Conditioning Matters: Training Diffusion Policies is Faster Than You Think

Zibin Dong[♡], **Yicheng Liu**[♣], **Yinchuan Li**[♢], **Hang Zhao**[*♣], **Jianye Hao**[*♡,♢]
[♡]Tianjin University, [♣]Tsinghua University, [♢]Huawei Noah's Ark Lab

Abstract

Diffusion policies have emerged as a mainstream paradigm for building vision-language-action (VLA) models. Although they demonstrate strong robot control capabilities, their training efficiency remains suboptimal. In this work, we identify a fundamental challenge in conditional diffusion policy training: when generative conditions are hard to distinguish, the training objective degenerates into modeling the marginal action distribution, a phenomenon we term *loss collapse*. To overcome this, we propose Cocos, a simple yet general solution that modifies the source distribution in the conditional flow matching to be condition-dependent. By anchoring the source distribution around semantics extracted from condition inputs, Cocos encourages stronger condition integration and prevents the loss collapse. We provide theoretical justification and extensive empirical results across simulation and real-world benchmarks. Our method achieves faster convergence and higher success rates than existing approaches, matching the performance of large-scale pre-trained VLAs using significantly fewer gradient steps and parameters. Cocos is lightweight, easy to implement, and compatible with diverse policy architectures, offering a general-purpose improvement to diffusion policy training.

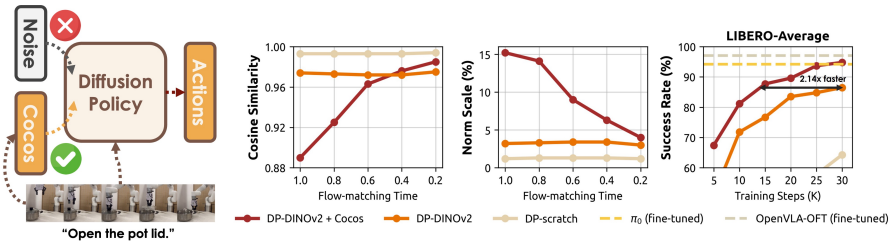


Figure 1: **Fusing generative condition into the source distribution greatly simplifies diffusion policy training.** Diffusion policy trained with our method achieves π_0 performance on the LIBERO benchmarks with only 30K gradient steps, which is 2.14x faster than the vanilla model. We also show the cosine similarity and the norm scale change between the policy hidden states before and after injecting condition information, demonstrating that our method fundamentally compels the policy network to utilize condition information, rather than simply embedding conditions into the source distribution.

1 Introduction

Denoising generative models have emerged as a scalable approach for high-dimensional data generation [28, 6, 26, 17, 18, 12, 30, 16], significantly advancing Vision-Language-Action (VLA) models in robotics (VLAs[2]) [19, 15, 1, 3, 36, 11]. Mainstream VLA models frame robot control as a conditional generation problem: given vision-language inputs as conditions, the model generates appropriate robot action sequences. Despite action sequences being substantially lower-dimensional than typical generative content like images or text, training VLAs remains unexpectedly challenging and

[*]Corresponding authors: Hang Zhao (hangzhao@mail.tsinghua.edu.cn), Jianye Hao (jianye.hao@tju.edu.cn).
[2]Unless otherwise specified, VLAs in this paper refer specifically to denoising generative VLA models.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

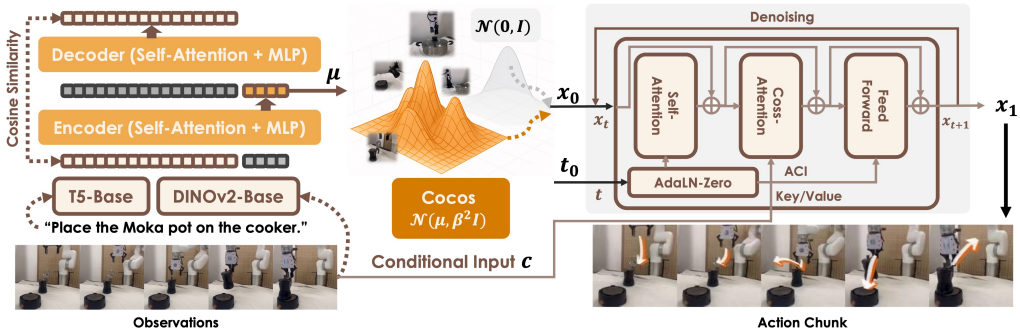


Figure 2: **Diffusion Policy w/ Cocos.** Our approach requires only replacing the standard Gaussian source distribution with a condition-conditioned Gaussian. An autoencoder compresses condition representations to match dimensionality, providing the mean while maintaining a fixed standard deviation.

resource-intensive. Recent efforts to enhance VLA training efficiency have explored various directions, including leveraging more powerful vision-language encoders [9, 1, 24, 10, 33, 32], designing compact and expressive action tokenizers [22, 31], and incorporating richer conditioning strategies [7, 8, 35, 14, 23]. These approaches converge on a critical insight: the core challenge may not lie in action generation itself, but rather in how models interpret and utilize conditional inputs.

To further investigate how condition inputs affect policy training, we empirically compare diffusion policies with visual encoders trained from scratch versus those initialized with DINOv2. We analyze the cosine similarity and norm scale change between policy network hidden states before and after condition information injection, correlating these metrics with LIBERO task success rates Figure 1. Higher cosine similarity and lower norm scale change indicate weaker condition influence on the policy network, directly correlating with degraded performance (as observed comparing DP-DINOv2 versus DP-scratch). The results demonstrate a fundamental issue: policies facing difficult-to-interpret conditions *actively omit conditional inputs*, producing actions disconnected from observations. This phenomenon persists even when using high-quality representations, though to a lesser degree.

While empirical results reveal this condition omission phenomenon, we conduct a theoretical analysis to identify its root cause in VLA training. We discover that when policy networks struggle to differentiate between generative conditions, the flow-matching objective degrades into an unconditional one that merely models the marginal action distribution. This loss collapse creates a destructive feedback loop: as the policy network begins ignoring conditional inputs, the training objective further degrades, reinforcing the network's tendency to discard conditions altogether rather than attempting to interpret them. To prevent this loss collapse, we introduce a novel conditional flow-matching approach with a **co**ndition-**co**nditioned **s**ource distribution (**Cocos**). Rather than adopting a standard Gaussian prior $q(z)$, Cocos anchors the source distribution around the semantics of each condition $q(z|c)$, theoretically preventing training loss collapse and forcing the policy network to remain responsive to condition inputs. As demonstrated in Figure 1, diffusion policies trained with Cocos exhibit 2.14x faster training and substantially higher success rates. Critically, the lower cosine similarity and increased norm scale change indicate that Cocos fundamentally compels the policy network to utilize condition information, rather than simply embedding conditions into the source distribution.

Specifically, we implement Cocos by formulating the source distribution as a Gaussian with fixed standard deviation, where the mean derives from latent representations produced by a vision-language autoencoder (see Figure 2). This approach introduces minimal architectural overhead, making it compatible with diverse VLA architectures and model scales. This simplicity, combined with strong theoretical guarantees and empirical benefits, positions Cocos as a general-purpose solution for preventing condition omission in VLA training.

To comprehensively validate Cocos, we conduct extensive evaluations across diverse settings: 70 simulation tasks from the LIBERO and MetaWorld benchmarks [13, 34], 10 real-world tasks on the low-cost open-source SO-100 robot platform [2], and 10 tasks on the high-performance xArm robot platform. Our results demonstrate significant improvements in both manipulation success rates and learning efficiency. Through detailed case studies, we analyze loss collapse manifestations and the mechanisms through which Cocos enhances performance. Our contributions include:

- We formulate the mathematical framework of flow matching with generative conditions and demonstrate that policy networks omit conditions when they are difficult to distinguish.

- We introduce **Cocos**, a simple yet effective source distribution modification that prevents loss collapse and significantly improves diffusion policy training efficiency and performance.
- We establish a comprehensive evaluation benchmark across diverse settings, including simulation tasks from LIBERO and MetaWorld, real-world tasks on a low-cost open-source robot (SO100), and tasks on a high-performance robot (xArm). Our results demonstrate consistent performance improvements across these varied platforms, validating Cocos as a general-purpose, plug-and-play solution for enhancing diffusion policy training.

2 Preliminaries

Problem Formulation. Our goal is to train a VLA $\pi_\theta(a_{1:H}|\mathcal{E}(o))$ that maps an observation o to sequences of future robot actions $a_{1:H}$ by imitating expert demonstrations. Here, o typically includes current and historical images as well as language instructions, while $\mathcal{E}$ represents a pre-trained vision-language encoder. In this paper, we focus on denoising generative model VLAs.

