VITA: VISION-TO-ACTION FLOW MATCHING POLICY

Anonymous authors
Paper under double-blind review

ABSTRACT

Conventional flow matching and diffusion-based policies sample through iterative denoising from standard noise distributions (e.g., Gaussian), and require conditioning mechanisms to incorporate visual information during the generative process, incurring substantial time and memory overhead. To reduce the complexity, we develop VITA (**VI**sion-**T**o-**A**ction policy), a noise-free and conditioning-free policy learning framework that directly maps visual representations to latent actions using flow matching. VITA treats latent visual representations as the source of the flow, thus eliminating the need of conditioning. As expected, bridging vision and action is challenging, because actions are lower-dimensional, less structured, and sparser than visual representations; moreover, flow matching requires the source and target to have the same dimensionality. To overcome this, we introduce an action autoencoder that maps raw actions into a structured latent space aligned with visual latents, trained jointly with flow matching. To further prevent latent space collapse, we propose flow latent decoding, which anchors the latent generation process by backpropagating the action reconstruction loss through the flow matching ODE (ordinary differential equations) solving steps. We evaluate VITA on 8 simulation and 2 real-world tasks from ALOHA and Robomimic. VITA outperforms or matches state-of-the-art generative policies, while achieving $1.5\times$-$2.3\times$ faster inference compared to conventional methods with conditioning.

1 INTRODUCTION

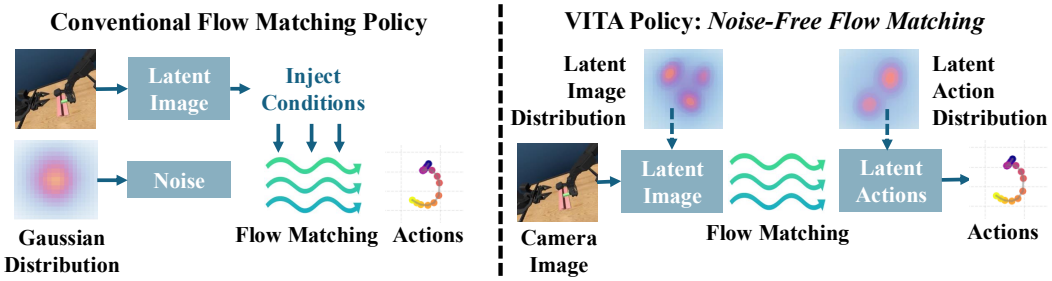

Figure 1: A comparison between VITA and conventional flow matching and diffusion policies. Unlike conventional methods that sample noise from standard distributions and inject source input modalities via conditioning during generation, VITA poses no constraints on the source distributions, and flow directly from latent visual representations to latent actions, eliminating the need for conditioning.

Flow matching and diffusion models have demonstrated remarkable success across a wide range of cross-modal generation tasks, from text-to-image generation Rombach et al. (2022); Peebles & Xie (2023); Ma et al. (2024); Liu et al. (2024a); He et al. (2025); Zhang et al. (2023), text-to-video generation Ho et al. (2022); Li et al. (2023); Jin et al. (2024), to visuomotor (vision-to-action) policy Chi et al. (2023); Ren et al. (2024a); Gao et al. (2025); Su et al. (2025); Zhang et al. (2025); Rouxel et al. (2024); Braun et al. (2024); Black et al.. Conventional flow matching and diffusion methods Lipman et al. (2023); Sohl-Dickstein et al. (2015) generate samples by starting with noise sampled from a basic source distribution (often Gaussian) and progressively "denoising" them into the target modality, requiring additional conditioning mechanisms Rombach et al. (2022); Zhang

et al. (2023); Chi et al. (2023); Dasari et al. (2024) to inject visual modality at denoising steps, which incurs increased time and space complexity Liu et al. (2024a); He et al. (2025).

Clearly, minimizing complexity is critical for effective real-time robot control. To overcome the inefficiencies inherent to conditioning mechanisms in conventional methods, we develop VITA (**VI**sion-**T**o-**A**ction flow matching), a noise-free policy learning framework that directly maps visual representations to latent actions. As depicted in Figure 1, unlike conventional methods that flow from a Gaussian prior and inject visual contexts via conditioning mechanisms during generation, VITA poses no constraints on the source distribution, and directly flows from latent visual representations, obviating the need for injecting visual inputs via conditioning modules. Consequently, VITA significantly reduces inference latency and memory overhead, simplifying the network architecture.

Learning vision-to-action flow matching, however, presents several new challenges. In general, bridging two distinct modalities is inherently difficult Liu et al. (2024a); He et al. (2025), particularly in robot learning where action data is relatively limited, unstructured and sparse, whereas the visual representations exhibit rich structures and semantics and have far higher dimensions. Additionally, flow matching requires that the source and target have equal dimensionalities, which prevents directly pairing raw actions with visual representations.

To address these challenges, we propose two key designs for VITA. 1) **Learned target latent actions for flow matching.** We introduce a structured latent action space, learned via an action autoencoder, that 'lifts' action representations to match the higher dimensionality of visual representations and serves as a structured target distribution for flow matching. The action encoder up-samples raw actions into target latent actions, and a decoder reconstructs raw actions from these latents. 2) **Flow latent decoding**. In conventional flow matching methods such as latent diffusion for image generation (Rombach et al., 2022), the target latent space can be pre-trained with abundant image data and then frozen as reliable flow matching targets; in contrast, we show that a pre-trained and frozen latent action space for learning vision-to-action flows yields poor performance, since action data can be too sparse and limited to learn reliable target latents and frozen latent spaces cannot be corrected. It is therefore plausible to jointly train the flow model with the action autoencoder. To prevent model collapse of targe latent action space (which happens by naively reducing the flow matching and autoencoder loss), we introduce flow latent decoding which backpropagates the reconstruction loss of latent actions generated by solving flow matching ordinary differential equations (ODE), anchoring the latent generation process using ground-truth actions.

We evaluate VITA on both real-world and simulated tasks using ALOHA Chuang et al. (2024); Fu et al. (2024b); Zhao et al. (2024) and Robomimic Mandlekar et al. (2021). Our experiments demonstrate that VITA outperforms or matches state-of-the-art generative policies. We implement VITA with only MLP layers. Compared to state-of-the-art flow matching and diffusion policies that predominantly leverage complex architecture such as transformers Ma et al. (2024) or U-Net Ronneberger et al. (2015), and introduce conditioning modules Zhao et al. (2023); Zhang et al. (2025), the MLP-only and noise-free VITA policy achieves, $1.5\times$–$2.3\times$ faster inference compared to conventional flow matching. To our knowledge, VITA is the first MLP-only flow matching policy to succeed on tasks as challenging as ALOHA bi-manual manipulation.

Our main contributions are summarized as follows:

Noise-Free Flow Matching for Visuomotor Learning. We propose VITA, a noise-free policy that directly evolves latent visual representations into latent actions via flow matching. To bridge the modality gap, VITA incorporates an action autoencoder that maps raw actions into structured latent spaces aligned with visual representations. To prevent latent space collapse, we propose flow latent decoding, which anchors latent generation using ground-truth actions by backpropagating the action reconstruction loss through the flow matching ODE (ordinary differential equations) solving steps.

An Efficient Policy Architecture. By treating vision as the source distribution, VITA obviates costly conditioning mechanisms required by conventional diffusion and flow matching policies. VITA also enables a lightweight MLP-only implementation that achieves superior performance on challenging bi-manual manipulation tasks with high-dimensional visual inputs.

State-of-the-Art Performance in Real and Simulated Environments. We validate VITA's effectiveness on 8 simulation and 2 real-world tasks. VITA surpasses or matches state-of-the-art generative

policies in success rates while delivering $1.5\times$–$2.3\times$ faster inference compared to conventional flow matching approaches.

2 Related Work

Imitation Learning for Visuomotor Policy. Imitation learning enables robots to learn complex behaviors by mimicking expert demonstrations. Behavior cloning is a prominent imitation learning paradigm that learns a policy that maps observations to actions (Zhao et al., 2023; Lee et al., 2024) via supervised learning. Recent advancements in behavioral cloning have widely adopted autoregressive modeling (Fu et al., 2024a; Gong et al., 2024; Su et al., 2025) and generative modeling (Chi et al., 2023). Generative modeling learns a conditional distribution of actions given an observation, leveraging conditional variational autoencoders (CVAEs) (Zhao et al., 2023; Lee et al., 2024), diffusion (Dasari et al., 2024; Chi et al., 2023), or flow matching (Zhang & Gienger, 2024; Zhang et al., 2025). Generative models ubiquitously require conditioning modules (e.g., cross-attention (Dasari et al., 2024), AdaLN (Dasari et al., 2024), FiLM (Perez et al., 2018; Chi et al., 2023)) to inject observations at each step of the generation process. VITA removes the visual conditioning module by developing a noise-free and conditioning-free vision-to-action flow.

Diffusion and Flow matching for Generative Modeling. Diffusion and flow matching models have shown strong performance across diverse generative tasks, such as image and video generation (Rombach et al., 2022; Ho et al., 2022), and visuomotor policies (Chi et al., 2023). Unlike diffusion which samples from a Gaussian distributions, flow matching theoretically places no constraints on the choice of source distribution (Tong et al., 2024). A few works have explored this property to learn the direct transport within the same modality (Albergo & Vanden-Eijnden, 2022; Tong et al., 2023b). Recently, Liu et al. (2024a) and He et al. (2025) extended this to more challenging cross-modal generation between text and image. VITA focuses on learning to bridge vision and action for visuomotor control, where the action modality presents unique challenges because of limited data and its unstructured nature. Different from flow matching for image generation, which typically pre-trains and freezes the latent image space when learning the flow (Rombach et al., 2022; He et al., 2025; Liu et al., 2024a), VITA resorts to a fully end-to-end pipeline training to effectively learn the latent action space from limited and sparse action data along with flow matching. We propose flow latent decoding to backpropagate action reconstruction losses through the the latent action generation process (ODE solving steps) during training.

