

# Fast Goal-Conditioned Flow-Matching Policy for RoboMimic Manipulation

Gavin Chen(gc487), Bopeng Zhang (bz292)

## 1 Introduction

Diffusion-based policies have become a strong tool for imitation learning in robotic manipulation because they can represent complex, multi-step action distributions conditioned on observations and goals. The main barrier to real-time deployment is inference latency: standard diffusion sampling relies on many iterative denoising steps, which is expensive when control requires high-rate action updates and small delays or errors can quickly compound in closed-loop execution.

This project studies whether we can retain the expressivity of diffusion policies while making inference fast and controllable. We propose a *Fast Goal-Conditioned Flow-Matching Policy (FGDP)* that models fixed-horizon action chunks as samples from a conditional *rectified flow* in trajectory space. FGDP learns a goal-conditioned velocity field via flow matching between expert action chunks and Gaussian noise along a simple linear interpolation path. At test time, FGDP generates an action chunk by deterministically integrating the learned velocity field for only  $K$  steps, where  $K$  provides a direct speed–accuracy knob: smaller  $K$  minimizes latency, while larger  $K$  improves integration fidelity and temporal coherence.

We evaluate FGDP on four RoboMimic proficient-human manipulation tasks (LIFT, CAN, SQUARE, TRANSPORT) under both low-dimensional and image-based observations. Our experiments quantify the quality–speed trade-off as  $K$  varies, benchmark against standard behavioral cloning (BC), and probe conditioning reliance through goal and image ablations. Results show that increasing  $K$  consistently improves action accuracy and, more strongly, chunk smoothness, while ablations confirm that FGDP meaningfully uses both goal and visual inputs. Overall, FGDP provides a simple and effective mechanism for fast goal-conditioned trajectory generation in imitation learning with an explicit deployment knob for real-time control.

## 2 Related Works

### General Fast Diffusion Sampling and Distillations

Motivated by the heavy compute cost of standard diffusion models, a large body of work has focused on accelerating the iterative denoising process while preserving sample quality. A representative line of approach is teacher–student distillation, where a fast student is trained to match the output of a slower, high-quality teacher while progressively collapsing many denoising steps into a small number of steps [1]. Closely related ideas appear in *consistency*-based formulations, which aim to learn mappings that are stable across noise levels so that high-quality samples can be produced in only a few evaluations; Latent Consistency Models adapt this principle to latent diffusion backbones and demonstrate few-step synthesis without running long denoising chains [2]. More recently, one-step generative transport mappings have pushed this idea further: InstaFlow shows that, under a rectified/flow-style perspective, a single network evaluation can often be sufficient for high-quality generation [3, 4]. While these methods provide the conceptual and algorithmic foundation for making diffusion inference fast—a prerequisite for our setting—they are largely developed in the context of image generation, where minor perceptual differences are tolerable and errors do not compound through closed-loop execution. For control, the same kinds of approximation and step-skipping that are benign for pixels can translate into systematic temporal drift or unstable behavior, motivating

architectures and training objectives that explicitly respect the structure and requirements of policy inference.

### Accelerations and Improvements to Diffusion Policies

Robotics research has begun to adapt these acceleration and conditioning ideas to diffusion-based control. One-Step Diffusion Policy adopts the distillation approach and distills a standard diffusion policy into a fast student that can synthesize actions in a single (or very small number of) forward passes, directly targeting the inference bottleneck that makes vanilla diffusion policies difficult to deploy at control rates [5, 6]. Another direction of acceleration, Streaming Diffusion Policy exploits the structure of receding-horizon control by *streaming* generation: rather than restarting a full denoising process for each timestep, it incrementally updates action sequences under variable-noise dynamics, reusing computation across timesteps [7]. Aside from general accelerations, other works have also explored strengthening controllability at inference time. DynaGuide introduces an active, dynamics-aware guidance mechanism that steers diffusion policies during sampling, aiming to bias generations toward trajectories that are more consistent with task dynamics or desired outcomes without retraining the base model [8]. Collectively, these methods make clear progress toward real-time diffusion control—either by collapsing denoising into one-step inference or amortizing computation over time, but they also highlight a remaining tension: aggressive acceleration can weaken the policy’s ability to represent multimodal, high-precision behavior, while heavy guidance and per-step optimization can reintroduce latency. Our approach is motivated by closing this gap, keeping diffusion-level expressivity and conditioning while making inference fast enough for practical closed-loop execution.