Conditional Flow Matching. Assume a smooth time-varying vector field $u : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ defines an ordinary differential equation (ODE) $\mathrm{d}x = u_t(x)\mathrm{d}t$, which pushforward samples from source distribution p_0 to the target distribution p_1. The density transported along u from time 0 to t is denoted as $p_t : [0,1] \times \mathbb{R}^d \to \mathbb{R}$. p_t and u_t satisfy the *continuity equation*: $\partial p/\partial t = -\nabla(p_t \cdot u_t)$. Suppose that the marginal probability path $p_t(x)$ is a mixture of probability paths $p_t(x|z)$, that is $p_t(x) = \int p_t(x|z)q(z)\mathrm{d}z$, where $q(z)$ is some distribution over the conditioning variable. If $p_t(x|z)$ is generated by the vector field $u_t(x|z)$ from initial conditions $p_0(x|z)$, then the vector field

$$u_t(x) := \mathbb{E}_{q(z)} \frac{u_t(x|z)p_t(x|z)}{p_t(x)} \tag{1}$$

generates the probability path $p_t(x)$. Let $v_\theta : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ be a time-dependent vector field parameterized as a neural network θ. Define the conditional flow matching (CFM) objective:

$$\mathcal{L}_{\text{CFM}}(\theta) := \mathbb{E}_{t,q(z),p_t(x|z)} \left\| v_\theta(t,x) - u_t(x|z) \right\|^2. \tag{2}$$

In VLA models, a typical choice sets $q(z) = q(x_0)q(x_1)$, corresponding to an independent coupling of the source and target distributions, and the conditional flow is defined by

$$p_t(x|z) = p_t(x|x_1, x_0),\ u_t(x|z) = u_t(x|x_1, x_0), \tag{3}$$

where the source distribution is a standard Gaussian. By optimizing CFM objective, $v_\theta(t,x)$ can converge to the vector field $u_t(x)$, with which we can solve the ODE and sample action chunks. *Note that this is a general formulation covering a wide range of denoising generative models, e.g., Var. Exploding [27], Var. Preserving [6] diffusion models, flow matching models [12], rectified flow models [16], etc. So our following discussion can be easily transferred to various policy backbones.*

3 How Generative Conditions Can Lead to Loss Collapse

While conditional flow matching offers a general framework for transporting probability distributions, it does not inherently consider condition inputs. Prior works adopt a straightforward strategy: injecting the generative condition directly into the policy network [4, 37, 38]. However, the implications of this approach have received little theoretical attention. In this section, we first analyze how the underlying flow formulation changes when generative conditions are introduced, then demonstrate that this seemingly benign modification can lead to a critical failure mode: loss collapse, where the training objective degenerates and the model learns to omit the generative condition altogether.

3.1 Flow with Generative Conditions

We consider a conditional vector field $u_t(x|c)$ that transports the source distribution $p_0(x|c)$ to the target $p_1(x|c)$, where c is the generative condition. Let $p_t(x|z,c)$ denote the time-varying intermediate density under generation condition c and flow condition z, with its associated vector field $u_t(x|z,c)$. Then, the overall conditional velocity can be expressed as:

$$u_t(x|c) := \mathbb{E}_{q(z)} \frac{u_t(x|z,c)p_t(x|z,c)}{p_t(x|c)}. \tag{4}$$

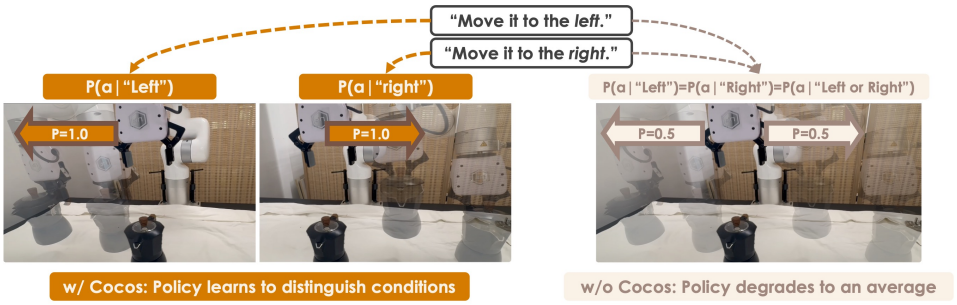


Figure 3: **Loss collapse causes the policy to degrade to an average.** The policy omits language instructions ('Move it to the left' or 'Move it to the right') and yields actions based on their frequency in the training data. Cocos prevents loss collapse, enabling the policy to produce distinct actions corresponding to 'Left' and 'Right'.

Suppose we introduce c into the neural estimator $v_\theta(t, x, c)$ of the conditional velocity $u_t(x|c)$. The corresponding training objective becomes:

$$\mathcal{L}_{\text{CFMc}}(\theta) := \mathbb{E}_{t,q(z),p_t(x|z,c)} \left\| v_\theta(t, x, c) - u_t(x|z, c) \right\|^2 . \tag{5}$$

Lemma 1. *If $p_t(x|c) > 0$ for all $x \in \mathbb{R}^d$ and $t \in [0, 1]$, up to a constant independent of θ, objective $\mathbb{E}_{t,p_t(x|c)} \left\| v_\theta(t, x, c) - u_t(x|c) \right\|^2$ and $\mathbb{E}_{t,q(z),p_t(x|z,c)} \left\| v_\theta(t, x, c) - u_t(x|z, c) \right\|^2$ are equal.*

By optimizing Equation (5), we can use $v_\theta(t, x, c)$ to estimate $u_t(x|c)$ and solve the ODE for generation. In practice, a common formulation of $q(z)$ in Equation (5) is $q(z) = q(x_0)q(x_1, c)$, $z = (x_1, x_0, c)$. This setup corresponds to sampling a condition-action pair from the dataset and perturbing the action with Gaussian noise during training. Under this formulation, the conditional flow and velocity field are defined by:

$$p_t(x|z, c) = \begin{cases} p_t(x|x_1, x_0) & , \ c \in z \\ 0 & , \ c \notin z \end{cases} , \text{ and } u_t(x|z, c) = \begin{cases} u_t(x|x_1, x_0) & , \ c \in z \\ 0 & , \ c \notin z \end{cases} \tag{6}$$

This reduces the objective to that implemented in previous works [4, 37, 38]:

$$\mathcal{L}_{\text{CFMc}}(\theta) := \mathbb{E}_{t,q(x_0),q(x_1,c),p_t(x|x_1,x_0)} \left\| v_\theta(t, x, c) - u_t(x|x_1, x_0) \right\|^2 . \tag{7}$$

3.2 Loss Collapse

While this conditional objective encourages the model to learn condition-dependent vector fields, a critical failure mode can occur: if the model fails to distinguish the condition information in $\mathcal{C}$ (e.g., for any $c_1, c_2 \in \mathcal{C}$, $\|v_\theta(t, x, c_1) - v_\theta(t, x, c_2)\| \leq \epsilon$), the objective effectively collapses into a marginal form on $\mathcal{C}$ that no longer depends on c.

Theorem 1 (Gradient Contraction under Independent Sampling)**.** *Under the independent sampling measure*

$$\mu(\mathrm{d}y) = q(x_1, c)\, q(x_0)\, p_t(x|x_1, x_0)\, \mathrm{d}x_0 \mathrm{d}x_1 \mathrm{d}x,$$

the difference of the score gradients for any two conditions $c_1, c_2 \in \mathcal{C}$ admits the bound

$$\left\| \nabla_\theta \mathcal{L}_{\text{CFMc}}(\theta, c_1) - \nabla_\theta \mathcal{L}_{\text{CFMc}}(\theta, c_2) \right\| \ \leq \ 2\,(M + KD)\,\epsilon,$$

provided that $\|\nabla_\theta v_\theta\| \leq M$, $\|d\| \leq D$, and the model output satisfies $\|v_\theta(t, x, c_1) - v_\theta(t, x, c_2)\| \leq \epsilon$ with its gradient being output-sensitive Lipschitz of constant K. The velocity estimator $v_\theta(t, x, c)$ tends to converge to a c-independent function $v^*(t, x)$ that minimizes the averaged objective:

$$v^*(t, x) := \arg\min_v \ \mathbb{E}_{c\in\mathcal{C}} \mathbb{E}_{z\sim\mu} \left[\|v(t, x) - u_t(x|x_1, x_0)\|^2 \right] .$$

By contrast, when samples are drawn from the conditional measure

$$\mu_c(\mathrm{d}y) = q(x_1, c)\, q(x_0|c)\, p_t(x|x_1, x_0)\, \mathrm{d}x_0 \mathrm{d}x_1 \mathrm{d}x,$$

the same gradient difference can become arbitrarily large unless additional coupling assumptions on $q(x_0|c)$ are imposed.