3 Preliminaries

Flow matching models learn to transport samples from a source probability distribution p_0 to a target distribution p_1 by learning a velocity vector field v_θ (Lipman et al., 2023; Liu et al., 2022c). This generative process is defined by an ordinary differential equation (ODE), $\frac{d\boldsymbol{z}_t}{dt} = v_\theta(\boldsymbol{z}_t, t)$, where $t \in [0, 1]$ is a continuous time variable, $\boldsymbol{z}_t$ represents a sample at time t, and the model v_θ is a neural network. The goal of training is for v_θ to generate a path that transforms a sample $\boldsymbol{z}_0 \sim p_0$ into a corresponding sample $\boldsymbol{z}_1 \sim p_1$.

For a simple, straight probability path between two samples, the interpolated state at time t is defined as $\boldsymbol{z}_t = (1-t)\boldsymbol{z}_0 + t\boldsymbol{z}_1$ The corresponding ground-truth velocity of this path given by $u_t = \frac{d\boldsymbol{z}_t}{dt} = \boldsymbol{z}_1 - \boldsymbol{z}_0$. The network v_θ is trained by minimizing the Mean Squared Error (MSE) between its predicted velocity and the ground-truth velocity:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,\boldsymbol{z}_0,\boldsymbol{z}_1}\left[\|v_\theta(\boldsymbol{z}_t, t) - (\boldsymbol{z}_1 - \boldsymbol{z}_0)\|^2\right] \tag{1}$$

During inference, starting from a source sample $\boldsymbol{z}_0$, the model generates a target sample $\hat{\boldsymbol{z}}_1$ by solving the ODE initial value problem from $t = 0$ to $t = 1$:

$$\hat{\boldsymbol{z}}_1 = \boldsymbol{z}_0 + \int_0^1 v_\theta(\boldsymbol{z}_t, t)\, dt \tag{2}$$

Notably, conventional flow matching or diffusion policies for visuomotor control generate actions by evolving a sample from a simple prior distribution, such as Gaussian noise ($\boldsymbol{z}_0 \sim \mathcal{N}(0, I)$), and

require conditioning on visual inputs to guide the generation process (Zhang & Gienger, 2024; Chi et al., 2023; Dasari et al., 2024). However, flow matching does not assume the source distribution must be simple noise; it can theoretically connect two arbitrary distributions. We exploit this flexibility to re-formulate generative visuomotor policy learning: we directly use the latent distribution of visual observations as the source distribution p_0, yielding a noise-free flow matching process that eliminates the need for costly conditioning modules and therefore boosts time and space efficiency.

4 VITA: VISION-TO-ACTION FLOW MATCHING

The key challenge in VITA is the large dimensionality gap between vision and action, compounded by the sparsity and unstructured nature of action data. In this section, we present the core designs of VITA developed to address these issues. We first introduce the mathematical formulation of VITA (Section 4.1) and its overall architecture (Section 4.2). We then show why constructing a latent action space is essential for resolving dimensionality mismatch (Section 4.3), and propose flow latent decoding to address model collapse. Finally, we describe the objectives that enable effective end-to-end VITA learning from scratch (Section 4.4).

4.1 FLOWING FROM VISION TO ACTION

VITA learns a policy $\pi(A|O)$ that directly maps observations O to a corresponding sequence of future actions A. The observations O encompasses raw visual inputs $I \in \mathbb{R}^{H\times W\times C}$ and, optionally, the robot's proprioceptive state vector S. Actions are represented as temporal sequences over a prediction horizon, formally defined as $A \in \mathbb{R}^{T_{\text{pred}}\times D_{\text{action}}}$, where T_{pred} is the prediction horizon and D_{action} is the dimensionality of the action space. We employ action chunking with $T_{\text{pred}} > 1$ to enhance temporal consistency (Zhao et al., 2023).

Unlike conventional flow matching policies that generate actions through conditional denoising of random noise given visual inputs, VITA formulates the visuomotor policy as an unconditional flow between the source distribution of latent visual representations p_0 and the target distribution of latent actions p_1. The flow matching framework learns to evolve a latent visual representation $\boldsymbol{z}_0$ into an latent action representation $\boldsymbol{z}_1$ by learning a velocity field $v_\theta(\boldsymbol{z}_t, t)$, where $t \in [0, 1]$ represents the continuous time parameter interpolating between the source state at $t = 0$ and the target state at $t = 1$. Critically, flow matching requires $\boldsymbol{z}_0$ and $\boldsymbol{z}_1$ to share identical dimensionality, necessitating the construction of a latent action space that matches the dimensionality of visual representations (Section 4.3).

During inference, the current observation O_{curr} is first encoded into its latent visual representation $\boldsymbol{z}_0 = \mathcal{E}_v(O_{\text{curr}})$. This latent vector is subsequently evolved into a predicted action latent representation, $\hat{\boldsymbol{z}}_1$, by numerically solving the ODE from $t = 0$ to $t = 1$ using the learned velocity field v_θ: $\hat{\boldsymbol{z}}_1 = \boldsymbol{z}_0 + \int_0^1 v_\theta(\boldsymbol{z}_t, t)\, dt$. In other words, $\hat{\boldsymbol{z}}_1$ is obtained by numerically solving the flow matching ODE, and serves as an approximation to the target latent $\boldsymbol{z}_1$. The resulting latent action vector $\hat{\boldsymbol{z}}_1$ is then decoded through the action decoder to yield the final action sequence $\hat{A} = \mathcal{D}_a(\hat{\boldsymbol{z}}_1)$.

4.2 VITA ARCHITECTURE DESIGN

As depicted in Figure 2, VITA is composed of the three primary components: 1) The **Vision Encoder** ($\mathcal{E}_v$) maps raw camera images into a vector representation of visual latents I. The observation O to the policy consists of both image vector representations I and, optionally, the robot's proprioceptive states S. In short, O is mapped into a latent vector $\boldsymbol{z}_0 = \mathcal{E}_v(O), \quad$ where $\boldsymbol{z}_0 \in \mathbb{R}^{D_{\text{latent}}}$ is the source of flow matching. 2) The **Action Autoencoder (AE)** consists of the Action Encoder and the Action Decoder, and learns a compact representation for action chunks. The Action Encoder ($\mathcal{E}_a$) maps the ground-truth action chunk A to latent actions $\boldsymbol{z}_1 = \mathcal{E}_a(A), \quad$ where $\boldsymbol{z}_1 \in \mathbb{R}^{D_{\text{latent}}}$ serves as the target for flow matching; the Action Decoder ($\mathcal{D}_a$) reconstructs an action chunk $\hat{A} = \mathcal{D}_a(\hat{\boldsymbol{z}}_1)$ from latent actions $\hat{\boldsymbol{z}}_1$. 3) The **Flow Matching Network** (v_θ) is learned to predict the velocity field at arbitrary t.

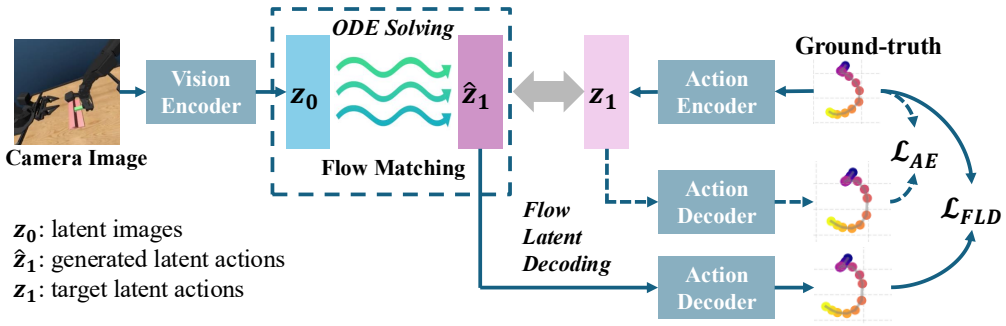

Figure 2: An overview of the VITA architecture: The vision encoder maps observations into a source latent representation z_0 for the flow; the action encoder provides a target latent representation z_1 for flow matching training. The action decoder learns to decode $\hat{z}_1$ (latent actions generated by solving ODEs) to actions via flow latent decoding losses, and decode z_1 to actions (latent actions from action encoder) via AE losses. The flow matching network is designed to learn the velocity field over a continuous flow matching path from z_0 to $\hat{z}_1$.

4.3 Bridging the Modality Gap between Vision and Action

A key constraint of flow matching is that the source and target distributions must share the same dimensionality. This poses a critical challenge for vision-to-action policies, since action spaces are typically much lower-dimensional than visual representations. For example, action dimensionalities range from 2 on `PushT` to 21 on `ThreadNeedle`, whereas visual representations can be 512-dimensional (Zhao et al., 2024) or even higher when using grid-based tokenizers (Chi et al., 2023).

To bridge the gap, one naive option is to down-sample latent visual representations to action chunk dimensionalities which, however, causes severe information loss and degrades performance. Alternatively, one can up-sample actions with zero-padding, yielding sparse, unstructured targets that hinder flow matching learning (see Appendix B.1). A third alternative is a pre-trained, frozen action AE, akin to common practice in latent diffusion for image generation (Rombach et al., 2022), but this proves ineffective for learning vision-to-action flow: with sparse, limited action data the induced latent space is unreliable as a flow target and cannot be corrected once frozen. As another alternative, jointly training the action AE with flow matching may still fail, and our empirical studies identify the root cause as latent space collapse induced by a training–inference gap in the latent actions used for decoding as detailed below.

