\mathcal{R}_t, \mathcal{S}_{t+1}), \tag{1}$$

where $\mathcal{T}$ denotes the environmental transition dynamics. The action chunk $\mathcal{A}_t$ is generated by an inner denoising MDP. Starting from Gaussian noise $\mathcal{A}_t^K \sim \mathcal{N}(0, \mathbf{I})$, the diffusion policy iteratively denoises the actions through K steps:

$$\text{The inner MDP:} \quad \mathcal{A}_t^K \xrightarrow{\pi_\theta^K} \mathcal{A}_t^{K-1} \xrightarrow{\pi_\theta^{K-1}} \cdots \xrightarrow{\pi_\theta^1} \mathcal{A}_t^0 = \mathcal{A}_t. \tag{2}$$

At each denoising step k, the transition is parameterized by a Gaussian policy:

$$\mathcal{A}_t^{k-1} \sim \pi_\theta^k \left(\mathcal{A}_t^{k-1} \mid \mathcal{S}_t, \mathcal{A}_t^k \right), \tag{3}$$

which allows the task-level reward obtained after executing $\mathcal{A}_t^0$ to propagate through the entire denoising process. Thus, the environment MDP and the denoising MDP together constitute the two-layer Diffusion Policy MDP underlying policy optimization for the target online task. DexPrior extends this formulation by introducing an additional latent-skill MDP transition between the environment state and the denoising variables.

Latent skill representation. We consider a VAE-like framework with a learned conditional prior, where the latent space captures temporally structured behaviors. Let $\mathbf{x}_t$ denote the information available to the policy at deployment time, and let $\mathbf{y}_t$ denote additional information available during skill learning that reveals the behavior being executed. A latent skill $\mathbf{z}_t \in \mathcal{Z}$ is inferred by an encoder $q_\phi(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{y}_t)$, while a conditional prior $p_\psi(\mathbf{z}_t \mid \mathbf{x}_t)$ predicts the distribution of plausible skills using only deployment-available information. A decoder $\pi_\theta(\mathcal{A}_t \mid \mathbf{x}_t, \mathbf{z}_t)$ then maps the latent skill back to executable behavior. The three components can be jointly optimized through a variational objective:

$$\mathcal{L}_{\text{skill}} = -\mathbb{E}_{q_\phi(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{y}_t)} \left[\log \pi_\theta(\mathcal{A}_t \mid \mathbf{x}_t, \mathbf{z}_t) \right] + \beta D_{\text{KL}} \left(q_\phi(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{y}_t) \,\|\, p_\psi(\mathbf{z}_t \mid \mathbf{x}_t) \right).$$

The joint optimization injects future information into the latent space while ensuring that the learned skills remain independently inferable by the prior from $\mathbf{x}_t$ alone.

3 METHOD

In this section, we present the details of the proposed DexPrior, a practical offline-to-online algorithm for solving dexterous manipulation problems. Unlike previous methods (Ren et al., 2025; Wagenmaker et al., 2025; Chandra et al., 2025) that rely on in-domain offline data for policy initialization, we learn a task-agnostic latent skill prior from multi-task demonstrations. Rather than initializing online adaptation with a task-specific behavioral policy, DexPrior extracts reusable skills shared across diverse tasks and uses them as a structured prior for task-specific online fine-tuning.

As shown in Figure 2, DexPrior learns task-agnostic latent skills from multi-task demonstrations using an Encoder–Prior–Decoder architecture. The encoder leverages privileged information to infer latent skills, while the prior is jointly optimized to match the encoder's latent distribution, enabling skill inference using only deployment-time information. The decoder adopts diffusion as its backbone to synthesize action chunks conditioned on the sampled latent skill. After policy initialization, DexPrior retains the pretrained prior in a frozen state and performs online adaptation by keeping the diffusion decoder trainable and treating the denoising process as an MDP. In parallel, a latent residual network is randomly initialized to enable exploration in the latent skill space. We incorporate the latent residual as an additional transition at the beginning of the inner MDP, extending the denoising process while maintaining the original two-layer MDP structure (Ren et al., 2025) and enabling consistent credit assignment across latent-skill exploration and diffusion denoising.

3.1 MULTI-TASK LATENT SKILL LEARNING

Instead of directly learning a behavioral cloning policy π_{BC} that maps the state $\mathcal{S}_t$ to an action chunk $\mathcal{A}_t$ from multi-task demonstrations, we explicitly learn a latent space $\mathcal{Z} \subseteq \mathbb{R}^n$ that captures task-dependent variations in temporally structured behaviors. Specifically, our offline policy consists of

[1] Strictly speaking, the action chunk $\mathcal{A}_{t:t+T}$ is executed sequentially, resulting in a sequence of rewards $\mathcal{R}_{t:t+T}$ and ultimately transitioning the environment to $\mathcal{S}_{t+T}$. We abstract this process using the standard RL notation for simplicity.