## 3 Problem Formulation

We study *goal-conditioned action-chunk generation* for robotic manipulation under two observation modalities: (i) low-dimensional state and (ii) RGB images with accompanying robot state. At each decision time  $t$ , the policy outputs a horizon- $H$  open-loop action sequence (an *action chunk*)

$$\mathbf{a}_{t:t+H-1} \in \mathbb{R}^{H \times d_a}, \quad (1)$$

where  $d_a$  is the continuous action dimension. We flatten the chunk as  $\mathbf{x} \in \mathbb{R}^{Hd_a}$  when defining the generative model.

### 3.1 Observations and Goal Conditioning

Each sample provides an observation and a goal, written as

$$o = (I^{\text{obs}}, s^{\text{obs}}), \quad g = (I^{\text{goal}}, s^{\text{goal}}), \quad (2)$$

where  $I \in \mathbb{R}^{3 \times H_{\text{img}} \times W_{\text{img}}}$  is an RGB image (if available) and  $s$  denotes a low-dimensional state vector (robot proprioception and task-relevant state features). In the low-dimensional setting, the image components may be absent; in the image setting, both image and state components are available. The goal  $g$  specifies the desired end condition, and the policy is explicitly conditioned on both  $o$  and  $g$ .

### 3.2 Goal-Conditioned Rectified Flow Policy

We represent the policy as a conditional rectified flow (flow-matching) model that defines a velocity field over action chunks. Let  $\mathbf{x}_0 \in \mathbb{R}^{Hd_a}$  be an expert action chunk from the dataset and let  $\mathbf{x}_1 \sim \mathcal{N}(0, I)$  be a noise sample. We define an interpolation path

$$\mathbf{x}_t = (1 - t)\mathbf{x}_0 + t\mathbf{x}_1, \quad t \sim \text{Uniform}(0, 1), \quad (3)$$

whose target (rectified) velocity is constant:

$$\mathbf{v}^* = \frac{d\mathbf{x}_t}{dt} = \mathbf{x}_1 - \mathbf{x}_0. \quad (4)$$

Our model learns a goal-conditioned velocity field

$$\mathbf{v}_\theta(\mathbf{x}_t, t \mid o, g) \in \mathbb{R}^{Hd_a}, \quad (5)$$

which predicts the instantaneous transport direction in action-chunk space given the current point  $\mathbf{x}_t$ , the scalar time  $t$ , and the conditioning  $(o, g)$ .

### 3.3 Training Objective (Flow Matching)

We learn  $\theta$  by minimizing a flow-matching regression loss between the predicted velocity and the rectified target velocity:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{(\mathbf{x}_0, o, g), \mathbf{x}_1, t} \left[ \|\mathbf{v}_\theta(\mathbf{x}_t, t \mid o, g) - (\mathbf{x}_1 - \mathbf{x}_0)\|_2^2 \right], \quad (6)$$

where  $\mathbf{x}_0$  is sampled from demonstrations,  $\mathbf{x}_1 \sim \mathcal{N}(0, I)$ , and  $t \sim U[0, 1]$ .

### 3.4 Inference with $K$ -Step Integration

At test time, we generate an action chunk by integrating the learned velocity field starting from noise at  $t = 1$  back to  $t = 0$ . Given  $(o, g)$ , we initialize  $\mathbf{x}^{(0)} \sim \mathcal{N}(0, I)$  and perform  $K$  explicit integration steps with  $\Delta t = 1/K$ :

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \Delta t \mathbf{v}_\theta(\mathbf{x}^{(i)}, t_i \mid o, g), \quad t_i = 1 - i\Delta t, \quad i = 0, \dots, K-1. \quad (7)$$

The final  $\hat{\mathbf{x}} = \mathbf{x}^{(K)}$  is reshaped into  $\hat{\mathbf{a}}_{t:t+H-1} \in \mathbb{R}^{H \times d_a}$  and executed (typically in a receding-horizon manner). The parameter  $K$  directly controls the inference-time compute/latency: smaller  $K$  yields faster sampling, while larger  $K$  yields a finer numerical integration of the flow.

### 3.5 Offline Metrics Used in This Project

To quantify behavior cloning fidelity and the effect of reducing  $K$ , we report: (i) action-chunk mean squared error  $\mathbb{E}[\|\hat{\mathbf{x}} - \mathbf{x}_0\|_2^2]$  on held-out samples, (ii) an action smoothness score based on mean squared first differences across the horizon, and (iii) measured inference latency as a function of  $K$ .