Training–Inference Gap between Encoder-Based and ODE-Generated Latent Actions. During training, the decoder reconstructs actions from encoder-based latent actions z_1, whereas at inference it decodes $\hat{z}_1$ generated by solving the flow matching ODE. Since $\hat{z}_1$ is an approximation, and does not always align with z_1, the decoder can fail to map them into meaningful actions. To address this gap, we propose flow latent decoding, which enforces the model to decode from ODE-generated latent actions $\hat{z}_1$ during training, anchoring the latent generation process with ground-truth actions.

4.4 VITA Learning Objectives

Building upon our analysis of the training-inference gap, we now formulate a comprehensive learning framework for VITA that prevents latent collapse while ensuring effective end-to-end optimization. Our framework includes three essential objectives: flow latent decoding (FLD), flow matching (FM), and action autoencoder (AE) losses, each addressing distinct aspects of the learning challenge.

Flow Latent Decoding (FLD). Flow latent decoding addresses the training–inference gap by anchoring ODE-generated actions using ground-truth actions during training. Formally, FLD is defined as the reconstruction loss using ODE-generated latent actions, $\mathcal{L}_{\text{FLD}} = \left\| \mathcal{D}_a(\hat{z}_1) - A \right\|$, where $\hat{z}_1$ is obtained by solving the flow ODE with an Euler solver during training. By decoding $\hat{z}_1$ into actions and measuring reconstruction error directly in action space, FLD propagates gradients through the decoder and the ODE solver into both v_θ and $\mathcal{E}_v$, effectively updating the vision-to-flow-to-action pipeline.

Flow Latent Consistency (FLC). To gain deeper insight into the mechanics of FLD, we introduce flow latent consistency (FLC), a minimalist alternative that directly aligns ODE-generated and encoder-based latents. FLC directly aligns ODE-generated latents with encoder-based targets without decoding, i.e., $\mathcal{L}_{\text{FLC}} = \left\|\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1\right\|$. Under mild local regularity assumptions on $\mathcal{D}_a$ (stated below), FLC and FLD provide locally equivalent training signals for the same $\hat{\boldsymbol{z}}_1$. Empirically, FLC also prevents collapse, though convergence is slightly slower than with FLD (Section 5.3). While FLC operates purely in latent space without explicit action reconstruction, we establish that under mild regularity conditions, FLD and FLC provide locally equivalent optimization signals. This theoretical connection not only validates our approach but also offers computational flexibility in implementation. A sketch of the analysis is given below, with full proofs and corollaries deferred to Appendix A.

Assumption 1 (Decoder locally well-behaved)**.** *$\mathcal{D}_a$ is C^1 in a neighborhood of $\boldsymbol{z}_1$, with Jacobian singular values bounded as $m \le \sigma_{\min} \le \sigma_{\max} \le L$ on that neighborhood. Let $\varepsilon_{\text{AE}} := \|\mathcal{D}_a(\boldsymbol{z}_1) - A\|$ denote the local AE reconstruction error.*

Theorem 1 (Local equivalence of FLD and FLC)**.** *Under Assumption 1, for any $\hat{\boldsymbol{z}}_1$ in the neighborhood,*

$$m\,\|\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1\| - \varepsilon_{\text{AE}} \;\le\; \|\mathcal{D}_a(\hat{\boldsymbol{z}}_1) - A\| \;\le\; L\,\|\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1\| + \varepsilon_{\text{AE}}.$$

If $\varepsilon_{\text{AE}} = 0$, the minimizers of $\mathcal{L}_{\text{FLD}}$ and $\mathcal{L}_{\text{FLC}}$ coincide and equal $\{\boldsymbol{z}_1\}$; if $\varepsilon_{\text{AE}} > 0$, any minimizer of $\mathcal{L}_{\text{FLD}}$ lies within radius $\varepsilon_{\text{AE}}/m$ of $\boldsymbol{z}_1$.

This theoretical result confirms that FLD and FLC target the same underlying optimization objective.

We provide further discussion in Section 5.3, showing that including the flow latent decoding loss with a non-zero λ_{FLD} and backpropagating through the ODE solving steps is critical for training successful VITA policies. We also ablate the effects of λ_{FLD} and λ_{AE} in Figure 6.

Flow Matching and Autoencoder Losses. The flow matching loss supervises the flow network v_θ by minimizing the MSE between the predicted velocity and the ground-truth velocity as shown in Equation (1). The action autoencoder loss trains $(\mathcal{E}_a, \mathcal{D}_a)$ to reconstruct action chunks using an L1 loss, $\mathcal{L}_{\text{AE}} = \|A - \mathcal{D}_a(\mathcal{E}_a(A))\|_1$, where $\boldsymbol{z}_1 = \mathcal{E}_a(A)$ serves as a structured target latent with small reconstruction bias and good local conditioning. This structured latent space strengthens the theoretical link between FLD and FLC and further stabilizes training.

The training objective can be written as a weighted sum of all three losses: $\mathcal{L}_{\text{VITA}} = \lambda_{\text{FM}}\mathcal{L}_{\text{FM}} + \lambda_{\text{FLD}}\mathcal{L}_{\text{FLD}} + \lambda_{\text{AE}}\mathcal{L}_{\text{AE}}$.

5 Experiments

We evaluate VITA on ALOHA (Chuang et al., 2024; Fu et al., 2024b) across 5 simulation tasks (Figure 3) and 2 real-world tasks (Figure 4). The simulation environments and datasets are from AV-ALOHA (Chuang et al., 2024), which extend ALOHA with an additional 7-DoF arm equipped with an active-vision camera that dynamically adjusts its viewpoint. This setup increases task difficulty due to the larger action space and non-stationary observation dynamics. In addition, we evaluate on 2 single-arm Robomimic simulation tasks.

For simulation tasks, each environment provides 100–200 demonstrations. Demonstrations were collected from human experts in simulation using VR headsets, with the left-eye image used as input. VITA was trained at $25/3$ FPS on AV-ALOHA and 20 FPS on Robomimic. For real-world tasks on AV-ALOHA, (Figure 4). Each task is trained with 50 demonstrations, using the left image from the stereo active-vision camera as input. VITA was trained at 11 FPS, and linearly interpolated to 33 FPS for deployment on the physical robot.

5.1 Implementation

Flow Matcher. We adopt OT-CFM (Tong et al., 2023a), a conditional flow matching method based on optimal transport, and solve the Euler ODE with 6 steps using linearly interpolated t for both VITA and FM baselines.

Networks. We use the widely adopted ResNet-18 (He et al., 2016; Su et al., 2025; Chi et al., 2023) as the vision encoder for all the experiments. We use the 512-dimensional feature vector after the average

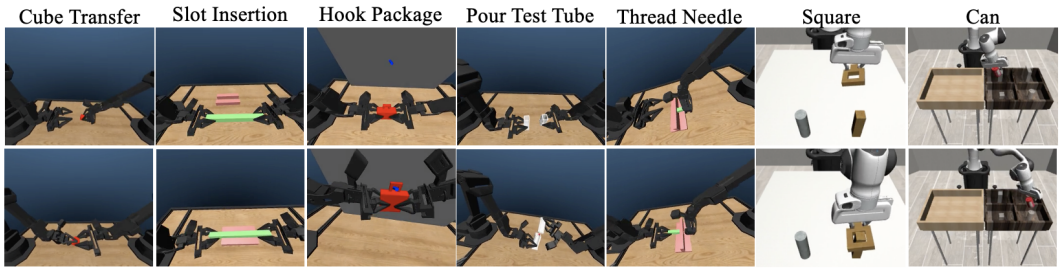

Figure 3: An illustration of 5 AV-ALOHA tasks (`CubeTransfer`, `SlotInsertion`, `HookPackage`, `PourTestTube`, `ThreadNeedle`), and 2 Robomimic tasks (`Square`, `Can`).

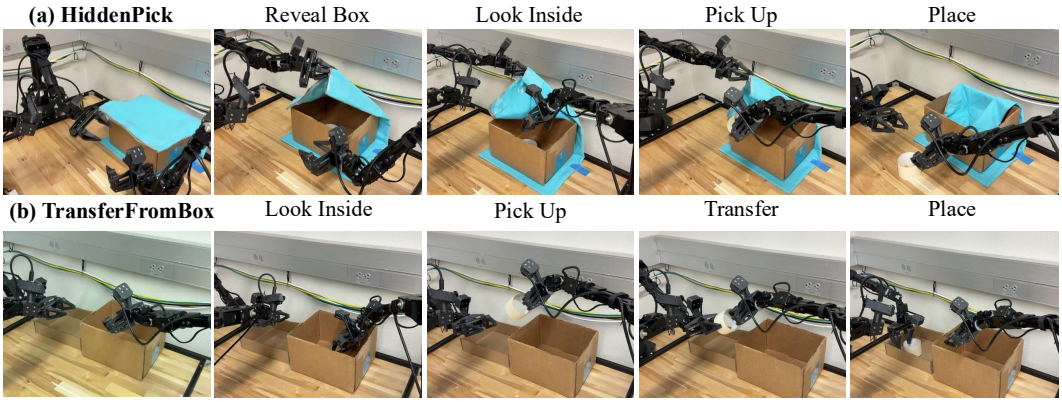

Figure 4: An illustration of two challenging real-world AV-ALOHA tasks, `HiddenPick`, and `TransferFromBox`. The pictures are taken from autonomous rollouts by the VITA policy.

pooling layer as the visual representations for VITA. Therefore, the source $\boldsymbol{z}_0$ and target latents $\boldsymbol{z}_1$ are both 512-dimensional vectors ($D_{\text{latent}} = 512$). The vector representations and conditioning-free design is amenable to simple and efficient implementation. For example, we use 4-layer MLP-only networks for both flow matching and decoding.

Training & Evaluation. We train VITA on each task for 50,000 steps with a batch size of 128. The policy predicts action chunks of length 16, of which the first 8 actions are executed. In simulation, we evaluate the policy every 500 steps, with each evaluation consisting of 50 episodes. We report the best success rates (SRs), averaged over 3 random seeds, for both VITA and baselines (Table 1). For real-world tasks, we evaluate 3 checkpoints over 20 episodes each and report the best SRs (Table 2).