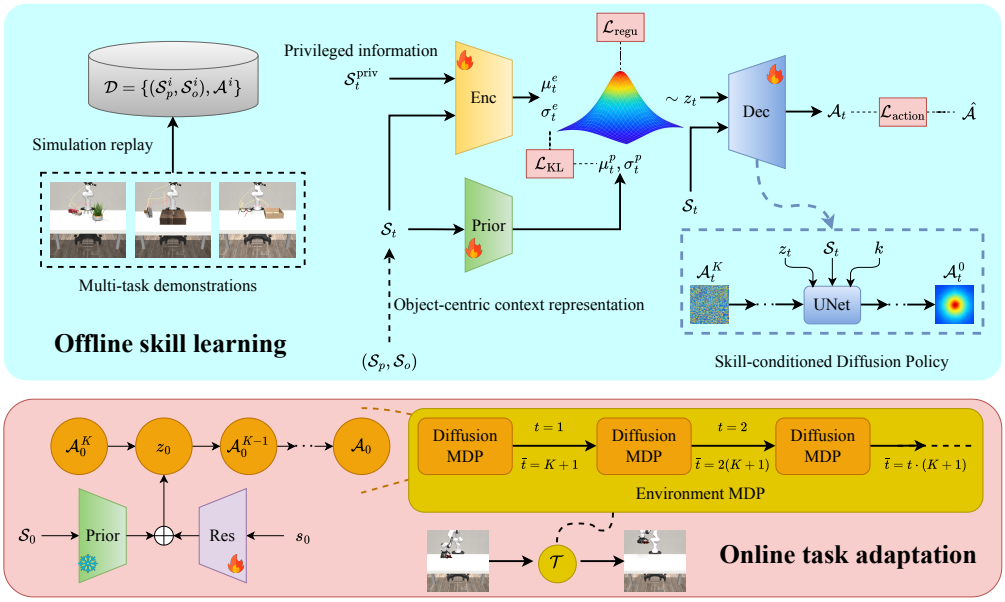


Figure 2: The two-stage training procedure of DexPrior.

a prior that infers a latent skill $\mathbf{z}_t \in \mathcal{Z}$ and a decoder that generates the action chunk $\mathcal{A}_t$ conditioned on the state $\mathcal{S}_t$ and latent skill $\mathbf{z}_t$. Formally:

$$\pi(\mathcal{A}_t \mid \mathcal{S}_t) = \int_{\mathcal{Z}} \pi_\theta(\mathcal{A}_t \mid \mathcal{S}_t, \mathbf{z}_t)\, p_\psi(\mathbf{z}_t \mid \mathcal{S}_t)\; d\mathbf{z}_t, \tag{4}$$

where p_ψ and π_θ denote the prior and the decoder, respectively. To learn the latent skill, we introduce an encoder $q_\phi(\mathbf{z}_t \mid \mathcal{S}_t, \mathcal{S}_t^{\text{priv}})$ with access to the additional privileged information $\mathcal{S}_t^{\text{priv}}$, which is crucial for identifying the latent skills underlying the action chunks. We construct a unified object-centric context representation to accommodate heterogeneous object states, the details of which is presented in Appendix B.

Privileged Information for Skill Abstraction. The privileged information $\mathcal{S}_t^{\text{priv}}$ provides the encoder with short-horizon future information that is unavailable at deployment time. For an action chunk with a horizon H, it consists of the fingertip motion and contact trajectories over the next H steps, together with the robot-state residuals at the end of the horizon:

$$\mathcal{S}_t^{\text{priv}} = \left[\mathcal{F}_{t+1:t+H}, \Delta\mathcal{S}_{(p)t+H}\right]. \tag{5}$$

Here, $\mathcal{F}_{t+1:t+H}$ contains the relative fingertip pose changes and contact forces over the future horizon, while $\Delta\mathcal{S}_{(p)t+H}$ captures the terminal robot-state changes. By providing these future motion and contact cues to the posterior encoder, $\mathcal{S}_t^{\text{priv}}$ facilitates the abstraction of latent skills underlying the demonstrated action chunks. In contrast, the learned prior does not have access to this privileged information and must infer the corresponding latent skills solely from the deployment-time state $\mathcal{S}_t$.

Learning objective. For offline training, DexPrior first converts the multi-task demonstrations $\mathcal{D} = \{(\mathcal{S}_p^i, \mathcal{S}_o^i), \hat{\mathcal{A}}^i\}$ into a homogenized and chunked dataset $\widetilde{\mathcal{D}} = \{(\mathcal{S}_t, \mathcal{S}_t^{\text{priv}}), \hat{\mathcal{A}}_t\}$, yielding a unified data representation across heterogeneous tasks. Following Luo et al. (2024), we jointly optimize the encoder, prior, and decoder with an objective that combines action reconstruction, latent distribution alignment, and latent smoothness regularization. The overall objective is as follows:

$$\min_{\phi,\psi,\theta} \mathcal{L}_{\text{offline}} = \mathbb{E}_{(\mathcal{S}_t, \mathcal{S}_t^{\text{priv}}, \hat{\mathcal{A}}_t)\sim\widetilde{\mathcal{D}}}\left[\mathcal{L}_{\text{action}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}} + \lambda_{\text{regu}}\mathcal{L}_{\text{regu}}\right], \tag{6}$$

where $\mathcal{L}_{\text{action}}$ denotes the diffusion action reconstruction loss, $\mathcal{L}_{\text{KL}}$ aligns the encoder posterior with the state-conditioned prior, and $\mathcal{L}_{\text{regu}}$ encourages temporal smoothness in the latent skill space.

