DYNAMIC NEURAL KOOPMAN DISTILLATION FOR FAST ROBOT CONTROL USING DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion models excel at generating diverse, multimodal trajectories for robotic control, yet their iterative denoising process introduces latency that can constrain real-time closed-loop execution. To address this problem, we propose a Dynamic Neural Koopman (DNK) distillation framework, which distills multistep diffusion inference into a single forward pass conditioned on observations and sampled noise. Specifically, we introduce a Factorized Dynamic Koopman (FDK) layer that models the denoising process through a latent linear transition. The noisy trajectory input is lifted into a latent space, where the FDK layer parameterizes a factorized linear operator with condition-dependent eigenvalues that adapt the noisy-to-denoised transition to the current context. We evaluate DNK across robot-control benchmarks that span locomotion, state-based, and image-based manipulation, comparing against accelerated generative-policy baselines. The results demonstrate that DNK achieves competitive or improved closed-loop performance relative to accelerated generative-policy baselines, while substantially reducing inference latency relative to the strongest high-performing one-step baselines. On long-horizon manipulation tasks, DNK achieves competitive performance with substantially fewer inference parameters than the one-step policy baselines. Hardware experiments on a physical Kinova manipulator further demonstrate millisecond-scale inference and reduced task completion time. A project page is available at https://fdkoopman.github.io/.

1 INTRODUCTION

Diffusion models have emerged as a promising approach for robotic motion generation and control (Urain et al., 2025; Peng et al., 2025). By shifting policy learning from deterministic state-action mappings to stochastic trajectory generation, diffusion-based policies have shown strong performance across a broad range of continuous tasks, including locomotion (Pan et al., 2024), manipulation (Chi et al., 2025), motion planning (Yang et al., 2026), and exploration (Samavi et al., 2025; Cao et al., 2025). Unlike traditional unimodal policies, these models effectively generate diverse and context-appropriate plans, which makes them well-suited to robots operating in dynamic environments.

Despite this high expressivity, inference latency remains a major obstacle to deploying diffusion policies on nonlinear physical systems. During inference, a diffusion policy generates each trajectory through multiple reverse denoising steps, where a learned network sequentially transforms an initial noise sample into a task-conditioned trajectory (Song et al., 2021b; Ho et al., 2020). As a result, these sequential updates can lead to substantial per-decision latency. This delay is incompatible with the low-latency updates required for high-frequency closed-loop control (Hwangbo et al., 2019).

To alleviate this latency bottleneck, researchers have explored diffusion acceleration and distillation frameworks, such as Consistency Models (Song et al., 2023; Song and Dhariwal, 2024) and Progressive Distillation (Salimans and Ho, 2022). These methods substantially reduce the number of network evaluations required for diffusion sampling, thereby lowering inference cost. Recent studies have extended this paradigm to robot control (Prasad et al., 2024; Wang et al., 2025). However, existing one-step approaches typically parameterize the student policy as a direct conditional generator without an explicit context-dependent latent transition. Such context dependence is particularly relevant to robot control, where the noisy-to-denoised mapping can vary substantially across observations, contact conditions, and motion regimes (Huang et al., 2025b).

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.

Koopman operator theory provides a principled way to introduce dynamical structure into this setting by lifting nonlinear dynamics into a linear latent space (Bevanda et al., 2021; Shi et al., 2026). In robotics, Bi et al. (2025) combine deep Koopman models with diffusion-based planning to enable dynamics learning and control under limited action supervision. However, such architectures still rely on iterative upstream diffusion planners. To streamline the denoising process itself, recent efforts in image synthesis have proposed using a fixed global latent transition to compress diffusion inference into a single step (Berman et al., 2025). However, directly translating this static distillation framework to robot control is challenging since robotic trajectories can vary substantially across states, contact conditions, and motion regimes. A single global latent transition does not explicitly adapt its latent evolution to the current control context, which may limit its flexibility across heterogeneous trajectory regimes.

In this paper, we propose Dynamic Neural Koopman (DNK) distillation, a Koopman-inspired framework that compresses multistep diffusion inference into a one-step, context-conditioned latent transition for fast robot control. Using Koopman-inspired latent dynamics (Strässer et al., 2026; Mezić, 2005), we approximate the multistep noisy-to-denoised transformation of the teacher through a context-conditioned factorized transition in a learned latent space. This design enables millisecond-level inference latency while supporting diverse candidate generation. The main contributions of this paper are summarized as follows:

1. We present a one-step diffusion distillation framework for robot control by approximating the iterative reverse denoising process through a single linear transition in a learned latent space. This largely reduces inference cost relative to multistep diffusion sampling, enabling high-frequency closed-loop control for nonlinear robotic systems.
2. We introduce the Factorized Dynamic Koopman (FDK) layer, which parameterizes the latent transition using a shared modal basis and condition-dependent modal gains. Compared with a fixed global latent transition (Berman et al., 2025), the FDK layer provides a more flexible parameterization of denoising dynamics through context-dependent spectral adaptation, with a spectral regularizer that discourages large modal gains.
3. We evaluate the proposed method on D4RL locomotion and state-based and image-based manipulation benchmarks, and a physical Kinova Gen3 manipulator. Across these tasks, our approach maintains competitive teacher-level performance with substantially lower inference latency and a compact student architecture relative to high-capacity one-step baselines. Hardware experiments further demonstrate millisecond-scale inference, reduced task completion time, and obstacle-aware receding-horizon control.

2 Related Work

Diffusion Models for Robot Control: Recent work has increasingly applied diffusion models to robot control, where actions or trajectories are generated from a conditional distribution rather than as single deterministic outputs (Ho et al., 2020; Urain et al., 2025). This formulation is particularly useful when multiple feasible behaviors are compatible with the current context. Existing work has applied diffusion-based policies to manipulation (Chi et al., 2025), locomotion (Pan et al., 2024), motion planning (Zhong et al., 2026; Peng et al., 2025), and exploration (Samavi et al., 2025; Cao et al., 2025). Unified software frameworks, such as CleanDiffuser, have facilitated more accessible, modular, and reproducible development of diffusion-based methods in robotics (Dong et al., 2024). Most of these methods generate actions through iterative denoising conditioned on the current observation or task context. While this procedure enables expressive conditional policies, it also entails substantial inference latency because multiple denoising steps are required before an action is generated. Such latency becomes a major limitation in fast closed-loop robot control, where actions must be updated at high frequency under tight latency constraints (Urain et al., 2025). This motivates the development of faster diffusion-based policies that remain suitable for closed-loop control.

Acceleration and Distillation of Diffusion Models: Prior work on accelerating inference for diffusion models can be broadly grouped into two directions: fast samplers and distillation-based generators (Wang et al., 2025; Berman et al., 2025). Fast samplers reduce the number of denoising evaluations while retaining the original diffusion model. Representative examples include Denoising Diffusion Implicit Models (DDIM) (Song et al., 2021a), DPM-Solver (Lu et al., 2022), and EDM-style fast-sampling formulations (Karras et al., 2022). These methods improve efficiency, but they remain iterative and may be too slow for high-frequency closed-loop control.

Distillation-based methods instead train a student model to reproduce the teacher with fewer denoising steps, or in a single forward pass. Progressive Distillation (Salimans and Ho, 2022) reduces the number of sampling steps through teacher-student compression, while Consistency Models support direct or near-direct generation through Consistency Training (CT) and Consistency Distillation (CD) (Song et al., 2023; Song and Dhariwal, 2024). These approaches can provide larger reductions in inference cost than sampler acceleration, making them attractive for fast control. Several recent works have extended these distillation paradigms to robotics (Prasad et al., 2024; Wang et al., 2025). Although their training objectives exploit diffusion states or teacher-generated samples, the resulting students are typically parameterized as direct conditional generators without an explicit context-dependent latent transition. This distinction is relevant to robot control, where the noisy-to-denoised mapping can vary across observations, contact conditions, and motion regimes. Our work instead introduces context dependence directly into a structured latent transition of the one-step student.

Koopman-Based Models for Nonlinear Dynamics: Koopman operator theory provides a principled way to represent nonlinear dynamics through linear evolution of observables in a lifted space (Bevanda et al., 2021; Strässer et al., 2026). This perspective has motivated a broad literature on data-driven prediction and control, including neural variants that jointly learn lifted representations and approximate latent dynamics (Bi et al., 2025; Li et al., 2025). In robotics, Koopman-based models have been used for system identification, prediction, and feedback control, especially when nonlinear dynamics are difficult to model directly (Shi et al., 2026).

Recently, Koopman ideas have been introduced into generative modeling. The Koopman Distillation Model (KDM) shows that diffusion denoising can be compressed into a single latent linear evolution step for image generation (Berman et al., 2025). This result suggests that Koopman-based distillation is a promising approach to fast generative inference. However, existing Koopman diffusion formulations are developed primarily for image synthesis and rely on a static global operator. In robot control, the required behavior depends strongly on the current state and task context, and may involve changing contact modes. This motivates our FDK parameterization, which retains a shared modal basis while adapting the modal gains to the conditioning context.

3 Problem Statement and Preliminaries

We consider trajectory generation for high-frequency closed-loop robot control. Let $\mathbf{s}_t \in \mathcal{S} \subseteq \mathbb{R}^m$ and $\mathbf{a}_t \in \mathcal{A} \subseteq \mathbb{R}^n$ denote the robot state and action at control step t, respectively. Over a prediction horizon H, we use $\boldsymbol{\tau}$ to denote a predicted trajectory. Depending on the task and policy parameterization, $\boldsymbol{\tau}$ may represent a state–action trajectory $(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_H, \mathbf{a}_H)$ or an action trajectory $(\mathbf{a}_1, \ldots, \mathbf{a}_H)$. At each control step, the planner models a conditional trajectory distribution $\pi(\boldsymbol{\tau} \mid \mathbf{c})$, where $\mathbf{c}$ denotes the information available at the current step, such as the current observation, observation history, and task specification.

Let T_{inf} denote the inference time required to generate trajectory samples, and let T_{ctrl} denote the control period. For high-frequency closed-loop control, the generated action must be available within the control period, i.e., $T_{\text{inf}} \leq T_{\text{ctrl}}$. Our goal is to learn a one-step student generator that substantially reduces T_{inf} relative to the teacher diffusion policy while approximating the teacher-conditioned outputs and supporting stochastic candidate generation.