Baselines. We compare VITA against state-of-the-art generative policies, including flow matching (FM) policy (Zhang & Gienger, 2024), diffusion policy (DP)(Chi et al., 2023), and action chunking transformer (ACT)(Zhao et al., 2023). We follow the ACT and DP implementation in LeRobot Cadene et al. (2024). We found DP requires significantly more training steps than FM; ACT yields sub-optimal performance on most tasks; therefore, we increase training steps to 200,000 for both DP and ACT, and then report the best success rates. For DP, each inference takes 10 DDPM (Denoising Diffusion Probabilistic Model) steps (Chi et al., 2023). We evaluate both the task performance and inference efficiency of VITA relative to these baselines in Section 5.2.

5.2 Performance

Success Rates. We evaluate VITA against state-of-the-art conditional generative policies on 8 simulated tasks: 5 bimanual AV-ALOHA tasks with active vision (Chuang et al., 2024), 2 single-arm Robomimic tasks (Mandlekar et al., 2021), and 1 PushT task. Results are reported in Table 1. Across all simulation tasks, VITA consistently outperforms or matches state-of-the-art baselines. ACT shows limited performance even after convergence, while DP requires significantly more training steps and remains sub-optimal despite extended training. We further assess VITA in real-world settings on two challenging AV-ALOHA tasks, with performance summarized in Table 2 and examples of

autonomous rollouts in Figure 4, highlighting the precision of vision-to-action flow in manipulation tasks with high-dimensional visual inputs and large action spaces.

Table 1: Task success rates (SR) across simulation tasks including ALOHA, Robomimic, and `PushT`. We report the best SR during validation, and compute the mean across 3 random seeds.

Task	VITA	FM	DP	ACT
`ThreadNeedle`	**0.92**	0.88	0.00	0.52
`SlotInsertion`	0.76	**0.83**	0.20	0.50
`PourTestTube`	0.79	**0.84**	0.06	0.14
`HookPackage`	**0.86**	0.83	0.12	0.22
`CubeTransfer`	**1.00**	**1.00**	0.76	0.98
`PushT`	**0.88**	0.84	0.80	0.30
`Square`	**1.00**	**1.00**	**1.00**	0.46
`Can`	**1.00**	**1.00**	**1.00**	0.78

Table 2: Success rates of VITA on two real-world AV-ALOHA tasks. Each task has 3 subtasks.

	`HiddenPick`			`TransferFromBox`		
Subtask	Reveal	Pick	Place	Pick	Transfer	Place
SR	1.00	0.65	0.65	1.00	0.95	0.90

Efficiency. The VITA policy is highly efficient due to its architectural simplicity. The core components used during inference, the flow matching network and the action decoder, are both lightweight MLP-only networks. We use vector representations for both vision and action, further reducing time and space overhead. VITA contains 31M parameters, including the ResNet-18. We benchmark the inference latency and policy throughput of VITA and compare it against conventional flow matching policy baselines. We implement two different flow matching parameterizations, including diffusion transformer (Peebles & Xie, 2023), and U-Net (Ronneberger et al., 2015), and three different conditioning mechanisms, including AdaLN (Peebles & Xie, 2023; Dasari et al., 2024), cross-attention (Vaswani et al., 2017; Dasari et al., 2024; Gong et al., 2024), and FiLM (Perez et al., 2018; Chi et al., 2023), as the baselines. FM and DP are typically parameterized using U-Nets (Ronneberger et al., 2015) or diffusion transformers (DiTs) (Peebles & Xie, 2023), and are trained to predict velocity fields or noise at each denoising step. While U-Nets and DiTs are highly expressive, they can be large or computationally expensive, posing a drawback for real-world robotic deployments that require low-latency inference. Most importantly, these methods inevitably incorporate visual conditioning mechanisms, such as cross, further increasing both inference time and memory footprint.

With a control frequency of $25/3 \approx 8.33$ Hz, our MLP-only VITA policy achieves an inference wall-clock time of 0.22 ms on an NVIDIA 4090 GPU. In contrast, a comparable transformer-based flow matching policy incurs higher latency (0.33–0.51 ms) and requires more parameters. A detailed breakdown of efficiency metrics is given in Table 3.

Table 3: Model sizes and inference latency of different policy implementations.

Model	Conditioning	Architecture	Num of Params	Latency (ms/chunk)
VITA	***N/A***	**MLP**	31.09M	**0.2215**
FM	AdaLN	DiT	31.16M	0.3307
FM	Cross-Attn	DiT	29.06M	0.5102
FM	FiLM	U-Net	84.05M	0.3650
DDPM	FiLM	U-Net	81.82M	2.5985

5.3 Ablation of Flow Latent Decoding

We investigate the importance of flow latent decoding ($\mathcal{L}_{\text{FLD}}$) for effectively training VITA policies. During training, the decoder is trained by reconstructing from encoder-based latent actions $\boldsymbol{z}_1$, while during inference, the decoder needs to reconstruct from ODE-generated latent actions $\hat{\boldsymbol{z}}_1$. As a result, the generated latent actions at inference may not decode into meaningful actions, as demonstrated by Figure 5. To bridge this gap, we propose two objectives to minimize the discrepancy between encoder-based latents $\boldsymbol{z}_1$ and ODE-generated latents $\hat{\boldsymbol{z}}_1$: (1) FLD, which decodes $\hat{\boldsymbol{z}}_1$ into the action space and computes the reconstruction error against the ground-truth action; and (2) FLC, which directly aligns the predicted latent action with the encoded latent target $\boldsymbol{z}_1 = \mathcal{E}_a(A)$ using $\|\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1\|$.

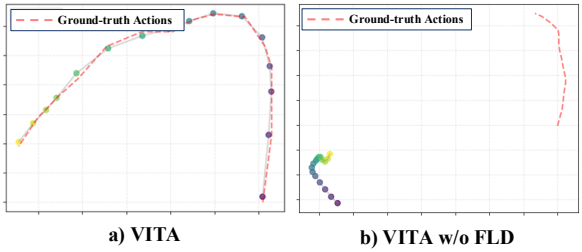

Figure 5: Comparison of reconstructed actions between (a) VITA, and (b) VITA without FLD. Reconstruction fails without flow latent decoding.

As shown in Figure 6, without flow latent decoding (i.e., $\lambda_{\text{FLD}} = 0$), the model completely fails to learn a reasonable policy. In contrast, applying FLD succeeds in learning the policy. FLC provides a weaker signal compared to FLD (because FLD directly anchors using the ground-truth actions), resulting in slightly slower convergence. However, we found that using a combination of both objectives yields the best performance due to richer learning signals on both action space and latent space. In short, FLD and FLC are crucial for learning VITA, and in practice a combination of both can achieve optimal results.

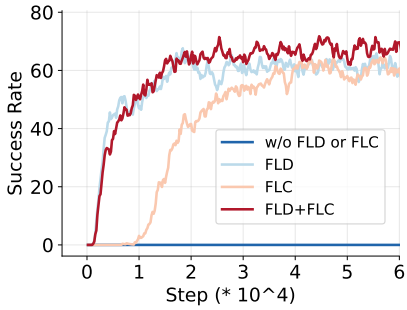

Figure 6: Comparison of success rates using different objectives. Curves are smoothed using exponential moving average for clarity.

6 Conclusion

In this work, we developed VITA, an efficient and high-performing visuomotor policy. VITA generates actions in a noise-free manner and directly evolves latent visual representations into latent actions. By leveraging latent image representations as the flow's source distribution, VITA eliminates the need for complex conditioning mechanisms such as cross-attention, significantly simplifying model architecture and enhancing efficiency. VITA employs vector representations for both latent images and actions, thereby enabling a simple MLP-only architecture for vision-to-action flow. Two key designs are critical to VITA's success — structuring the latent action space using an action autoencoder and enabling backpropagation of flow latent decoding losses across ODE solving steps — allowing VITA to effectively learn latent action spaces with flow matching in a fully end-to-end manner. Extensive experiments demonstrate that VITA achieves state-of-the-art performance on ALOHA and Robomimic in both real and simulation tasks. VITA achieves $1.5\times$-$2.3\times$ faster inference compared to conventional flow matching using different conditioning mechanisms.

7 Ethics Statement

All experiments were conducted in simulation environments or on standard robotic platforms without involving human subjects, sensitive user data, or any form of personal information. Thus, there are no privacy, security, or human participant concerns. The datasets we use are publicly available benchmark datasets, and no proprietary or restricted data were employed. No conflicts of interest or external sponsorships influence the reported findings.

8 Reproducibility Statement

We take multiple steps to ensure reproducibility of our results. A detailed description of our model architecture, training objectives, and algorithmic choices is provided in the main text. Hyper-

parameters, training configurations, and ablations are reported in the Appendix. For theoretical derivations (e.g., flow matching formulation), complete proofs and assumptions are included in the supplementary materials. To facilitate replication, we include anonymous source code with training scripts, evaluation pipelines, and configuration files as part of the supplementary material during review. All datasets used are publicly available (Robomimic, ALOHA).

REFERENCES

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. arXiv preprint arXiv:2209.15571, 2022.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. $\pi 0$: A vision-language-action flow model for general robot control, 2024. URL https://arxiv. org/abs/2410.24164.

Max Braun, Noémie Jaquier, Leonel Rozo, and Tamim Asfour. Riemannian flow matching policy for robot motion learning. In 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5144–5151. IEEE, 2024.

Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, Martino Russi, Francesco Capuano, Caroline Pascal, Jade Choghari, Jess Moss, and Thomas Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. `https://github.com/huggingface/lerobot`, 2024.

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, pp. 02783649241273668, 2023.