This result reveals a vicious cycle: as the policy network becomes confused by condition inputs, the objective degenerates into an unconditional one. This further reinforces the network's tendency to discard difficult-to-interpret conditions, leading to models that appear to be well-trained but fail to comprehend conditional cues during deployment. As shown in Figure 3, loss collapse causes the policy to average actions. In the next section, we introduce Cocos, a simple method to prevent it.

4 Method

In Theorem 1, we identify that loss collapse arises when c becomes indistinguishable to the policy network. This collapse is exacerbated by the standard choice of sampling z from $q(z) = q(x_1, c)q(x_0)$, which decouples x_0 from the condition. If $v_\theta(t, x, c) \approx v_\theta(t, x)$, the expectation in the objective effectively marginalizes over c, causing it to degenerate into an unconditional one.

To prevent it, we propose a simple yet effective modification: using a **co**ndition-**co**nditioned **s**ource distribution (**Cocos**) by sampling z from $q(z) = q(x_1, c)q(x_0|c)$. As shown in Theorem 1, this change directly avoids the loss collapse and enforces condition-awareness during training. While $q(x_0|c)$ can be designed in various ways, in this work, we choose to maintain the overall framework's simplicity by setting it as a Gaussian with fixed standard deviation, with the following objective:

$$\mathcal{L}_{\text{Cocos}}(\theta) := \mathbb{E}_{t, q(x_0|c), q(x_1,c), p_t} \left\| v_\theta - u_t \right\|^2, \text{ where } q(x_0|c) = \mathcal{N}(x_0; \alpha F_\phi(\mathcal{E}(c)), \beta^2 I), \quad (8)$$

where F_ϕ is a feature encoder that maps the condition representations to the action space dimensionality. The scalar α controls the strength of the condition prior, and β adjusts the uncertainty. Setting $\alpha = 0, \beta = 1$ recovers the commonly used formulation with a standard Gaussian prior. Incorporating Cocos requires only replacing the standard Gaussian source distribution with a condition-conditioned one. Algorithm 1 clearly shows the differences in training and inference pipelines.

In practice, we adopt an autoencoding objective on condition embeddings to train F_ϕ. Specifically, we minimize the negative cosine similarity between the original and reconstructed embeddings:

$$\mathcal{L}(\phi) = -\mathbb{E}_c \text{Sim}\left(G_\phi(F_\phi(\mathcal{E}(c))), \mathcal{E}(c)\right), \quad (9)$$

where F_ϕ and G_ϕ are the encoder and decoder networks, respectively. As a default setting, we adopt a two-stage training process: we first train the source distribution (i.e., the autoencoder) and then fix it during policy training. However, in practical scenarios, this two-step pipeline may introduce additional inflexibility. To address this, we also explore a joint training alternative. Simultaneous training of this autoencoder and the policy can be unstable due to the evolving distribution of x_0. To mitigate this, we adopt an Exponential Moving Average (EMA) strategy. We maintain a target network F_{ϕ^-}, updated via EMA from F_ϕ, and use this EMA copy during policy training, ensuring stability. In practice, this approach achieves performance comparable to the two-stage strategy, which we will discuss in detail in the experimental section.

We implement our policy network using a compact Robot Diffusion Transformer (RDT) [15]. The diffusion process follows a linear interpolation schedule [12]:

$$p_t(x|x_1, x_0) = \mathcal{N}(x; tx_1 + (1-t)x_0, \sigma^2), \quad u_t(x|x_1, x_0) = x_1 - x_0. \quad (10)$$

We select these design choices as they have been widely adopted in advanced VLA models [15, 1].

Algorithm 1 Training and Inference Pseudocode for Diffusion Policy with Cocos

Require: Policy Network v_θ, Condition Network $\mathcal{E}$, Pre-trained condition encoder F_ϕ

while not coverge **do** `# Training`
  $(x_1, c) \sim q(x_1, c)$, $t \sim \mathcal{U}(0, 1)$, $x_0 \sim q(x_0|c)$ Equation (8) `#` $x_0 \sim \mathcal{N}(0, I)$ `(without Cocos)`
  $\theta \leftarrow \theta + \nabla_\theta \mathcal{L}_{\text{CFMc}}(\theta)$ Equation (7)
end while
while not done **do** `# Inference`
  Observe c, $x_0 \sim q(x_0|c)$ Equation (8) `#` $x_0 \sim \mathcal{N}(0, I)$ `(without Cocos)`
  Solve ODE $\mathrm{d}x_t = v_\theta(t, x_t, c)\mathrm{d}t$ from $t = 0$ to $t = 1$.
  $a \leftarrow x_1$; Execute action a
end while

5 Experiments

We conduct comprehensive experiments to evaluate the effectiveness of Cocos in both simulated and real-world robot manipulation tasks. The experiments are designed to answer three core questions:

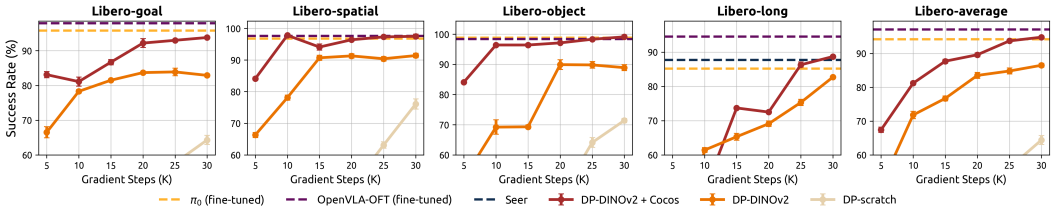


Figure 4: **Learning curve on LIBERO benchmark.** Dashed line scores are reported by Kim et al. [10].

Table 1: **Success rate on LIBERO benchmark.** Each of the four task suites includes 10 tasks, and we evaluate 150 trials for each task. The 1st, 2nd highest scores are emphasized with bold and the 3rd with underline. Cocos (0.2 std) is our default configuration, and other Cocos variants serve for an ablation study.

		DP-scratch	DP-DINOv2	Cocos (0.2 std)	Cocos (0.1 std)	Cocos (0.4 std)	Cocos (VAE)	Cocos (EMA)
LIBERO	Goal	64.3±1.3	82.9±0.4	**93.8±0.3**	50.9±1.8	92.3±0.9	**92.5±0.3**	90.5±0.4
	Spatial	76.1±1.6	91.4±0.7	**97.5±0.7**	97.1±0.4	96.7±0.5	96.1±0.7	**97.5±0.2**
	Object	71.4±0.7	88.9±1.0	**99.1±0.6**	79.3±0.8	97.6±1.0	**98.1±0.7**	96.9±0.5
	Long	45.9±1.6	82.7±0.2	**88.7±0.1**	82.7±0.5	**89.5±0.4**	88.2±0.6	87.3±0.6
Average		64.4	86.5	**94.8**	77.5	**94.0**	93.8	93.0

(RQ1) Does Cocos improve diffusion policies' training efficiency and final performance?

(RQ2) Does it mitigate the loss collapse phenomenon?

(RQ3) Does it facilitate policy learning in real-world settings across heterogeneous robot platforms?

5.1 Experimental Setup

Diffusion Policy. Our diffusion policy (DP) adopts a compact RDT policy network of approximately 40M parameters. The vision-language condition inputs are encoded using a DINOv2-Base [20] and a T5-Base [25]. We denote the resulting model as DP-DINOv2. To evaluate the effectiveness of Cocos, we compare three key variants: (1) DP-scratch, which is trained without any pre-trained vision encoder; (2) DP-DINOv2, which incorporates frozen DINOv2 features but uses the standard source distribution; and (3) DP-DINOv2 with Cocos, our full method with a condition-conditioned source distribution. The autoencoder used to define the source distribution in Cocos is implemented using a single-layer Transformer for both encoder and decoder components.

Benchmarks. Our simulation evaluations are based on the LIBERO and MetaWorld benchmarks. LIBERO includes 40 tasks in four task suites: *Goal*, *Spatial*, *Object*, and *Long*, to test different policy generalizations. MetaWorld includes 30 tasks from various difficulty levels. For real-world experiments, we deploy the models on two robot platforms: The SO100 robot (low-cost, open-sourced, equipped with dual RGB cameras) evaluated on 10 tasks in four suites: *Pick&Place*, *MoveTo*, *Wipe*, and *Unfold*; The xArm robot (higher-precision, equipped with one Intel RealSense L515 LiDAR camera) evaluated on 10 tasks in suites: *Pick&Place*, *Pot*, *Pour*, and *Moka*. We show detailed task configurations in Appendix B.

5.2 Overall Comparison (RQ1)

We evaluate the effectiveness of Cocos on two widely-used simulation benchmarks: LIBERO and MetaWorld. For LIBERO, we also include comparisons with fine-tuned π_0 [1], OpenVLA-OFT [10], and Seer [29]. π_0 and OpenVLA-OFT are SOTA large-scale pre-trained VLA models, featuring flow-matching and next-token prediction, respectively. Seer is not an end-to-end VLA model but predicts future frames to infer actions. We report the results in Table 1, Table 2 and Figure 4.

