L1 Sample Flow for Efficient Visuomotor Learning

Weixi Song[1,2,3] Zhetao Chen[1,2] Tao Xu[2] Xianchao Zeng[2] Xinyu Zhou[2] Lixin Yang[2,4]
Donglin Wang[3†] Cewu Lu[2,4] Yong-Lu Li[2,4†]
[1]Zhejiang University [2]Shanghai Innovation Institute [3]Westlake University
[4]Shanghai Jiao Tong University
songweixi@zju.edu.cn yonglu_li@sjtu.edu.cn

Abstract

Denoising-based models, such as diffusion and flow matching, have been a critical component of robotic manipulation for their strong distribution-fitting and scaling capacity. Concurrently, several works have demonstrated that simple learning objectives, such as L1 regression, can achieve performance comparable to denoising-based methods on certain tasks, while offering faster convergence and inference. In this paper, we focus on how to combine the advantages of these two paradigms: retaining the ability of denoising models to capture multi-modal distributions and avoid mode collapse while achieving the efficiency of the L1 regression objective. To achieve this vision, we reformulate the original v-prediction flow matching and transform it into sample-prediction with the L1 training objective. We empirically show that the multi-modality can be expressed via a single ODE step. Thus, we propose ***L1 Flow****, a two-step sampling schedule that generates a suboptimal action sequence via a single integration step and then reconstructs the precise action sequence through a single prediction. The proposed method largely retains the advantages of flow matching while reducing the iterative neural function evaluations to merely two and mitigating the potential performance degradation associated with direct sample regression. We evaluate our method with varying baselines and benchmarks, including 8 tasks in MimicGen, 5 tasks in RoboMimic & PushT Bench, and one task in the real-world scenario. The results show the advantages of the proposed method with regard to training efficiency, inference speed, and overall performance. Project Website.*

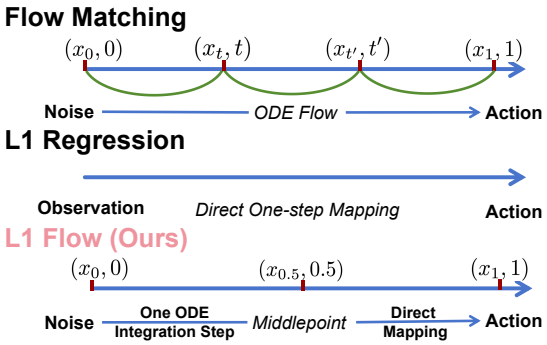


Figure 1. **Overview of the proposed method. L1 Flow** employs a 2-step denoising paradigm which combines the efficiency of L1 regression and the strong distribution-modeling capacity of standard flow matching. Compared with the iterative denoising process of the standard flow matching and the direct mapping of L1 regression, L1 Flow decouples the modeling of multi-modal distribution and the reconstruction of the precise actions. Starting from a random noise, L1 Flow performs one integration step towards the middle timestep and predicts the precise action x_1 from the coarse action $x_{0.5}$, which are based on the reformulated sample-prediction type flow matching.

1. Introduction

Learning from demonstrations to map the observations to actions formulates the basic paradigm of visuomotor policy learning, casting the cloning of expert behavior as supervised learning. Denoising-based methods, like diffusion and flow matching, play a significant role in robot manipulation imitation learning due to their powerful distribution modeling capabilities, with applications ranging from lightweight visuomotor policies[2, 29, 30, 37, 38] to large scale Visual-Language-Action models[1, 7, 14, 22, 28, 34–36]. However, the multi-timestep noise prediction in training and the iterative denoising process during inference lead to slow training convergence and high inference latency, respectively, which pose a challenge for their practical application in robotics tasks. To accelerate the inference process, Wang et al. [33] distills the diffusion policy into a one-step action generator by minimizing the KL divergence along the diffusion chain, which introduces additional post-training cost after the initial training of the diffusion policy. Non-

†Corresponding author.

denoising approaches[3, 16, 21, 39] have long been proposed for visuomotor policy learning, which map observations directly to actions through simple regression, allowing for efficient policy training. These methods were once considered inferior to denoising-based approaches due to their limited ability to capture complex, especially multi-modal, action distributions. However, with appropriate architectural modifications[5, 26], the direct action regression objective can achieve performance comparable to denoising-based methods. Recently, Kim et al. [11] demonstrates that simple learning objectives, such as L1 regression, can achieve performance comparable to diffusion-based methods on certain tasks, but Kim et al. [11] also points out that it may struggle to capture the multi-modality in human demonstrations.

The deterministic sampling, *e.g.*, DDIM[23] and flow matching[13], are commonly used to accelerate the generation of action sequences for real-time inference, which means the model's ability to capture multi-modal behaviors is derived solely from the stochastic sampling of the initial noise. Therefore, we are motivated to model the multi-modality via the minimal integration step and reconstruct the precise action sequences through the direct L1 regression conditioned on the integral result, which combines the strong distributional expressiveness of denoising models with the efficient training and inference features of L1 regression. To achieve this, we reformulate the original v-prediction form of flow matching into an equivalent sample-prediction formulation, enabling the direct regression on the target samples. Based on this sample-prediction variant, we introduce a two-step sampling procedure. In the first step, a noisy sample drawn at $t = 0$ is transformed into a coarse sample by converting the model's prediction into the corresponding velocity of the original flow matching and performing a single-step ODE integration. Empirically, we find that this single integration step is sufficient to capture the modality of the target distribution as shown in Figure 2. In the second step, the model directly predicts the final clean sample conditioned on the coarse sample, resulting in efficient reconstructions that preserve the multimodality of the data distribution.

We first compare the proposed two-step sampling strategy with 2 standard denoising-based methods *i.e*. DDPM (100 steps) and Flow Matching (10 steps), and L1 regression in 8 tasks of MimicGen Benchmark. The results (Figure 4) and Table 2) exhibit the better training efficiency and comparable overall performance of L1 Flow, though it is sampled with much fewer times. Then, as an efficient paradigm aiming at speeding up, L1 Flow is compared to distillation-based methods *i.e*. Consistency Policy (CP)[19] and OneDP[33] in total 5 tasks of Robomimic and PushT Bench. In contrast to the pretraining-distillation training of distillation-based methods, we apply end-to-end direct training, achieving faster training while outperforming in most tasks. Further, we establish a 2-stage task in real-world scenarios, mainly focusing on multi-modality and prediction precision. The results show the advantage of L1 Flow in overall performance and about $10\times$ to $70\times$ faster inference speed than the common diffusion policy.

In conclusion, the main contributions are as follows:

- We reformulate the origin velocity prediction flow matching to sample prediction and empirically show that the multi-modality can be expressed via a single integration step.
- Based on the reformulated sample-prediction flow matching, we propose an efficient two-step denoising strategy to reconstruct the action sequence via one-step integration and direct sample prediction and validate its effectiveness in both simulation and real-world scenarios.

2. Related Work

Denoising-based Models in Robotics Manipulation

Recent advances have shown that denoising models can serve as powerful policies for robot manipulation, including imitation learning[1, 2, 14, 28, 29, 36, 37] and reinforcement learning[18, 20, 27, 32], due to their strong distribution mapping capacity. Diffusion Policy[2] formulates visuomotor control as conditional denoising of action sequences, enabling robust and multi-modal policy learning across diverse tasks, with extensions like DP3[37] incorporating the 3D visual representations into diffusion policies and EquiDP[29] considering the domain symmetries in manipulation. Subsequent works [14, 34–36] scale this paradigm to large pre-trained models, achieving strong generalization across varied robot embodiment and deployment scenes. More recently, flow-matching[13, 15] approaches have been widely applied to large robotics foundation models[1, 7, 22, 28] for faster, deterministic sampling, but they still need about 10 NFE.