Ian Chuang, Andrew Lee, Dechen Gao, M Naddaf-Sh, Iman Soltani, et al. Active vision might be all you need: Exploring active vision in bimanual robotic manipulation. arXiv preprint arXiv:2409.17435, 2024.

Sudeep Dasari, Oier Mees, Sebastian Zhao, Mohan Kumar Srirama, and Sergey Levine. The ingredients for robotic diffusion transformers. arXiv preprint arXiv:2410.10088, 2024.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis, 2024. URL https://arxiv. org/abs/2403.03206, 2.

Johannes S Fischer, Ming Gui, Pingchuan Ma, Nick Stracke, Stefan A Baumann, and Björn Ommer. Boosting latent diffusion with flow matching. arXiv preprint arXiv:2312.07360, 2023.

Letian Fu, Huang Huang, Gaurav Datta, Lawrence Yunliang Chen, William Chung-Ho Panitch, Fangchen Liu, Hui Li, and Ken Goldberg. In-context imitation learning via next-token prediction. arXiv preprint arXiv:2408.15980, 2024a.

Zipeng Fu, Tony Z Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation using low-cost whole-body teleoperation. In 8th Annual Conference on Robot Learning, 2024b.

Dechen Gao, Hang Wang, Hanchu Zhou, Nejib Ammar, Shatadal Mishra, Ahmadreza Moradipari, Iman Soltani, and Junshan Zhang. In-ril: Interleaved reinforcement and imitation learning for policy fine-tuning. arXiv preprint arXiv:2505.10442, 2025.

Zhefei Gong, Pengxiang Ding, Shangke Lyu, Siteng Huang, Mingyang Sun, Wei Zhao, Zhaoxin Fan, and Donglin Wang. Carp: Visuomotor policy learning via coarse-to-fine autoregressive prediction. arXiv preprint arXiv:2412.06782, 2024.

Ju He, Qihang Yu, Qihao Liu, and Liang-Chieh Chen. Flowtok: Flowing seamlessly across text and image tokens. arXiv preprint arXiv:2503.10772, 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.

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

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. Advances in Neural Information Processing Systems, 35:8633–8646, 2022.

Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. arXiv preprint arXiv:2410.05954, 2024.

Andrew Lee, Ian Chuang, Ling-Yuan Chen, and Iman Soltani. Interact: Inter-dependency aware action chunking with hierarchical attention transformers for bimanual manipulation. arXiv preprint arXiv:2409.07914, 2024.

Xin Li, Wenqing Chu, Ye Wu, Weihang Yuan, Fanglong Liu, Qi Zhang, Fu Li, Haocheng Feng, Errui Ding, and Jingdong Wang. Videogen: A reference-guided latent diffusion approach for high definition text-to-video generation. arXiv preprint arXiv:2309.00398, 2023.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

Qihao Liu, Xi Yin, Alan Yuille, Andrew Brown, and Mannat Singh. Flowing from words to pixels: A framework for cross-modality evolution. arXiv preprint arXiv:2412.15213, 2024a.

Qihao Liu, Zhanpeng Zeng, Ju He, Qihang Yu, Xiaohui Shen, and Liang-Chieh Chen. Alleviating distortion in image generation via multi-resolution diffusion models and time-dependent layer normalization. Advances in Neural Information Processing Systems, 37:133879–133907, 2024b.

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. arXiv preprint arXiv:2410.07864, 2024c.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. arXiv preprint arXiv:2209.03003, 2022a.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022b. URL https://arxiv.org/abs/2209.03003.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. arXiv preprint arXiv:2209.03003, 2022c.

Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In European Conference on Computer Vision, pp. 23–40. Springer, 2024.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In Conference on Robot Learning (CoRL), 2021.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In Proceedings of the IEEE/CVF international conference on computer vision, pp. 4195–4205, 2023.

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.

Allen Z Ren, Justin Lidard, Lars L Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. arXiv preprint arXiv:2409.00588, 2024a.

Sucheng Ren, Qihang Yu, Ju He, Xiaohui Shen, Alan Yuille, and Liang-Chieh Chen. Flowar: Scale-wise autoregressive image generation meets flow matching. arXiv preprint arXiv:2412.15205, 2024b.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 10684–10695, 2022.

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

Quentin Rouxel, Andrea Ferrari, Serena Ivaldi, and Jean-Baptiste Mouret. Flow matching imitation learning for multi-support manipulation. In 2024 IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids), pp. 528–535. IEEE, 2024.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In International conference on machine learning, pp. 2256–2265. pmlr, 2015.

Yue Su, Xinyu Zhan, Hongjie Fang, Han Xue, Hao-Shu Fang, Yong-Lu Li, Cewu Lu, and Lixin Yang. Dense policy: Bidirectional autoregressive learning of actions. arXiv preprint arXiv:2503.13217, 2025.

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. arXiv preprint arXiv:2302.00482, 2023a.

Alexander Tong, Nikolay Malkin, Kilian Fatras, Lazar Atanackovic, Yanlei Zhang, Guillaume Huguet, Guy Wolf, and Yoshua Bengio. Simulation-free schr\" odinger bridges via score and flow matching. arXiv preprint arXiv:2307.03672, 2023b.

Alexander Tong, Nikolay Malkin, Kilian Fatras, Lazar Atanackovic, Yanlei Zhang, Guillaume Huguet, Guy Wolf, and Yoshua Bengio. Simulation-free schrödinger bridges via score and flow matching, 2024. URL https://arxiv.org/abs/2307.03672.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

Fan Zhang and Michael Gienger. Affordance-based robot manipulation with flow matching. arXiv preprint arXiv:2409.01083, 2024.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In Proceedings of the IEEE/CVF international conference on computer vision, pp. 3836–3847, 2023.

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 Proceedings of the AAAI Conference on Artificial Intelligence, volume 39, pp. 14754–14762, 2025.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. arXiv preprint arXiv:2304.13705, 2023.

Tony Z Zhao, Jonathan Tompson, Danny Driess, Pete Florence, Kamyar Ghasemipour, Chelsea Finn, and Ayzaan Wahid. Aloha unleashed: A simple recipe for robot dexterity. arXiv preprint arXiv:2410.13126, 2024.

A Proof of FLD and FLC Equivalence

Preliminaries. All norms below are vector norms with induced operator norms. We use the ball $B(\boldsymbol{z}_1, r) = \{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{z}_1\| < r\}$. Assumption 1 in the main text holds throughout.

Lemma 1 (Local bi-Lipschitzness from Jacobian bounds)**.** *For any $\hat{\boldsymbol{z}}_1 \in B(\boldsymbol{z}_1, r)$,*

$$m\,\|\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1\| \;\le\; \|\mathcal{D}_a(\hat{\boldsymbol{z}}_1) - \mathcal{D}_a(\boldsymbol{z}_1)\| \;\le\; L\,\|\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1\|.$$

Proof of Lemma 1. Let $\gamma(s) = \boldsymbol{z}_1 + s(\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1)$ for $s \in [0, 1]$. The mean-value integral formula gives

$$\mathcal{D}_a(\hat{\boldsymbol{z}}_1) - \mathcal{D}_a(\boldsymbol{z}_1) = \int_0^1 J_{\mathcal{D}_a}(\gamma(s))\,(\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1)\,ds.$$

Taking norms and using, for any matrix J and vector $v \neq 0$, $\sigma_{\min}(J)\|v\| \le \|Jv\| \le \sigma_{\max}(J)\|v\|$, together with $m \le \sigma_{\min}(J_{\mathcal{D}_a}(\gamma(s)))$ and $\sigma_{\max}(J_{\mathcal{D}_a}(\gamma(s))) \le L$ for all s, yields the bounds.

Proof of Theorem 1. Add and subtract $\mathcal{D}_a(\boldsymbol{z}_1)$ and apply triangle/reverse–triangle inequalities:

$$\|\mathcal{D}_a(\hat{\boldsymbol{z}}_1) - A\| \le \|\mathcal{D}_a(\hat{\boldsymbol{z}}_1) - \mathcal{D}_a(\boldsymbol{z}_1)\| + \|\mathcal{D}_a(\boldsymbol{z}_1) - A\|,$$

$$\|\mathcal{D}_a(\hat{\boldsymbol{z}}_1) - A\| \ge \|\mathcal{D}_a(\hat{\boldsymbol{z}}_1) - \mathcal{D}_a(\boldsymbol{z}_1)\| - \|\mathcal{D}_a(\boldsymbol{z}_1) - A\|.$$

Invoke Lemma 1 and set $\varepsilon_{\mathrm{AE}} = \|\mathcal{D}_a(\boldsymbol{z}_1) - A\|$ to obtain the two-sided inequality stated in Theorem 1. The minimizer claims follow immediately: if $\varepsilon_{\mathrm{AE}} = 0$, both losses are minimized at $\hat{\boldsymbol{z}}_1 = \boldsymbol{z}_1$; otherwise any minimizer of FLD must satisfy $\|\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1\| \le \varepsilon_{\mathrm{AE}}/m$.

Corollary A.1 (squared-loss version). Assume $\varepsilon_{\mathrm{AE}} = 0$. Then

$$m^2\,\|\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1\|^2 \;\le\; \|\mathcal{D}_a(\hat{\boldsymbol{z}}_1) - A\|^2 \;\le\; L^2\,\|\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1\|^2.$$

With $\varepsilon_{\mathrm{AE}} = 0$, Lemma 1 gives $m\|\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1\| \le \|\mathcal{D}_a(\hat{\boldsymbol{z}}_1) - \mathcal{D}_a(\boldsymbol{z}_1)\| \le L\|\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1\|$. Since both sides are nonnegative, squaring preserves the inequalities. Thus the squared FLD objective is sandwiched between m^2 and L^2 times the squared FLC objective. Consequently, the map $\hat{\boldsymbol{z}}_1 \mapsto \|\mathcal{D}_a(\hat{\boldsymbol{z}}_1) - A\|_2^2$ is locally L^2-smooth and m^2-strongly convex along latent directions (intuitively, its curvature is controlled by $J_{\mathcal{D}_a}^\top J_{\mathcal{D}_a}$ whose eigenvalues lie in $[m^2, L^2]$).