Finding 1: **Cocos significantly improves training efficiency and policy performance.** Across all LIBERO suites, Diffusion policy (DP) trained with Cocos outperforms its counterparts without Cocos. Notably, Cocos-enabled models reach high success rates (e.g., >95% on *Spatial* and *Object* suites) within only 5–10K training steps, achieving convergence roughly **2.14x** faster than DP-DINOv2. On average, Cocos improves LIBERO success rates by **8.3%**. Across all MetaWorld tasks, it also outperforms the counterpart by **25.7%** gain, showing consistent performance gains.

Table 2: **Success rate on MetaWorld benchmark.** We evaluate 150 trials for each task. The highest scores are emphasized in bold. We take a multi-task setting, using language instructions to differentiate each task.

	button-press	button-press-td	button-press-td-w	button-press-w	coffee-button	door-open	door-lock	door-unlock	drawer-close	drawer-open	**Average**
w/ Cocos	**100.0±0.0**	95.3±0.9	**94.7±3.4**	97.3±2.5	**100.0±0.0**	**99.3±0.9**	**44.7±1.9**	**93.3±4.1**	**100.0±0.0**	**100.0±0.0**	**74.8±0.9**
w/o Cocos	97.3±2.5	**96.0±1.6**	91.3±4.1	**99.3±0.9**	97.3±2.5	77.3±5.7	3.3±0.9	67.3±5.2	**100.0±0.0**	**100.0±0.0**	59.5±3.8
scratch	48.7±8.1	41.3±6.6	32.7±6.6	82.0±5.9	53.3±0.9	50.7±4.7	23.3±2.5	24.7±3.4	88.7±0.9	57.3±5.0	33.6±3.4
	faucet-close	faucet-open	handle-press	handle-press-s	window-close	window-open	handle-pull	handle-pull-s	basketball	bin-picking	
w/ Cocos	**82.0±2.8**	**100.0±0.0**	**99.3±0.9**	**100.0±0.0**	**100.0±0.0**	**94.0±3.3**	**2.7±0.9**	**9.3±4.7**	**80.7±8.6**	**64.0±6.2**	
w/o Cocos	80.7±3.8	84.0±4.3	98.0±1.6	98.0±1.6	**100.0±0.0**	89.3±4.1	1.3±0.9	0.0±0.0	0.7±0.9	46.0±5.9	
scratch	36.7±3.4	79.3±1.9	76.0±2.8	53.3±3.4	60.7±1.9	26.0±0.0	2.0±1.6	0.0±0.0	0.0±0.0	10.0±4.3	
	coffee-pull	coffee-push	dial-turn	hammer	sweep	sweep-into	soccer	shelf-place	disassemble	stick-push	
w/ Cocos	**79.3±6.2**	**60.7±5.0**	**60.0±2.8**	**76.7±6.6**	**72.0±8.5**	**66.0±8.6**	**14.0±3.3**	**26.7±0.9**	39.3±1.9	**94.0±4.3**	
w/o Cocos	72.7±4.7	55.3±5.2	24.0±4.3	72.7±1.9	9.3±2.5	20.0±3.4	**14.0±0.9**	3.3±2.5	**72.7±4.9**	12.7±1.9	
scratch	21.3±6.8	19.3±5.7	2.7±0.9	25.3±2.5	10.0±4.9	18.0±3.3	6.7±1.9	2.7±0.9	14.7±2.5	40.0±2.8	

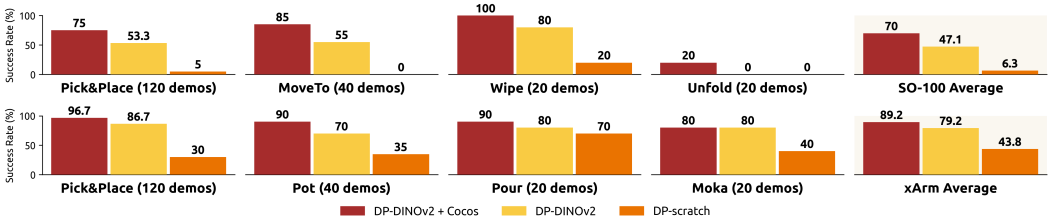


Figure 5: **Evaluation results on SO100 and xArm platforms.** We collect 4 task suites or 10 tasks for each robot platform. Each task provides 20 demonstrations and is tested over 10 trials.

Finding 2: **Cocos enables compact models to rival large-scale pre-trained VLAs.** Despite using fewer parameters and fewer gradient steps, DP with Cocos performs comparably to large-scale models like π_0 and OpenVLA-OFT. As described in Kim et al. [10], while OpenVLA-OFT takes 50-150K gradient steps for fine-tuning and π_0 takes 100-250K steps, our model reaches similar performance within 30K steps. On the challenging *Long* suite, Cocos even outperforms Seer, a non-end-to-end model with a larger parameter count. These results demonstrate that proper condition integration, rather than model scale alone, can be a major driver of performance in VLA training.

Finding 3: **Cocos encourages deeper condition utilization rather than just injecting priors.** To understand the nature of the improvement, we examine internal representations of the policy network before and after condition injection (Figure 1). Specifically, we measure cosine similarity and norm scale changes of hidden states (see Appendix C for details). A lower similarity and a higher norm shift reflect greater responsiveness to conditioning inputs. As shown in Figure 1, these metrics are consistently aligned with higher task success rates. This suggests that the benefit of Cocos does not stem merely from injecting condition priors into the source distribution. Instead, it actively compels the policy network to differentiate between and respond to condition inputs. If the gains were due only to initialization, we would expect weaker condition dependence, contrary to what we observe.

5.3 Real-World Experiments (RQ2)

To assess the practicality of our approach, we deploy diffusion policies trained with and without Cocos on two real-world robotic platforms: **SO100**, a low-cost and open-source arm, and **xArm**, a widely used high-precision industrial robot. The two platforms offer complementary insights: SO100 lowers the entry barrier for research and reproducibility, while xArm reflects deployment scenarios with more demanding task complexity and control fidelity.

As shown in Figure 5 and Figure 6, Cocos consistently improves task success rates across all task suites on both robots. The improvement is particularly pronounced on SO100, where the relatively lightweight structure makes it more sensitive to inaccuracies in action generation. We observe that both policy variants, with and without Cocos, are capable of generating smooth and coherent trajectories. However, the main failure modes of the baseline model are closely linked to misinterpretation of condition inputs, for example, confusing spatial references in language, or failing to correctly localize an object in the visual scene. In contrast, these types of errors are reduced when using Cocos, indicating improved condition understanding and robustness. These trends align with our theoretical motivation: condition misalignment during training can lead to loss collapse and unreliable behavior at inference time. The fact that Cocos provides consistent improvements across two distinct robotic platforms further highlights its practical value. In the next section, we present detailed case studies to examine these qualitative differences in greater depth.

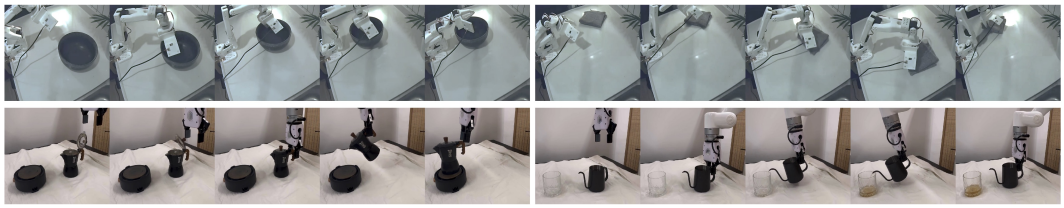


Figure 6: **Rollout samples of DP with Cocos on real-robot manipulation,** including *MoveTo (Bowl)* (Top Left), *Wipe* (Top Right), *Moka* (Bottom Left), and *Pour* (Bottom Right).

Figure 7: **Real robot case study.** Top left rollout comes from policy w/ Cocos and the others are from baselines.

5.4 Case Studies (RQ3)

To better understand the mechanisms behind Cocos's effectiveness, we conduct qualitative case studies on both simulation and real-world tasks, focusing on common failure patterns and condition sensitivity.

Finding 1: **Cocos improves utilization of 3rd-person visual inputs in simulation.** In the LIBERO benchmark, we find that policies trained without Cocos tend to omit 3rd-person views, relying almost exclusively on the wrist-mounted 1st-person camera. To quantify this, we compute cosine similarity between randomly sampled visual features extracted before fusion into the policy network. As shown in Figure 8 bottom table, models without Cocos produce highly similar 3rd-person embeddings across diverse scenes, indicating weak discriminability, while Cocos-trained models show significantly greater variation, suggesting stronger visual grounding.