## 4 Method

### 4.1 Overview

We propose a *Fast Goal-Conditioned Flow-Matching Policy* (FGDP) for Robomimic manipulation. Similar to diffusion-based policies, FGDP models a distribution over fixed-horizon action trajectories conditioned on the current observation and a goal observation. However, instead of running a long denoising chain, FGDP learns a *rectified flow* in action-trajectory space and performs *deterministic* sampling using only  $K$  evaluations of a learned velocity field. This yields a simple and controllable speed-accuracy knob via  $K$ .

### 4.2 Goal-conditioned action-trajectory modeling

At each timestep  $t$ , we predict a horizon- $H$  action chunk

$$\mathbf{a}_{t:t+H-1} \in \mathbb{R}^{H \times d_a}, \quad (8)$$

which we flatten into a single vector

$$\mathbf{x}_0 \triangleq \text{vec}(\mathbf{a}_{t:t+H-1}) \in \mathbb{R}^D, \quad D = H \cdot d_a. \quad (9)$$

The policy is conditioned on both the current observation and a goal:

$$\mathbf{c}_t = \phi(o_t, g_t). \quad (10)$$

In our dataset construction, a goal index  $t_g$  is chosen from the same demonstration trajectory. When enabling future-goal conditioning,  $t_g$  is sampled from a future timestep with a minimum temporal offset; otherwise the goal is the final timestep of the trajectory. The resulting supervised sample is

$$(o_t, g_t, \mathbf{x}_0), \quad (11)$$

where  $o_t$  provides the current state/image and  $g_t$  provides the goal state/image.

### 4.3 Conditional encoder

FGDP supports both low-dimensional states and images through a shared conditioning interface. We encode:

1. **Current state**  $\mathbf{s}_t \in \mathbb{R}^{d_s}$  and **goal state**  $\mathbf{s}_{t_g} \in \mathbb{R}^{d_g}$  using separate MLP encoders.
2. **Current image**  $\mathbf{I}_t$  and **goal image**  $\mathbf{I}_{t_g}$  using a lightweight CNN encoder (three strided convolutions, global average pooling, and a linear projection).

All encoded features are concatenated and passed through a fusion MLP to produce a fixed-dimensional conditioning vector:

$$\mathbf{c}_t = \text{MLP}_{\text{fuse}}\left(\left[\psi_s(\mathbf{s}_t), \psi_g(\mathbf{s}_{t_g}), \psi_I(\mathbf{I}_t), \psi_I(\mathbf{I}_{t_g})\right]\right) \in \mathbb{R}^{d_c}. \quad (12)$$

(For state-only datasets, images are zero-filled by construction to preserve an identical input interface.)

### 4.4 Rectified flow parameterization

We learn a time-dependent velocity field over action trajectories:

$$\mathbf{v}_\theta(\mathbf{x}, \tau, \mathbf{c}) : \mathbb{R}^D \times [0, 1] \times \mathbb{R}^{d_c} \rightarrow \mathbb{R}^D, \quad (13)$$

implemented as an MLP that takes the concatenation of (i) the current trajectory point  $\mathbf{x}$ , (ii) a sinusoidal time embedding  $\gamma(\tau)$ , and (iii) the conditioning vector  $\mathbf{c}$ :

$$\mathbf{v}_\theta(\mathbf{x}, \tau, \mathbf{c}) = \text{MLP}_v([\mathbf{x}, \gamma(\tau), \mathbf{c}]). \quad (14)$$

### 4.5 Flow-matching objective (velocity field training)

Training uses a straight-line coupling between expert trajectories and Gaussian noise. Given a data point  $\mathbf{x}_0$  and an independent noise sample

$$\mathbf{x}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad (15)$$

we sample a scalar  $\tau \sim \mathcal{U}(0, 1)$  and define the interpolated point

$$\mathbf{x}_\tau = (1 - \tau)\mathbf{x}_0 + \tau\mathbf{x}_1. \quad (16)$$

Along this linear path, the target velocity is constant:

$$\mathbf{v}^*(\mathbf{x}_\tau, \tau) = \frac{d\mathbf{x}_\tau}{d\tau} = \mathbf{x}_1 - \mathbf{x}_0. \quad (17)$$

We then train  $\mathbf{v}_\theta$  by minimizing mean squared error between predicted and target velocities:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{(o_t, g_t, \mathbf{x}_0), \mathbf{x}_1, \tau} \left[ \left\| \mathbf{v}_\theta(\mathbf{x}_\tau, \tau, \phi(o_t, g_t)) - (\mathbf{x}_1 - \mathbf{x}_0) \right\|_2^2 \right]. \quad (18)$$

This objective directly learns a rectified flow (velocity field) that maps noise to data while remaining conditioned on  $(o_t, g_t)$ .