To ease the time-consuming iterative denoising process, distillation-based methods [19, 33] accelerate the sampling by training one-/ few-step student models with pre-trained teacher models as prior. Consistency Policy(CP)[19] adapts the Consistency Model frameworks[10, 24, 25] and achieves faster inference speed while maintaining comparable performances against the baseline. To address slow convergence and occasional performance degradation in CP, OneDP[33] distills the diffusion policy into a one-step action generator by minimizing the KL divergence along the diffusion chain. Distillation-based methods introduces additional post-training cost after the initial training. For end-to-end training, DM1[40] applies MeanFlow[4], a recent one-step generation framework, to achieve one-step generation. However, it also brings an additional training burden in the computation of Jacobian–Vector Product (JVP) bringing

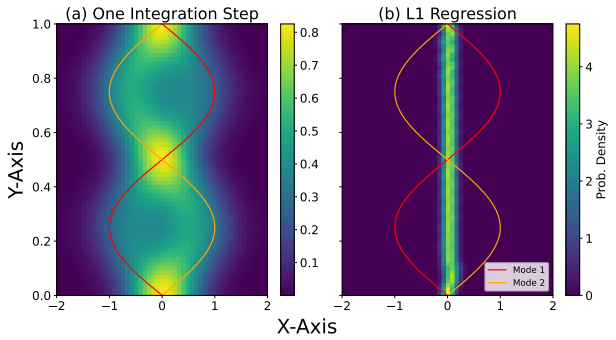


Figure 2. **Visualization of the sample distribution.** Apply the proposed one-step integration to model two sine curves with different phases and compare with L1 Regression. **(a)** One-step integration effectively captures the multi-modality. **(b)** The direct regression exhibits the average of two modes, the so-called mode collapse.

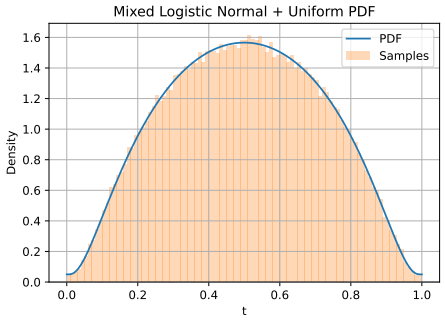


Figure 3. **PDF of the mixed distribution of Logistic Normal and Uniform distribution.** The Logistic Normal distribution emphasizes sampling around intermediate timesteps, while we additionally incorporate a low-level uniform distribution to ensure that the probabilities at the boundary timesteps remain non-zero.

more 15% overhead.

Non-Denoising Models in Robotics Manipulation

There has long been a series of non-denoising methods for learning visuomotor policy[3, 16, 21, 39], which directly map observations to actions via simple regression, achieving efficient policy training. For a period of time, such alternatives were considered inferior to denoising-based methods for their poor capacity to model complex, especially, multi-modal distributions. Several architectural modifications[5, 26] *e.g.* auto-regressive multi-scale prediction, together with the direct action regression objective, can achieve competitive performance against denoising-based approaches. However, OpenVLA-OFT[11] points out that using L1 regression with parallel decoding can achieve performance comparable to diffusion-based methods on certain tasks, and VLA-Adapter[31] also achieves state-of-the-art results on various benchmarks with L1 training objective. But the fact is that L1 regression is hard to capture the multi-modality in human demonstrations and is easy to encountering mode collapse when trained with similar inputs but largely different outputs. This work aims to maintain the training and inference efficiency of simple regression while including the strong distribution-mapping capacity of denoising-based methods. In the following section, we establish a sample prediction type of flow matching and introduce L1 Flow to bridge simple regression and multi-modality modeling via two-step denoising.

3. L1 Sample Flow

3.1. Preliminary

Flow matching [13] formulates generative modeling as learning a continuous-time deterministic flow that transports a simple base distribution $p(x_0)$, *e.g.* a Gaussian distribution, toward a complex target distribution $p(x_1)$. The goal of flow matching is to learn a velocity field $v_\theta(x_t, t)$ such that the solution of the following ordinary differential equation (ODE) satisfies $x_1 \sim p(x_1)$:

$$\frac{dx_t}{dt} = v_\theta(x_t, t). \tag{1}$$

A common flow choice is the linear interpolation path [15]:

$$x_t = (1-t)x_0 + tx_1, t \in [0, 1], \tag{2}$$

which yields the ground-truth velocity:

$$v_\theta(x_t, t) = \frac{dx_t}{dt} = x_1 - x_0. \tag{3}$$

The training objective minimizes the discrepancy between the predicted flow and the target flow:

$$\mathcal{L}(\theta) = \mathbb{E}_{x_0, x_1, t} ||f_\theta(x_t, t) - (x_1 - x_0)||^2. \tag{4}$$

Samples are generated by integrating the learned flow field from $t = 0$ to $t = 1$, starting from random noise:

$$x_1 = x_0 + \int_0^1 f_\theta(x_t, t) dt. \tag{5}$$

3.2. Methodology

We consider a variant of Flow Matching where the model directly predicts the terminal sample rather than the instantaneous velocity. Instead of learning the instantaneous velocity field $x_1 - x_0$, the model $f_\theta(x_t, t)$ predicts the corresponding terminal sample x_1 conditioned on the intermediate state x_t and time t:

$$f_\theta(x_t, t) = \frac{dx_t}{dt} = x_1 - x_0 \Rightarrow x_1. \tag{6}$$

The instantaneous velocity can then be implicitly recovered as:

$$\begin{aligned} v = x_1 - x_0 &= \frac{(1-t)x_1 - (1-t)x_0}{1-t} \\ &= \frac{x_1 - ((1-t)x_0 + tx_1)}{1-t} \\ &= \frac{f_\theta(x_t, t) - x_t}{1-t}. \end{aligned} \tag{7}$$

This yields a sample-prediction flow whose dynamics are defined by the ODE:

$$\frac{dx}{dt} = \frac{f_\theta(x_t, t) - x_t}{1-t}. \tag{8}$$

At training time, the model is optimized to minimize the discrepancy between the predicted terminal sample x_{pred} and the true target sample x_1. Similar to [11, 31], we also employ L1 loss to supervise the target samples:

$$\mathcal{L} = \mathbb{E}_{x_0 \sim \mathcal{N}(0,1), x_1 \sim data, t} \, ||f_\theta(x_t, t) - x_1||_1. \tag{9}$$

Empirically, we observe that under this setting, the L1 loss performs better than the MSE loss, which is commonly used in standard flow matching to supervise the velocity field. A further comparison is included in the ablation study in Section 4.4.

During inference, we introduce a two-step denoising schedule that combines continuous flow integration with direct sample prediction. Starting from a pure noise initialization $x_0 \sim \mathcal{N}(0, 1)$, we first integrate the learned flow field from $t = 0$ to an intermediate time $t = 0.5$:

$$x_{0.5} = x_0 + \frac{f_\theta(x_0, 0) - x_0}{2}. \tag{10}$$

This partial integration allows the model to evolve the noisy sample toward an intermediate representation $x_{0.5}$ that captures the coarse structural and modality information about the target distribution. At the midpoint, instead of continuing the ODE integration until $t = 1$, we directly invoke the model's sample prediction capability to obtain the final output:

$$x_1 = f_\theta(x_{0.5}, 0.5). \tag{11}$$

We leverage the multi-modal capture ability of flow matching to predict a coarse sample at mid-timestep while utilizing the efficiency of simple sample prediction that avoids time-consuming integration. Note that our two-step schedule is mathematically equivalent to the standard 2-step integration. The numerical errors in the calculation of the velocity bring the performance gap(see ablation study in Section 4.4.4). Therefore, we apply the current 2-step version. We apply the proposed sample-prediction flow matching to model two sine curves with different phases (simulating different trajectories of human demonstrations under the same observations). Samples are generated via a single-step integration, and the resulting sample distribution is visualized as the heatmap shown in Figure 2(a). For comparison, we also perform the direct L1 regression on the curves, where Gaussian noise is used as input to enable one-to-many mapping in the regression. The results demonstrate that our one-step integration effectively captures the multi-modality of the distribution, while the direct regression exhibits the average of two modes, the so-called mode collapse.