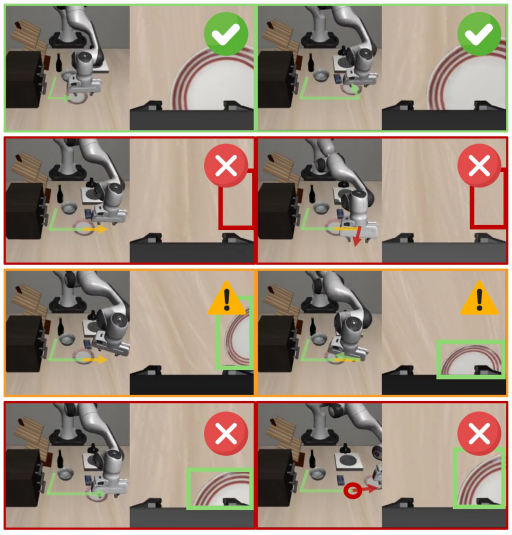
This contrast is clearly illustrated through the rollout comparisons in Figure 8. (Row 1) shows a successful execution by the model trained with Cocos: the robot pushes the plate forward, turns smoothly to the left, and finally pushes the plate backward to the stove, demonstrating accurate spatial reasoning and multi-step planning. This trajectory serves as a reference for the expected behavior. In (Row 2), a failure case from the model without Cocos, the robot accidentally releases the plate mid-way and attempts to recover by locating it again. However, since the plate is no longer visible in the 1st-person view, the policy fails to re-establish contact, even though the 3rd-person view clearly shows the plate's new location. This indicates that the model underutilizes the auxiliary camera input. (Row 3) presents a similar failure condition, but the plate remains within the first-person field of view after being released. In this case, the robot successfully relocates and completes the task. Comparing Row 2 and Row 3 highlights the model's strong reliance on the 1st-person view: if the object leaves that view, recovery becomes unlikely. (Row 4) shows another typical failure from the non-Cocos model: after reaching the stove, the robot continues to push the plate beyond the goal region. This happens because the 1st-person view offers no clear signal indicating proximity to the stove, while the 3rd-person view does. Again, the model fails to leverage this available information. Taken together, these cases reveal a consistent pattern: policies trained without Cocos tend to over-rely on the wrist-mounted camera and struggle to utilize 3rd-person observations. In contrast, models trained with Cocos seldom exhibit these failures, suggesting that the policy network learns to better incorporate multi-view context and is more robust to visual uncertainty.

Finding 2: **Cocos reduces modality over-reliance and improves contextual awareness in real-world settings.** In real-robot experiments, we observe that models without Cocos frequently exhibit modality over-reliance, particularly favoring language instructions or arm states over visual cues. As shown in Figure 7, when tasks begin in an already-completed configuration (e.g., lid already open), the baseline policy often executes a full rollout regardless of the visual scene (left bottom). This indicates

that the model has learned to follow the instruction without verifying whether it is still applicable. By contrast, policy trained with Cocos correctly remains idle in such states, indicating improved contextual awareness (left top). Similar trends appear in other tasks such as *SpongeMoveRight* (right top) and *ScissorMoveRight* (right bottom), where the baseline struggles to locate target objects or misinterprets spatial references. These errors are less frequent with Cocos, which appears to enhance the policy's sensitivity to both visual and semantic nuances in the scene. Together, these case studies provide concrete support for our theoretical claims: Cocos prevents loss collapse not just in training metrics, but in how models behave under ambiguous or weakly conditioned scenarios.

5.5 Ablation Studies

Cocos only requires the source distribution to be condition-dependent, i.e., $q(z|c)$, and admits flexibility in its concrete form. For simplicity, we instantiate it as a Gaussian with fixed standard deviation and a condition-dependent mean (Equation (8)). Our default setting uses $\beta = 0.2$, and we compare it against $\beta = 0.1$ and $\beta = 0.4$ to investigate the impact of the standard deviation. Inspired by the similarity between our formulation and the variational autoencoder (VAE) posterior, we evaluate a VAE-based version of Cocos, where the standard deviation is automatically learned from data. As shown in Table 1, we observe that the default ($\beta = 0.2$) and the wider variant ($\beta = 0.4$), and the VAE version all achieve comparable performance across LIBERO task suites. However, the model with $\beta = 0.1$ performs substantially worse, suggesting that overly concentrated $q(z|c)$ hinders training. Overall, these results indicate that Cocos is robust to the choice of β, as long as $q(z|c)$ remains sufficiently spread.



Average feature cosine similarities (↓)

view	w/ Cocos	w/o Cocos	scratch
3rd-person	0.514	0.843	0.971
1st-person	0.206	0.324	0.820

Figure 8: **Case study of LIBERO plate movement.**

We also compare the default two-stage training approach, where the source distribution is pretrained and then fixed, with a joint training strategy using exponential moving average (EMA) updates. In this version, a slowly updated target encoder F_{ϕ^-} is used during policy training to provide stable condition-dependent sampling. This online approach achieves performance on par with the two-stage method while simplifying the pipeline, making it a practical choice for integrated training scenarios.

6 Conclusion, Limitations, and Future Works

In this paper, we identify and address a critical failure mode in diffusion policy training, loss collapse, where the model fails to distinguish between generative conditions and degenerates into modeling marginal action distributions. We propose Cocos, a simple yet effective modification that injects condition-awareness directly into the source distribution. We provide a theoretical analysis showing how Cocos prevents loss collapse and demonstrate its empirical benefits across extensive simulations and real-world tasks. Our results show that Cocos improves both training efficiency and policy performance, enabling even compact models to match or exceed the performance of large-scale pretrained VLA systems. Despite its effectiveness, Cocos leaves open several important directions. First, while the method only requires the source distribution to be condition-dependent ($q(z|c)$), we have so far instantiated it using a fixed-variance Gaussian derived from autoencoding the condition. Exploring more expressive or learnable source distributions, such as flow-based or attention-guided priors, may further enhance its adaptability. Second, our current experiments are focused on multi-task imitation learning. Extending Cocos to large-scale pretraining and evaluating its ability to generalize across broad VLA domains remains a promising area for future work.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant Nos. 62422605, 92370132). We would like to thank all the anonymous reviewers for their valuable comments and constructive suggestions, which have greatly improved the quality of this paper.

References

[1] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *arXiv preprint arXiv:2410.24164*, 2024.

[2] Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, and Thomas Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. https://github.com/huggingface/lerobot, 2024.

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

[4] Eugenio Chisari, Nick Heppert, Max Argus, Tim Welschehold, Thomas Brox, and Abhinav Valada. Learning robotic manipulation policies from point clouds with conditional flow matching. In *8th Annual Conference on Robot Learning, CoRL*, 2024.

[5] 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. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track, NIPS*, 2024.

[6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems, NIPS*, 2020.

[7] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *8th Annual Conference on Robot Learning, CoRL*, 2024.

[8] Yuanchen Ju, Kaizhe Hu, Guowei Zhang, Gu Zhang, Mingrun Jiang, and Huazhe Xu. Robo-abc: Affordance generalization beyond categories via semantic correspondence for robot manipulation. In *European Conference on Computer Vision, ECCV*, 2025.

[9] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *8th Annual Conference on Robot Learning, CoRL*, 2024.

[10] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *arXiv preprint arXiv:2502.19645*, 2025.

[11] Yinchuan Li, Xinyu Shao, Jianping Zhang, Haozhi Wang, Leo Maxime Brunswic, Kaiwen Zhou, Jiqian Dong, Kaiyang Guo, Xiu Li, Zhitang Chen, Jun Wang, and Jianye Hao. Generative models in decision making: A survey. In *arXiv preprint arXiv:2502.17100*, 2025.

[12] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023.

[13] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, qiang liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track, NIPS*, 2023.

[14] Jiaming Liu, Hao Chen, Pengju An, Zhuoyang Liu, Renrui Zhang, Chenyang Gu, Xiaoqi Li, Ziyu Guo, Sixiang Chen, Mengzhen Liu, Chengkai Hou, Mengdi Zhao, KC alex Zhou, Pheng-Ann Heng, and Shanghang Zhang. Hybridvla: Collaborative diffusion and autoregression in a unified vision-language-action model. In *arXiv preprint arXiv:2503.10631*, 2025.

[15] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1b: a diffusion foundation model for bimanual manipulation. In *The Thirteenth International Conference on Learning Representations, ICLR*, 2025.

[16] Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations, ICLR*, 2023.

[17] 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. In *Advances in Neural Information Processing Systems, NIPS*, 2022.

[18] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. In *arXiv preprint arXiv:2211.01095*, 2022.

[19] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems, RSS*, 2024.

[20] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research, TMLR*, 2024.

[21] William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022.

[22] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. In *arXiv preprint arXiv:2501.09747*, 2025.