3.2 SKILL-CONDITIONED DIFFUSION POLICY

To capture the multi-modal nature of action chunks in dexterous manipulation, DexPrior adopts a diffusion model as the decoder backbone. Given the current state $\mathcal{S}_t$ and a latent skill $\mathbf{z}_t$, the decoder models the conditional distribution over an action chunk $\mathcal{A}_t$ through an iterative denoising process:

$$\pi_\theta(\mathcal{A}_t \mid \mathcal{S}_t, \mathbf{z}_t) = \prod_{k=1}^{K} \pi_\theta^k \left(\mathcal{A}_t^{k-1} \mid \mathcal{A}_t^k; \mathcal{S}_t, \mathbf{z}_t\right), \tag{7}$$

where $\mathcal{A}_t^K$ is initialized from a standard Gaussian distribution and progressively denoised to obtain $\mathcal{A}_t^0$, which is the generated action chunk. Following Chi et al. (2025), we employ a temporal 1D UNet (Ronneberger et al., 2015) to perform noise prediction over the action sequence. The state and skill information are encoded into a shared conditioning vector and injected into the UNet throughout the denoising process, allowing the generated action chunk to be jointly determined by the current environment state and the selected latent skill.

For each denoising step k, the decoder predicts the noise component in the noisy action chunk:

$$\hat{\boldsymbol{\epsilon}}_t = \boldsymbol{\epsilon}_\theta \left(\mathcal{A}_t^k; \mathcal{S}_t, \mathbf{z}_t, k\right). \tag{8}$$

Specifically, the action chunk consists of H consecutive actions, which are flattened and reshaped into a temporal sequence before being processed by the 1D UNet. The conditioning features derived from $\mathcal{S}_t$ and $\mathbf{z}_t$ are kept fixed throughout the denoising chain, while the denoising step k is encoded using a sinusoidal timestep embedding. We use feature-wise linear modulation (FiLM) (Perez et al., 2018) to inject the conditioning features into the residual blocks of the UNet. This design enables the decoder to decode the latent skill into a temporally coherent action chunk while conditioning its execution on the current state.

Diffusion training objective. We train the decoder using the standard ϵ-prediction objective of DDPM (Ho et al., 2020). Following Nichol & Dhariwal (2021), we adopt a cosine noise schedule for the forward diffusion process. Given a clean action chunk $\hat{\mathcal{A}}_t$ and a randomly sampled diffusion step k, the forward process produces

$$\mathcal{A}_t^k = \sqrt{\bar{\alpha}_k}\hat{\mathcal{A}}_t + \sqrt{1-\bar{\alpha}_k}\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{9}$$

During offline training, $\mathbf{z}_t$ is sampled from the encoder posterior $q_\phi(\mathbf{z}_t \mid \mathcal{S}_t, \mathcal{S}_t^{\text{priv}})$, such that the decoder learns to associate inferred latent skills with their corresponding demonstrated behaviors. The denoising objective is defined by matching the predicted noise to the noise injected during the forward diffusion process:

$$\mathcal{L}_{\text{action}} = \mathbb{E}_{\substack{(\mathcal{S}_t, \mathcal{S}_t^{\text{priv}}, \hat{\mathcal{A}}_t) \sim \widetilde{\mathcal{D}} \\ \mathbf{z}_t \sim q_\phi(\mathbf{z}_t \mid \mathcal{S}_t, \mathcal{S}_t^{\text{priv}}),\, k, \boldsymbol{\epsilon}}} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\mathcal{A}_t^k; \mathcal{S}_t, \mathbf{z}_t, k\right) \right\|_2^2 \right]. \tag{10}$$

The action, KL, and latent smoothness losses are jointly optimized to train the encoder, prior, and decoder, with the action loss also propagating through the sampled latent skill to the encoder.

3.3 ONLINE ADAPTATION WITH LATENT PRIOR

We present the two-stage training pipeline of DexPrior in Algorithm 1. After offline pretraining on multi-task demonstrations, the prior and decoder together form the initial policy for exploration on the target online task. A latent residual network is then introduced to explore and adapt the pretrained skill space through online reinforcement learning. This results in an extended inner MDP, where an additional latent-skill transition precedes the diffusion denoising steps while preserving the original two-layer MDP structure.

Online skill adaptation. After offline pretraining, we freeze the learned prior to preserve the task-agnostic skill distribution and introduce a lightweight latent residual policy π_{res} for online adaptation. Given the current state s_t[2], the frozen prior predicts the parameters $\boldsymbol{\mu}_t^p$ and $\boldsymbol{\sigma}_t^p$ of the

[2]Here, lowercase s_t denotes the task-specific state returned by the online environment, while $\mathcal{S}_t$ denotes the corresponding state representation with object-centric context used by the prior.