Algorithm 1 L1 Sample Flow Training

Require: Dataset of pairs of observations and trajectories $\mathcal{D} = \{(o, x_1)\}$, model f_θ

while not converged **do**
  Sample a batch of pairs of observations and trajectories $(o, x_1) \sim \mathcal{D}$
  Sample a batch of noise $x_0 \sim \mathcal{N}(0, 1)$
  Sample t by Equation(12)
  Compute interpolated state: $x_t = (1-t)x_0 + tx_1$
  Predict terminal sample: $\hat{x}_1 = f_\theta(x_t, t, o)$
  Compute L1 loss: $\mathcal{L} = \|\hat{x}_1 - x_1\|_1$
  Update $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
end while

Algorithm 2 L1 Sample Flow Inference

Require: model f_θ, observation o

Sample noise $x_0 \sim \mathcal{N}(0, 1)$
$x_{pred} = f_\theta(x_0, 0, o)$
$x_{0.5} = x_0 + \frac{1}{2}(f_\theta(x_0, 0) - x_0)$ ▷ One-step Integration
$x_1 = f_\theta(x_{0.5}, 0.5)$ ▷ Direct Sample Prediction

With coarse samples, the subsequent prediction is to accurately reconstruct samples from different modalities. To further enhance the model's capability of predicting samples at the intermediate timestep, we sample the timestep t from the mixed distribution of logistic-normal distribution and the uniform one:

$$t = \begin{cases} \frac{1}{1+e^{-x}}, x \sim \mathcal{N}(0, 1), & p = 1 - \alpha, \\ x, x \sim U(0, 1), & \text{otherwise.} \end{cases} \tag{12}$$

While increasing the sampling probability at intermediate time steps, the sampling probabilities at the boundary timesteps are maintained at appropriately low levels, as shown in the visualization in Figure 3. As a result, the model develops stronger representational capacity around intermediate timesteps, which directly benefits the midpoint-prediction sampling procedure described above. We summarize the training and inference outlines of our proposed method in Algorithm.1 and 2.

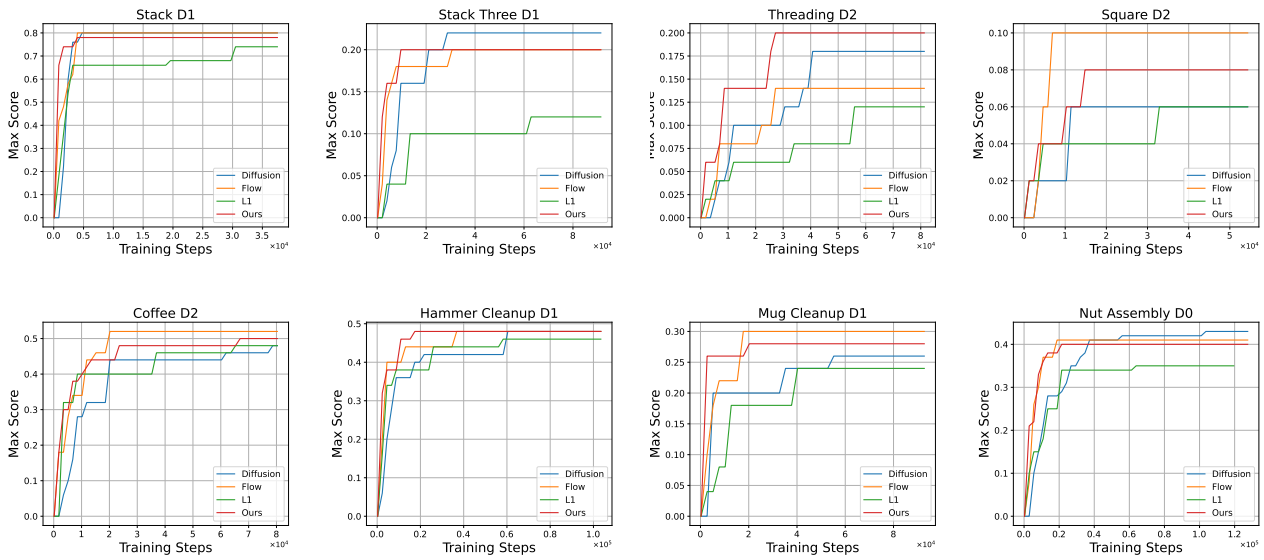


Figure 4. **The trend of the maximum success rate throughout the training.** We experiment with our method with baselines in the 8 tasks of MimicGen and report the maximum success rate among 50 evaluations throughout the training. The overall results demonstrate that our method achieves performance comparable to or even surpassing the baselines with only two neural function evaluations (NFE), while also exhibiting higher training efficiency by reaching performance saturation in success rate with fewer training steps. L1 Flow largely retains the advantages of flow matching with less inference budget while mitigating the potential performance degradation associated with direct L1 regression.

4. Experiment

In this section, we comprehensively evaluate the proposed method with varying baselines in both simulation and real-world settings. The analysis focuses mainly on the following aspects:

- Compared with standard flow matching, diffusion, and L1 regression, does our method outperform in terms of training efficiency, inference efficiency, and performance?
- Compared with related accelerated denoising-based methods *e.g.*, One-DP[33] and Consistency Policy[19], does our approach demonstrate advantages?
- When applied to real-world scenarios, how does our approach remain effective and benefit from faster inference?
- What are the contributions of each component in our method, and how does our approach connect to standard flow matching?

4.1. Comparison with Standard Methods

To verify the effectiveness of the proposed method, we first evaluate our method with several standard baselines, including 2 denoising-based methods and direct L1 regression: **Diffusion**: We follow the initial setting of Diffusion Policy[2] and apply DDPM sampling with 100 denoising steps. **Flow Matching**: We follow the implementation details of flow matching described in [1]. Compared with the standard flow matching approach, [1] replaces the uniform sampling distribution of timesteps with a Beta distribution, which emphasizes lower (noisier) timesteps sampling. We adopt the same design and likewise use 10 integration steps in our implementation. **L1 Regression**: We adapt the L1 regression objectives to directly supervise the ground truth actions without other designs.