[23] Zekun Qi, Wenyao Zhang, Yufei Ding, Runpei Dong, Xinqiang Yu, Jingwen Li, Lingyun Xu, Baoyu Li, Xialin He, Guofan Fan, Jiazhao Zhang, Jiawei He, Jiayuan Gu, Xin Jin, Kaisheng Ma, Zhizheng Zhang, He Wang, and Li Yi. Sofar: Language-grounded orientation bridges spatial reasoning and object manipulation. In *arXiv preprint arXiv:2503.10631*, 2025.

[24] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. Spatialvla: Exploring spatial representations for visual-language-action model. In *arXiv preprint arXiv:2501.15830*, 2025.

[25] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research, JMLR*, 2020.

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

[27] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *arXiv preprint arXiv:1907.05600*, 2020.

[28] 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, ICLR*, 2021.

[29] Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. In *The Thirteenth International Conference on Learning Representations, ICLR*, 2025.

[30] Alexander Tong, Kilian FATRAS, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research, TMLR*, 2024.

[31] Zihao Wang, Shaofei Cai, Zhancun Mu, Haowei Lin, Ceyao Zhang, Xuejie Liu, Qing Li, Anji Liu, Xiaojian Ma, and Yitao Liang. OmniJARVIS: Unified vision-language-action tokenization enables open-world instruction following agents. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems, NIPS*, 2024.

[32] Junjie Wen, Minjie Zhu, Yichen Zhu, Zhibin Tang, Jinming Li, Zhongyi Zhou, Chengmeng Li, Xiaoyu Liu, Yaxin Peng, Chaomin Shen, and Feifei Feng. Diffusion-vla: Scaling robot foundation models via unified diffusion and autoregression. In *arXiv preprint arXiv:2412.03293*, 2024.

[33] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. In *IEEE Robotics and Automation Letters, RAL*, 2025.

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

[35] Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction in robotics. In *Proceedings of The 8th Conference on Robot Learning, CoRL*, 2025.

[36] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *Proceedings of Robotics: Science and Systems, RSS*, 2024.

[37] Fan Zhang and Michael Gienger. Affordance-based robot manipulation with flow matching. In *arXiv preprint arXiv:2409.01083*, 2025.

[38] Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. Flowpolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation. In *arXiv preprint arXiv:2412.04987*, 2024.

A Proofs of Theorems

Proof of Lemma 1. As in Tong et al. [30] we assume that $q, p_t(x|z)$ are decreasing to zero at sufficient speed as $\|x\| \to \infty$ and that $u_t, v_t, \nabla_\theta v_t$ are bounded.

$$\begin{aligned}\nabla_\theta \mathbb{E}_{p_t(x|c)} \|v_\theta(t,x,c) - u_t(x|c)\|^2 = \\ \nabla_\theta \mathbb{E}_{p_t(x|c)} \left(\|v_\theta(t,x,c)\|^2 - 2\left\langle v_\theta(t,x,c), u_t(x|c)\right\rangle + \|u_t(x|c)\|^2 \right) \\ = \nabla_\theta \mathbb{E}_{p_t(x|c)} \left(\|v_\theta(t,x,c)\|^2 - 2\left\langle v_\theta(t,x,c), u_t(x|c)\right\rangle \right) \end{aligned} \tag{11}$$

$$\begin{aligned}\nabla_\theta \mathbb{E}_{q(z),p_t(x|z,c)} \|v_\theta(t,x,c) - u_t(x|z,c)\|^2 = \\ \nabla_\theta \mathbb{E}_{q(z),p_t(x|z,c)} \left(\|v_\theta(t,x,c)\|^2 - 2\left\langle v_\theta(t,x,c), u_t(x|z,c)\right\rangle + \|u_t(x|z,c)\|^2 \right) \\ = \nabla_\theta \mathbb{E}_{q(z),p_t(x|z,c)} \left(\|v_\theta(t,x,c)\|^2 - 2\left\langle v_\theta(t,x,c), u_t(x|z,c)\right\rangle \right) \end{aligned} \tag{12}$$

By bilinearity of the 2-norm and since u_t is independent of θ. Next,

$$\mathbb{E}_{p_t(x|c)} \|v_\theta(t,x,c)\|^2 = \int \|v_\theta(t,x,c)\|^2 p_t(x|c) \mathrm{d}x \tag{13}$$

$$= \iint \|v_\theta(t,x,c)\|^2 p_t(x|z,c) q(z) \mathrm{d}z \mathrm{d}x \tag{14}$$

$$= \mathbb{E}_{q(z),p_t(x|z,c)} \|v_\theta(t,x,c)\|^2 \tag{15}$$

Finally,

$$\mathbb{E}_{p_t(x|c)} \left\langle v_\theta(t,x,c), u_t(x|c)\right\rangle = \int \left\langle v_\theta(t,x,c), u_t(x|c)\right\rangle p_t(x|c) \mathrm{d}x \tag{16}$$

$$= \int \left\langle v_\theta(t,x,c), \mathbb{E}_{q(z)} \frac{u_t(x|z,c) p_t(x|z,c)}{p_t(x|c)} \right\rangle p_t(x|c) \mathrm{d}x \tag{17}$$

$$= \int \left\langle v_\theta(t,x,c), \int u_t(x|z,c) p_t(x|z,c) q(z) \right\rangle \mathrm{d}z \mathrm{d}x \tag{18}$$

$$= \iint \left\langle v_\theta(t,x,c), u_t(x|z,c)\right\rangle p_t(x|z,c) q(z) \mathrm{d}z \mathrm{d}x \tag{19}$$

$$= \mathbb{E}_{q(z),p_t(x|z,c)} \left\langle v_\theta(t,x,c), u_t(x|z,c)\right\rangle \tag{20}$$

Since the gradients of the two objectives are equal for any time t, the two objectives are equal. □

Proof of Theorem 1. Define

$$d(t,x,c) \coloneqq v_\theta(t,x,c) - u_t\big(x|x_1,x_0\big), \quad f(z,c) \coloneqq \nabla_\theta v_\theta(t,x,c)^\top \, d(t,x,c), \tag{21}$$

where we write $y = (x_0, x_1, x)$. Introduce two measures:

- *Independent measure*

$$\mu(\mathrm{d}y) = q(x_1,c)\, q(x_0)\, p_t\big(x|x_1,x_0\big)\, \mathrm{d}x_0\, \mathrm{d}x_1\, \mathrm{d}x, \tag{22}$$

  which does *not* depend on the particular choice of c, corresponding to the design choice where $q(z) = q(x_1,c)q(x_0)$.

- *Conditional measure*

$$\mu_c(\mathrm{d}y) = q(x_1,c)\, q\big(x_0|c\big)\, p_t\big(x|x_1,x_0\big)\, \mathrm{d}x_0\, \mathrm{d}x_1\, \mathrm{d}x, \tag{23}$$

  which *does* depend on c, corresponding to the design choice where $q(z) = q(x_1,c)q(x_0|c)$.

We assume the following bounds hold for all y and $c \in \mathcal{C}$:

$$\left\|\nabla_\theta v_\theta(t,x,c)\right\| \le M, \quad \left\|d(t,x,c)\right\| \le D, \tag{24}$$

and that the model's gradient is *output-sensitive Lipschitz*, namely there exists $K > 0$ such that

$$\left\|\nabla_\theta v_\theta(t,x,c_1) - \nabla_\theta v_\theta(t,x,c_2)\right\| \le K \left\|v_\theta(t,x,c_1) - v_\theta(t,x,c_2)\right\| \le K\,\epsilon. \tag{25}$$

(i) Independent measure. Under μ, the gradient is

$$\nabla_\theta \mathcal{L}_{\mathrm{CFMc}}(\theta,c) = 2\int f(z,c)\,\mu(\mathrm{d}y). \tag{26}$$

Hence for any c_1, c_2:

$$\left\|\nabla_\theta \mathcal{L}_{\mathrm{CFMc}}(\theta,c_1) - \nabla_\theta \mathcal{L}_{\mathrm{CFMc}}(\theta,c_2)\right\| \tag{27}$$

$$= 2\left\|\int f(z,c_1)\,\mu(\mathrm{d}y) - \int f(z,c_2)\,\mu(\mathrm{d}y)\right\| \tag{28}$$

$$= 2\left\|\int \left[f(z,c_1) - f(z,c_2)\right]\mu(\mathrm{d}y)\right\|. \tag{29}$$

Expanding the integrand,

$$f(z,c_1) - f(z,c_2) = \nabla_\theta v_\theta(t,x,c_1)^\top\, d(t,x,c_1) - \nabla_\theta v_\theta(t,x,c_2)^\top\, d(t,x,c_2) \tag{30}$$

$$= \nabla_\theta v_\theta(t,x,c_1)^\top \left[d(t,x,c_1) - d(t,x,c_2)\right] \tag{31}$$

$$+ \left[\nabla_\theta v_\theta(t,x,c_1) - \nabla_\theta v_\theta(t,x,c_2)\right]^\top d(t,x,c_2). \tag{32}$$