### 4.6 Fast $K$ -step sampling (inference)

At test time, we generate an action trajectory by integrating the learned flow from noise to data using an explicit Euler solver with  $K$  steps. Initialize:

$$\mathbf{x}^{(0)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \Delta = \frac{1}{K}. \quad (19)$$

For  $i = 0, 1, \dots, K - 1$ , set  $\tau_i = 1 - i\Delta$  and update:

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \Delta \cdot \mathbf{v}_\theta\left(\mathbf{x}^{(i)}, \tau_i, \mathbf{c}_t\right). \quad (20)$$

The final prediction is  $\hat{\mathbf{x}}_0 = \mathbf{x}^{(K)}$ , which is reshaped back to  $\hat{\mathbf{a}}_{t:t+H-1} \in \mathbb{R}^{H \times d_a}$ . Crucially, sampling cost scales linearly with  $K$ , enabling a direct trade-off between speed ( $K$  small) and trajectory fidelity ( $K$  larger).

#### 4.7 Training pipeline

We train FGDP using minibatch stochastic optimization over the preprocessed RoboMimic-derived .pt datasets. Each dataset item returns:

$$\{\mathbf{s}_t, \mathbf{s}_{t_g}, \mathbf{I}_t, \mathbf{I}_{t_g}, \mathbf{x}_0\}, \quad (21)$$

where images are converted from uint8 to float and normalized to  $[0, 1]$  (with channel-first layout). The training loop repeatedly samples  $(\mathbf{x}_0, \mathbf{x}_1, \tau)$ , evaluates  $\mathcal{L}_{\text{FM}}$ , updates  $\theta$  using AdamW, and periodically saves checkpoints for evaluation.

### 5 Experiments

We evaluate our fast goal-conditioned flow policy (FGDP) on RoboMimic manipulation tasks under both low-dimensional and image-based observations. Our experiments are designed to answer four practical questions: (i) the *quality-latency trade-off* induced by  $K$ -step flow integration (i.e., how accuracy, smoothness, and inference time change as  $K$  varies), (ii) how prediction quality (MSE and smoothness) changes as  $K$  varies, (iii) whether the learned policy truly leverages the *goal* signal, and (iv) for image policies, how sensitive the controller is to visual inputs. We additionally benchmark against standard behavioral cloning (BC) checkpoints under the same offline chunked evaluation.

#### 5.1 Experimental Setup

**Tasks and datasets.** We use four RoboMimic proficient-human (PH) tasks: LIFT, CAN, SQUARE, and TRANSPORT. For each task, we consider two observation modalities. In the *low-dimensional* setting, the policy conditions on the compact state vector provided by RoboMimic. In the *image-based* setting, the policy additionally conditions on an RGB camera stream of resolution  $84 \times 84$ , alongside the same low-dimensional state. We train and evaluate one model per task and modality.

**Trajectory chunk prediction.** Following our formulation, the policy predicts a fixed-horizon action *chunk* of length  $H = 10$  from the current observation and a goal derived from a future timestep (denoted `future_goal` in preprocessing). Chunk prediction is a particularly revealing regime for flow-based policies: it exposes temporal coherence and integration artifacts (e.g., jitter) that may be hidden in single-step prediction.

**Offline evaluation metrics.** We report two offline metrics computed over action chunks: (1) action MSE between predicted and expert action sequences, and (2) smoothness of the predicted chunk, measured by the mean squared finite difference between consecutive actions (reported as `smooth(pred)`; the expert reference is `smooth(gt)`). We additionally measure inference latency using repeated forward sampling with batch size 1 to mimic real-time control (one observation at a time).

**$K$ -step inference (quality-speed knob).** At test time, we vary the number of flow integration steps  $K \in \{1, 5\}$ . Intuitively, larger  $K$  better approximates continuous rectified-flow integration, while smaller  $K$  targets minimal latency. Throughout the paper,  $K = 1$  serves as our *fast inference baseline* for FGDP, and  $K = 5$  represents a modest increase in computation to recover integration accuracy.