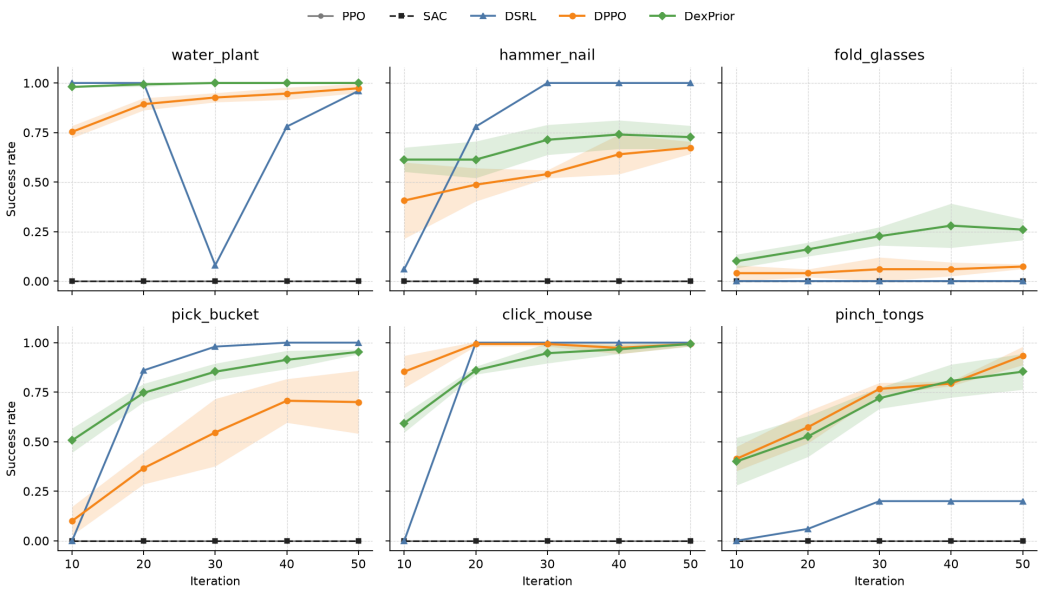


Figure 3: Close-loop evaluation curves on online training checkpoints.

state-conditioned latent distribution, while the residual policy predicts the parameters of a Gaussian residual distribution:

$$(\boldsymbol{\mu}_t^{\text{res}}, \boldsymbol{\sigma}_t^{\text{res}}) = \pi_{\text{res}}(s_t), \qquad \boldsymbol{\delta z}_t \sim \mathcal{N}\left(\boldsymbol{\mu}_t^{\text{res}}, \text{diag}\left((\boldsymbol{\sigma}_t^{\text{res}})^2\right)\right). \tag{11}$$

The latent skill is then obtained by modulating the residual with the state-dependent scale of the pretrained prior:

$$\mathbf{z}_t = \boldsymbol{\mu}_t^p + \boldsymbol{\sigma}_t^p \odot \boldsymbol{\delta z}_t. \tag{12}$$

Thus, the pretrained prior provides the state-dependent scale for latent exploration, while the residual policy learns to steer the latent skill toward task-specific behaviors.

Extended inner MDP. Following DPPO (Ren et al., 2025), we formulate the diffusion denoising process as an inner MDP. Starting from $\mathcal{A}_t^K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, each denoising step constitutes a transition

$$\mathcal{A}_t^k \to \mathcal{A}_t^{k-1} \sim \pi_\theta^k\left(\mathcal{A}_t^{k-1} \mid \mathcal{S}_t, \mathbf{z}_t, \mathcal{A}_t^k\right), \tag{13}$$

until the final action chunk $\mathcal{A}_t^0$ is obtained and executed in the environment. DexPrior extends this inner MDP by introducing the latent-skill sampling as an additional transition before the diffusion denoising process:

$$\mathcal{S}_t \to \mathbf{z}_t \to \mathcal{A}_t^K \to \mathcal{A}_t^{K-1} \to \cdots \to \mathcal{A}_t^0. \tag{14}$$

Here, $\mathbf{z}_t$ is sampled from the latent distribution induced by the frozen prior and the residual policy and is held fixed throughout the subsequent denoising steps. The latent-skill transition and the K denoising transitions therefore form a single inner-MDP trajectory. During online optimization, PPO assigns the same environment-level advantage to these transitions through their respective log-probability ratios, allowing the task reward to jointly update the latent residual policy and the diffusion decoder. This preserves the original two-layer MDP structure of DPPO while extending the inner MDP with an additional latent-skill transition.

4 Experiments

We evaluate DexPrior on the challenging DexJoCo (Wang et al., 2026) benchmark, built on the MuJoCo (Todorov et al., 2012) physics simulation engine. The embodiment consists of a Franka Panda manipulator equipped with an Allegro Hand. We consider six tasks—Hammer Nail, Click Mouse, Pick Bucket, Pinch Tongs, Fold Glasses, and Water Plant—covering fine-grained manipulation, tool use, long-horizon execution, and task reasoning. The state-based multi-task demonstrations are collected by replaying human demonstration trajectories in the simulation environment.

Table 1: Success rates at iteration 50 across six dexterous manipulation tasks. Results for DPPO and DexPrior correspond to the mean and standard error over 3 online training seeds. We have bolded the maximum success rate per task.

Task	PPO	SAC	DSRL	DPPO	DexPrior
Water Plant	0	0	0.96	0.97 ± 0.02	$\mathbf{1.00} \pm 0.00$
Hammer Nail	0	0	**1.00**	0.67 ± 0.03	0.73 ± 0.06
Fold Glasses	0	0	0.00	0.07 ± 0.01	$\mathbf{0.26 \pm 0.05}$
Pick Bucket	0	0	**1.00**	0.70 ± 0.16	0.95 ± 0.01
Click Mouse	0	0	**1.00**	0.99 ± 0.01	0.99 ± 0.01
Pinch Tongs	0	0	0.20	$\mathbf{0.93 \pm 0.05}$	0.85 ± 0.09
Average	0	0	0.693	0.724	**0.798**