Corollary A.2 (gradient scaling for squared losses). Let $J := J_{\mathcal{D}_a}(\hat{\boldsymbol{z}}_1)$ and assume $\varepsilon_{\mathrm{AE}} = 0$. For the squared losses,

$$\nabla_{\hat{\boldsymbol{z}}_1}\mathcal{L}_{\mathrm{FLD}}^{(2)} = 2\,J^\top\big(\mathcal{D}_a(\hat{\boldsymbol{z}}_1) - A\big), \qquad \nabla_{\hat{\boldsymbol{z}}_1}\mathcal{L}_{\mathrm{FLC}}^{(2)} = 2\,(\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1).$$

Then

$$m^2\,\big\|\nabla\mathcal{L}_{\mathrm{FLC}}^{(2)}\big\| \;\le\; \big\|\nabla\mathcal{L}_{\mathrm{FLD}}^{(2)}\big\| \;\le\; L^2\,\big\|\nabla\mathcal{L}_{\mathrm{FLC}}^{(2)}\big\|.$$

Use $\|J^\top y\| \in [m\|y\|,\, L\|y\|]$ (by singular-value bounds) with $y = \mathcal{D}_a(\hat{\boldsymbol{z}}_1) - A = \mathcal{D}_a(\hat{\boldsymbol{z}}_1) - \mathcal{D}_a(\boldsymbol{z}_1)$, and Lemma 1 to get $m\,\|y\| \le \|J^\top y\| \le L\,\|y\|$ and $m\,\|\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1\| \le \|y\| \le L\,\|\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1\|$. Multiplying the bounds yields $m^2\|\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1\| \le \|J^\top y\| \le L^2\|\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1\|$. Since $\|\nabla\mathcal{L}_{\mathrm{FLC}}^{(2)}\| = 2\|\hat{\boldsymbol{z}}_1 - \boldsymbol{z}_1\|$, this gives the stated inequality (up to the common factor 2). It follows that step-size sensitivity is governed by the squared condition number $(L/m)^2$.

B Extended Ablations

B.1 Dimensionality Matching for Vision-to-Action Flow

A constraint of flow matching is that the source and target must have identical shapes. In visuomotor contexts, the visual latent representations ($\boldsymbol{z}_0$) are typically much higher-dimensional than raw action chunks (A). A naive solution would be to down-sample the visual representation to match the action dimensionality, which, nevertheless, leads to significant information loss and poor performance, particularly when the dimensional gap is large.

Therefore, we adopt the opposite strategy: we map the raw action chunks into a higher-dimensional latent space that matches the dimensionality of the visual latent representations.

A naive approach is to use fixed linear transformations. When the action dimension is smaller than the latent dimension, we construct a lossless mapping by embedding the actions into the higher-dimensional latent space through zero-padding. The inverse mapping simply discards the padding. Our experiments show that action representations produced by such transformations are insufficient for learning reasonable flow matching policies. We found that learning well-structured action latent spaces via autoencoders as the target distributions for flow matching can be crucial for the success of vision-to-action flow. We develop an action encoder ($\mathcal{E}_a$), which does not simply remap dimensions, but also learns structured latent action spaces, making the complex flow from vision to action more tractable.

Table 4: Task SR (%) on `ThreadNeedle` with different action up-sampling strategies.

Up-sampling Strategy	SR (%)
Zero-Padding	0
Action AE (w/o FLD)	0
Action AE (w/ FLD)	**92**

Autoencoder Loss. We found L1 loss outperforms L2 because L2 tends to average points.

Sampling Stochasticity. We examined VITA variants with different sources of sampling stochasticity. We found that introducing dropout or variance (σ) in flow matching—where σ injects Gaussian noise into the interpolation path between source and target latents as in OT-CFM—degrades performance. In contrast, adding covariance to the source distribution yields results comparable to the deterministic baseline. Similarly, using a variational objective in the action autoencoder performs on par with the standard design, whereas applying a variational formulation to the image encoder significantly hurts performance.

C Experiment Results

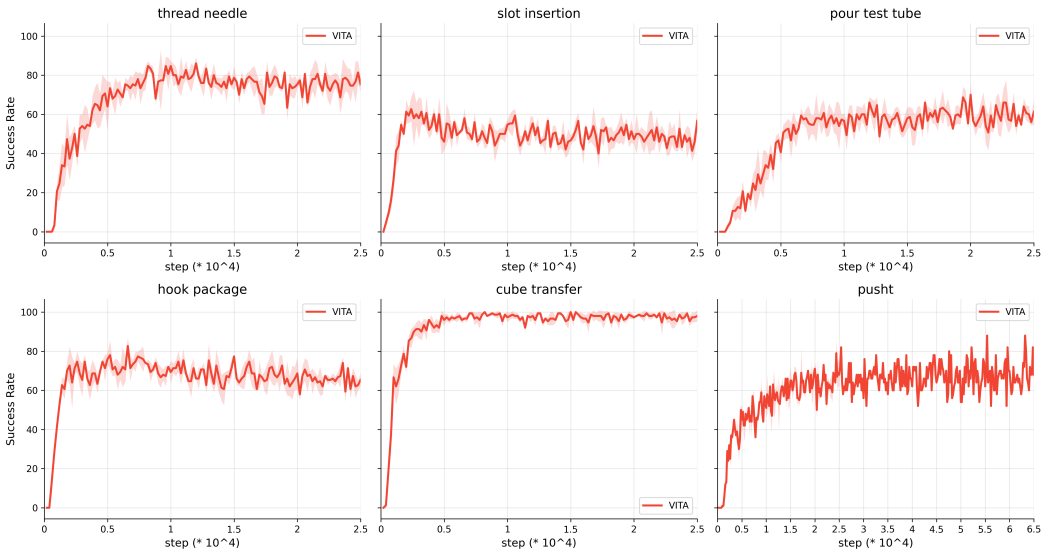

Figure 7: **Training curves on six tasks (five AV-ALOHA + `PushT`).** Each plot reports success rate (%) versus training step ($\times 10^4$) for VITA. The solid line is the mean across three random seeds; the shaded region is ± 1 standard deviation.

As shown in Fig. 7, VITA exhibits fast early learning and stable plateaus on AV-ALOHA and `PushT`; Fig. 8 shows consistent improvements on RoboMimic `Can`/`Square` under the same protocol.

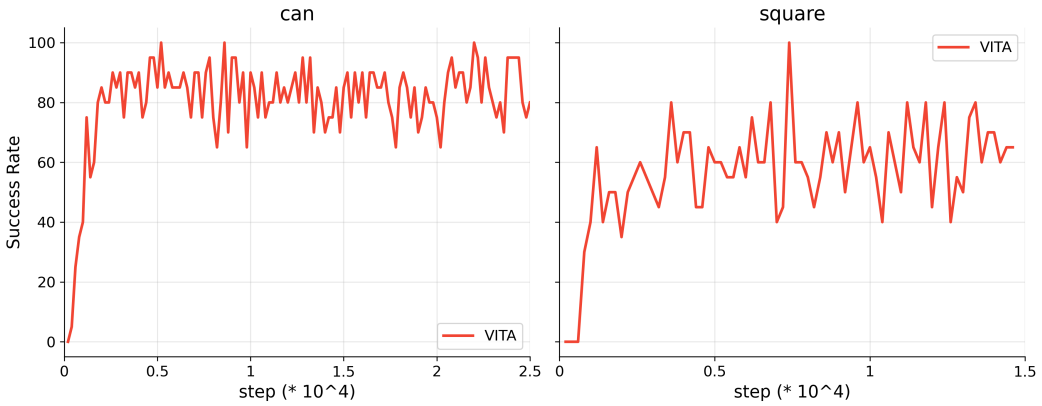

Figure 8: **Training curves on two RoboMimic tasks.** Success rate (%) versus training step ($\times 10^4$) on `Can` and `Square`.

D Extended Related Works

Imitation Learning for Visuomotor Policy. Imitation learning enables robots to learn complex behaviors by mimicking expert demonstrations. Behavioral cloning, a prominent imitation learning paradigm, frames this as a supervised learning problem, learning a policy that maps observations to actions (Zhao et al., 2023; Lee et al., 2024). Recent advancements in behavioral cloning have widely adopted two modeling methods: generative modeling and autoregressive modeling. Generative methods learn a conditional distribution of actions given an observation. This category includes policies based on conditional variational autoencoders (CVAEs) (Zhao et al., 2023; Lee et al., 2024), as well as diffusion (Dasari et al., 2024; Chi et al., 2023) and flow matching (Zhang & Gienger, 2024; Zhang et al., 2025). On the other hand, autoregressive methods tokenize actions and frame policy learning as a sequence modeling task. These methods predict action tokens sequentially, using next-token prediction (Fu et al., 2024a), next-scale prediction (Gong et al., 2024), or bi-directional prediction (Su et al., 2025). Generative models ubiquitously require extra conditioning modules (e.g., cross-attention (Dasari et al., 2024), AdaLN (Dasari et al., 2024), FiLM (Perez et al., 2018; Chi et al., 2023)) to inject observations at each step of the generation process. Furthermore, generative and autoregressive methods commonly employ large, expressive networks such as U-Nets or transformers to succeed on complex, high-dimensional robotics tasks. VITA reduces these complexity by formulating the policy as a noise-free and conditioning-free vision-to-action flow; by using vector representations for vision and action, VITA also enables simple and lightweight MLP-only network for policy parameterization, even for challenging bi-manual manipulation tasks.