Following the setting in [2], all baselines share the same non-pretrained ResNet-18[6] visual encoder and 1-D Conditional Temporal UNet [8] architecture (except for L1 objective, which doesn't take the timestep as condition), while employing different objectives. We build the evaluation on 8 manipulation tasks in MimicGen [17]. We train all methods with the same 100 expert trajectories in relative control mode and report the best success rate until the current evaluation. The success rate of each evaluation is averaged over 50 different runs with varied random seeds. Figure 4 shows the trend of the best success rate over steps throughout the training process and Table 2 reports the overall results. The results indicate that flow matching exhibits better convergence efficiency and overall performance than diffusion across most tasks. Although L1 regression performs comparably to denoising-based methods in some cases and offers higher inference efficiency, it often suffers from significant performance degradation. L1 Flow achieves superior convergence efficiency and maintains performance comparable to or even surpassing of the majority of tasks with

Table 1. **Performance comparison with fast denoising baselines on RoboMimic.** We compare our method **L1 Flow (Ours)** against representative fast denoising baselines, including full Diffusion Policies (DP) trained under DDPM, DDIM, and EDM schedulings, as well as distillation-based approaches—Consistency Policy (CP) and OneDP. We report the mean and standard deviation of success rates over five independent training runs, each evaluated under 100 randomized environment initializations (500 trials in total). Results marked with * are taken from [33] and the best results are marked in **bold**. As shown in the table, **L1 Flow** achieves the best average success rates, converging about **5-7× faster** than the baselines while requiring only two inference steps (NFE = 2).

METHODS	EPOCHS	NFE	PUSHT	SQUARE-MH	SQUARE-PH	TOOLHANG-PH	TRANSPORT-PH	AVG.
DP (DDPM)*	1000	100	0.863±0.040	0.846±0.023	0.926±0.023	0.822±0.016	0.896±0.032	0.871
DP (DDIM)*	1000	10	0.823±0.023	0.850±0.013	0.918±0.009	0.828±0.016	0.908±0.011	0.865
	1000	1	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000
DP (EDM)*	1000	35	0.861±0.030	0.810±0.026	0.898±0.033	0.828±0.019	0.890±0.012	0.857
	1000	19	0.851±0.012	0.828±0.015	0.880±0.014	0.794±0.012	0.860±0.013	0.843
	1000	1	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000±0.000	0.000
CP (EDM)*	1450	3	0.839±0.037	0.710±0.018	0.874±0.022	0.626±0.041	0.848±0.028	0.779
	1450	1	0.828±0.055	0.646±0.047	0.776±0.055	0.650±0.046	0.754±0.120	0.731
ONEDP-D (EDM)*	1020	1	0.829±0.052	0.776±0.023	0.902±0.040	0.762±0.056	0.898±0.019	0.833
ONEDP-S (EDM)*	1020	1	0.841±0.042	0.774±0.003	0.910±0.041	0.824±0.039	0.910±0.027	0.852
ONEDP-D (DDPM)*	1020	1	0.802±0.057	0.846±0.028	0.926±0.011	0.808±0.046	0.896±0.013	0.856
ONEDP-S (DDPM)*	1020	1	0.816±0.058	0.864±0.042	0.926±0.018	**0.850±0.033**	0.914±0.021	0.874
L1 FLOW (OURS)	200	2	**0.869±0.007**	**0.868±0.018**	**0.948±0.013**	0.788±0.029	**0.932±0.013**	**0.881**

Table 2. **Overall results of the comparison with the standard methods.** Results show our method largely retains the advantages of flow matching, *i.e.* fast convergence and better performance, with much fewer neural function evaluations (NFE), while mitigating the potential performance degradation associated with direct L1 regression.

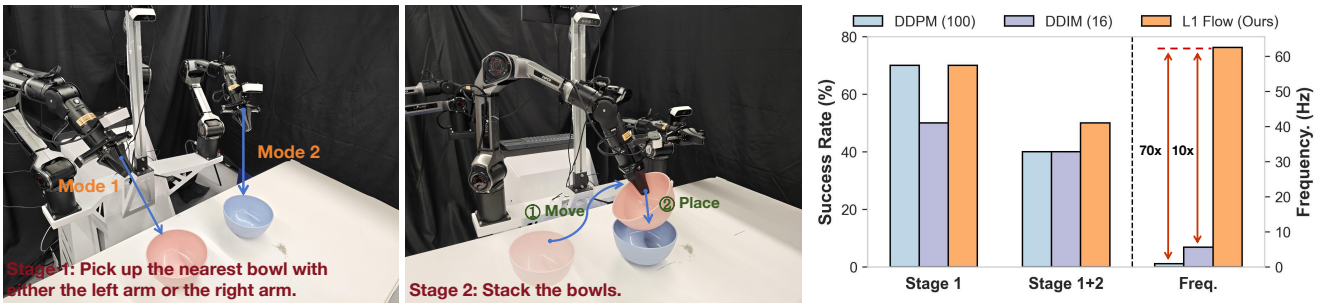
TASK	DP	FLOW	L1	OURS
STACK D1	**0.80**	**0.80**	0.74	0.78
STACK THREE D1	**0.22**	0.20	0.14	0.20
THREADING D2	0.18	0.14	0.12	**0.20**
SQUARE D2	0.06	**0.10**	0.06	0.08
COFFEE D2	0.48	**0.52**	0.48	0.50
HAMMER CLEANUP D1	**0.48**	**0.48**	0.46	**0.48**
MUG CLEANUP D1	0.26	**0.30**	0.24	0.28
NUT ASSEMBLY D0	0.43	0.41	0.35	0.40
NFE	100	10	1	2
AVERAGE.	0.364	**0.369**	0.324	0.365

much less neural function evaluations (NFE), while also demonstrating a consistent performance advantage over L1 regression. The performance comparison with DDPM (100 steps) shows that our method largely retains the advantages of flow matching, while mitigating the potential performance degradation associated with direct L1 regression.

4.2. Comparison with Other Fast Denoising Models

To further evaluate the performance and efficiency of our approach, we compare it with two representative fast denoising baselines, **Consistency Policy (CP)** [19] and **OneDP** [33]. Both CP and OneDP are distillation-based methods that rely on a pretrained diffusion model. Specifically, CP depends on the boundary conditions of the EDM[9] noise scheduling, whereas OneDP is compatible with both DDPM and EDM schedulers. Therefore, diffusion policies with different schedulers (DDPM, DDIM, and EDM) are also included as baselines for comparison. In addition, OneDP includes two variants: a stochastic version (**OneDP-S**) and a deterministic version (**OneDP-D**). OneDP-S employs an auxiliary network to estimate the score of the generator distribution, enhancing the performance of the one-step policy in complex environments at the cost of a more computationally expensive training process. Conversely, OneDP-D removes the generator score network by directly optimizing a simplified score loss, yielding a deterministic observation-to-action policy. These two variants are included in the evaluation.

Following the setting in OneDP, we conduct the evaluations on two established benchmarks, RoboMimic [16] and PushT [3]. We evaluate 5 challenging tasks in Robomimic with high-quality human demonstrations: Square-mh,



(a) Stage 1 (b) Stage 2 (c) Overall Results

Figure 5. **The real-world evaluation setup.** The evaluation is conducted in AGILEX Mobile Aloha platform, and we establish a task with two stages, emphasizing modeling the action multi-modality and prediction precision. **(a) Stage 1:** The robot is required to pick up the bowl with either the left or the right arm, requiring the policy to model the multi-modal action distribution. **(b) Stage 2:** The dual-arm is required to move the grasped bowl on top of the other bowls and place it, emphasizing the transformation from multi-modal to single-modal action prediction and also demands action precision, requiring the policy to accurately localize the bowl's edge. **(c) Performance comparison with the standard diffusion policy**, including DDPM (100 steps) and DDIM (16 steps) in real-world task. The results show that L1 Flow achieves comparable performance against the baselines and has a superior inference speed, achieving about **10-70× speed-up**.

Square-ph, ToolHang-ph, and Transport-ph, where "ph" denotes proficient human demonstrations and "mh" denotes mixed proficient/non-proficient human demonstrations. For PushT, we adopt the standard setting with 200 expert trajectories and use the RGB observations as input. During the evaluation, we report the peak success rate averaged over five different training runs and 100 random environment initializations (500 trials per task). The RoboMimic task metric is measured by binary success rates, while PushT is evaluated using goal-region coverage ranging from 0 to 1. The baseline results are taken from the original comparison in OneDP[33] and marked with * in Table 1.