3.1 Denoising Diffusion Probabilistic Models

Denoising diffusion probabilistic models (DDPMs) learn a target distribution by reversing a Markovian noise-addition process (Ho et al., 2020; Song et al., 2021b). Let $\boldsymbol{\tau}_0 \sim q(\boldsymbol{\tau}_0 \mid \mathbf{c})$ denote a denoised trajectory target conditioned on $\mathbf{c}$. The forward diffusion process adds Gaussian noise over N steps according to a variance schedule $\{\beta_k\}_{k=1}^N$, with $\beta_k \in (0, 1)$:

$$q(\boldsymbol{\tau}_k \mid \boldsymbol{\tau}_{k-1}) = \mathcal{N}\big(\boldsymbol{\tau}_k; \sqrt{1-\beta_k}\,\boldsymbol{\tau}_{k-1}, \beta_k \mathbf{I}\big), \tag{1}$$

where $\mathbf{I}$ denotes the identity matrix. Equivalently, $\boldsymbol{\tau}_k = \sqrt{\bar{\alpha}_k}\,\boldsymbol{\tau}_0 + \sqrt{1-\bar{\alpha}_k}\,\boldsymbol{\epsilon}$, where $\bar{\alpha}_k = \prod_{i=1}^k (1-\beta_i)$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. For sufficiently large N, the terminal distribution approaches $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The reverse process sequentially denoises $\boldsymbol{\tau}_N$ to recover $\boldsymbol{\tau}_0$. The resulting conditional diffusion policy can be written as

$$\pi_\theta(\boldsymbol{\tau}_0 \mid \mathbf{c}) := \int \mathcal{N}(\boldsymbol{\tau}_N; \mathbf{0}, \mathbf{I}) \prod_{k=1}^{N} p_\theta(\boldsymbol{\tau}_{k-1} \mid \boldsymbol{\tau}_k, \mathbf{c})\, d\boldsymbol{\tau}_{1:N}. \tag{2}$$

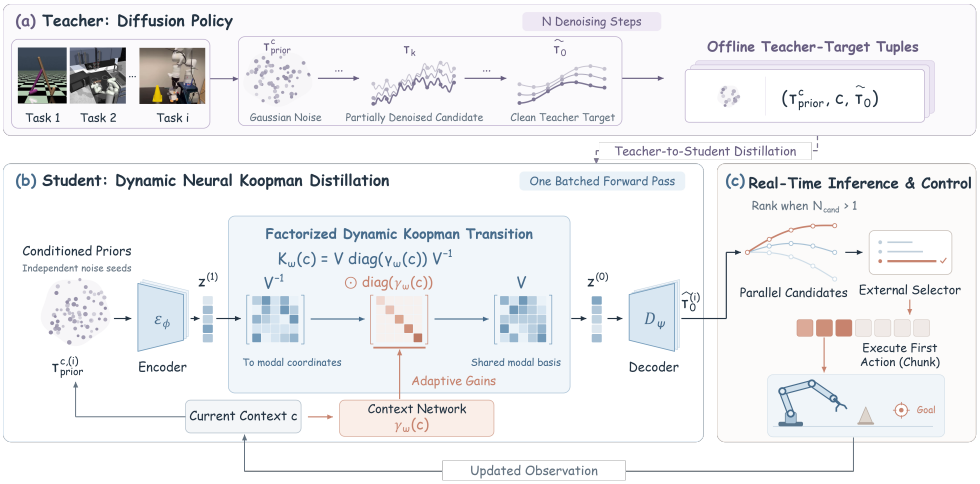


Figure 1: Overview of the proposed DNK distillation framework. A pretrained diffusion teacher generates offline teacher–target tuples $(\boldsymbol{\tau}^{\mathbf{c}}_{\text{prior}}, \mathbf{c}, \tilde{\boldsymbol{\tau}}_0)$ through multistep reverse diffusion. The lightweight student encodes the conditioned prior samples and performs a single FDK transition, using a shared modal basis and context-dependent latent spectrum. When multiple candidates are generated, the decoded sequences are ranked by an external selector and executed in closed loop with updated observations.

Here, $d\boldsymbol{\tau}_{1:N}$ denotes integration over the intermediate diffusion variables $d\boldsymbol{\tau}_1 \cdots d\boldsymbol{\tau}_N$. The term $p_\theta(\boldsymbol{\tau}_{k-1} \mid \boldsymbol{\tau}_k, \mathbf{c})$ denotes the learned reverse denoising transition, defined as

$$p_\theta(\boldsymbol{\tau}_{k-1} \mid \boldsymbol{\tau}_k, \mathbf{c}) = \mathcal{N}\big(\boldsymbol{\tau}_{k-1}; \boldsymbol{\mu}_\theta(\boldsymbol{\tau}_k, \mathbf{c}, k), \boldsymbol{\Sigma}_\theta(\boldsymbol{\tau}_k, \mathbf{c}, k)\big).$$

The denoising network $\boldsymbol{\epsilon}_\theta(\boldsymbol{\tau}_k, \mathbf{c}, k)$ is trained with the standard DDPM noise-prediction objective $\mathcal{L}_{\text{DDPM}} = \mathbb{E}_{\boldsymbol{\tau}_0, \boldsymbol{\epsilon}, k}\big[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{\tau}_k, \mathbf{c}, k)\|^2\big]$, where k is sampled uniformly from $\{1, \ldots, N\}$. This objective teaches the network to approximate the reverse denoising process through noise prediction.

The conditional diffusion policy in (2) reveals the main computational bottleneck: generating one trajectory sample requires N sequential network evaluations, which limits real-time deployment in high-frequency receding-horizon control.

3.2 Koopman Operator Theory and Neural Koopman Models

Koopman operator theory represents nonlinear dynamics through a linear evolution of observables in a lifted space (Strässer et al., 2026; Mezić, 2005). For a discrete-time nonlinear system $\mathbf{x}^+ = \Phi(\mathbf{x})$, where $\mathbf{x}^+$ denotes the next state and Φ is the nonlinear state-transition map, the Koopman operator acts on a scalar observable g as $(\mathcal{K}g)(\mathbf{x}) = g(\Phi(\mathbf{x}))$. Although Φ is generally nonlinear, $\mathcal{K}$ is linear on the space of observables. In practice, this infinite-dimensional operator is approximated in a finite-dimensional lifted space (Strässer et al., 2026).

Neural Koopman models parameterize the observable lifting and reconstruction maps with neural networks and learn a latent space in which the dynamics are approximately linear. In this setting,

$$\mathbf{z}_k = \mathcal{E}_\phi(\boldsymbol{\tau}_k), \qquad \hat{\boldsymbol{\tau}}_k = \mathcal{D}_\psi(\mathbf{z}_k), \tag{3}$$

where $\mathbf{z}_k \in \mathbb{R}^L$ denotes the lifted latent representation of the trajectory variable $\boldsymbol{\tau}_k$, with L being the latent dimension; $\mathcal{E}_\phi$ and $\mathcal{D}_\psi$ denote the encoder and decoder with trainable parameters ϕ and ψ, respectively. Since diffusion inference proceeds from $\boldsymbol{\tau}_N$ to $\boldsymbol{\tau}_0$, we write the latent transition in the reverse direction. The learned finite-dimensional latent evolution is then approximated by

$$\mathbf{z}_k = \mathbf{A}\mathbf{z}_{k+1}, \tag{4}$$

where $\mathbf{A}$ denotes a finite-dimensional linear operator in the lifted latent space. For a time-homogeneous latent system, multi-step evolution can be represented compactly through powers of the same operator, i.e., $z_0 = A^k z_k$. As outlined by Bevanda et al. (2023), under suitable assumptions, finite-dimensional lifted linear representations can approximate the original nonlinear map arbitrarily well. This motivates our use of Koopman-inspired latent modeling to approximate the multistep noisy-to-denoised transformation induced by diffusion inference.

4 DYNAMIC NEURAL KOOPMAN DISTILLATION FRAMEWORK

This section presents the proposed DNK Distillation framework, as illustrated in Fig. 1. The framework first constructs an offline teacher-target dataset by querying a pretrained DDPM teacher. It then trains a one-step student to map a task-conditioned Gaussian prior to a denoised target sequence through a context-conditioned lifted transition. The student is built on an FDK layer and trained using teacher-fidelity losses, control-oriented supervision, and structural regularization. At test time, the distilled student is deployed in a receding-horizon control loop.

4.1 OFFLINE TEACHER-TARGET PAIR GENERATION

Let π_θ denote the pretrained teacher diffusion policy. We construct an offline distillation dataset by sampling conditioning variables $\mathbf{c}$ from the dataset used to train the teacher. For each context, we sample a Gaussian prior $\boldsymbol{\tau}_N \sim \mathcal{N}(\mathbf{0}, \sigma_N^2 \mathbf{I})$ with the same horizon and per-step feature layout as the teacher output, where σ_N follows the sampling configuration of the teacher. For state–action trajectories, we obtain conditioned prior samples $\boldsymbol{\tau}^{\mathbf{c}}_{\text{prior}}$ by fixing the observed state entries according to the teacher's conditioning scheme. For action-only policies, $\boldsymbol{\tau}^{\mathbf{c}}_{\text{prior}} = \boldsymbol{\tau}_N$, while the observation history, or its encoded representation for high-dimensional observations, is provided separately as conditioning to the denoising network.

Starting from $\boldsymbol{\tau}^{\mathbf{c}}_{\text{prior}}$, the teacher generates the target through multistep reverse diffusion:

$$\tilde{\boldsymbol{\tau}}_0 \sim \pi_\theta\big(\cdot \mid \mathbf{c}; \boldsymbol{\tau}^{\mathbf{c}}_{\text{prior}}\big), \tag{5}$$

where the semicolon specifies the initial trajectory. Each distillation sample therefore takes the form $(\boldsymbol{\tau}^{\mathbf{c}}_{\text{prior}}, \mathbf{c}, \tilde{\boldsymbol{\tau}}_0)$, with $\mathbf{c}$ retained for conditioning the student.

4.2 CONTEXT-CONDITIONED LATENT TRANSITION VIA THE FDK LAYER

We propose an FDK layer to model one-step denoising as a context-dependent transition in a shared lifted latent space. The student first maps the conditioned prior samples to a latent state

$$\mathbf{z}^{(1)} = \mathcal{E}_\phi\big(\boldsymbol{\tau}^{\mathbf{c}}_{\text{prior}}\big) \in \mathbb{R}^L, \tag{6}$$

where L is the lifted latent dimension. The same encoder is applied to conditioned prior samples and denoised teacher targets, providing a shared latent representation for distillation.

A static Koopman distillation model (Berman et al., 2025) applies a single global linear operator to all latent samples. This assumption can be restrictive for robotic state or action trajectory generation because the mapping from a noisy prior to a denoised trajectory may vary across observations, motion regimes, and task specifications. To address this issue, we introduce an FDK layer that replaces the fixed global transition with a factorized, context-conditioned family of latent linear operators. Rather than predicting a dense $L \times L$ operator independently for each condition, FDK uses a shared learned modal basis together with condition-dependent latent eigenvalues:

$$\mathbf{z}^{(0)} = \mathbf{V}\left(\mathbf{V}^{-1}\mathbf{z}^{(1)} \odot \boldsymbol{\gamma}_\omega(\mathbf{c})\right), \tag{7}$$

where $\odot$ denotes elementwise multiplication. The matrix $\mathbf{V} \in \mathbb{R}^{L\times L}$ denotes a shared nonsingular modal basis, while $\mathbf{V}^{-1}$ maps the lifted representation $\mathbf{z}^{(1)}$ into the corresponding modal coordinates. The network $\boldsymbol{\gamma}_\omega : \mathbb{R}^{d_c} \to \mathbb{R}^L$ maps the conditioning representation $\mathbf{c}$ to a set of condition-dependent latent eigenvalues, where d_c denotes the dimension of the conditioning representation. These coefficients modulate the latent modes according to the information available at the current control step, such as the current observation and task specification. The transformed modal coordinates are then mapped back to the shared latent space through $\mathbf{V}$. Equivalently, the FDK layer defines

$$\mathbf{z}^{(0)} = \mathbf{K}_\omega(\mathbf{c})\mathbf{z}^{(1)}, \qquad \mathbf{K}_\omega(\mathbf{c}) = \mathbf{V}\,\mathrm{diag}\big(\boldsymbol{\gamma}_\omega(\mathbf{c})\big)\mathbf{V}^{-1}. \tag{8}$$

The context-dependent gains modulate the shared latent modes, providing a structured form of adaptation across conditioning contexts. For any fixed condition $\mathbf{c}$, $\mathbf{K}_\omega(\mathbf{c})$ defines a linear transition in the lifted state. Since $\mathbf{K}_\omega(\mathbf{c})$ is similar to $\mathrm{diag}(\boldsymbol{\gamma}_\omega(\mathbf{c}))$, its eigenvalues are exactly $\gamma_{\omega,\ell}(\mathbf{c})$, while the columns of $\mathbf{V}$ define the shared modal basis. Thus, FDK adapts the spectrum of the latent transition to the current context while retaining a shared modal basis. This structure introduces context dependence

without predicting a dense $L \times L$ operator for each context: only the L modal gains $\boldsymbol{\gamma}_\omega(\mathbf{c})$ vary with the conditioning context, whereas the modal basis $\mathbf{V}$ is shared. Consequently, the condition-dependent degrees of freedom scale as L, rather than L^2 for an unrestricted context-specific dense transition. Compared with the fixed global transition used in static Koopman distillation (Berman et al., 2025), FDK provides structured context adaptation while preserving a shared latent representation.