**Ablations (conditioning reliance).** To probe reliance on conditioning, we perform controlled dataset-level perturbations. For both low-dim and image policies, `zero_goal` sets the goal inputs to zero and `shuffle_goal` permutes goals across samples, breaking the observation–goal correspondence. For image policies only, `zero_image` sets image tensors to zero (observation and goal images when present) and `shuffle_image` permutes images across samples while preserving the low-dim state. Low-dim experiments do not include image ablations.

**BC baselines.** We additionally evaluate RoboMimic BC checkpoints under the same offline chunk protocol ( $H = 10$ ) and report their chunk-wise MSE, predicted smoothness, and microbenchmarked latency. Since BC is a single forward-pass predictor, we report it as a single setting (no  $K$ ).

Table 1: **BC vs. FGDP (offline chunk metrics).** BC is a single-pass predictor; FGDP is shown at  $K=1$  (fast inference) and  $K=5$  (more accurate integration). Lower is better.

Task	Modality	Action MSE ↓			Smooth(pred) ↓		
		BC	FGDP@1	FGDP@5	BC	FGDP@1	FGDP@5
Lift	low-dim	0.072797	0.1322	0.1125	0.352872	0.758	0.478
Lift	image	0.173332	0.1117	0.0871	0.147348	0.545	0.310
Can	low-dim	0.049610	0.1175	0.0774	0.235197	0.700	0.131
Can	image	0.565127	0.1151	0.0726	0.351361	0.707	0.129
Square	low-dim	0.036935	0.0879	0.0579	0.191228	0.600	0.078
Square	image	0.350668	0.0876	0.0567	0.041410	0.604	0.080
Transport	low-dim	0.046071	0.0598	0.0323	0.535777	1.054	0.087
Transport	image	0.316031	0.0569	0.0300	0.616192	1.031	0.085

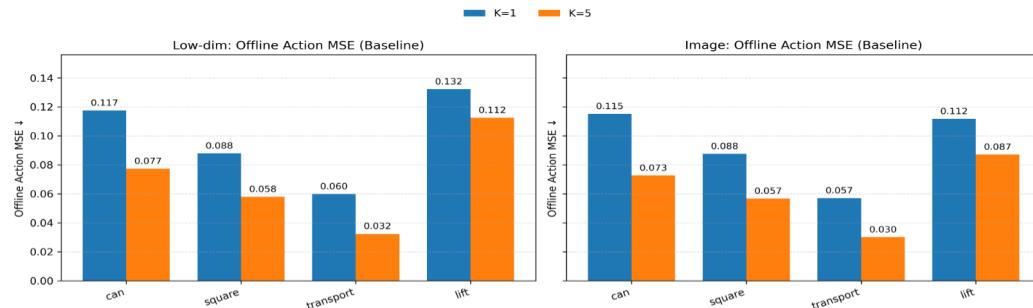


Figure 1: Bar chart of action MSE for  $K=1$  vs.  $K=5$  across tasks (separate panels for low-dim and image)

## 5.2 Quantitative Results

**Effect of  $K$ -step inference in FGDP.** Table 1 summarizes performance across tasks and modalities for  $K \in \{1, 5\}$ . Increasing  $K$  consistently improves both action accuracy and smoothness. In particular,  $K=1$  often produces noticeably higher smooth(pred) (jittery action chunks), while  $K=5$  brings smooth(pred) much closer to expert smoothness across CAN, SQUARE, and TRANSPORT. This pattern matches the intended role of  $K$  as a deployment knob: a small number of steps enables fast sampling, while additional steps better resolve rectified-flow integration and yield more coherent multi-step action chunks.

**Benchmark comparison against BC.** To contextualize FGDP relative to a standard imitation baseline, Table 1 compares BC checkpoints against FGDP under the same offline chunked evaluation. Two trends stand out. First, BC is highly competitive in the low-dimensional setting for LIFT, CAN, and SQUARE, whereas FGDP benefits more clearly from additional flow steps (e.g., TRANSPORT-low-dim where  $K=5$  improves substantially and surpasses BC). Second, in the image-based setting, FGDP maintains strong performance and dramatically outperforms BC across all four tasks, suggesting that the goal-conditioned flow formulation and  $K$ -step sampling provide a robust mechanism for producing accurate multi-step chunks under visual inputs. This is especially evident in tasks that require precise geometric alignment (notably SQUARE), where the image-conditioned FGDP remains accurate while BC incurs much larger error.