Implementation details. For offline training, the six tasks are trained jointly for 100,000 Adam steps with a batch size of 1024. We use a learning rate of 1×10^{-4} with cosine decay to 1×10^{-5}. The diffusion decoder uses $T = 100$ training steps on 8-step action chunks and a 32-dimensional latent space. The KL weight is warmed up to 3×10^{-4} over 5000 steps, with a free-bits threshold of 0.02, and the temporal latent regularizer is weighted by 1×10^{-3}. At deployment, we use the prior mean and deterministic DDIM with $K = 10$ and $\eta = 0$. For online training, each task is fine-tuned for 50 PPO iterations. We optimize the residual policy and diffusion decoder with learning rates of 1×10^{-4} and 1×10^{-5}, respectively. The residual standard deviation is initialized at 0.10 and constrained to $[0.02, 0.20]$. We use a 5-step stochastic DDIM sampler with $\eta = 1$, PPO clipping of 0.10, and a denoising discount of 0.9. Each iteration collects 50 parallel environments with 500 decisions per environment.

4.1 Main Results on DexJoCo

Baselines. We implement PPO (Ren et al., 2025) and SAC (Haarnoja et al., 2018) as from-scratch RL baselines, and adopt RND (Burda et al., 2018) as their exploration bonus. We implement DPPO (Ren et al., 2025) and DSRL-NA (Wagenmaker et al., 2025) as pretrained-policy RL baselines. Both methods share the same multi-task offline demonstrations and diffusion decoder architecture as DexPrior. For online adaptation, DPPO fine-tunes the diffusion policy with PPO for 50 iterations per task, while DSRL-NA freezes the pretrained BC diffusion policy and adapts its initial noise using a SAC policy. Unless otherwise specified, both methods use the same online rollout and optimization settings as DexPrior. We denote DSRL-NA as DSRL for simplicity.

Online evaluation. For each task, we evaluate each checkpoint on 50 episodes initialized with deterministic seeds, where the policy generates and executes action chunks in a closed loop until the task succeeds or the environment reaches its episode horizon. The closed-loop success rate is computed as the fraction of successful episodes among the 50 trials. The results are shown in Figure 3 and Table 1. PPO and SAC achieve zero success rates on all six tasks, highlighting the difficulty of learning from scratch under sparse rewards and long-horizon dynamics. DexPrior exhibits the strongest overall performance, achieving the highest average success rate throughout training. It achieves the fastest improvement at iteration 10 on three of the six tasks and consistently outperforms DPPO on five of the six tasks. Although DSRL reaches a 100% success rate on Hammer Nail, Pick Bucket, and Click Mouse, it performs substantially worse on Fold Glasses and Pinch Tongs. This behavior can be attributed to the fact that DSRL keeps the diffusion denoising policy frozen during online adaptation. As a result, the multi-task offline policy may be biased toward certain task subsets, while its online adaptation mechanism has limited capacity to further refine the denoising process for tasks that are less well represented or poorly captured by the pretrained policy.

Offline evaluation. We evaluate each offline checkpoint using open-loop chunk RMSE and closed-loop success rate. The chunk RMSE measures the root-mean-square error over the entire 8-step predicted action chunk on the validation set. The overall chunk RMSE is computed over all validation windows rather than averaged across tasks. The closed-loop success rate follows the same evaluation protocol as the online evaluation. Since DPPO and DSRL share the same offline checkpoint, we report the offline performance only once to represent the pretrained-policy RL baselines. As shown in Table 2, DexPrior consistently outperforms DPPO in terms of Chunk RMSE on all six

Table 2: Offline evaluation. Chunk RMSE is computed over the full 8-step action chunk on the validation set, while success rate is measured over 50 closed-loop evaluation episodes.

	Chunk RMSE ↓		Success Rate ↑	
Task	DPPO	DexPrior	DPPO	DexPrior
Water Plant	0.213	**0.168**	0.20 (10/50)	**0.96 (48/50)**
Hammer Nail	0.290	**0.287**	0.08 (4/50)	**0.20 (10/50)**
Fold Glasses	0.351	**0.325**	0.00 (0/50)	0.00 (0/50)
Pick Bucket	0.399	**0.378**	0.04 (2/50)	**0.18 (9/50)**
Click Mouse	0.517	**0.431**	**0.52 (26/50)**	0.38 (19/50)
Pinch Tongs	0.329	**0.295**	**0.44 (22/50)**	0.22 (11/50)
Overall	0.376	**0.336**	0.213 (64/300)	**0.323 (97/300)**

tasks. Moreover, DexPrior achieves a higher overall closed-loop success rate of 0.323, indicating that the learned latent skills provide a more effective initialization for downstream online adaptation.

4.2 Model Analysis

Decoder backbone. We replace the diffusion decoder with a deterministic MLP that predicts the action chunk in a single forward pass, without iterative DDIM denoising. Both variants use the same latent skill prior, allowing us to isolate the effect of the decoder backbone. The results are obtained on all six tasks. As shown in Table 3, the MLP decoder achieves a slightly lower chunk RMSE than the diffusion decoder, but obtains a substantially lower closed-loop success rate, indicating the discrepancy between open-loop action prediction error and ong-horizon task execution. These results suggest that the iterative diffusion decoder provides more effective action generation for closed-loop manipulation.

Decoder	Chunk RMSE ↓	Success Rate ↑
MLP	**0.322**	0.267
DexPrior	0.340	**0.323**

Table 3: Ablation study on the decoder backbone.