The baseline Diffusion Policy (DP) is trained for 1,000 epochs and sampled under DDPM, DDIM, and EDM noise scheduling. Both distillation-based methods require additional training stages: OneDP performs an extra 20 epochs for distillation, while CP requires around 450 additional epochs to converge. Accordingly, for distillation-based methods, the reported number of epochs includes both the pretraining epochs and the additional epochs required for the distillation. For comparison, our method is a single-stage approach trained from scratch with only 200 epochs.

In Table 1, our method achieves the highest success rates, outperforming the standard diffusion policies, CP, and OneDP in most tasks except ToolHang-ph. Although it shows degradation towards the initial DP in ToolHang-ph, it still outperforms CP and OneDP-D (EDM). In particular, it achieves strong performance with only two denoising steps (NFE=2) and only requires 200 epochs to converge, which is about 5× to 7× faster than the baselines.

4.3. Real World Experiments

In this section, we study the multi-modal action in real-world scenarios to evaluate the proposed method. Beyond the confirmed improvement (only 2 NFE) in inference latency, which is crucial for real-world deployment, we are also interested in whether this training objective offers other advantages or drawbacks in other aspects of the real deployment. Therefore, the task design primarily focuses on two topics: multi-modality modeling and precision single modality prediction.

Platform. Our experiments are conducted on the AGILEX ALOHA dual-arm platform, which includes two 6-DoF PIPER arms with grippers, two wrists, and one front-facing Orbbec DABAI depth camera. All policies are executed on a PC workstation equipped with an NVIDIA RTX 4090 GPU and an Intel I7-13700 CPU.

Task Definition. The evaluation includes one task with 2 stages as follows. **Stage 1: Pick up the Bowl.** This task requires the dual-arm robot to use either left or right arm to pick up the nearest bowl on the table, *i.e.* pick up the left bowl with the left arm or pick up the right bowls with arm arm. The provided demonstrations include an equal number of trajectories performed with the left and right arms, highlighting the policy's capability to model multi-modal action distributions under the same observation. **Stage 2: Stack the Bowl.** The dual-arm robot is required to place the grasped bowl on the other bowl on the table. This stage highlights the common single modality modeling and the prediction precision, so as to avoid the accumulation of errors during execution. The overview of the task scene and setup is shown in Figure 5a and 5b.

Details. We collect 50 episodes via teleoperation for the task and record the data at 25 Hz. We keep most of the policy configurations used in the simulation and specifically make several modifications as follows: (1) input RGB images are resized to 224×224 and without random crop, (2) action/proprioceptive state is defined in the 6D absolute

Table 3. **Ablation study on key components of the proposed method.** Results show the contributions of the training objectives and timestep sampling strategy to the overall performance. We also compare the performance across the initial flow matching with different integration steps. The best results are marked in **bold** and the second best is underlined.

METHOD	NFE	PUSHT	SQUARE-PH
MSE	2	0.821±0.012 (-0.048)	0.916±0.009 (-0.032)
UNIFORM	2	0.854±0.033 (-0.015)	0.928±0.011 (-0.020)
BETA	2	0.857±0.013 (-0.012)	0.940±0.012 (-0.008)
FLOW	1	0.755±0.023 (-0.114)	0.928±0.013 (-0.020)
FLOW	2	0.777±0.008 (-0.092)	**0.964±0.005** (+0.016)
FLOW	10	0.812±0.015 (-0.057)	0.946±0.011 (-0.002)
FLOW	100	0.815±0.010 (-0.054)	0.932±0.004 (-0.016)
OURS	2	**0.869±0.007**	0.948±0.013

joint-angle and 1D gripper space, (3) prediction action horizon is 32, and the execution action horizon is 16, (4) vision encoder is replaced by the pretrained ResNet18[6] for better generalization in the real world. We train the diffusion policy for 200 epochs, while the L1 flow is trained for only 50 epochs for its fast convergence.

Results. Each result is obtained by averaging the results of 10 executions. Figure 5c shows that L1 Flow achieves comparable performance against the baselines, showcasing the potential of modeling multi-modal action distribution via this paradigm. We measure the computing time required for each method to generate one action chunk and convert it to the chunk frequency as shown in Figure 5c. Taking advantage of its 2-step denoising strategy and the simplification in each flow step compared with diffusion, L1 Flow achieves extremely fast inference speed, approximately 70× faster than the original 100-step DDPM, and about 10× faster than the accelerated 16-step DDIM.

4.4. Ablation and Discussion

To further analyze the contribution of each component in our proposed framework, we conduct a comprehensive ablation study on two representative tasks, **PushT** and **Square-ph**, in Section 4.2. All experiments are trained and evaluated in a consistent setting. Two key components are analyzed: **(1)** The impact of different regression objectives (L1 *vs*. MSE) and **(2)** The effect of different timestep sampling strategies (Uniform *vs*. Beta *vs*. Logistic Normal Mixture). In addition, we also include the discussion of **(3)** the comparison between the standard flow matching with different integration steps and our proposed 2-step L1 Flow. The results are summarized in Table 3.

4.4.1. L1 vs MSE Loss

We first compare the L1 and MSE training objective under identical configurations. As the L1 training objective is increasingly adopted by large vision-language-action models[11, 31] for direct action regression, we explore replacing the commonly used MSE loss for noise or velocity regression in denoising-based methods with L1 loss to supervise the ground-truth samples in our sample-prediction flow matching. As shown in Table 3, the L1 objective yields consistently higher success rates on both tasks. The empirical results shown in [11, 31], along with our comparison, suggest the advantages of the L1 objective in supervising the ground-truth sample. However, it is still unclear where this superiority stems from. One of the possible future directions will focus on delving into the distributional difference between sample prediction and noise-like prediction, which could offer guidance for choosing more appropriate learning objectives rather than empirically.

4.4.2. Uniform vs Beta vs Mixed Logistic Normal

To investigate the effect of the timestep sampling strategy, we compare the original *Uniform distribution* in flow matching and *Beta distribution* used in [1] with our applied *Mixed Logistic Normal Distribution* (Equation 12). Specifically, the *Uniform distribution* samples timesteps evenly across the entire range, the *Beta strategy* places greater emphasis on high-noise level, and our *Mixed Logistic Normal* prioritizes sampling around the intermediate timesteps. The results show that replacing the *Mixed Logistic Normal* with the other two leads to varying degrees of performance degradation on both tasks. The variation indicates that emphasizing intermediate timesteps can benefit our two-step sampling.

4.4.3. L1 Flow vs Standard Flow Matching

We compare our method with standard flow matching under varying inference budgets. In **PushT**, 1-step flow matching exhibits non-trivial performance compared with 1 step diffusion shown in Table 1, and there are performance gains with increasing number of integration steps. In **Square-ph**, standard flow matching gains the best success rate in a 2-step setting and shows degradation as the integration step number increases, which is likely due to the accumulation error. This indicates that the optimal inference budget for flow matching varies widely between different task settings. In comparison, our method exhibits highly competitive performance in both tasks with fixed 2-step sampling. Therefore, from another perspective, our method can be viewed as a stable alternative that avoids the need for dynamically selecting the number of integration steps based on the task while simultaneously addressing the performance degradation caused by a small number of steps and the accumulated error arising from a large number of steps.