Using the bounds $\|d(c_1) - d(c_2)\| \le \epsilon$ and $\|\nabla_\theta v(c_1) - \nabla_\theta v(c_2)\| \le K\epsilon$, we obtain

$$\left\|\nabla_\theta \mathcal{L}_{\mathrm{CFMc}}(\theta,c_1) - \nabla_\theta \mathcal{L}_{\mathrm{CFMc}}(\theta,c_2)\right\| \le 2\int \Big(M\,\epsilon + K\,\epsilon\,D\Big)\,\mu(\mathrm{d}y) \tag{33}$$

$$= 2\,(M + KD)\,\epsilon. \tag{34}$$

The above bound shows that when all $c \in \mathcal{C}$ induce similar gradients, the network update over this region becomes nearly invariant to c. In the limit of gradient descent, the parameter θ thus evolves according to a shared direction, regardless of the exact c value. As a result, the learned model $v_\theta(t,x,c)$ tends to converge, for all $c \in \mathcal{C}$, to a c-independent function $v^*(t,x)$ that minimizes the averaged objective:

$$v^*(t,x) := \arg\min_v\ \mathbb{E}_{c\in\mathcal{C}}\mathbb{E}_{z\sim\mu}\left[\|v(t,x) - u_t(x|x_1,x_0)\|^2\right]. \tag{35}$$

That is, the network effectively collapses the conditional dependency on c and behaves like a shared estimator v^* optimized for average performance across $\mathcal{C}$.

(ii) Conditional measure. Under μ_c, the gradient becomes

$$\nabla_\theta \mathcal{L}_{\mathrm{CFMc}}(\theta,c) = 2\int f(z,c)\,\mu_c(\mathrm{d}y). \tag{36}$$

For c_1, c_2, write

$$\nabla_\theta \mathcal{L}_{\mathrm{CFMc}}(\theta,c_1) - \nabla_\theta \mathcal{L}_{\mathrm{CFMc}}(\theta,c_2) \tag{37}$$

$$= 2\Big[\int f(z,c_1)\,\mu_{c_1}(\mathrm{d}z) - \int f(z,c_2)\,\mu_{c_2}(\mathrm{d}z)\Big] \tag{38}$$

$$= 2\Big[\Big(\int f(z,c_1)\,\mu_{c_1}(\mathrm{d}z) - \int f(z,c_1)\,\mu_{c_2}(\mathrm{d}z)\Big) + \Big(\int f(z,c_1)\,\mu_{c_2}(\mathrm{d}z) - \int f(z,c_2)\,\mu_{c_2}(\mathrm{d}z)\Big)\Big] \tag{39}$$

$$= T_1 + T_2. \tag{40}$$

The second term T_2 coincides with the independent-measure analysis and thus is bounded by $2(M + KD)\,\epsilon$. The first term

$$T_1 = 2\Big(\int f(z, c_1)\,\mu_{c_1}(\mathrm{d}z) - \int f(z, c_1)\,\mu_{c_2}(\mathrm{d}z)\Big) \tag{41}$$

is the difference of the same function under two *different* measures. Without any further assumption on the relationship between μ_{c_1} and μ_{c_2}, this difference can be made arbitrarily large. For instance, if $q(x_0|c_1) = \delta_a$ and $q(x_0|c_2) = \delta_b$, then $\|T_1\| = 2|f(a, c_1) - f(b, c_1)|$ can diverge independently of ϵ. Hence, under the conditional measure μ_c, no bound depending only on ϵ exists for $\|\nabla_\theta L(\theta, c_1) - \nabla_\theta L(\theta, c_2)\|$. □

B Details of Benchmarks

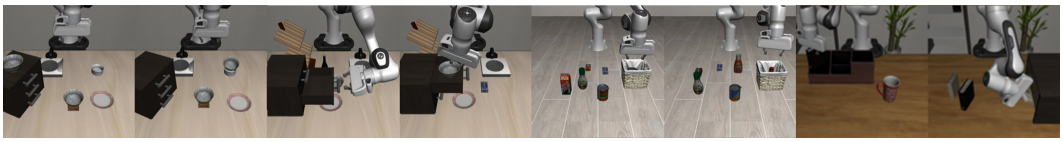


Figure 9: **LIBERO simulation benchmark.** We conduct experiments on 40 tasks from four task suites in the LIBERO benchmark. We show two task examples for each suite here.

Figure 10: **MetaWorld simulation benchmark.** We conduct experiments on 30 tasks of three difficulty levels in the MetaWorld benchmark. We show all task examples here.

Figure 11: **Real-world experimental setups.** We conduct experiments on both SO100 and xArm platform. For each robot, we design a suite of 10 tabletop tasks involving diverse objects.

LIBERO. The LIBERO simulation benchmark [13] features a Franka Emika Panda arm in simulation across four challenging task suites: *Goal*, *Spatial*, *Object*, and *Long*. Each suite comprises 10 tasks with 500 demonstrations and is designed to investigate controlled knowledge transfer related to goal variations, spatial configurations, object types, and long-horizon tasks. Unlike prior work [9, 10], we do not filter out unsuccessful demonstrations, aiming for a more realistic evaluation setting. For policy training, the model predicts action chunks of length 16; after each chunk prediction, 8 steps are executed before generating the next chunk. The observation space includes 2-view RGB images at the current time step, without historical observations. During evaluation, following Liu et al. [13], each task is tested over 50 trials with 3 different random seeds, and success rates are reported. To

provide a clearer understanding of the task suites, we present agent-view observations in Figure 9 and detailed task descriptions in Table 3.

MetaWorld. The MetaWorld simulation benchmark [34] includes 50 distinct tabletop manipulation tasks using a Sawyer robot arm. We select 30 tasks from *easy*, *medium*, and *very hard* difficulty levels to evaluate VLA models. We use a scripted policy to collect 20 demonstrations for each task. For policy training, the model predicts action chunks of length 16; after each chunk prediction, 16 steps are executed before generating the next chunk. The observation space consists of a single RGB image at the current time step, without historical observations. During evaluation, each task is tested over 50 trials with 3 different random seeds, and success rates are reported. To better illustrate the task suites, we show agent-view observations in Figure 10 and task descriptions in Table 4.

SO100 Robot Manipulation. The SO100 robot [2] is a low-cost, open-source 6-DoF manipulator, with both the leader and follower arms costing approximately $250. We assemble the hardware using a 3D-printed kit provided by the open-source community. The robot has two RGB cameras: one mounted on the wrist and the other positioned to provide a third-person view. Both cameras operate at a resolution of 640×480 and 25 FPS. The robot controller runs at 30Hz, and actions are defined as target absolute joint angles. Due to its low-cost design, the platform has several hardware limitations, including significant arm jitter, low load capacity, and occasional camera lag, which present practical challenges for developing embodied AI systems. However, given the increasing adoption of such affordable open-source robots by the research community, we believe that evaluating models on these lower-performance systems offers valuable insights and broader applicability. We design four categories of tabletop manipulation tasks for the SO100 setup: ***(1) Pick&Place:*** involving 3 objects and 2 placement zones (6 tasks), ***(2) MoveTo:*** navigating 2 objects to a single target zone (2 tasks), ***(3) Wipe:*** picking up a cloth and wiping the table (1 task), and ***(4) Unfold:*** unfolding a cloth (1 task). In total, we evaluate performance on 10 distinct tasks. Language instructions for each task are listed in Table 5, and visual examples of the task environments are shown in Figure 11.

During data collection, we record 20 demonstrations per task. For policy training, the model predicts an action chunk of length 64; after each chunk prediction, 40 steps are executed before generating the next chunk. The observation space includes 2-view RGB images at the current time step, along with the absolute joint angles from the current and previous 10 steps. During evaluation, each task is tested over 10 trials, and success rates are reported.

xArm Robot Manipulation. xArm is a high-performance 7-DoF manipulator. The robot is equipped with a third-person view Intel RealSense L515 LiDAR camera, operating at 640×480 resolution and 30 FPS. We collect both RGB and depth images from the camera. The robot controller runs at 30Hz, and actions are defined as target absolute joint angles. We design four categories of tabletop manipulation tasks for the xArm setup: ***(1) Pick&Place:*** involving 3 objects and 2 placement zones (6 tasks), ***(2) Pot:*** taking off or putting on the pot lid (2 tasks), ***(3) Pour:*** pouring water from the kettle into the cup (1 task), and ***(4) Moka:*** placing the Moka pot on the cooker (1 task). In total, we evaluate performance on 10 distinct tasks. Language instructions for each task are listed in Table 5, and visual examples of task environments are shown in Figure 11.