Latent skill importance. We further investigate the contribution of the learned latent skill by comparing the original prior with two perturbations: *zero*, which sets the latent skill $\mathbf{z}$ to zero, and *shuffle*, which randomly permutes the prior mean within each task. All variants use the same DexPrior checkpoint and share the same DDIM noise for a fair comparison. The results are obtained on all six tasks. As shown in Table 4, removing or mismatching the learned latent skill substantially degrades both open-loop and closed-loop performance. These results demonstrate that the learned latent skill provides essential task-relevant information for both action prediction and successful closed-loop execution.

Variant	Chunk RMSE ↓	Success Rate ↑
Zero	0.535	0.000
Shuffle	0.516	0.003
Prior	**0.340**	**0.323**

Table 4: Ablation on the learned latent skill.

Latent dimension. We further investigate the effect of the latent dimension by evaluating $z_{\mathrm{dim}} \in \{16, 32, 48\}$ and evaluate the success rate on Click Mouse, Pinch Tongs and Water Plant. As shown in Table 5, $z_{\mathrm{dim}} = 32$ provides consistently strong online adaptation across the evaluated tasks. In comparison, $z_{\mathrm{dim}} = 16$ fails on Pinch Tongs, suggesting that a lower-dimensional latent space may limit the representation of task-relevant skills required for downstream adaptation.

Task	$z_{\mathrm{dim}} = 16$	32	48
Click Mouse	0.86	**0.98**	0.76
Pinch Tongs	0.00	0.82	**0.88**
Water Plant	1.00	1.00	1.00

Table 5: Ablation on the latent dimension.

5 Related Work

Dexterous manipulation with demonstrations. Demonstration-based dexterous learning needs to address both difficult exploration and the embodiment gap between human and robot hands. DAPG

combines imitation-based initialization with demonstration-augmented policy gradients to learn dexterous skills (Rajeswaran et al., 2017). DexCap collects portable human hand motion data and learns point-cloud-conditioned diffusion policies after retargeting (Wang et al., 2024). DexUMI reduces kinematic and visual embodiment gaps through a wearable exoskeleton and robot-hand inpainting (Xu et al., 2025a). Dexplore instead uses motion-capture demonstrations as soft spatial guidance for reinforcement learning, then distills the resulting tracking policy into a skill-conditioned generative controller (Xu et al., 2025b). Together, these works address demonstration acquisition, embodiment transfer, and policy learning. DexPrior focuses on extracting a reusable skill prior from heterogeneous multi-task dexterous demonstrations to guide subsequent task-specific adaptation.

Diffusion policy fine-tuning. Fine-tuning a pretrained diffusion policy with online reinforcement learning is challenging because action generation involves iterative denoising. Recent methods address this through policy optimization, noise steering, or model-based learning. Recent methods adapt pretrained diffusion policies through policy optimization, noise steering, or model-based learning. DPPO applies policy gradients to an MDP over denoising steps (Ren et al., 2025), whereas DSRL optimizes the latent-noise distribution while keeping the base policy fixed (Wagenmaker et al., 2025). OTPR uses optimal transport with a Q-function-based cost to guide diffusion-policy fine-tuning (Sun et al., 2025), while DiWA performs offline adaptation through reinforcement learning in a learned world model (Chandra et al., 2025). More recently, OGPO combines off-policy critics with a modified PPO objective to fine-tune the full generative policy (Patil et al., 2026). REFINE-DP jointly adapts a diffusion motion planner and a humanoid tracking controller to reduce their distributional mismatch (Gu et al., 2026). DexPrior explores a complementary adaptation interface: a latent-skill space learned from multi-task demonstrations that conditions diffusion action generation and supports downstream reinforcement learning.

Latent skill learning. SPiRL jointly learns latent skills and a state-conditioned prior from offline experience, using the prior to guide downstream reinforcement learning (Pertsch et al., 2021). Residual Skill Policies add low-level residual control to adapt pretrained skills to unseen task variations (Rana et al., 2023). For humanoid control, PULSE distills a motion imitator into a variational latent representation and learns a proprioception-conditioned prior for skill reuse (Luo et al., 2024). QueST learns quantized temporal action abstractions with a causal structure for multi-task and few-shot imitation (Mete et al., 2024). EXTRACT uses vision-language models to discover discrete semantic skills with continuous arguments, then learns downstream skill-selection and argument policies guided by learned priors (Zhang et al., 2024). Building on these approaches, DexPrior couples a task-agnostic latent-skill prior with diffusion-based dexterous control, learning from heterogeneous demonstrations without explicit task labels and reusing the resulting representation for online adaptation.

6 Conclusion

We presented DexPrior, an offline-to-online framework that learns reusable latent skills from heterogeneous multi-task dexterous manipulation demonstrations and reuses them for task-specific reinforcement learning. DexPrior employs an Encoder–Prior–Decoder architecture to abstract temporally structured behaviors from privileged future motion and contact information, while learning a deployment-time conditional skill prior and a skill-conditioned diffusion decoder. During online adaptation, a latent residual policy explores the pretrained skill space and jointly adapts the diffusion decoder. By introducing latent-skill selection as an additional transition within the diffusion-policy MDP, DexPrior provides a structured interface for coordinating high-level skill exploration with low-level action generation.

Limitation. Currently, DexPrior is evaluated only in state-based settings, and its effectiveness in vision-based tasks remains to be investigated. Future work could also explore scaling up the multi-task demonstration dataset.