**Inference latency (microbenchmark).** We microbenchmark inference with batch size 1 and horizon  $H = 10$  to assess real-time feasibility. Table 2 shows that FGDP achieves sub-millisecond *chunk-level* inference at  $K=1$  (approximately 0.75–0.78 ms per chunk) and scales approximately linearly to  $K=5$  (about 3.56–3.78 ms per chunk). Behavioral cloning (BC), in contrast, operates at the *step level* and requires one forward pass per action. As a result, while BC step latency is comparable to FGDP at  $K=1$  (about 0.8–0.9 ms per step in the low-dimensional setting), generating

an  $H$ -step action chunk incurs  $O(H)$  sequential cost, leading to substantially higher chunk-level latency (about 8.6–9.4 ms per chunk). This gap becomes more pronounced for image-based policies, where BC incurs 50–90 ms per chunk. Overall, these results highlight  $K$  as an explicit speed-quality knob for FGDP: increasing  $K$  improves action accuracy and smoothness (Table 1) while incurring a roughly linear increase in computation, yet remaining competitive with or faster than BC at comparable control semantics.

Table 2: **Inference latency (microbenchmark).** Batch size 1, horizon  $H = 10$  (lower is better), and inputs preloaded on the target device. FGDP latency is reported for  $K \in \{1, 5\}$ ; step latency is computed as chunk latency divided by  $H$ .

Task	Modality	BC		FGDP@1		FGDP@5	
		Chunk (ms) ↓	Step (ms) ↓	Chunk (ms) ↓	Step (ms) ↓	Chunk (ms) ↓	Step (ms) ↓
Lift	low-dim	8.751	0.875	0.768	0.077	3.779	0.378
Lift	image	52.914	5.291	0.757	0.076	3.706	0.371
Can	low-dim	8.645	0.864	0.760	0.076	3.622	0.362
Can	image	50.065	5.007	0.757	0.076	3.755	0.376
Square	low-dim	8.835	0.883	0.772	0.077	3.783	0.378
Square	image	51.704	5.170	0.759	0.076	3.691	0.369
Transport	low-dim	9.402	0.940	0.754	0.075	3.556	0.356
Transport	image	90.349	9.035	0.778	0.078	3.718	0.372

**Goal-conditioning ablation.** We next test whether FGDP genuinely relies on the goal signal, rather than behaving as a reactive behavioral cloning model that ignores future targets. Table 3 shows that disrupting goals via `zero_goal` or `shuffle_goal` consistently degrades MSE relative to the baseline ( $K = 1$ ), indicating that the learned flow field meaningfully incorporates goal information. The effect is strongest on tasks where the goal disambiguates the intended behavior (notably LIFT, SQUARE, and TRANSPORT), supporting the interpretation that goal-conditioning is essential for producing correct multi-step chunks.

**Image ablation (image policies only).** Finally, we probe whether the image-conditioned FGDP policy truly leverages visual inputs. Table 4 shows that zeroing images can substantially harm performance, most dramatically on SQUARE, suggesting that vision is critical for precise alignment/assembly under our chunked evaluation. Shuffling images also degrades performance on TRANSPORT, indicating sensitivity to consistent visual context. In contrast, CAN is comparatively insensitive to image perturbations under this offline metric, consistent with the low-dim state already carrying rich object and robot information in RoboMimic.

### 5.3 Qualitative Results

Comparing predicted chunks against expert sequences for  $K = 1$  versus  $K = 5$  typically reveals that the dominant benefit of additional flow steps is reduced high-frequency oscillation:  $K = 1$  often captures the correct coarse direction but exhibits jitter between consecutive timesteps, whereas  $K = 5$  yields smoother, more coherent action sequences that track expert trends over the horizon.

Ablation case studies provide additional evidence that FGDP is genuinely goal- and perception-conditioned. In goal ablations, removing or mismatching the goal produces visibly inconsistent chunk trajectories, particularly in tasks where the goal disambiguates multiple plausible futures. In the SQUARE image-based task, zeroing images should visibly disrupt the fine corrective motions needed for insertion/alignment, even when the low-dimensional state is preserved, supporting the quantitative sensitivity in Table 3 and Table 4.

## 6 Limitations and Future Work

**Limitations.** Our study focuses on offline chunk-level fidelity and inference latency as a function of the integration step count  $K$ . While this isolates the speed-quality trade-off of rectified-flow

Table 3: **Goal ablation (MSE,  $K = 1$ )**. Removing or mismatching the goal increases prediction error across tasks and modalities.