During data collection, we record 20 demonstrations per task. For policy training, the model predicts an action chunk of length 64; after each chunk prediction, 40 steps are executed before generating the next chunk. The observation space includes a third-person view RGB image at the current time step, as well as the absolute joint angles from the current and previous 10 steps. During evaluation, each task is tested over 10 trials, and success rates are reported.

C Details of Evaluation Metrics

Cosine similarity. To quantify the influence of condition inputs on our policy network, we measure the cosine similarity between hidden states before and after condition injection. Given an L-layer RDT policy network where conditions are integrated via cross-attention in each transformer block, we denote the hidden states before and after condition injection as $h^l, \bar{h}^l \in \mathbb{R}^{S\times D}$, where S is the sequence length and D is the transformer hidden dimension. Since earlier layers' transformations propagate through the network, affecting all subsequent computations, we calculate an exponentially weighted sum: $\sum_{l=0}^{L-1} w^l \min_S\{\text{Sim}(h^l, \bar{h}^l)\}$, where $\min_S$ operates across the sequence dimension, Sim computes cosine similarity across the hidden dimension, and $w = 0.5$ gives greater weight to

Table 3: Task description of each task in the LIBERO benchmark.

Task Suite	Task Description
LIBERO-Goal	open the middle layer of the drawer put the bowl on the stove put the wine bottle on the top of the drawer open the top layer of the drawer and put the bowl inside put the bowl on the top of the drawer push the plate to the front of the stove put the cream cheese on the bowl turn on the stove put the bowl on the plate put the wine bottle on the rack
LIBERO-Spatial	pick the akita black bowl between the plate and the ramekin and place it on the plate pick the akita black bowl next to the ramekin and place it on the plate pick the akita black bowl from table center and place it on the plate pick the akita black bowl on the cookies box and place it on the plate pick the akita black bowl in the top layer of the wooden cabinet and place it on the plate pick the akita black bowl on the ramekin and place it on the plate pick the akita black bowl next to the cookies box and place it on the plate pick the akita black bowl on the stove and place it on the plate pick the akita black bowl next to the plate and place it on the plate pick the akita black bowl on the wooden cabinet and place it on the plate
LIBERO-Object	pick the alphabet soup and place it in the basket pick the cream cheese and place it in the basket pick the salad dressing and place it in the basket pick the bbq sauce and place it in the basket pick the ketchup and place it in the basket pick the tomato sauce and place it in the basket pick the butter and place it in the basket pick the milk and place it in the basket pick the chocolate pudding and place it in the basket pick the orange juice and place it in the basket
LIBERO-Long	put both the alphabet soup and the tomato sauce in the basket put both the cream cheese box and the butter in the basket turn on the stove and put the moka pot on it put the black bowl in the bottom drawer of the cabinet and close it put the white mug on the left plate and put the yellow and white mug on the right plate pick up the book and place it in the back compartment of the caddy put the white mug on the plate and put the chocolate pudding to the right of the plate put both the alphabet soup and the cream cheese box in the basket put both moka pots on the stove put the yellow and white mug in the microwave and close it

earlier layers. Lower similarity values indicate stronger condition influence on network representations.

Norm scale change. Similarly, we quantify the magnitude of change induced by conditional inputs using an exponentially weighted sum of relative norm changes: $\sum_{l=0}^{L-1} w^l \max_S \left| \frac{\|\bar{h}^l\| - \|h^l\|}{\|h^l\|} \right|$, where $\max_S$ operates across the sequence dimension, $\|\cdot\|$ computes the norm across the hidden dimension, and $w = 0.5$. Larger values indicate that conditional information substantially alters the magnitude of hidden representations, suggesting stronger condition integration within the policy network.

Cosine similarity in case study. In Figure 8, we also calculate a cosine similarity metric to quantify the degree to which the policy network differentiates between conditions. This cosine similarity is computed between visual features *before* they are fused into the policy network (i.e., after the vision encoder's output has been processed by positional encoding and an MLP, yielding the features ultimately used for fusion with the policy network). We calculate this metric by randomly sampling image pairs from the dataset and then averaging the cosine similarity of their corresponding features.

Table 4: Task description of each task in the MetaWorld benchmark.

Task Name	**Task Description**
basketball	Dunk the basketball into the basket.
bin-picking	Grasp the puck from one bin and place it into another bin.
button-press	Press a button.
button-press-topdown	Press a button from the top.
button-press-topdown-wall	Bypass a wall and press a button from the top.
button-press-wall	Bypass a wall and press a button.
coffee-button	Push a button on the coffee machine.
coffee-pull	Pull a mug from a coffee machine.
coffee-push	Push a mug under a coffee machine.
dial-turn	Rotate a dial 180 degrees.
disassemble	Pick a nut out of the peg.
door-lock	Lock the door by rotating the lock clockwise.
door-open	Open a door with a revolving joint.
door-unlock	Unlock the door by rotating the lock counter-clockwise.
drawer-close	Push and close a drawer.
drawer-open	Open a drawer.
faucet-close	Rotate the faucet clockwise.
faucet-open	Rotate the faucet counter-clockwise.
hammer	Hammer a screw on the wall.
handle-press	Press a handle down.
handle-press-side	Press a handle down sideways.
handle-pull	Pull a handle up.
handle-pull-side	Pull a handle up sideways.
shelf-place	Pick and place a puck onto a shelf.
soccer	Kick a soccer into the goal.
stick-push	Grasp a stick and push a box using the stick.
sweep	Sweep a puck off the table.
sweep-into	Sweep a puck into a hole.
window-close	Push and close a window.
window-open	Push and open a window.

Table 5: **Task description of each task in the SO100 and xArm benchmark.** As each parameter combination introduces one task, each task suite has 10 tasks in total. For each task, we test the model for 10 trials.

Task Suite	Task Description	Parameter
SO100	pick `[A]` and place it on the `[B]` side of the table	`[A]: ["screwdriver", "sponge", "charger"], [B]: ["left", "right"]`
	move `[A]` to the center of the table	`[A]: ["cup", "bowl"]`
	pick the cloth and wipe the table	`None`
	unfold the cloth	`None`
xArm	pick `[A]` and place it on the `[B]` side of the table	`[A]: ["scissor", "plier", "tap"], [B]: ["left", "right"]`
	open the pot lid or put the lid on the pot	`[open, close]`
	pour the water from the kettle into the cup	`None`
	place the Moka pot on the cooker	`None`

D Policy Backbone Recipe

D.1 Condition Network

For vision encoding, we utilize the pre-trained DINOv2-Base model, and for language encoding, we employ the pre-trained T5-Base model. Both of these encoders are frozen during training. The outputs from these pre-trained modality encoders are processed by a linear projector and then augmented with positional encodings before being fused into the policy network. These positional encodings are trainable but are initialized with sin-cos positional encoding and scaled down by a factor of 0.2.

D.2 Policy Network

We employ a compact Robot Diffusion Transformer (RDT) architecture, which is fundamentally a Diffusion Transformer incorporating cross-attention layers. Diffusion timestamps and robot kinematic information are integrated into the policy network using AdaLN-Zero [21]. The vision and language embeddings are used as the Keys and Values in the cross-attention layers to be integrated into the policy network alternately [15]. The Transformer architecture has a hidden dimension of 384, with 6 attention heads, and 12 layers.

D.3 Diffusion Model

We use a flow matching model defined by Equation (10). Diffusion timestamps are treated as continuous values within the range $[0, 1]$; we do not discretize them. Instead, they are represented using a Fourier embedding with a scale of 0.2 [5]. During training, diffusion timestamps are sampled from a uniform distribution over the interval $[0, 1]$. For inference, we solve the corresponding ODE using the Euler method, dividing the interval $[0, 1]$ into equal-sized steps.

D.4 Computation Resources

All policy training and testing are conducted on a server equipped with 4 NVIDIA GeForce RTX 4090 GPUs and an Intel(R) Xeon(R) CPU E5-2678 v3 @ 2.50GHz.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We have discussed the limitations of the work in Section 6.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: We provide the full set of assumptions and a complete proof for each theoretical result.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide every detail to reproduce the main experimental results.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We will open-source the code in a few days after submission.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We provide every detail in training and testing.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Yes. We report error bars in the learning curve.

   Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Yes. We list the GPU and CPU resources.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Yes, we do.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Yes, we do.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: Yes, they are.

    Guidelines:

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

13. **New assets**

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

Answer: [Yes]

Justification: Yes.

Guidelines:

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

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

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

Answer: [NA]

Justification: We do not include crowdsourcing experiments or research with human subjects.

Guidelines:

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

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

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

Answer: [NA]

Justification: We do not include crowdsourcing experiments or research with human subjects.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [NA]

Justification: The core method development in this research does not involve LLMs.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.