AI use statement

Generative AI tools and large language models were used solely for language editing, grammatical correction, and drafting figures during preparation. They were not used to formulate research questions, develop the methodology, conduct experiments, or interpret the results. The authors developed

and verified all algorithmic formulations and experimental analyses, reviewed the entire manuscript, and take full responsibility for its accuracy and scientific content.

ETHICS STATEMENT

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide the problem formulation and two-layer diffusion-policy optimization setup in Section 2. Section 3 specifies the unified representation of heterogeneous/multi-task demonstrations, the encoder–prior–decoder training objectives, and the mechanism for online adaptation; the two-stage procedure is summarized in Algorithm 1. Section 4 describes the DexJoCo tasks, demonstration collection, offline training hyperparameters, diffusion sampling settings, and online reinforcement learning budget. Together, these details specify how to reproduce the offline pretraining and task-specific adaptation experiments.

REFERENCES

Shan An, Ziyu Meng, Chao Tang, Yuning Zhou, Tengyu Liu, Fangqiang Ding, Shufang Zhang, Yao Mu, Ran Song, Wei Zhang, et al. Dexterous manipulation through imitation learning: A survey. *IEEE Transactions on Automation Science and Engineering*, 23:1760–1792, 2025.

Sridhar Pandian Arunachalam, Sneha Silwal, Ben Evans, and Lerrel Pinto. Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation. In *2023 ieee international conference on robotics and automation (icra)*, pp. 5954–5961. IEEE, 2023.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Akshay L Chandra, Iman Nematollahi, Chenguang Huang, Tim Welschehold, Wolfram Burgard, and Abhinav Valada. Diwa: Diffusion policy adaptation with world models. *arXiv preprint arXiv:2508.03645*, 2025.

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. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Shutong Ding, Ke Hu, Shan Zhong, Haoyang Luo, Weinan Zhang, Jingya Wang, Jun Wang, and Ye Shi. Genpo: Generative diffusion models meet on-policy reinforcement learning. *Advances in neural information processing systems*, 38:130443–130474, 2026.

Zhaoyuan Gu, Yipu Chen, Zimeng Chai, Alfred Cueva, Thong Nguyen, Yifan Wu, Huishu Xue, Minji Kim, Isaac Legene, Fukang Liu, et al. Refine-dp: Diffusion policy fine-tuning for humanoid loco-manipulation via reinforcement learning. *IEEE Robotics and Automation Letters*, 2026.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105):eads5033, 2025.

Zhengyi Luo, Jinkun Cao, Josh Merel, Alexander Winkler, Jing Huang, Kris Kitani, and Weipeng Xu. Universal humanoid motion representations for physics-based control. In *International Conference on Learning Representations*, volume 2024, pp. 56766–56782, 2024.

Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. Quest: Self-supervised skill abstractions for learning continuous control. *Advances in Neural Information Processing Systems*, 37:4062–4089, 2024.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PmLR, 2021.

Sarvesh Patil, Mitsuhiko Nakamoto, Manan Agarwal, Shashwat Saxena, Jesse Zhang, Giri Anantharaman, Cleah Winston, Chaoyi Pan, Douglas Chen, Nai-Chieh Huang, et al. Ogpo: Sample efficient full-finetuning of generative control policies. *arXiv preprint arXiv:2605.03065*, 2026.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, pp. 188–204. PMLR, 2021.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Krishan Rana, Ming Xu, Brendan Tidd, Michael Milford, and Niko Sünderhauf. Residual skill policies: Learning an adaptable skill-based action space for reinforcement learning for robotics. In *Conference on Robot Learning*, pp. 2095–2104. PMLR, 2023.

Allen Ren, Justin Lidard, Lars Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, volume 2025, pp. 77288–77329, 2025.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Mingyang Sun, Pengxiang Ding, Weinan Zhang, and Donglin Wang. Score-based diffusion policy compatible with reinforcement learning via optimal transport. *arXiv preprint arXiv:2502.12631*, 2025.

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

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.

Chen Wang, Haochen Shi, Weizhuo Wang, Ruohan Zhang, Li Fei-Fei, and C Karen Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. *arXiv preprint arXiv:2403.07788*, 2024.

Hanwen Wang, Weizhi Zhao, Xiangyu Wang, Siyuan Huang, He Lin, Boyuan Zheng, Rongtao Xu, Gang Wang, Yao Mu, He Wang, et al. Dexjoco: A benchmark and toolkit for task-oriented dexterous manipulation on mujoco. *arXiv preprint arXiv:2605.16257*, 2026.

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

Mengda Xu, Han Zhang, Yifan Hou, Zhenjia Xu, Linxi Fan, Manuela Veloso, and Shuran Song. Dexumi: Using human hand as the universal manipulation interface for dexterous manipulation. *arXiv preprint arXiv:2505.21864*, 2025a.

Sirui Xu, Yu-Wei Chao, Liuyu Bian, Arsalan Mousavian, Yu-Xiong Wang, Liangyan Gui, and Wei Yang. Dexplore: Scalable neural control for dexterous manipulation from reference scoped exploration. In *Conference on Robot Learning*, pp. 2184–2199. PMLR, 2025b.