4.4.4. Timestep Strategy

We examines the empirical optimality of targeting the temporal midpoint in the first flow integration. In our two-step denoising schedule, inference consists of two stages: the

first is to perform one step integration to a selective time point t_{first}, followed by direct model prediction at t_{first} to obtain the final sample x_1. We sweep $t_{\text{first}} \in [0.1, 0.9]$ to empirically verify whether the temporal midpoint ($t_{\text{first}} = 0.5$) indeed yields optimal performance. The second study analyzes the efficiency of our 2-step sample approach relative to the standard integration inference paradigm with our sample prediction flow matching, which integrates with different step sizes *i.e.* NFE $\in \{1, 2, 5, 10\}$. All models are trained using the default L1 Flow configuration on the PushT task.

Results in Table 4 reveal two insights:

(1) Our two-step inference achieves best performance at $t_{\text{first}} = 0.5$, confirming the empirical optimality of the temporal midpoint in our method. Notably, it outperforms all standard multi-step baselines, demonstrating that our two-step design eliminates redundant integration steps while preserving accuracy. This validates that our strategy efficiently captures optimal flow behavior with minimal computational overhead.

(2) Among standard baselines, NFE=2 yields the highest success rate (0.863 ± 0.008), closely approaching our method's best (0.869 ± 0.007). This is expected, the standard NFE=2 integration is mathematically equivalent to our two-step schedule under exact ODE solving. The minor performance gap ($\sim$0.006) therefore arises primarily from numerical integration errors in the calculation of the velocity, whereas our approach bypasses intermediate integration and directly predicts x_1.

4.4.5. Loss Formulation

Recently, [12] has also applied the sample-prediction variant of flow matching in the image generation domain, while adopting to regress velocity as the final objective. To examine how different loss formulations affect learning dynamics and downstream control performance, we perform ablation studies comparing velocity-space loss (v-loss) and sample-space loss (x-loss) under identical experimental settings.

The sample-preditcion model directly predicts the target sample x_1 and the sample-space loss (x-loss) is naturally obtained as

$$\mathcal{L}_x = \mathbb{E}_{t,x_0,x_1} \left\| f_\theta(x_t, t) - x_1 \right\| \tag{13}$$

Following the notation in Section 3.2, we denote x_0 as Gaussian noise, x_1 as the ground-truth target sample, and define the linear interpolation path $x_t = tx_1 + (1-t)x_0$. Along this trajectory, the ground-truth velocity field is as follows:

$$v_{\text{gt}} = x_1 - x_0 = \frac{x_1 - x_t}{1 - t}. \tag{14}$$

Because the model predicts x_1 directly (denoted as $f_\theta(x_t, t)$), the induced velocity prediction becomes

Table 4. **Ablation study on timestep strategy.** All models are trained using the default L1 Flow configuration on the **PushT** task. "Ours" means using our proposed two-step denoising schedule, while "Standard" refers to the conventional multi-step Flow Matching inference method. We report the mean and standard deviation of success rates over five independent training runs, each evaluated under 100 randomized environment initializations (500 trials in total). The best results are marked in **bold** and the second best is underlined.

Strategy	t_{first}	NFE	PushT
Ours	0.1	2	0.846 ± 0.022
	0.2		0.853 ± 0.016
	0.3		0.843 ± 0.009
	0.4		0.843 ± 0.020
	0.5		$\mathbf{0.869 \pm 0.007}$
	0.6		0.855 ± 0.012
	0.7		0.855 ± 0.018
	0.8		0.849 ± 0.019
	0.9		0.848 ± 0.017
Standard	—	1	0.845 ± 0.017
	—	2	$\underline{0.863 \pm 0.008}$
	—	5	0.847 ± 0.016
	—	10	0.856 ± 0.007

$$v_{\text{pred}} = \frac{f_\theta(x_t, t) - x_t}{1 - t}. \tag{15}$$

The velocity-space loss is thus given by the discrepancy between predicted and ground-truth velocities:

$$\begin{aligned} \mathcal{L}_v &= \mathbb{E}_{t,x_0,x_1} \left\| v_{\text{pred}} - v_{\text{gt}} \right\| \\ &= \mathbb{E}_{t,x_0,x_1} \left\| \frac{f_\theta(x_t, t) - x_1}{1 - t} \right\|. \end{aligned} \tag{16}$$

This derivation reveals that $\mathcal{L}_v$ is essentially a time-dependent reweighting of $\mathcal{L}_x$, where the factor $\frac{1}{1-t}$ increasingly amplifies prediction errors as $t \to 1$, *i.e.*, the low-level noisy sample.

All experiments follow the configuration described in Section 4.4.4. We evaluate four loss variants, combining velocity-space loss (v-loss) and sample-space loss (x-loss) with either L1 Loss or Mean Squared Error (MSE). Furthermore, we conduct comprehensive evaluations across multiple inference strategies mentioned in Section 4.4.4, and the full results are summarized in Table 5.

Experiments on PushT yield three main observations:

(1) **L1 loss consistently outperform their MSE counterparts**. It is still unclear where this superiority stems from, but we recommend L1 loss as the default loss for sample prediction flow matching. MSE remains a viable alternative when smooth gradient signals are prioritized, though

Table 5. **Ablation study on loss type.** (v-loss vs. x-loss) under the same configuration as Table 4, where "2-step" adopts $t_{\text{first}} = 0.5$. "v-loss" and "x-loss" denote losses computed in velocity space and action(x) space, respectively, while both targeting action prediction. For each loss type, the best results are marked in **bold** and the second best is underlined.

Loss Type	Strategy	NFE	PushT
x-loss (MSE)	Standard	1	0.842 ± 0.014
	Standard	2	0.817 ± 0.014
	Standard	5	0.836 ± 0.013
	Standard	10	**0.843 ± 0.013**
	2-step	2	0.821 ± 0.012
v-loss (MSE)	Standard	1	**0.851 ± 0.023**
	Standard	2	0.832 ± 0.031
	Standard	5	0.844 ± 0.010
	Standard	10	0.850 ± 0.020
	2-step	2	0.847 ± 0.017
x-loss (L1)	Standard	1	0.845 ± 0.017
	Standard	2	0.863 ± 0.008
	Standard	5	0.847 ± 0.016
	Standard	10	0.856 ± 0.007
	2-step (Ours)	2	**0.869 ± 0.007**
v-loss (L1)	Standard	1	0.865 ± 0.012
	Standard	2	0.865 ± 0.019
	Standard	5	0.870 ± 0.014
	Standard	10	**0.872 ± 0.018**
	2-step	2	0.868 ± 0.013

it generally yields slightly lower robustness in our experiments.

(2) **The v-loss generally achieves higher best performance than x-loss.** Specifically, v-loss (L1) attains the highest overall success rate (0.872 at NFE=10), surpassing our method, x-loss (L1) with 2-step inference (0.869), by a marginal +0.003. Nevertheless, our approach offers improved inference efficiency (5× fewer function evaluations), illustrating a clear trade-off between maximum accuracy and practical deployment speed.

(3) **The one-step prediction of sample-prediction flow matching yields a non-trivial outcome.** Under the same setting, the one-step prediction performance of sample-prediction flow matching is substantially higher than that of v-prediction (refer to Table 3) and in some cases even surpasses the multi-step prediction results. This suggests that multi-step denoising may be redundant for robotic action modeling, and that understanding how to booster one-step prediction to its optimal performance could be an intriguing direction for future research.

5. Conclusion

This work targets at the key trade-off between multi-modal distribution modeling capability and efficiency in visuomotor policy learning. We propose L1 Flow, a novel framework that reforms velocity-prediction flow matching into a sample-prediction paradigm with an L1 training objective, enabling the fusion of denoising models' strengths and L1 regression's efficiency. It achieves comparable or better performance than standard denoising-based methods (e.g., diffusion, flow matching) with 10–70× accelerated inference and distillation-based approaches (e.g., Consistency Policy, OneDP) with 5–7× faster training convergence, which provides a practical alternative for real-time robotic manipulation by balancing expressiveness and efficiency.