For a fixed latent representation and shared modal basis, the optimal context-dependent modal gains attain no larger population latent-prediction risk than the best constant gain vector, with strict improvement whenever the optimal gains vary with the conditioning context. Appendix A.1 formalizes this result. After the FDK transition, the decoder reconstructs the predicted denoised trajectory: $\hat{\boldsymbol{\tau}}_0 = \mathcal{D}_\psi\big(\mathbf{z}^{(0)}\big)$.

4.3 Training Objective

Given an offline teacher-target sample $(\boldsymbol{\tau}^{\mathbf{c},(i)}_{\text{prior}}, \tilde{\boldsymbol{\tau}}^{(i)}_0)$ from the dataset defined in Section 4.1, the student predicts a denoised trajectory $\hat{\boldsymbol{\tau}}^{(i)}_0$. We propose a training objective that aligns the student with the teacher while emphasizing control-relevant predictions and regularizing the latent transition.

Teacher-Fidelity Losses: To distill the teacher into a one-step student, we align the student with the teacher using denoised-trajectory reconstruction, latent-target alignment, and trajectory prediction:

$$\mathcal{L}_{\text{rec}} = \mathbb{E}\big[\|\mathcal{D}_\psi(\mathcal{E}_\phi(\tilde{\boldsymbol{\tau}}_0)) - \tilde{\boldsymbol{\tau}}_0\|_2^2\big]\,, \tag{9}$$

$$\mathcal{L}_{\text{lat}} = \mathbb{E}\left[\left\|\mathbf{z}^{(0)} - \text{sg}(\mathcal{E}_\phi(\tilde{\boldsymbol{\tau}}_0))\right\|_2^2\right], \tag{10}$$

$$\mathcal{L}_{\text{pred}} = \mathbb{E}\Big[\|\mathcal{D}_\psi(\boldsymbol{z}^{(0)}) - \tilde{\boldsymbol{\tau}}_0\|_2^2\Big]\,. \tag{11}$$

Here, $\mathcal{L}_{\text{rec}}$ regularizes the encoder–decoder pair so that denoised teacher trajectories are represented in a shared latent space; $\mathcal{L}_{\text{lat}}$ aligns the predicted denoised latent $\mathbf{z}^{(0)} = \mathbf{K}_\omega(\mathbf{c})\mathcal{E}_\phi\big(\boldsymbol{\tau}^{\mathbf{c}}_{\text{prior}}\big)$ with the encoded teacher target, providing direct supervision for the output of the context-conditioned FDK transition. The stop-gradient operator $\text{sg}(\cdot)$ detaches the encoded teacher target, providing a latent target for the FDK-evolved state $\mathbf{z}^{(0)}$ at each optimization step. Finally, $\mathcal{L}_{\text{pred}}$ directly supervises the decoded trajectory to match the teacher target.

Control-Oriented Supervision: Since only the first action or action chunk is executed in a receding-horizon manner, control performance depends strongly on near-term action prediction. We therefore augment the objective with a weighted action-sequence loss defined directly on the predicted and teacher-target trajectories. Let $\Pi_a(\boldsymbol{\tau}) = [\mathbf{a}_1^\top, \ldots, \mathbf{a}_H^\top]^\top \in \mathbb{R}^{Hn}$ denote the action sequence associated with $\boldsymbol{\tau}$. We define

$$\mathcal{L}_{\text{act}} = \mathbb{E}\Big[\|\Pi_a(\hat{\boldsymbol{\tau}}_0) - \Pi_a(\tilde{\boldsymbol{\tau}}_0)\|^2_{\mathbf{W}_a}\Big]\,, \tag{12}$$

where $\|\mathbf{x}\|^2_{\mathbf{W}_a} = \mathbf{x}^\top \mathbf{W}_a \mathbf{x}$, and $\mathbf{W}_a$ denotes a diagonal action-weighting matrix.

Structural Regularization: To regularize the factorized latent transition, we introduce a structural regularizer that aims at limiting excessive modal amplification:

$$\mathcal{L}_{\text{spec}} = \mathbb{E}\left[\frac{1}{L}\sum_{\ell=1}^{L} \max\big(0, |\gamma_{\omega,\ell}(\mathbf{c})| - 1\big)\right], \tag{13}$$

where $\gamma_{\omega,\ell}(\mathbf{c})$ denotes the ℓ-th coefficient produced by the FDK modulation network. This term penalizes modal eigenvalue magnitudes that exceed unity. The corresponding spectral properties and the role of the modal-basis conditioning are discussed in Appendix A.2.

Total Objective: Thus, the final training objective is

$$\mathcal{L} = \alpha_{\text{rec}}\mathcal{L}_{\text{rec}} + \alpha_{\text{lat}}\mathcal{L}_{\text{lat}} + \alpha_{\text{pred}}\mathcal{L}_{\text{pred}} + \alpha_{\text{spec}}\mathcal{L}_{\text{spec}} + \mathcal{L}_{\text{act}}, \tag{14}$$

where α_{rec}, α_{lat}, α_{pred}, and α_{spec} are the corresponding loss weights. The active loss terms and their benchmark-specific weights are summarized in Appendix B.3.

4.4 Inference: Batched One-Step Receding-Horizon Control

After training, the distilled student is deployed in a receding-horizon control manner, as illustrated in Fig. 1. At each control step, a conditioned Gaussian prior trajectory $\boldsymbol{\tau}_{\text{prior}}$ is constructed by sampling Gaussian noise and encoding current task-relevant context into the prior representation.

When multi-candidate generation is used, we draw N_{cand} independent noise seeds and evaluate the student on the resulting batch of conditioned prior samples. This yields a set of candidate trajectories in parallel. These candidates are then ranked by an external selector, such as an Implicit Q-Learning (IQL) critic with argmax selection (Kostrikov et al., 2022) or a classifier (Dong et al., 2024), which assigns each candidate a task-dependent control score. The controller executes the first action of the highest-scoring candidate and replans at the next control step using updated observations. This inference scheme separates stochastic candidate generation from control-oriented candidate selection: independent noise seeds provide multiple candidate trajectories, while the external selector favors candidates with higher task-dependent control scores (Huang et al., 2025a).

5 Experiments

In this section, we evaluate the proposed DNK distillation framework on D4RL MuJoCo locomotion, state-based and image-based Push-T, long-horizon Franka manipulation benchmarks, and a physical Kinova Gen3 control task.

5.1 Simulation Setup

Benchmarks and Teacher: We evaluate the proposed method on the following benchmarks. The detailed experimental setup is given in Appendix B.

1. **D4RL locomotion** (Fu et al., 2020): we consider halfcheetah and walker2d (17D observations, 6D actions) tasks. They are useful for testing the performance under strongly nonlinear dynamics and rapidly changing motion regimes, where the required behavior varies substantially across states.
2. **Push-T** (Florence et al., 2022; Chi et al., 2025): a contact-rich planar manipulation task in which a circular end-effector pushes a T-shaped block to a fixed target pose. We evaluate both 5D state and high-dimensional 96×96 RGB observations to examine contact-rich trajectory generation.
3. **Franka Kitchen** (Gupta et al., 2020): a long-horizon manipulation benchmark with seven kitchen subtasks, where performance is measured by completion of up to four subtasks using 60D state observations and 9D actions. It is useful for evaluating whether the distilled policy can maintain effective behavior over long horizons and across changing task stages.

Student Models and Baselines: We compare the proposed method against three groups of baselines. First, we include the diffusion teacher: DDPM (Ho et al., 2020) and EDM (Karras et al., 2022) as high-quality references using the official CleanDiffuser (Dong et al., 2024) implementation. Second, we compare with two static Koopman baselines, KDM and KDM-F (Berman et al., 2025). Third, we compare with one-step diffusion policies, OneDP-S (Wang et al., 2025), and consistency-based baselines, including CT (Song and Dhariwal, 2024), CD (Song et al., 2023), and Consistency Policy (CP) (Prasad et al., 2024).

Evaluation Protocol: Each method is evaluated over three evaluation groups of 50 rollouts, using identical environment initializations and evaluation settings across methods. Benchmark-specific details, such as rollout horizons and candidate budgets, are given in Appendix B. We report closed-loop task performance, inference latency, and model size. Locomotion performance is the normalized return and manipulation performance is the corresponding closed-loop metric (normalized task score or success rate), both summarized as mean $\pm$ standard deviation across seeds. Latency is the per-step policy decision time on an NVIDIA RTX 5090 GPU with CUDA synchronization, and model size is the number of parameters.

Table 1: Quantitative results on D4RL locomotion tasks. T_D and T_E denote task-specific DDPM and EDM teachers, respectively. NFE denotes the number of sampling network evaluations per generated action chunk.

		HalfCheetah		Walker2D	
Method	NFE	Score ↑	Latency (ms) ↓	Score ↑	Latency (ms) ↓
DDPM Teacher T_D (Dong et al., 2024)	20	0.867 ± 0.004	36.94 ± 0.63	1.109 ± 0.000	49.65 ± 0.12
EDM Teacher T_E (Dong et al., 2024)	18	0.842 ± 0.012	6.72 ± 0.02	1.108 ± 0.001	6.67 ± 0.01
CD (Song et al., 2023)	1	0.428 ± 0.013	1.08 ± 2.32	0.689 ± 0.010	1.09 ± 2.33
CT (Song and Dhariwal, 2024)	1	0.403 ± 0.025	1.08 ± 2.31	0.387 ± 0.017	1.09 ± 2.37
CP (Prasad et al., 2024) (T_E)	1	0.676 ± 0.012	0.981 ± 0.01	1.059 ± 0.013	0.99 ± 0.01
OneDP-S (Wang et al., 2025) (T_D)	1	0.748 ± 0.030	2.03 ± 0.11	1.105 ± 0.000	2.03 ± 0.11
KDM (Berman et al., 2025) (T_D)	1	0.553 ± 0.024	0.34 ± 0.03	0.620 ± 0.011	0.35 ± 0.04
KDM-F (Berman et al., 2025) (T_D)	1	0.510 ± 0.004	0.86 ± 0.03	0.396 ± 0.066	0.87 ± 0.03
DNK (Ours) (T_D)	1	$\mathbf{0.873 \pm 0.007}$	$\mathbf{0.68 \pm 0.39}$	$\mathbf{1.095 \pm 0.001}$	$\mathbf{0.60 \pm 0.42}$

5.2 Simulation Results

Locomotion. Table 1 summarizes the quantitative results in the agile locomotion benchmarks. DNK achieves competitive normalized returns on both tasks, reaching 0.873 ± 0.007 on HalfCheetah and 1.095 ± 0.001 on Walker2d. Compared with OneDP-S, DNK achieves a higher return on HalfCheetah while requiring substantially lower inference latency. On Walker2d, OneDP-S attains a

Table 2: Closed-loop performance comparison of different approaches on Push-T. NFE denotes the number of sampling network evaluations per generated action chunk. Params. counts deployed neural-network parameters in millions, including frozen visual encoders. Image DNK includes its 2.56M action network and 11.37M frozen visual encoder.