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

A PSEUDOCODE

Algorithm 1 DexPrior

function TRAINOFFLINE($\mathcal{D}$)
  Input: multi-task demonstrations $\mathcal{D} = \{(\mathcal{S}_p^i, \mathcal{S}_o^i), \hat{\mathcal{A}}^i\}$
  $\widetilde{\mathcal{D}} = \{(\mathcal{S}_t, \mathcal{S}_t^{\text{priv}}, \mathcal{A}_t)\} \leftarrow \mathcal{D}$ // homogenization and chunking
  initialize encoder Enc, prior Prior, diffusion decoder Dec
  while not converged **do**
    sample a minibatch of windows from $\widetilde{\mathcal{D}}$
    $\mu_t^e, \sigma_t^e \leftarrow \text{Enc}(\mathcal{S}_t^{\text{priv}}, \mathcal{S}_t), \quad z_t \sim \mathcal{N}(\mu_t^e, \sigma_t^e)$ // encode skill posterior
    $\mu_t^p, \sigma_t^p \leftarrow \text{Prior}(\mathcal{S}_t)$ // prior from the shared state
    calculate the noising action chunk $\mathcal{A}_t^k$ using equation 9
    $\hat{\epsilon} \leftarrow \text{Dec}(\mathcal{A}_t^k \mid \mathcal{S}_t, \mathbf{z}_t, k)$ // skill-conditioned denoising
    $\mathcal{L}_{\text{action}} \leftarrow \|\hat{\epsilon} - \epsilon\|^2$
    $\mathcal{L}_{\text{KL}} \leftarrow \text{KL}(\mathcal{N}(\mu_t^e, \sigma_t^e) \,\|\, \mathcal{N}(\mu_t^p, \sigma_t^p))$
    $\mathcal{L}_{\text{regu}} \leftarrow \|\mu_t^e - \text{sg}(\mu_{t-1}^e)\|^2$ // latent smoothness
    update Enc, Dec, Prior on $\mathcal{L}_{\text{action}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}} + \lambda_{\text{regu}}\mathcal{L}_{\text{regu}}$
  end while
  return Enc, Dec, Prior
end function

function TRAINONLINE(Dec, Prior, $\mathcal{T}_H$)
  Input: pretrained Dec and Prior, H-step state transition $\mathcal{T}_H$.
  freeze Prior and initialize latent residual network π_{res}
  while not converged **do**
    $\mathcal{M} \leftarrow \emptyset$ // initialize rollout memory
    while $\mathcal{M}$ is not full **do**
      $s_0 \leftarrow$ reset the downstream task
      for $t \leftarrow 0, \ldots, T$ **do**
        $\mu_t^p, \sigma_t^p \leftarrow \text{Prior}(\mathcal{S}_t)$ // frozen prior
        $\delta\mathbf{z}_t \sim \pi_{\text{res}}(s_t)$ // latent-skill residual
        $\mathbf{z}_t \leftarrow \mu_t^p + \sigma_t^p \odot \delta\mathbf{z}_t$ // held fixed for all k
        $\mathcal{A}_t^K \sim \mathcal{N}(0, I)$
        for $k \leftarrow K, \ldots, 1$ **do**
          $\mathcal{A}_t^{k-1} \leftarrow \text{DDIM}(\text{Dec}, \mathcal{A}_t^k, \mathcal{S}_t, \mathbf{z}_t, k)$ // diffusion MDP
        end for
        $s_{t+1}, r_t \leftarrow \mathcal{T}_H(s_t, \mathcal{A}_t)$ // environment MDP
        store $(s_t, \delta\mathbf{z}_t, \{\mathcal{A}_t^k\}_{k=0}^K, r_t, s_{t+1})$ into $\mathcal{M}$
      end for
    end while
    PPO update of π_{res} and Dec on $\mathcal{M}$ // one env advantage over the $1{+}K$ inner steps
  end while
  return π_{res}, Dec
end function

B OBJECT-CENTRIC CONTEXT REPRESENTATION

Multi-task demonstrations involve a shared robot embodiment interacting with diverse objects to accomplish different task goals. While keeping the definition of the robot state $\mathcal{S}_p$ consistent across tasks, we construct a unified object-centric context representation to accommodate heterogeneous object states:

$$c = (o_1, \ldots, o_K), \qquad o_i = (p_i, m_i, r_i, v_i), \tag{15}$$

where K denotes the maximum number of objects involved in any task, $p_i \in \mathbb{R}^7$ denotes the object pose, consisting of its position and orientation quaternion, $m_i \in \{0, 1\}$ indicates whether the object is movable, r_i denotes its discrete role selected from a shared vocabulary, and $v_i \in 0, 1$ is a validity

indicator. A shared encoder maps the object-centric context c into the policy condition. For each valid object token o_i, an MLP encodes its pose and mobility indicator (p_i, m_i), which is combined with a learned embedding of its discrete role r_i. A single learned query then attends to the resulting object tokens to aggregate them into a fixed-dimensional context representation, while a key-padding mask excludes invalid tokens with $v_i = 0$:

$$\mathbf{c}_{\mathrm{obj}} = \mathrm{MHA}(\mathbf{q}_{\mathrm{learn}}, \mathbf{E}, \mathbf{E}; \mathrm{Mask}(\mathbf{v})), \qquad \mathbf{E}_i = \mathrm{MLP}(p_i, m_i) + E_{\mathrm{role}}(r_i). \tag{16}$$

When all tokens are invalid, the mask on the first slot is relaxed to ensure that the attention operation remains well-defined. Finally, the deployment-time state $\mathcal{S}$ is obtained by concatenating the object-context embedding with the robot state embedding.