References

[1] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. 1, 2, 5, 8

[2] 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*, page 02783649241273668, 2023. 1, 2, 5

[3] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on robot learning*, pages 158–168. PMLR, 2022. 2, 3, 6, 1

[4] Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025. 2

[5] 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13460–13470, 2025. 2, 3

[6] 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*, pages 770–778, 2016. 5, 8

[7] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz,

James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. 1, 2

[8] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022. 5

[9] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022. 6

[10] Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In *The Twelfth International Conference on Learning Representations*, 2024. 2, 1

[11] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025. 2, 3, 4, 8

[12] Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025. 9

[13] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 2, 3

[14] 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*, 2024. 1, 2

[15] 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*, 2022. 2, 3

[16] 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. *arXiv preprint arXiv:2108.03298*, 2021. 2, 3, 6, 1

[17] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023. 5, 1

[18] Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *Forty-second International Conference on Machine Learning*, 2025. 2

[19] Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. *arXiv preprint arXiv:2405.07503*, 2024. 2, 5, 6

[20] Allen Z. Ren, Justin Lidard, Lars Lien Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. 2

[21] Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022. 2, 3

[22] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025. 1, 2

[23] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 2

[24] Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *The Twelfth International Conference on Learning Representations*, 2024. 2

[25] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, 2023. 2

[26] 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. 2, 3

[27] Mingyang Sun, Pengxiang Ding, Weinan Zhang, and Donglin Wang. Score-based diffusion policy compatible with reinforcement learning via optimal transport. In *Forty-second International Conference on Machine Learning*, 2025. 2

[28] RDT Team. Rdt2: Enabling zero-shot cross-embodiment generalization by scaling up umi data, 2025. 1, 2

[29] Dian Wang, Stephen Hart, David Surovik, Tarik Kelestemur, Haojie Huang, Haibo Zhao, Mark Yeatman, Jiuguang Wang, Robin Walters, and Robert Platt. Equivariant diffusion policy. *arXiv preprint arXiv:2407.01812*, 2024. 1, 2

[30] Yixiao Wang, Yifei Zhang, Mingxiao Huo, Ran Tian, Xiang Zhang, Yichen Xie, Chenfeng Xu, Pengliang Ji, Wei Zhan, Mingyu Ding, et al. Sparse diffusion policy: A sparse, reusable, and flexible policy for robot learning. *arXiv preprint arXiv:2407.01531*, 2024. 1

[31] Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, et al. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model. *arXiv preprint arXiv:2509.09372*, 2025. 3, 4, 8

[32] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023. 2

[33] Zhendong Wang, Zhaoshuo Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, et al. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *arXiv preprint arXiv:2410.21257*, 2024. 1, 2, 5, 6, 7

[34] Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025. 1, 2

[35] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin

Shen, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025.

[36] Junjie Wen, Yichen Zhu, Minjie Zhu, Zhibin Tang, Jinming Li, Zhongyi Zhou, Xiaoyu Liu, Chaomin Shen, Yaxin Peng, and Feifei Feng. DiffusionVLA: Scaling robot foundation models via unified diffusion and autoregression. In *Forty-second International Conference on Machine Learning*, 2025. 1, 2

[37] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024. 1, 2

[38] Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. Flowpolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 14754–14762, 2025. 1

[39] 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. 2, 3

[40] Guowei Zou, Haitao Wang, Hejun Wu, Yukun Qian, Yuhang Wang, and Weibing Li. Dm1: Meanflow with dispersive regularization for 1-step robotic manipulation. *arXiv preprint arXiv:2510.07865*, 2025. 2

L1 Sample Flow for Efficient Visuomotor Learning

Supplementary Material

In the supplementary material, the additional experimental details with regard to 2 simulation experiments and the real-world experiment.

6. Experiment Details

6.1. MimicGen

6.1.1. Simulation Setup

We select 8 representative tasks from MimicGen[17] which contains an automatically synthesizing large-scale dataset. As shown in Figure 6, these tasks span a wide spectrum of difficulty—as evidenced by their varying baseline success rates. Since the previous evaluations of OneDP[33] and Consistency Policy[10] utilize the absolute action space in RoboMimic, we adopt the complementary relative action representation in this part of the experiments to evaluate the adaptability of our method to different action parameterizations.

6.1.2. Hyperparameters

We adopt the 1D-Unet backbone and ResNet-18 observation encoder architecture from Diffusion Policy [2] for all policies. In the implementation of our L1 regression policy, we remove the timestep component: the U-Net takes only the observation embedding as the conditioning input, rather than concatenating observation embeddings with timestep embeddings as in other timestep-dependent policies. Apart from the components intrinsic to each algorithm, all remaining hyperparameters are kept consistent. The details of the key hyperparameters for L1 Flow training are summarized in Table 7.

6.2. Robomimic

6.2.1. Simulation Setup

RoboMimic[16] is a large-scale robotic manipulation benchmark designed to study imitation learning and offline reinforcement learning. The dataset includes 5 distinct manipulation tasks, each with a dataset of demonstrations teleoperated by proficient humans. These tasks are designed to enhance the learning effectiveness of robots through real human demonstrations.

For the experiments in Section 4.2, we choose three representative tasks from RoboMimic—**Square**, **ToolHang**, and **Transport**—as well as the **PushT** task from IBC [3]. Visualizations of these tasks in simulation are provided in Figure 7.

6.2.2. Hyperparameters

Following the implementation in Section 6.1.2, we adopt the identical backbone in Diffusion Policy [2]. All optimization hyperparameters are kept identical to the official implementation, except for a few task-specific settings. This ensures that any observed performance differences are attributable solely to the training objective, rather than architectural or optimization variations. Key hyperparameters and task-specific training configurations are summarized in Tables 7 and 6, respectively.

For certain tasks, unstable training dynamics were observed under the default learning rate (1×10^{-4}). To promote convergence, the learning rate was reduced where needed, and the final values are reported in Table 6. Higher maximum epoch limits are adopted specifically for **Square-mh** (1000 epochs) and **ToolHang-ph** (500 epochs) to ensure a well-shaped cosine annealing schedule and prevent premature decay of the learning rate to near-zero. In practice, early stopping is triggered at 200 epochs for both tasks, as validation performance typically saturates by then. All models are trained on NVIDIA RTX 4090 GPUs with mixed-precision training enabled, and the corresponding wall-clock training times are reported in Table 6.

Table 6. **Training configurations across tasks.** Reported values include the learning rate, training epoch, and wall-clock training time (trained on a single NVIDIA RTX 4090 GPU with mixed-precision training). For **Square-mh** and **ToolHang-ph**, early stopping is triggered at epoch 200 despite higher maximum epoch settings mentioned before.