Task	Modality	Baseline	Zero-goal	Shuffle-goal
Lift	low-dim	0.1322	0.1476	0.1573
Lift	image	0.1117	0.1120	0.1188
Can	low-dim	0.1175	0.1254	0.1242
Can	image	0.1151	0.1249	0.1216
Square	low-dim	0.0879	0.0955	0.1023
Square	image	0.0876	0.1148	0.1032
Transport	low-dim	0.0598	0.0671	0.0625
Transport	image	0.0569	0.0747	0.0596

Table 4: **Image ablation (MSE, image modality,  $K = 1$ )**. Vision is crucial for SQUARE and helpful for TRANSPORT, while CAN is less sensitive under this offline evaluation.

Task	Baseline	Zero-image	Shuffle-image
Lift	0.1117	0.1232	0.1108
Can	0.1151	0.1152	0.1132
Square	0.0876	0.4120	0.0947
Transport	0.0569	0.1348	0.0809

sampling, it does not fully capture closed-loop behavior where small action errors can accumulate over time. In particular, we do not evaluate online rollout success rates in simulation, nor evaluate robustness under observation noise, distribution shift, or delayed control execution. This means that the offline improvements we observe (especially in smoothness) can only be interpreted as evidence of chunk-level coherence, not as a guarantee of improved task success under feedback control.

Another practical limitation is the absence of a Diffusion Policy baseline in our comparison. We were unable to reliably train and evaluate diffusion policies in our compute environment within the project timeframe and our limited machine resources. As a result, we cannot directly benchmark FGDP against iterative diffusion sampling, nor can we use diffusion-policy checkpoints as teachers for distillation. This prevents us from studying teacher–student compression, which would be one of the most direct ways to connect FGDP to the broader literature on fast diffusion and distillation.

**Future works.** A direct next step is online evaluation in RoboMimic/Robosuite to measure success rates, stability, and error accumulation under receding-horizon execution. This would let us connect offline chunk metrics (e.g., MSE and smoothness) to task-level outcomes, and identify when improvements in offline fidelity actually translate into better closed-loop performance. It would also allow a more rigorous study of the inference–control tradeoff: how the optimal integration step count  $K$  varies with control frequency, latency budgets, observation noise, and task dynamics (e.g., contact-rich vs. free-space motion), and whether there are cases where shorter chunks stabilize recovery while longer chunks improve sample efficiency.

Naturally, a second direction is to incorporate distillation once a suitable teacher is available. Rather than relying solely on task supervision, we can train FGDP (or a one-step flow variant) to directly match a high-quality teacher’s action-chunk distribution, using progressive step-compression schedules or consistency-style objectives. This would provide a clean way to transfer the teacher’s multimodality and fine-grained action structure into a fast policy, and would let us test whether we can retain diffusion-level behavioral richness while still preserving an explicit  $K$ -step deployment knob for trading off latency and horizon.

## References

- [1] T. Salimans and J. Ho. Progressive distillation for fast sampling of diffusion models, 2022. URL <https://arxiv.org/abs/2202.00512>.
- [2] S. Luo, Y. Tan, L. Huang, J. Li, and H. Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference, 2023. URL <https://arxiv.org/abs/2310.04378>.
- [3] X. Liu, X. Zhang, J. Ma, J. Peng, and Q. Liu. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation, 2024. URL <https://arxiv.org/abs/2309.06380>.
- [4] X. Liu, C. Gong, and Q. Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022. URL <https://arxiv.org/abs/2209.03003>.
- [5] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2024. URL <https://arxiv.org/abs/2303.04137>.
- [6] Z. Wang, Z. Li, A. Mandlekar, Z. Xu, J. Fan, Y. Narang, L. Fan, Y. Zhu, Y. Balaji, M. Zhou, M.-Y. Liu, and Y. Zeng. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *arXiv preprint arXiv:2410.21257*, 2024.
- [7] S. H. Høeg, Y. Du, and O. Egeland. Streaming diffusion policy: Fast policy synthesis with variable noise diffusion models, 2024. URL <https://arxiv.org/abs/2406.04806>.
- [8] M. Du and S. Song. Dynaguide: Steering diffusion polices with active dynamic guidance, 2025. URL <https://arxiv.org/abs/2506.13922>.