		Push-T State			Push-T Image + proprioception		
Method	NFE	Score ↑	Latency (ms) ↓	Params. (M) ↓	Score ↑	Latency (ms) ↓	Params. (M) ↓
DDPM Teacher T_D (Dong et al., 2024)	5	0.937±0.047	30.592±0.077	68.88	0.845±0.027	31.554±0.073	80.39
EDM Teacher T_E (Dong et al., 2024)	158	**0.945±0.021**	1802.309±5.565	248.01	**0.913±0.024**	1809.775±4.711	262.71
CD (Song et al., 2023)	1	0.859±0.007	3.162±0.013	68.88	0.720±0.013	4.768±0.037	80.39
CT (Song and Dhariwal, 2024)	1	0.887±0.069	3.192±0.041	68.88	0.820±0.052	4.680±0.055	80.39
CP (Prasad et al., 2024) (T_E)	1	0.927±0.017	11.797±0.061	251.68	0.815±0.026	12.922±0.027	266.38
OneDP-S (Wang et al., 2025) (T_D)	1	0.923±0.021	6.162±0.009	68.88	0.825±0.013	7.053±0.028	80.39
KDM (Berman et al., 2025) (T_D)	1	0.378±0.049	0.618±0.006	1.12	0.515±0.056	1.347±0.012	12.75
KDM-F (Berman et al., 2025) (T_D)	1	0.272±0.012	0.903±0.003	1.91	0.519±0.074	1.530±0.009	13.54
DNK (Ours) (T_D)	1	**0.931±0.037**	**1.283±0.007**	**2.37**	**0.853±0.041**	**1.780±0.005**	**13.93**

slightly higher mean return (1.105 vs. 1.095), whereas DNK reduces latency from 2.03 to 0.60 ms, corresponding to a 3.38× speedup. DNK also achieves higher returns and lower latency than CP on both tasks. Although the static Koopman baselines KDM and KDM-F maintain low inference latency, they achieve substantially lower returns than DNK on both environments. As illustrated in Fig. 2, the static KDM policy fails to maintain a stable forward gait in the representative Walker2d rollout. Relative to the 20-step DDPM teacher, DNK reduces latency by 54.3× on HalfCheetah and 82.8× on Walker2d while retaining almost identical performance.

Push-T. Table 2 shows that DNK achieves the highest mean score among students who take one step under both observation settings, with substantially lower inference latency than non-Koopman methods. Notably, DNK obtains this competitive performance with a substantially smaller deployed model: the state-based student contains only 2.37M parameters, compared with 68.88M for OneDP-S and 251.68M for CP, corresponding to approximately 29× and 106× reductions, respectively. For image-based control, DNK reduces the deployed parameter count by approximately 5.8× and 19.1× compared to OneDP-S and CP, respectively. Although the smaller KDM variants require fewer parameters and lower latency, their substantially lower task scores indicate a less favorable performance–latency–capacity trade-off. Overall, these results show that DNK compresses both the iterative sampling procedure and the deployed policy architecture without requiring the student to inherit the large teacher network.

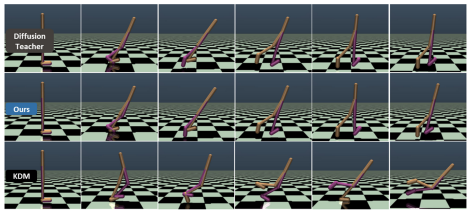


Figure 2: Representative Walker2d rollouts. The static KDM (Berman et al., 2025) fails to maintain a stable gait, whereas the diffusion teacher and DNK maintain coordinated forward motion.

Long-horizon Franka Kitchen. Table 3 summarizes the results on the Franka-kitchen benchmark. DNK achieves 0.973 ± 0.016, compared with 0.975 ± 0.013 for OneDP-S and 0.985 ± 0.000 for the DDPM teacher. Although OneDP-S attains a slightly higher mean score, DNK uses 2.46M parameters, approximately 28× fewer than OneDP-S. Moreover, DNK achieves a mean inference latency of 1.598 ms, corresponding to a 4.03× speedup over OneDP-S and a 7.53× speedup over CP. Additionally, DNK achieves higher task scores than CD, CT, CP, KDM, and KDM-F. These results demonstrate a favorable performance–latency–capacity trade-off in long-horizon manipulation.

Table 3: Quantitative comparison on Franka-Kitchen. NFE denotes the number of sampling-network evaluations per generated action chunk.

Task	Method	NFE	Score ↑	Latency (ms) ↓	Params. (M) ↓
Franka Kitchen	DDPM Teacher T_D (Dong et al., 2024)	50	0.985 ± 0.000	115.427 ± 0.809	68.92
	EDM Teacher T_E (Dong et al., 2024)	158	0.990 ± 0.010	681.062 ± 0.494	251.19
	CD (Song et al., 2023)	1	0.440 ± 0.040	3.690 ± 0.067	68.92
	CT (Song and Dhariwal, 2024)	1	0.803 ± 0.028	3.640 ± 0.010	68.92
	CP (Prasad et al., 2024) (T_E)	1	0.857 ± 0.035	12.026 ± 0.041	254.86
	OneDP-S (Wang et al., 2025) (T_D)	1	0.975 ± 0.013	6.434 ± 0.045	68.92
	KDM (Berman et al., 2025) (T_D)	1	0.462 ± 0.050	0.607 ± 0.008	1.78
	KDM-F (Berman et al., 2025) (T_D)	1	0.488 ± 0.031	0.818 ± 0.003	1.83
	DNK (Ours) (T_D)	**1**	**0.973±0.016**	**1.598±0.004**	**2.46**

5.3 Ablation Study

FDK Ablation: To assess the contribution of the proposed FDK transition, we compare DNK with a matched variant in which the learned latent transition is replaced by the identity map, $\mathbf{z}^+ = \mathbf{z}$. As shown in Fig. 3, removing FDK reduces the mean score from 0.973 to 0.738 on Franka Kitchen

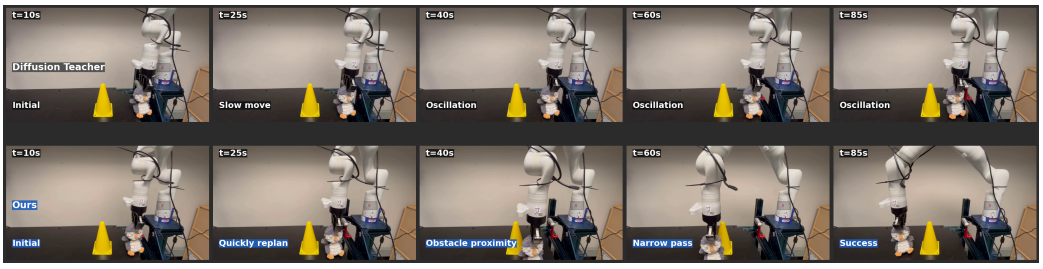


Figure 5: Consecutive frames from the real-world Kinova experiment. The diffusion teacher exhibits repeated corrective motion near the obstacle, consistent with the mismatch between its slower replanning rate and the faster low-level control loop. In contrast, DNK replans more frequently and traverses the narrow region with shorter goal-reaching time.

and from 0.931 to 0.563 on state-based Push-T. These results show that the learned FDK transition contributes substantially to closed-loop performance across both long-horizon and contact-rich manipulation tasks. Full experimental details and qualitative comparisons are provided in Appendix D.1.

Ablation on Internal Latent Refinement Depth: To investigate the effect of internal latent refinement, we recursively apply the FDK transition J times while keeping all other hyperparameters fixed on `halfcheetah-medium-expert-v2`. This ablation changes only the internal depth of the one-step student. Figure 4 illustrates the return and latency for $J \in \{1, 2, 3, 4\}$. Latency grows nearly linearly with J (from $0.680\,\text{ms}$ at $J = 1$ to $1.086\,\text{ms}$ at $J = 4$), while return stays within a band near 0.83–0.90. The highest mean return is at $J = 2$ (0.8966 ± 0.0083), but the gain over $J = 1$ (0.8731 ± 0.0070) is marginal relative to the added cost. This behavior is consistent with our Koopman-inspired evolution: repeated application of the same linear operator does not introduce new latent modes, but only rescales the modal gains. Hence, increasing J provides little benefit while adding computation. We therefore use $J = 1$ by default.

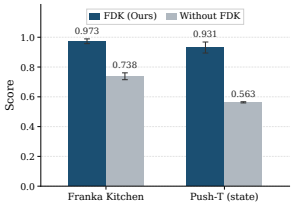


Figure 3: FDK component ablation on Franka Kitchen and state-based Push-T.

5.4 Real-World Experiment

To evaluate the proposed method in a real robotic setting, we consider an obstacle-aware reaching task on a Kinova Gen3 manipulator, as shown in Fig. 5. The task requires the robot to move its end-effector from an initial position to a goal position while avoiding static obstacles in the workspace.

Figure 5 shows a representative rollout in which the diffusion teacher progresses slowly and exhibits repeated corrective motion near the obstacle, whereas DNK replans faster and traverses the narrow passage to reach the goal. Across 50 hardware trials, DNK reduces mean inference latency from $151.00\,\text{ms}$ to $4.08\,\text{ms}$ and mean task completion time from $314.95\,\text{s}$ to $85.43\,\text{s}$ relative to the diffusion teacher. Detailed hardware settings and results are provided in Appendix C.

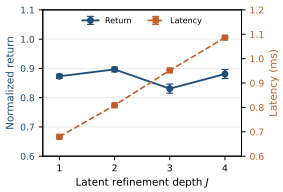


Figure 4: HalfCheetah ablation on latent refinement depth J: normalized return (left axis) and inference latency (right axis).

6 Conclusion and Future Work

This work presents a DNK distillation framework that distills a multistep diffusion policy into a one-step policy through an FDK layer with spectral regularization. The method achieves strong one-step performance across locomotion and manipulation benchmarks and on a real-world Kinova manipulator task, while substantially reducing inference latency and using a compact student architecture. Experiments on a physical Kinova Gen3 further demonstrate millisecond-scale inference and obstacle-aware reaching under receding-horizon control. The current framework does not explicitly enforce safety constraints or address robustness under broader distribution shifts. Future work will incorporate safety-critical constraints and investigate adaptation to more complex and unstructured robotic environments, such as human-robot interaction scenarios.

AI USE STATEMENT

Generative AI tools were used for translation, language editing, LaTeX formatting, and feedback on the technical presentation of the manuscript. The authors reviewed and verified all AI-assisted revisions, including mathematical statements and interpretations of experimental results, and take full responsibility for the final content.

ETHICS STATEMENT

This work involves simulation and controlled laboratory experiments on robotic systems and does not involve human participants or personal data. The proposed method does not provide formal safety guarantees; deployment in safety-critical settings would require application-specific validation and appropriate safety mechanisms.

REPRODUCIBILITY STATEMENT

The problem formulation and proposed DNK framework are described in Sections 3–4. Architectural details, training objectives, and implementation settings are provided in the corresponding method and experimental appendices. Benchmark configurations, hyperparameters, evaluation protocols, and hardware settings are documented in Appendix B and Appendix C. Proofs of the theoretical results are provided in Appendix A. We plan to release the source code, trained checkpoints, and evaluation scripts under an open-source license upon acceptance.