Diffusion and Flow matching for Generative Modeling. Diffusion (Ho et al., 2020), grounded in stochastic differential equations (SDEs), generates complex data distributions by sampling from a simple source distribution (typically Gaussian) and iteratively denoising it to the target distribution (Sohl-Dickstein et al., 2015). Flow matching (Lipman et al., 2023; Liu et al., 2022b) has been proposed to enable faster training and sampling (Liu et al., 2022b; Tong et al., 2024; Esser et al.; Lipman et al., 2023) by modeling the map between source and target distributions with an ordinary differential equation (ODE). Both diffusion and flow matching models have shown strong performance across diverse generative tasks, such as image generation (Rombach et al., 2022; Peebles & Xie, 2023; Ma et al., 2024; Zhang et al., 2023; Liu et al., 2024b; Ren et al., 2024b), video generation (Ho et al., 2022; Li et al., 2023), and visuomotor policies (Chi et al., 2023; Dasari et al., 2024; Black et al.; Liu et al., 2024c). Unlike diffusion, flow matching theoretically places no constraints on the choice of source distribution (Tong et al., 2024), and a few works have explored leveraging this property to learn the direct transport within the same modality (Albergo & Vanden-Eijnden, 2022; Tong et al., 2023b), e.g., for image-to-image generation tasks (Fischer et al., 2023; Liu et al., 2022a). Recently, Liu et al. (2024a) and He et al. (2025) extended this to more challenging cross-modal generation between text and image. VITA focuses on learning to bridge vision and action for visuomotor control, where the target modality, action, features sparser data, and lacks semantic structures, compared to text or images, presenting unique challenges. Different from flow matching for image generation,

which typically pre-trains and freezes the image encoder and decoder when learning flow matching or diffusion models for image generation (Rombach et al., 2022; He et al., 2025; Liu et al., 2024a), VITA resorts to a fully end-to-end pipeline training to effectively learn the latent action space from limited and sparse action data. Furthermore, to enable effective joint training of flow matching and target latent spaces, we propose flow latent decoding to backpropagate action reconstruction losses through the the latent action generation process (ODE solving steps) during training.

E VITA Generation Process

We compare the denoising processes of conventional flow matching and VITA in Figure 9. Conventional methods begin from Gaussian noise and iteratively denoise it into actions, whereas VITA starts directly from latent visual representations. By jointly learning flow matching and the action autoencoder, VITA constructs a structured latent space in which even the initial latent can be decoded into a coherent trajectory, which is then progressively refined along the flow matching ODE.

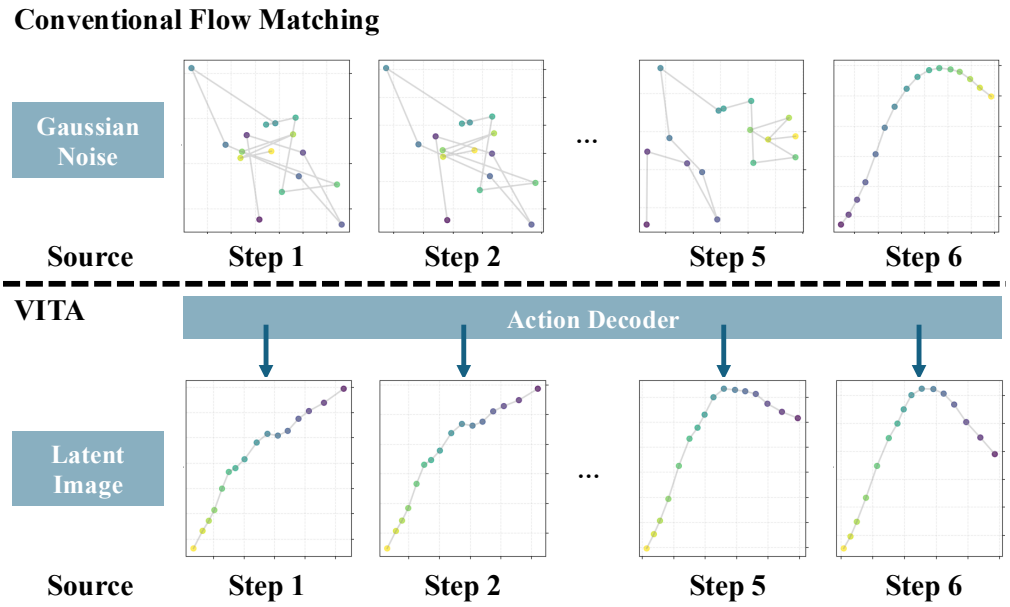

Figure 9: Comparison of action generation process: Conventional flow matching policies flow from random noise, and gradually "denoise" into raw action chunks by solving ODE; in contrast, VITA flows from latent images to latent actions across ODE steps; the latent actions are decoded to raw action chunks.

F Discussion of Training Objectives

F.1 Ablation: Action VAE and KL Regularization

Our main experiments use a deterministic action autoencoder as the target modality. To test whether imposing a prior on the action latent helps, we also trained a variational action autoencoder in which the encoder outputs $q(\boldsymbol{z}_1 \mid A)$ and we add a KL regularizer (weighted by λ_{KL})

$$\mathcal{L}_{\mathrm{KL}} \;=\; D_{\mathrm{KL}}\big(q(\boldsymbol{z}_1 \mid A) \,\big\|\, p(\boldsymbol{z}_1)\big), \qquad p(\boldsymbol{z}_1) = \mathcal{N}(0, I). \tag{3}$$

Figure 10(a) shows training curves for several λ_{KL} values, and the sweep in Fig. 11 (panel `action_kl_wt`) reports the broader trend. Introducing a nonzero KL weight generally hurts performance in our setting: convergence tends to slow and the final success rate typically drops relative to λ_{KL}=0. Very small weights (e.g., 10^{-4}) can track or even beat the zero-KL curve early and underperform at plateau, while moderate/large weights (e.g., 10^{-2}–5) yield noticeably lower plateaus; across seeds, the zero-KL setting is consistently among the best.

This behavior aligns with the training mechanics in §4.4: **FLD** anchors the ODE endpoint $\hat{z}_1$ via action reconstruction, and the quality of this anchor depends on the decoder being structured and locally well-behaved (small reconstruction bias ε_{AE} and decent local conditioning m; Theorem 1). An action KL prior pushes $q(z_1 \mid A)$ toward $\mathcal{N}(0, I)$, which in our sparse/low-entropy datasets reduces latent capacity and distorts local geometry—empirically increasing ε_{AE} and weakening m—so the link between latent and action errors becomes looser and the anchor noisier. We therefore adopt a non-variational autoencoder (λ_{KL}=0) by default.

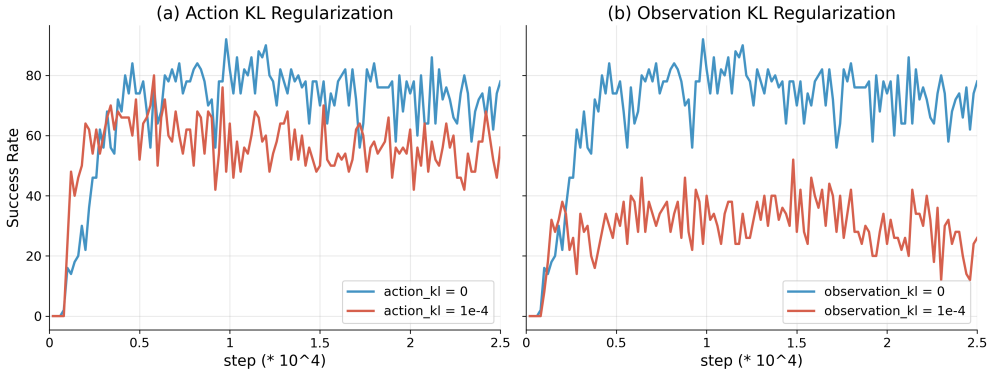

Figure 10: KL regularization ablation on action and observation latents on `ThreadNeedle`.

F.2 Variational Image Encoder

Recall the source latent is $z_0 = \mathcal{E}_v(O) \in \mathbb{R}^{D_{\text{latent}}}$ (Fig. 2). To assess whether imposing a prior on visual latents helps, we replace the deterministic encoder with a variational image encoder that produces a posterior $q(z_0 \mid O)$ and add a KL penalty (weighted by $\lambda_{\text{KL}}^{\text{obs}}$):

$$\mathcal{L}_{\text{KL}}^{\text{obs}} \;=\; D_{\text{KL}}\big(q(z_0 \mid O) \,\big\|\, p(z_0)\big), \qquad p(z_0) = \mathcal{N}(0, I). \tag{4}$$

Architecturally, $\mathcal{E}_v$ shares the same backbone as in §4.2 with mean/log-variance heads and reparameterization; all other components (flow, action AE, FLD/FLC) are unchanged.

As shown in Fig. 10(b), adding observation-KL degrades performance more strongly than action-KL: very small weights can initially track $\lambda_{\text{KL}}^{\text{obs}}$=0 but plateau lower.

This is consistent with §4.4's picture: in VITA, z_0 carries all task information for the flow (no separate conditioning). Pushing $q(z_0 \mid O)$ toward an isotropic prior reduces $I(z_0; O)$ and blurs task-relevant features, which weakens both the velocity supervision in FM (the straight-line target $z_1 - z_0$ becomes less informative) and the endpoint anchoring via FLD/FLC (the flow starts from a less informative z_0, making $\hat{z}_1$ harder to align). We therefore use a deterministic image encoder by default ($\lambda_{\text{KL}}^{\text{obs}}$=0).

G Task Settings

Dataset	State Dim	Action Dim	FPS	Image Size	Camera
AV-ALOHA (Sim)	21	21	8.33	240×320	`zed_cam_left`
AV-ALOHA (Real)	21	21	11	240×320	`left_eye_cam`
RoboMimic	43	7	20	256×256	`agentview_image`

Table 5: Comparison of dataset specifications.

G.1 AV-ALOHA Simulation tasks.