REFERENCES

Nimrod Berman, Ilan Naiman, Moshe Eliasof, Hedi Zisling, and Omri Azencot. One-step offline distillation of diffusion-based models via Koopman modeling. In *Advances in Neural Information Processing Systems*, 2025.

Petar Bevanda, Stefan Sosnowski, and Sandra Hirche. Koopman operator dynamical models: Learning, analysis and control. *Annual Reviews in Control*, 52:197–212, 2021.

Petar Bevanda, Max Beier, Armin Lederer, Stefan Sosnowski, Eyke Hüllermeier, and Sandra Hirche. Koopman kernel regression. *Advances in Neural Information Processing Systems*, 36:16207–16221, 2023.

Jianxin Bi, Kelvin Lim, Kaiqi Chen, Yifei Huang, and Harold Soh. Imitation learning with limited actions via diffusion planners and deep Koopman controllers. In *IEEE International Conference on Robotics and Automation*, pages 4861–4868, 2025.

Yuhong Cao, Jeric Lew, Jingsong Liang, Jin Cheng, and Guillaume Sartoretti. DARE: Diffusion policy for autonomous robot exploration. In *IEEE International Conference on Robotics and Automation*, pages 11987–11993, 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.

Zibin Dong, Yifu Yuan, Jianye Hao, Fei Ni, Yi Ma, Pengyi Li, and Yan Zheng. Cleandiffuser: An easy-to-use modularized library for diffusion models in decision making. *Advances in Neural Information Processing Systems*, 37:86899–86926, 2024.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A. Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Proceedings of the Conference on Robot Learning*, 2022.

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.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long horizon tasks via imitation and reinforcement learning. *Conference on Robot Learning*, 2020.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Tzu-Yuan Huang, Armin Lederer, Nicolas Hoischen, Jan Brudigam, Xuehua Xiao, Stefan Sosnowski, and Sandra Hirche. Toward near-globally optimal nonlinear model predictive control via diffusion models. In *Proceedings of the Annual Learning for Dynamics & Control Conference*, volume 283 of *Proceedings of Machine Learning Research*, pages 777–790, 2025a.

Xiaoyu Huang, Yufeng Chi, Ruofeng Wang, Zhongyu Li, Xue Bin Peng, Sophia Shao, Borivoje Nikolic, and Koushil Sreenath. DiffuseLoco: Real-time legged locomotion control with diffusion from offline datasets. In *In Proceedings of the Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 1567–1589. PMLR, 2025b.

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022.

Feihan Li, Abulikemu Abuduweili, Yifan Sun, Rui Chen, Weiye Zhao, and Changliu Liu. Continual learning and lifting of Koopman dynamics for linear control of legged robots. In *Proceedings of the Annual Learning for Dynamics & Control Conference*, page 136–148, 2025.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

Igor Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1):309–325, 2005.

Chaoyi Pan, Zeji Yi, Guanya Shi, and Guannan Qu. Model-based diffusion for trajectory optimization. *Advances in Neural Information Processing Systems*, 37:57914–57943, 2024.

Mingxing Peng, Kehua Chen, Xusen Guo, Qiming Zhang, Hui Zhong, Meixin Zhu, and Hai Yang. Diffusion models for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 26(12), 2025.

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.

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

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

Sepehr Samavi, Anthony Lem, Fumiaki Sato, Sirui Chen, Qiao Gu, Keijiro Yano, Angela P. Schoellig, and Florian Shkurti. SICNav-Diffusion: Safe and interactive crowd navigation with diffusion trajectory predictions. *IEEE Robotics and Automation Letters*, 10(9):8738–8745, 2025.

Lu Shi, Masih Haseli, Giorgos Mamakoukas, Daniel Bruder, Ian Abraham, Todd Murphey, Jorge Cortés, and Konstantinos Karydis. Koopman operators in robot learning. *IEEE Transactions on Robotics*, 42:1088–1107, 2026.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *International Conference on Learning Representations*, 2024.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pages 32211–32252, 2023.

Robin Strässer, Karl Worthmann, Igor Mezić, Julian Berberich, Manuel Schaller, and Frank Allgöwer. An overview of Koopman-based control: From error bounds to closed-loop guarantees. *Annual Reviews in Control*, 61:101035, 2026.

Julen Urain, Ajay Mandlekar, Yilun Du, Nur Muhammad, Danfei Xu, Katerina Fragkiadaki, Georgia Chalvatzaki, Jan Peters, et al. A survey on deep generative models for robot learning from multimodal demonstrations. *IEEE Transactions on Robotics*, 42:60–79, 2025.

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 *International Conference on Machine Learning*, 2025.

Rui Yang, Lei Zheng, Ruoyu Yao, and Jun Ma. DualShield: Safe model predictive diffusion via reachability analysis for interactive autonomous driving. *arXiv preprint arXiv:2601.15729*, 2026.

Ruiguo Zhong, Ruoyu Yao, Pei Liu, Xiaolong Chen, Rui Yang, and Jun Ma. CoPlanner: An interactive motion planner with contingency-aware diffusion for autonomous driving. In *IEEE International Conference on Robotics and Automation*, 2026.

A Theoretical Properties of the FDK Parameterization

A.1 Conditional versus Constant Modal Gains

To isolate the effect of the modal-gain parameterization, we analyze context-dependent modal gains with the encoder and shared modal basis $\mathbf{V}$ in Section 4.2 fixed. Let $\mathbf{C}$ denote the random conditioning variable, with realization $\mathbf{c}$, and let expectations be taken over the distillation data distribution

$$\mathbf{B}(\mathbf{z}^{(1)}) = \mathbf{V}\,\mathrm{diag}\Big(\mathbf{V}^{-1}\mathbf{z}^{(1)}\Big)\,, \tag{15}$$

so that, for a measurable gain function $g : \mathcal{C} \to \mathbb{R}^L$,

$$\mathbf{B}(\mathbf{z}^{(1)})g(\mathbf{c}) = \mathbf{V}\,\mathrm{diag}\big(g(\mathbf{c})\big)\mathbf{V}^{-1}\mathbf{z}^{(1)}.$$

The FDK transition corresponds to $g(\mathbf{c}) = \boldsymbol{\gamma}_\omega(\mathbf{c})$. We consider the population latent-prediction risk

$$R(g) = \mathbb{E}\left[\left\|\mathcal{E}_\phi(\tilde{\boldsymbol{\tau}}_0) - \mathbf{B}(\mathbf{z}^{(1)})g(\mathbf{C})\right\|_2^2\right]. \tag{16}$$

Let

$$\begin{aligned}\mathbf{M}(\mathbf{c}) &= \mathbb{E}\big[\mathbf{B}^\top\mathbf{B} \mid \mathbf{C} = \mathbf{c}\big]\,,\\ \mathbf{b}(\mathbf{c}) &= \mathbb{E}\big[\mathbf{B}^\top\mathcal{E}_\phi(\tilde{\boldsymbol{\tau}}_0) \mid \mathbf{C} = \mathbf{c}\big]\,,\end{aligned} \tag{17}$$

where $\mathbf{B} = \mathbf{B}(\mathbf{z}^{(1)})$.

Proposition 1 (Conditional versus constant modal gains)**.** *Assume the relevant second moments are finite and* $\mathbf{M}(\mathbf{C}) \succeq \mu\mathbf{I}$ *almost surely for some* $\mu > 0$*. Then*

$$g^\star(\mathbf{c}) = \mathbf{M}(\mathbf{c})^{-1}\mathbf{b}(\mathbf{c}) \tag{18}$$

minimizes $R(g)$ *over admissible conditional gains, and*

$$R(g) - R(g^\star) = \mathbb{E}\Big[\|g(\mathbf{C}) - g^\star(\mathbf{C})\|^2_{\mathbf{M}(\mathbf{C})}\Big]\,, \tag{19}$$

where $\|\mathbf{u}\|_{\mathbf{M}}^2 = \mathbf{u}^\top \mathbf{M}\mathbf{u}$.

If g is restricted to be constant, its optimum is

$$g_0 = \big(\mathbb{E}[\mathbf{M}(\mathbf{C})]\big)^{-1}\mathbb{E}[\mathbf{b}(\mathbf{C})], \tag{20}$$

and

$$R(g_0) - R(g^\star) \geq \mu\, \mathbb{E}\,\|g^\star(\mathbf{C}) - \mathbb{E}[g^\star(\mathbf{C})]\|_2^2\,. \tag{21}$$

Thus, the unrestricted conditional-gain class has strictly lower optimal population risk than the constant-gain class whenever $g^\star(\mathbf{C})$ is not almost surely constant.

Proof. Conditioned on $\mathbf{C} = \mathbf{c}$, the risk for $\mathbf{a} \in \mathbb{R}^L$ is

$$q_{\mathbf{c}}(\mathbf{a}) = \mathbb{E}\Big[\|\mathcal{E}_\phi(\tilde{\boldsymbol{\tau}}_0)\|_2^2 \mid \mathbf{C} = \mathbf{c}\Big] - 2\mathbf{a}^\top \mathbf{b}(\mathbf{c}) + \mathbf{a}^\top \mathbf{M}(\mathbf{c})\mathbf{a}.$$

Since $\mathbf{M}(\mathbf{c}) \succeq \mu\mathbf{I}$, this quadratic is strictly convex, with unique minimizer $g^\star(\mathbf{c}) = \mathbf{M}(\mathbf{c})^{-1}\mathbf{b}(\mathbf{c})$. Completing the square gives

$$q_{\mathbf{c}}(\mathbf{a}) - q_{\mathbf{c}}(g^\star(\mathbf{c})) = \|\mathbf{a} - g^\star(\mathbf{c})\|_{\mathbf{M}(\mathbf{c})}^2,$$

and taking expectations yields (19).

For constant $\mathbf{a}$,

$$R(\mathbf{a}) = \mathbb{E}\,\|\mathcal{E}_\phi(\tilde{\boldsymbol{\tau}}_0)\|_2^2 - 2\mathbf{a}^\top \mathbb{E}[\mathbf{b}(\mathbf{C})] + \mathbf{a}^\top \mathbb{E}[\mathbf{M}(\mathbf{C})]\mathbf{a},$$

whose unique minimizer is (20). Applying (19) to $g(\mathbf{C}) \equiv g_0$ gives

$$R(g_0) - R(g^\star) \geq \mu\, \mathbb{E}\|g_0 - g^\star(\mathbf{C})\|_2^2.$$

Using

$$\mathbb{E}\|g_0 - g^\star(\mathbf{C})\|_2^2 = \mathbb{E}\,\|g^\star(\mathbf{C}) - \mathbb{E}[g^\star(\mathbf{C})]\|_2^2 + \|g_0 - \mathbb{E}[g^\star(\mathbf{C})]\|_2^2$$

proves (21). □

The condition $\mathbf{M}(\mathbf{C}) \succeq \mu\mathbf{I}$ requires sufficient excitation of the modal coordinates. The result applies to a fixed latent representation and shared modal basis.

A.2 Spectral Properties of the FDK Operator

Proposition 2 (Spectral structure of the FDK operator)**.** *For any fixed context* $\mathbf{c}$*, the operator* $\mathbf{K}_\omega(\mathbf{c})$ *in* (8) *is diagonalizable, with eigenvalues*

$$\lambda_\ell(\mathbf{K}_\omega(\mathbf{c})) = \gamma_{\omega,\ell}(\mathbf{c}), \qquad \ell = 1, \ldots, L. \tag{22}$$

Consequently,

$$\rho(\mathbf{K}_\omega(\mathbf{c})) = \max_\ell |\gamma_{\omega,\ell}(\mathbf{c})|\,. \tag{23}$$

Moreover,

$$\|\mathbf{K}_\omega(\mathbf{c})\|_2 \leq \kappa_2(\mathbf{V}) \max_\ell |\gamma_{\omega,\ell}(\mathbf{c})|\,, \tag{24}$$

where

$$\kappa_2(\mathbf{V}) = \|\mathbf{V}\|_2 \|\mathbf{V}^{-1}\|_2$$

is the spectral condition number of the shared modal basis.

Proof. Let

$$\mathbf{D}_\omega(\mathbf{c}) = \mathrm{diag}\big(\boldsymbol{\gamma}_\omega(\mathbf{c})\big).$$

Since

$$\mathbf{K}_\omega(\mathbf{c}) = \mathbf{V}\mathbf{D}_\omega(\mathbf{c})\mathbf{V}^{-1},$$

$\mathbf{K}_\omega(\mathbf{c})$ is similar to $\mathbf{D}_\omega(\mathbf{c})$ and therefore has the same eigenvalues, which gives (23). By submultiplicativity of the induced 2-norm,

$$\|\mathbf{K}_\omega(\mathbf{c})\|_2 \leq \|\mathbf{V}\|_2\, \|\mathbf{D}_\omega(\mathbf{c})\|_2\, \|\mathbf{V}^{-1}\|_2.$$

Because

$$\|\mathbf{D}_\omega(\mathbf{c})\|_2 = \max_\ell |\gamma_{\omega,\ell}(\mathbf{c})|\,,$$

(24) follows. □

Table 4: Task-specific OneDP-S settings. Indices are zero-based. Each alternating iteration comprises one auxiliary and one generator optimizer step.

Task	Teacher levels	t_{init}	Score indices	Iterations
HalfCheetah / Walker2d	20	12	1–18	156,260
Push-T	5	2	0–3	6,840
Franka-Kitchen	50	32	1–47	37,580

Corollary 1 (Fixed-context output sensitivity)**.** *Fix a context* $\mathbf{c}$ *and a latent domain* $\mathcal{Z}$*. Suppose that the decoder* $\mathcal{D}_\psi$ *is* $L_{\mathcal{D}}$*-Lipschitz on* $\mathbf{K}_\omega(\mathbf{c})\mathcal{Z}$*. Then, for any* $\mathbf{z}, \mathbf{z}' \in \mathcal{Z}$*,*

$$\begin{aligned}&\|\mathcal{D}_\psi(\mathbf{K}_\omega(\mathbf{c})\mathbf{z}) - \mathcal{D}_\psi(\mathbf{K}_\omega(\mathbf{c})\mathbf{z}')\|_2 \\ &\quad \le L_{\mathcal{D}}\, \kappa_2(\mathbf{V})\, \|\boldsymbol{\gamma}_\omega(\mathbf{c})\|_\infty\, \|\mathbf{z} - \mathbf{z}'\|_2.\end{aligned} \tag{25}$$

Proof. The decoder Lipschitz condition gives an upper bound of $L_{\mathcal{D}}\|\mathbf{K}_\omega(\mathbf{c})(\mathbf{z} - \mathbf{z}')\|_2$. Applying the induced-norm inequality and (24) yields the result. □

Under the exact similarity parameterization, $\mathcal{L}_{\text{spec}}$ directly penalizes modal eigenvalue magnitudes exceeding unity. Moreover, for an $L_{\mathcal{D}}$-Lipschitz decoder, the fixed-context map $\mathbf{z} \mapsto \mathcal{D}_\psi(\mathbf{K}_\omega(\mathbf{c})\mathbf{z})$ has Lipschitz constant on $\mathcal{Z}$ bounded by

$$L_{\mathcal{D}}\, \kappa_2(\mathbf{V})\, \|\boldsymbol{\gamma}_\omega(\mathbf{c})\|_\infty.$$

Hence, the spectral penalty discourages large modal amplification, while the conditioning of $\mathbf{V}$ and the decoder sensitivity also determine perturbation amplification at the output.

B Experimental Setup

B.1 OneDP-S Implementation

Formulation. We adapt the stochastic OneDP-S objective of Wang et al. (2025) to our pretrained CleanDiffuser DDPM teachers. The generator and auxiliary score network adopt the same architecture as the teacher and are initialized as separate copies of the EMA weights from the teacher, while the teacher remains frozen. Each iteration first updates the auxiliary network with an unweighted noise-prediction MSE on detached generator samples, then updates the generator with a detached score-difference surrogate using $w(t) = \sigma_t^2$. Gradients through the forward-noised generator output are retained. Distillation uses observation contexts only and does not regress to demonstration actions or cached teacher endpoints.

Teacher adaptation. We retain each teacher's native diffusion schedule and normalization statistics. For Push-T and Franka Kitchen, the generator receives standard Gaussian noise and converts its noise prediction at a fixed initialization index t_{init} into a clean action sequence using the corresponding diffusion coefficients.

For locomotion, the inherited clean-sample predictor directly generates a state–action trajectory. The current normalized state is inpainted at the first trajectory step, and these fixed entries are excluded from both training losses. Teacher and auxiliary clean predictions are converted to noise predictions for loss computation. Table 4 summarizes the native diffusion grids; score-training indices are sampled uniformly.

Optimization. All students are trained for 20 epochs with batch size 64 and Adam with $(\beta_1, \beta_2) = (0, 0.999)$. The learning rates for the generator and auxiliary network are 10^{-6} and 2×10^{-5}, respectively, with gradient-norm clipped at 10. Locomotion training uses 500,000 observation contexts per task and retains frozen reward-classifier guidance with weights 10^{-4} (HalfCheetah) and 10^{-3} (Walker2d). In the locomotion implementation, latent noise, diffusion indices, and forward noise are resampled independently for the auxiliary and generator updates while sharing the same observation minibatch. In the manipulation implementation, these random draws are reused across the two updates.

Table 5: Numerical diagnostics of the FDK inverse implementation.

Checkpoint	r_{inv}	δ_{sd}
Push-T (state)	2.00×10^{-6}	2.37×10^{-6}
Push-T (image)	2.23×10^{-6}	2.58×10^{-6}
Franka Kitchen	1.71×10^{-6}	2.02×10^{-6}
Walker2d	7.31×10^{-6}	8.84×10^{-6}
HalfCheetah	1.14×10^{-5}	9.05×10^{-6}

Evaluation. For each task, we evaluate using 3 seeds, each one consisting of 50 episodes. Locomotion uses trajectory horizons of 4 (HalfCheetah) and 32 (Walker2d). A single batched generator evaluation produces 64 candidates, whose first actions are clipped to $[-1, 1]$ and ranked by the shared frozen IQL scorer. The selected action is then executed before replanning. Manipulation policies condition on two observation frames, predict 16 actions, and execute eight, using a single candidate without reranking. Image-based Push-T uses RGB observations and pusher position without privileged object state; its visual encoders remain trainable during distillation, with deterministic preprocessing. At inference time we deploy the generator and its conditioning components, together with the IQL scorer for locomotion.

Inverse implementation. The shared basis $\mathbf{V}$ is a trainable dense matrix initialized orthogonally. During training, we compute $\mathbf{V}^{-1}\mathbf{z}$ by solving $\mathbf{V}\mathbf{u} = \mathbf{z}$ with `torch.linalg.solve`, rather than learning an independent inverse. For frozen inference, $\mathbf{V}^{-1}$ is computed once using `torch.linalg.inv` and cached. The basis is otherwise unconstrained, and the spectral results assume nonsingularity.

Numerical checks of selected checkpoints using the exact similarity parameterization (CPU FP32) are reported in Table 5, where the normalized two-sided inverse residual is defined as

$$r_{\text{inv}} = \frac{\max\left\{\|\mathbf{V}\widehat{\mathbf{V}^{-1}} - \mathbf{I}\|_F,\ \|\widehat{\mathbf{V}^{-1}}\mathbf{V} - \mathbf{I}\|_F\right\}}{\sqrt{L}}, \tag{26}$$

where $\widehat{\mathbf{V}^{-1}}$ denotes the numerically computed cached inverse. The quantity δ_{sd} denotes the relative Frobenius discrepancy between solve-based and cached-inverse latent outputs over 256 synthetic test inputs. These diagnostics confirm numerically accurate inverse application and consistent FDK operator computation between training and deployment.

B.2 Deployed Parameter Counts Across Baselines

Code provenance. We use the public CleanDiffuser implementation for the DDPM teachers, CT, and CD,[1] and the official Consistency Policy codebase for CP and its EDM teacher.[2] Because OneDP-S has no public official release that we could run unchanged, we reimplement the published stochastic OneDP-S objective (Wang et al., 2025) on our frozen DDPM teachers, as detailed in Appendix B.1. KDM and KDM-F follow the open-source repository of Berman et al. (2025).[3]

Counting protocol. We count each unique parameter required for inference once, including frozen visual encoders, but excluding training-only teachers, auxiliary score networks, optimizer states, and additional weight copies used only during training. Unless otherwise stated, CD, CT, and OneDP-S retain the generator backbone of the corresponding local DDPM teacher T_D and therefore share its deployed parameter count. In particular, OneDP-S initializes the generator from the EMA weights of the teacher and subsequently optimizes it through distillation. This procedure reduces the number of sampling evaluations while keeping the backbone size unchanged. In contrast, CP uses an EDM-family architecture adapted to each task, and its parameter count does not necessarily match that of the corresponding teacher T_E.

[1] https://github.com/CleanDiffuserTeam/CleanDiffuser
[2] https://github.com/Aaditya-Prasad/Consistency-Policy
[3] https://github.com/azencot-group/KDM

Reported sizes. On low-dimensional Push-T and Franka-Kitchen, DDPM-backbone models contain approximately 69M parameters, whereas the EDM teacher and CP contain approximately 248–255M parameters. On Push-T, switching from state to image observations increases the total deployed count by approximately 11.50M for DDPM/OneDP-S models and 14.70M for EDM/CP models. All image-based counts include the visual conditioning modules.

B.3 Benchmarks and Setup

We evaluate DNK on D4RL MuJoCo locomotion and two manipulation benchmarks, Push-T and Franka Kitchen. Across benchmarks, DNK uses the general training objective in (14), with task-specific coefficients, training budgets, and teacher sampling settings summarized in Table 6.

MuJoCo locomotion. We use `halfcheetah-medium-expert-v2` and `walker2d-medium-expert-v2` from D4RL (Fu et al., 2020), both with 17D observations and 6D actions. Following CleanDiffuser (Dong et al., 2024), the planning horizon is 4 for HalfCheetah and 32 for Walker2d. The DDPM teacher uses 20 reverse steps at inference, whereas the EDM teacher uses 18 Euler sampling steps. Our method, KDM, and KDM-F are trained on the same teacher-generated distillation set under a matched configuration with latent size L=512, batch size 64, learning rate 3×10^{-4}, and 200 epochs. For each task, we generate 500,000 teacher–target pairs $(\boldsymbol{\tau}_{\text{prior}}, \tilde{\boldsymbol{\tau}}_0)$. Evaluation uses 64 candidates, IQL scoring, and argmax selection, with 3 seeds, 50 rollouts per seed, and a maximum episode length of 1000 steps. The encoder comprises two 1D convolutions (128/256 channels, kernel size 5, ReLU), adaptive average pooling, and a linear map to $z \in \mathbb{R}^L$. The decoder is a two-layer MLP with 1024 hidden units that reconstructs a trajectory.