`CubeTransfer`: Pick up a red cube with the right arm and transfer it to the left arm (200 episodes).

`SlotInsertion`: Use both arms to pick up a green stick and insert it into a pink slot (100 episodes).

`HookPackage`: Use both arms to pick up a red box and hook it onto a blue wall-mounted hook (100 episodes).

`PourTestTube`: Pick up two test tubes and pour a small red ball from one into the other (100 episodes).

`ThreadNeedle`: Pick up a green needle and thread it through the hole of a pink object (200 episodes).

G.2 AV-ALOHA Real tasks

`HiddenPick`: Lift and open a fabric cover from a box, then pick an object from inside (50 episodes).

`TransferFromBox`: Pick an object from a box with the right arm, transfer it to the left arm, and place it in another box (50 episodes).

G.3 Robomimic Tasks

Robomimic is a benchmark of single-arm imitation learning tasks Mandlekar et al. (2021). We adapt the environment for compatibility with the Cadene et al. (2024) codebase. The robot state includes arm joint positions (encoded with sin and cos), joint velocities, end-effector pose, gripper finger positions, and gripper finger velocities. The action space consists of six values for delta position control of the end-effector pose and one value for the absolute position of the gripper.

`Square`: Pick up a square nut and insert it onto a matching square peg (175 episodes).

`Can`: Pick up a red can and place it into a box (192 episodes).

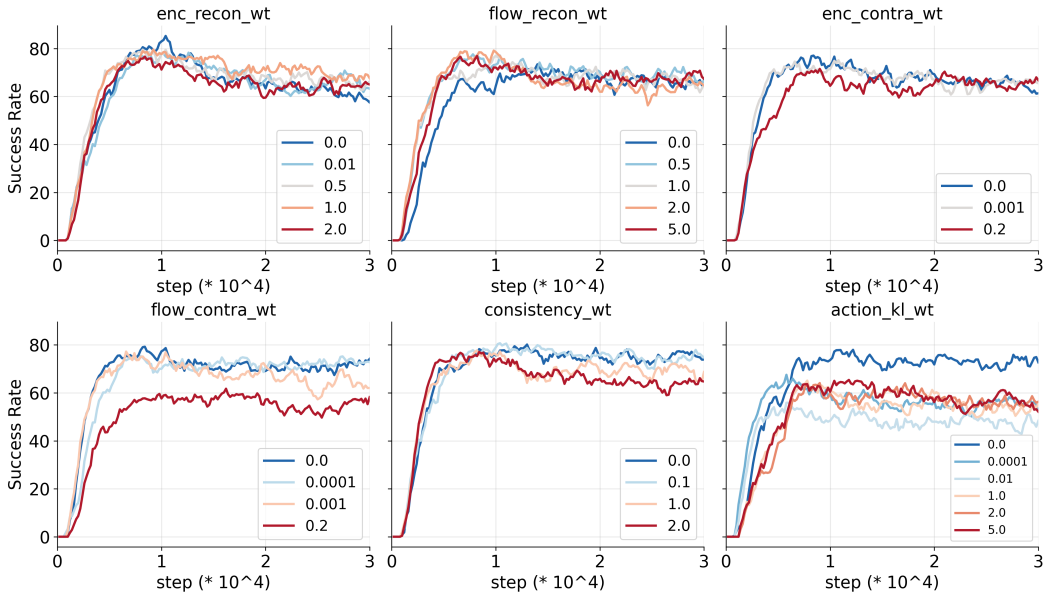

Figure 11: **Hyperparameter ablations on `ThreadNeedle`.** Each panel varies a single coefficient while holding all others at their default values; y-axis shows success rate (%), x-axis shows training steps ($\times 10^4$). Top: action encoder reconstruction weight, flow latent decoding weight, action encoder contrastive weight. Bottom: latent flow contrastive weight, latent flow consistency weight, action KL weight.

H Training

We run a one–factor-at-a-time ablation on `ThreadNeedle`: in each plot of Fig. 11, we sweep a single weight while holding all other hyperparameters at the default used in our main experiments.

Curves report success rate throughout training (mean ± 1 std across seeds); evaluations occur at fixed intervals. This isolates the marginal effect of each hyper-parameter choice.

With these experimental results, a robust configuration on `ThreadNeedle` is: moderate autoencoder reconstruction and FLD weights (typically in $[0.5, 1.0]$), small or morderate FLC, minimal or no contrastive penalties, and no KL regularization on the action latent. This setting consistently yields faster convergence and higher plateaus on `ThreadNeedle`.

I IMPLEMENTATIONS

I.1 VITA

VITA uses vector latents throughout: a ResNet-18 encodes observations into $\boldsymbol{z}_0$, a lightweight action autoencoder maps actions to $\boldsymbol{z}_1$, and a flow network transports $\boldsymbol{z}_0 \to \hat{\boldsymbol{z}}_1$ with a 6-step Euler ODE. Training combines FM with endpoint supervision via FLD (and optionally FLC); the action AE is deterministic by default (no KL). Core hyperparameters and loss weights are summarized in Table 6.

I.2 FLOW MATCHING POLICY

The FM baseline keeps the same horizons and ResNet-18 observer as VITA, but removes endpoint supervision and decoding: it learns a velocity field with a transformer backbone (AdaLN blocks, 4 layers, 8 heads) and integrates it for 6 steps. The matcher is the standard conditional FM variant used in our experiments. Core hyperparameters and loss weights are summarized in Table 7.

I.3 DIFFUSION POLICY

Our diffusion baseline follows DDPM training with a cosine β schedule (100 timesteps) and a FiLM-modulated U-Net (down dims $[512, 1024, 2048]$). Observations use the same ResNet-18 encoder; actions are generated by iterative denoising at inference. Optimization and scheduler choices mirror the other models. Core hyperparameters and loss weights are summarized in Table 8.

I.4 ACTION CHUNKING TRANSFORMER

ACT is a sequence-to-sequence transformer that predicts a 16-step action chunk from a 1-step observation context. It uses a 4-layer encoder (512 hidden, 8 heads, FFN 3200, dropout 0.1) and a small VAE head (latent 32) with a non-zero KL weight for regularization. This serves as a strong non-generative baseline. Core hyperparameters and loss weights are summarized in Table 9.

Table 6: **VITA hyperparameters**.

Horizons & Observation	
Observation horizon	`1`
Prediction horizon	`16`
Action horizon	`8`
Observer backbone	`resnet18`
Observer tokenize	`false`
VITA core (latent & losses)	
Latent dimension (D_{latent})	`512`
Decode flow latents	`true`
Consistency weight	`1.0`
Encoder contrastive weight	`1e-4`
Flow contrastive weight	`0.0`
Latent noise std	`0.0`
Flow matcher / ODE	
Matcher name	`exact`
σ	`0.0`
# sampling steps (Euler)	`6`
Action autoencoder (AAE)	
AE recon loss type	`l1`
Encoder recon weight	`0.5`
Flow recon (FLD) weight	`0.5`
Use variational (VAE)	`false`
KL weight (if variational)	`1e-6`
Freeze encoder / decoder	`false / false`
Pretrained path	`None`
AE network (encoder/decoder)	
Encoder type / hidden dim	`cnn / 512`
Decoder type / hidden dim	`simple / 512`
Latent dim (AE)	`${policy.vita.latent_dim} (=512)`
Num heads / MLP ratio	`8 / 4`
Dropout	`0.0`
Num layers	`4`

Table 7: **Flow Matching (FM) Policy hyperparameters**.

Horizons & Observation	
Observation horizon	`1`
Prediction horizon	`16`
Action horizon	`8`
Observer backbone	`resnet18`
Observer tokenize	`false`
Flow matcher / ODE	
Matcher name	`target`
σ	`0.0`
# sampling steps	`6`
Flow network architecture	
Backbone	`flow_transformer`
Block type	`adaln` (*options:* `adaln`, `cross`, `cross_adaln`)
Hidden dim	`512`
Num layers	`4`
Num heads	`8`
MLP ratio	`4`
Dropout	`0.1`

Table 8: **Diffusion policy hyperparameters**.

Horizons & Observation	
Observation horizon	1
Prediction horizon	16
Action horizon	8
Observer backbone	resnet18
Observer tokenize	false
Mask loss for padding	false
Diffusion scheduler	
Type	DDPM
Training timesteps	100
Beta schedule	squaredcos_cap_v2
Beta start / end	1e-4 / 2e-2
Prediction type	epsilon
Clip sample / range	true / 1.0
U-Net architecture	
Down dims	[512, 1024, 2048]
Kernel size	5
Group norm groups	8
Diffusion step embed dim	128
FiLM scale modulation	true
Optimization	
Adam LR / backbone scale	1e-4 / 0.1
Adam betas / eps	(0.95, 0.999) / 1e-8
Weight decay	1e-6
Scheduler	cosine, warmup 500 steps
Training & inference	
Total training steps	200000
Inference steps	null (defaults to training timesteps)

Table 9: Action Chunking Transformer (ACT): hyperparameters.

Sequence horizons	
Action horizon	8
Prediction horizon	16
Observation horizon	1
Observer / backbone	
Image encoder	ResNet-18
Tokenize	false
Transformer (policy head)	
Pre-norm	false
Model dimension d_{model}	512
Attention heads	8
Feedforward dim	3200
FFN activation	ReLU
Encoder layers	4
Decoder layers	1
Dropout	0.1
Latent / VAE block	
Use VAE	true
Latent dim	32
VAE encoder layers	4
ACT KL weight	10.0
Optimization	
Optimizer	Adam
Learning rate	1×10^{-5}
Betas	$(0.9,\ 0.999)$
ϵ	1×10^{-8}
Weight decay	1×10^{-4}
Backbone LR scale	0.1
LR scheduler & validation	
Scheduler	Cosine
Warmup steps	2000
Online validation frequency	2000 steps