Push-T. Push-T (Florence et al., 2022) requires a circular pusher to align a T-shaped block with a fixed target pose. We use the public Diffusion Policy demonstration archive (Chi et al., 2025) under the CleanDiffuser setup (Dong et al., 2024), which contains 206 episodes and 25,650 frames. We consider both state and image observations, using either a 5D state or a 96×96 RGB image together with the pusher position. Actions are 2D pusher targets. The episode score is $\max_t \text{clip}(c_{i,t}/0.95, 0, 1)$, where $c_{i,t}$ is target-area coverage. Episodes last at most 300 steps. The DDPM teacher uses 5 reverse steps and the `ChiUNet1d` backbone with FiLM observation conditioning (Perez et al., 2018).

State-based Push-T. We distill about 1.03×10^5 pairs for up to 1.5×10^5 updates (batch size 128). The student encodes a length-H trajectory that concatenates the current state with a noisy action sequence, applies a state-conditioned latent transition, and decodes the result to the same layout. Two-frame history is injected with gated FiLM, $z \leftarrow z + \alpha(z \odot s + b)$, where α=$\sigma(\alpha_0)$ starts near zero so training begins close to an identity map. Large FiLM scales and shifts are additionally penalized with weight 0.001.

Image-based Push-T. The image-based student receives two frames of 96×96 RGB observations together with the corresponding pusher positions. We freeze the visual encoder of the teacher, construct a 512D condition from the features of the two frames, and distill 1.0×10^5 pairs over 1.0×10^5 updates with a batch size of 256. A projection maps the condition to 128D. The temporal encoder processes the noisy action chunk together with this projection, while the latent transition uses the full 512D visual condition. The temporal decoder then predicts the action chunk. No FiLM branch is used. The visual history is incorporated only through the condition and the latent transition.

Franka Kitchen. Franka Kitchen (Gupta et al., 2020) is a long-horizon Franka manipulation benchmark. We use `kitchen-all-v0`, which contains seven subtasks that may be completed in any order. From the released multitask demonstrations, our preprocessing produces 603 episodes. Observations are 60D and actions are 9D absolute joint targets. Policies condition on two frames, predict 16 actions, execute eight, and then replan, with a maximum episode length of 280 steps. The DDPM teacher is a CleanDiffuser `ChiUNet1d`, trained for 10^6 updates and evaluated with EMA weights using 50 reverse steps. We build 100,000 training and 8,192 held-out pairs from the frozen teacher, with an episode-level train/validation split. Each pair contains a two-frame history, Gaussian action noise, and a teacher-generated 16×9 action sequence; zero-filled future-state slots are not used as targets.

Table 6: Student training settings by benchmark. For locomotion, the action weighting is the diagonal $\mathbf{W}_a$, with w_1=1.5 and $w_{k>1}$=0.6. For manipulation, action-loss entries denote $(\lambda_{\text{chunk}}, \lambda_{\text{full}}, \lambda_{\text{first}})$, corresponding to the executed action window, the full predicted action sequence, and additional emphasis on the first executed action, respectively.

	HalfCheetah / Walker2d	Push-T (state)	Push-T (image)	Franka Kitchen
Latent size L	512	128	128	128
Batch size	64	128	256	128
Learning rate	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}
EMA rate	0.9999	0.9999	0.9999	0.9999
Train pairs	5×10^5	$\approx 1.03 \times 10^5$	1.0×10^5	1.0×10^5
α_{rec}, α_{pred}	0.8, 1.0	0, 0	0, 0	0, 0
α_{lat}, α_{spec}	0.2, 0.01	0.15, 0.01	0.2, 0.01	0.15, 0.01
Action weighting	w_1=1.5, $w_{k>1}$=0.6	(1.0, 0.25, 0.5)	(1.0, 0.25, 0)	(1.0, 0.25, 0)

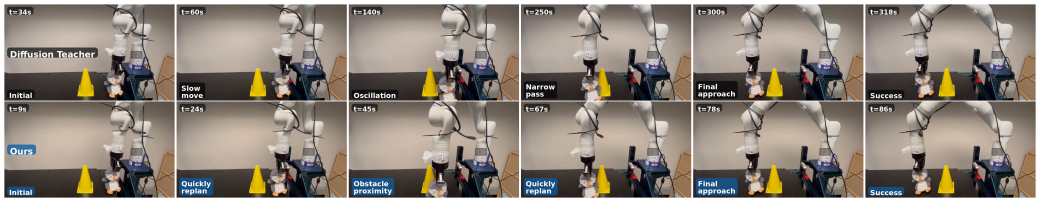


Figure 6: Consecutive snapshots of the Kinova manipulator in the real-world setup. The diffusion teacher exhibits slower and repeated corrective motion near the obstacles, while the proposed student replans much faster, enabling more continuous progress toward the goal.

The student is a temporal convolutional encoder–decoder with channel widths 128, 192, and 256 and L=128 (2.46M parameters vs. 68.92M for the teacher), with observation-gated FiLM and observation-dependent modal gains. We train with Adam for 150,000 updates (batch size 128, learning rate 3×10^{-4}, EMA 0.9999); closed-loop validation selects the EMA checkpoint at 80,000 updates. Following (Chi et al., 2025; Dong et al., 2024), let p_k denote the fraction of rollouts completing at least k distinct subtasks. The primary score is $S = \frac{1}{4}\sum_{k=1}^{4} p_k$, equivalently the mean of $\min(n, 4)/4$, where n is the number of distinct completed subtasks. We distinguish this normalized completion score from p_4, which measures the fraction of rollouts completing at least four subtasks.

Unless otherwise noted, Push-T and Franka Kitchen are evaluated on three evaluation groups of 50 episodes each. Notably, for these action-centric manipulation tasks, prediction supervision is retained through $\mathcal{L}_{\text{act}}$ on the decoded action sequence; Packed trajectories contain zero-filled future-state slots that are not valid teacher targets, and closed-loop performance depends on executable actions rather than full-state reconstruction. Hence, we set α_{rec}=α_{pred}=0.

C Real-World Experiments

C.1 Hardware Platform and Setup

To evaluate the proposed method in a real robotic setting, we consider an obstacle-aware reaching task on a Kinova Gen3 manipulator, as shown in Fig. 5. The robot moves its end-effector from an initial position to a goal position while avoiding static obstacles in the workspace. During execution, the planner operates online in a receding-horizon manner using the current robot state, goal specification, and obstacle representation.

The training data are generated offline using a custom-built B-spline planner over randomized start states, goal states, and obstacle layouts. The resulting collision-aware trajectories are used to train a trajectory diffusion teacher. The proposed student is then trained to distill the teacher policy under the same task conditioning, including the robot state, goal specification, and obstacle representation.

The teacher and student are evaluated with the same task specification and low-level controller. We use the Kinova Gen3 internal Cartesian position controller, with position targets published at 80

Table 7: Hardware comparison on the Kinova reaching task over 50 trials per method. Continuous metrics are reported as mean ± standard deviation; success rates are reported as percentages. Bold denotes the best results

Metric	Diffusion Teacher	Ours
Mean inference latency (ms) ↓	151.00 ± 3.73	**4.08 ± 0.20**
P95 inference latency (ms) ↓	158.27 ± 3.78	**4.73 ± 0.23**
Completion time (s) ↓	314.95 ± 3.54	**85.43 ± 2.53**
Terminal error (mm) ↓	**3.09 ± 1.42**	3.46 ± 1.75
Minimum clearance (mm) ↑	46.29 ± 12.91	**48.95 ± 12.12**
Success rate (%) ↑	0.0	**100.0**

Hz. Each planning request uses the latest available robot state, with at most one request in flight. Once planning completes, only the first action of the highest-scoring candidate is used to update the Cartesian position target at the next command cycle. Thus, the command publication rate is fixed, whereas the effective replanning rate depends on each method's computation time. During online planning, the DDPM teacher uses 30 reverse diffusion steps. Both methods use a prediction horizon of 32 steps and generate 64 candidates, which are ranked by a geometry-based score that rewards obstacle clearance and penalizes control effort. Success is defined as the end-effector reaching within $r_{\text{goal}} = 5\,\text{mm}$ of the goal position within $100\,\text{s}$, without end-effector collision with the obstacle during execution. This time threshold is used only to determine success; trials continue beyond $100\,\text{s}$ to measure the full task completion time.

Each method is evaluated over 50 hardware trials using inference latency, task completion time, minimum obstacle clearance, and success rate. For each trial, we compute the mean and 95th-percentile inference latency across recorded planning calls, and report each statistic as mean ± standard deviation across the 50 trials. Minimum end-effector clearance is the minimum distance between the end-effector position and the obstacle surface over the executed trajectory. All online inference is evaluated on a Titan workstation running Ubuntu 20.04 with two NVIDIA TITAN RTX GPUs, each with 24 GB of VRAM; this specification is reported for reproducibility and does not imply a deployment requirement.

C.2 Results

Table 7 reports the hardware results on the Kinova reconfiguration task. The proposed one-step student reduces the mean inference latency from $151.00\,\text{ms}$ to $4.08\,\text{ms}$, corresponding to an approximately $37.0\times$ reduction relative to the diffusion teacher. Its P95 inference latency is $4.73\,\text{ms}$, which remains well below the 12.5 ms command period. In contrast, the teacher cannot refresh the planned target within each 12.5 ms control period, resulting in a substantially lower effective replanning rate. This latency reduction is reflected in the task completion time, which decreases from $314.95\,\text{s}$ to $85.43\,\text{s}$.

DNK maintains comparable terminal accuracy and obstacle clearance while substantially reducing inference latency. Our method achieves a 100.0% success rate over 50 hardware trials. The terminal error remains comparable to the teacher, with $3.46 \pm 1.75\,\text{mm}$ for our method and $3.09 \pm 1.42\,\text{mm}$ for the teacher. Additionally, the proposed method achieves a slightly larger mean obstacle clearance, indicating that the faster inference does not compromise obstacle-avoidance behavior in this task.

Figure 6 further illustrates the difference in execution behavior. The diffusion teacher exhibits slower corrective motion and visible hesitation near the obstacle, whereas the proposed student replans more promptly and produces smoother obstacle-avoidance behavior before reaching the goal. These observations are consistent with the measured completion times and indicate that the distilled student is better suited to closed-loop real-time control on the physical system.

These results support the empirical effectiveness of factorized Koopman distillation for real-time hardware deployment.

D Discussions

D.1 FDK Component Ablation

To assess the contribution of the FDK transition, we replace $\mathbf{z}^+ = \mathbf{V}\,\text{diag}(\boldsymbol{\gamma}(\mathbf{c}))\mathbf{V}^{-1}\mathbf{z}$ with the identity map $\mathbf{z}^+ = \mathbf{z}$ on Franka Kitchen and state-based Push-T. Both variants share the same

Table 8: FDK component ablation on Franka Kitchen and state-based Push-T. Scores and latencies are reported as mean ± standard deviation across three evaluation groups of 50 episodes each. Every method uses one network evaluation per action chunk.

Task	Variant	Params. (M)	Score ↑	Latency (ms) ↓
Franka Kitchen	FDK (Ours)	2.457	0.973 ± 0.016	1.485 ± 0.004
Franka Kitchen	Without FDK	2.416	0.738 ± 0.023	1.128 ± 0.003
Push-T (state)	FDK (Ours)	2.367	0.931 ± 0.037	2.2 ± 1.0
Push-T (state)	Without FDK	2.333	0.563 ± 0.004	1.7 ± 0.9

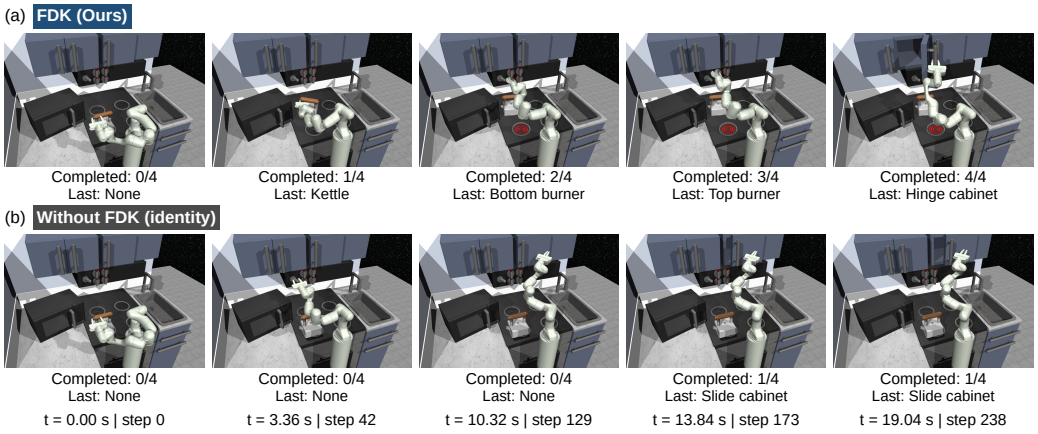


Figure 7: Qualitative FDK component ablation on Franka Kitchen. Time-aligned snapshots compare the full FDK student (top) with the identity variant ($z^+ = z$, bottom), starting from identical simulator states and initial observations. The FDK student sequentially completes the kettle, bottom-burner, top-burner, and hinge-cabinet subtasks by 19.04 s. The identity variant completes only the slide-cabinet subtask within the 280-step episode horizon (22.40 s). Timestamps denote simulated elapsed time rather than wall-clock inference latency.

retained-parameter initialization, data order, and action and latent-alignment losses. The identity variant additionally drops the spectral regularizer that applies only to FDK. Each run uses 150,000 updates and the task-specific checkpoint rule from Section 5.1.

Results. Table 8 shows that removing FDK sharply reduces task performance on both benchmarks, despite a smaller model and slightly lower latency. The mean score falls from 0.973 to 0.738 on Franka Kitchen (-0.235) and from 0.931 to 0.563 on Push-T (-0.368), while the parameter count decreases by only about 1.7% and 1.4%, respectively. These results show that the learned FDK transition contributes substantially to closed-loop performance relative to an identity latent mapping under the matched training setup.

Figure 7 compares the two Franka Kitchen policies using time-aligned snapshots from the same initial simulator states and observations. Ours sequentially completes the kettle, bottom-burner, top-burner, and hinge-cabinet subtasks by 19.04 s of simulated time. In contrast, the identity variant completes only the slide-cabinet subtask within the 280-step episode horizon (22.40 s). This example illustrates the observed performance difference across successive manipulation stages.

D.2 Ablation on Context-Dependent Modal Gains

To isolate the contribution of context-dependent modal adaptation, we compare the default FDK transition

$$\mathbf{K}_\omega(\mathbf{c}) = \mathbf{V}\,\mathrm{diag}\big(\boldsymbol{\gamma}_\omega(\mathbf{c})\big)\mathbf{V}^{-1}$$

against matched static-gain variants on Franka Kitchen.

All variants retain the same encoder, decoder, conditioned trajectory prior, FiLM conditioning, shared modal basis $\mathbf{V}$, training data, and optimization protocol, and differ only in the parameterization of the

Table 9: Matched ablation of modal-gain conditioning on Franka Kitchen. Scores and latencies are reported as mean ± standard deviation. Scores are computed across three evaluation groups of 50 episodes each. Bold denotes the best results.

Variant	Params. (M)	Score ↑	Latency (ms) ↓
Conditional $\boldsymbol{\gamma}_\omega(\mathbf{c})$	2.457	$\mathbf{0.973 \pm 0.016}$	1.485 ± 0.004
Static vector $\boldsymbol{\gamma}_0$	2.433	0.945 ± 0.018	$\mathbf{1.454 \pm 0.007}$
Static head (fixed reference)	2.457	0.910 ± 0.005	1.516 ± 0.014

Table 10: Ablation of the first-step weight in $\mathcal{L}_{\text{act}}$ on state-based Push-T. Scores are reported as mean ± standard deviation across three evaluation groups of 50 episodes each. "Worst group" denotes the minimum group mean, and "Lower 10%" denotes the mean score over the lowest-performing 10% of episodes. Latency follows the matched batch-1 host-to-host protocol in Table 2.

Variant	λ_{first}	Score ↑	Worst group ↑	Lower 10% ↑	Latency (ms) ↓	Params. (M) ↓
DNK (main)	0.5	$\mathbf{0.931 \pm 0.037}$	**0.904**	**0.378**	1.283 ± 0.007	2.37
DNK, no first-step emphasis	0.0	0.913 ± 0.030	0.887	0.212	1.283 ± 0.007	2.37

modal gains. The *static-vector* variant replaces $\boldsymbol{\gamma}_\omega(\mathbf{c})$ with a directly learned, context-independent vector $\boldsymbol{\gamma}_0 \in \mathbb{R}^{128}$, yielding $\mathbf{V}\,\mathrm{diag}(\boldsymbol{\gamma}_0)\mathbf{V}^{-1}$. To provide a parameter-matched control, the *static-head* variant retains the same gain-network architecture and parameter count as the conditional model, but replaces the current observation input with a fixed training-set reference. Its modal gains are therefore independent of the current observation while preserving the architecture of the conditional gain head.

Results. Table 9 shows that the context-dependent model achieves a score of 0.973 ± 0.016, compared with 0.945 ± 0.018 for the static-vector variant and 0.910 ± 0.005 for the parameter-matched static-head variant. Importantly, both static variants retain observation dependence through the conditioned trajectory prior and FiLM pathways; only the modal gains are prevented from adapting to the current observation. The resulting performance gap therefore provides evidence that, on the long-horizon Kitchen task, these alternative conditioning pathways do not fully substitute for observation-dependent adaptation of the modal transition.

The improved closed-loop performance of the conditional-gain model provides complementary empirical evidence for the benefit of context-dependent modal adaptation. At the latent level, Proposition 1 in Appendix A.1 shows that, for a fixed latent representation and shared modal basis, optimal context-dependent gains can achieve lower latent-prediction risk than constant gains when the optimal gains vary with context.

D.3 Ablation on Control-Oriented Supervision Loss $\mathcal{L}_{\text{act}}$

To isolate the effect of emphasizing the first action, we consider the control-oriented action loss in the Push-T task

$$\mathcal{L}_{\text{act}} = \mathcal{L}_{\text{chunk}} + \lambda_{\text{full}}\,\mathcal{L}_{\text{full}} + \lambda_{\text{first}}\,\mathcal{L}_{\text{first}}.$$

We retain the DNK architecture and training setup and modify only the first-step weight, setting $\lambda_{\text{first}} = 0.5 \to 0$ while keeping $\lambda_{\text{full}} = 0.25$ unchanged on state-based Push-T.

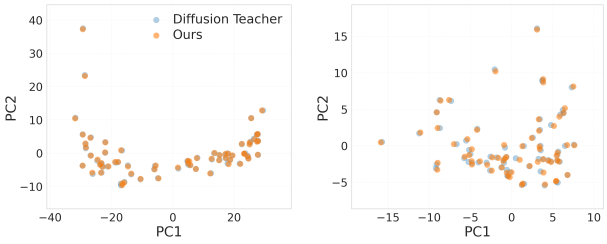
Table 10 reports the results over three evaluation groups. Because this ablation changes only the training-time loss weights, the deployed parameter count and inference latency are identical to those of the main DNK model in Table 2. Removing the first-step emphasis reduces the mean score from 0.931 to 0.913 and the worst-group score from 0.904 to 0.887. The degradation is more pronounced in the lower tail, where the lower-10% episode mean decreases from 0.378 to 0.212. These results suggest that first-step emphasis provides a modest improvement in average performance in push-T (state) task while more substantially improving performance on difficult episodes, without affecting inference cost. This behavior is consistent with chunked receding-horizon control: each decision executes an eight-step action chunk before replanning, so inaccuracies in the first executed action propagate through the remainder of the chunk.

Table 11: Reduced-capacity OneDP-S stress test on Franka Kitchen. Scores are reported as mean ± standard deviation across three evaluation groups of 50 episodes each.

Method	Student width	Params. (M) ↓	Score ↑	Latency (ms) ↓
OneDP-S (full)	256	68.92	0.975 ± 0.013	6.434 ± 0.045
OneDP-S (reduced)	128	19.50	0.055 ± 0.009	4.061 ± 0.837
OneDP-S (reduced)	64	6.22	0.068 ± 0.020	3.953 ± 0.789
OneDP-S (reduced)	32	2.44	0.045 ± 0.005	3.857 ± 0.816

D.4 Multimodal Candidate Generation

To assess whether the one-step student retains diversity across sampled candidates, we compare trajectories generated by DNK and the diffusion teacher from the same initial state. Fig. 8 shows that the candidate trajectories generated by our method occupy a broad, teacher-aligned region in the principal components. Across both locomotion tasks, DNK produces a broad set of candidates with substantial overlap with the teacher samples in the projected space. Together with the competitive closed-loop returns in Table 1, this result provides qualitative evidence that the distilled student retains teacher-aligned candidate diversity while using one-step inference.



(a) Walker2d. (b) HalfCheetah.

Figure 8: PCA projection of 64 candidate trajectories generated from the same initial state. The proposed method preserves broad candidate coverage with substantial overlap with teacher samples in the projected space.

D.5 Reduced-Capacity OneDP-S Stress Test

The one-step baselines differ substantially in deployed model capacity. On Franka Kitchen, OneDP-S (Wang et al., 2025) retains the 68.92M-parameter architecture of its diffusion teacher, whereas DNK uses only 2.46M parameters. To examine whether DNK's performance at this compact scale can be reproduced by simply narrowing OneDP-S, we evaluate reduced-width OneDP-S students using the same OneDP-S distillation objective and frozen full-size DDPM teacher.

All variants use the same observation and action horizons, execute eight actions per replanning step, use a single policy sample, and are evaluated on the same three groups of 50 episodes used in the main comparison. Changing the U-Net width makes the reduced students shape-incompatible with the teacher parameters, preventing direct initialization from the teacher's EMA weights. The reduced-width variants are therefore trained from random initialization. Accordingly, we treat this experiment as a capacity stress test of the OneDP-S training protocol rather than an architecture-matched comparison with DNK.

Results. Table 11 shows that reducing the OneDP-S backbone substantially degrades closed-loop performance. The 19.50M , 6.22M, and 2.44M-parameter variants achieve scores of 0.055, 0.068, and 0.045, respectively, whereas DNK achieves 0.973 with 2.46M parameters. Notably, even the 19.50M OneDP-S variant, which contains nearly eight times as many parameters as DNK, performs poorly under the reduced-width training protocol. These results show that simply reducing the OneDP-S backbone to a comparable parameter scale does not recover DNK's performance–capacity operating point under the tested training protocol.