Task	LR	Epochs	Time (h)
PushT	1e-4	200	2.5
Square-mh	2e-5	200	18
Square-ph	1e-4	200	10
ToolHang-ph	5e-5	200	53
Transport-ph	6e-5	200	76

6.3. Real-World Experiment

We provide the visualization of the policy rollouts with execution time stamp to clarify the efficiency of the proposed method in real-world, as shown in Figure 5. L1Flow achieves substantially faster inference which is roughly $10\times$ speed-up over DDIM (16 steps) and about $70\times$ over DDPM (100 steps). However, due to hardware limitations, the actual end-to-end completion time yields at most a $3\times$ speed-up. We also observe that, at the same spatial locations, DDIM is more prone to producing confusing actions

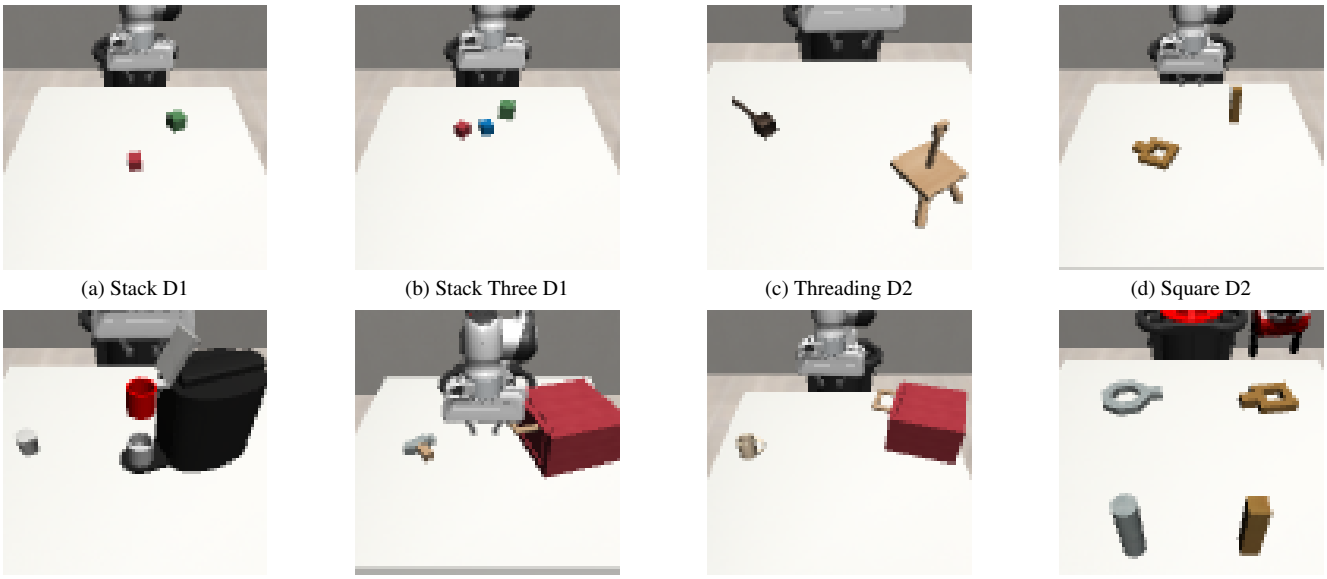
compared with DDPM and L1Flow. This often manifests as the policy getting "stuck" momentarily, which in turn leads to noticeably longer task completion times.

7. Code Example

We provide a minimal implementation of **L1 Flow**, as shown in the Figure 11, which can be directly substituted for the corresponding components in standard diffusion or flow-matching pipelines.

Table 7. **Key hyperparameters for L1 Flow training.**

Hyperparameters	**Values**
optimizer	AdamW($\beta_1 = 0.95, \beta_2 = 0.999$, weight decay=$1 \times 10^{-6}$)
learning rate	1e-4 in MimicGen, see Table 6 for MimicGen
learning rate scheduler	cosine annealing schedule with 500 warmup steps
batch size	128 in MimicGen, 64 in Robomimic&PushT
training epochs	500 in MimicGen, 200 in Robomimic&PushT
action chunk size (horizon)	16
executed actions per step	8
observed steps per decision	2
EMAs used	Yes ($\gamma = 1.0$, power=0.75)
abs action used	No in MimicGen, Yes in Robomimic&PushT



(a) Stack D1 (b) Stack Three D1 (c) Threading D2 (d) Square D2

(e) Coffee D2 (f) Hammer Cleanup D1 (g) Mug Cleanup D1 (h) Nut Assembly D0

Figure 6. **8 Tasks in MimicGen Benchmark.**

(a) Push T (b) Square (c) ToolHang (d) Transport

Figure 7. **The PushT Task and 3 Tasks in Robomimic Benchmark.**

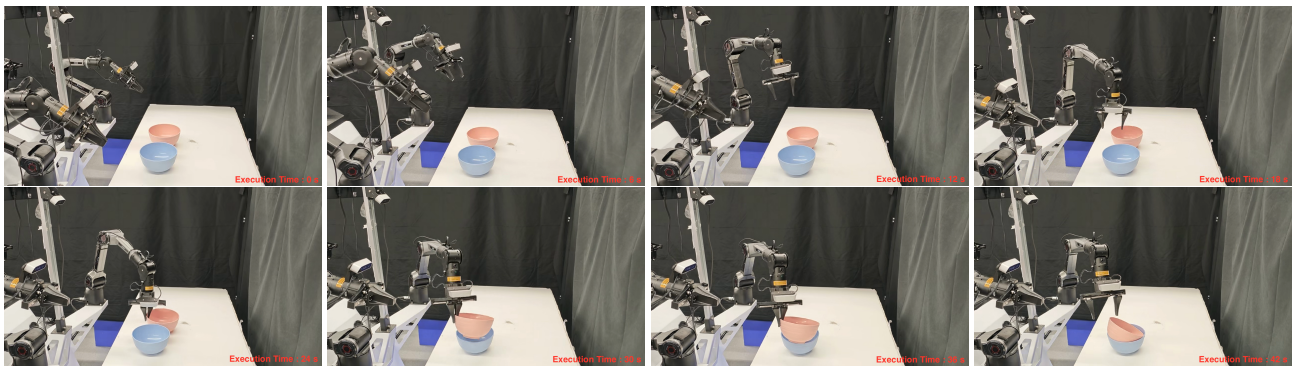


Figure 8. **A Real Execution Example of DDPM (100).** The total completion time is around 42 seconds. Each action chunk requires roughly 1 second for DDPM inference, leading to noticeable execution lag and prolonged task completion time.

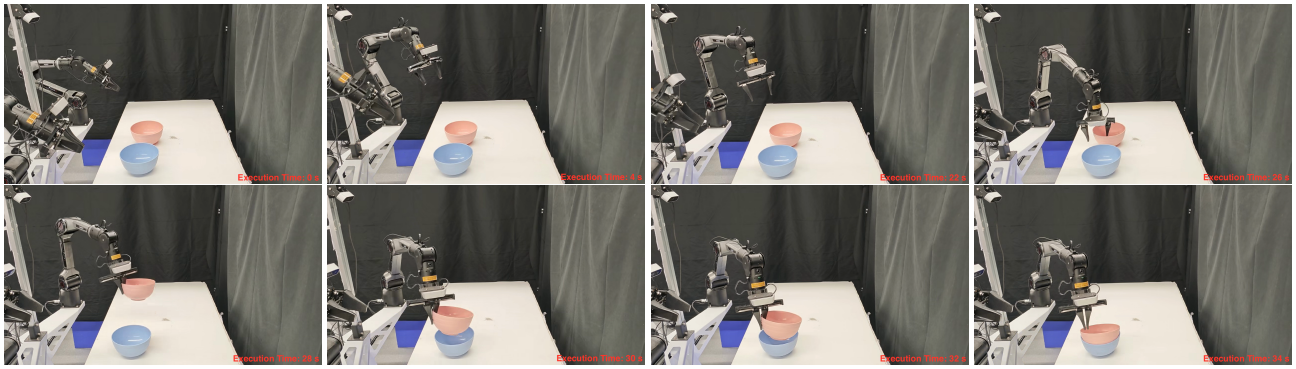


Figure 9. **A Real Execution Example of DDIM (16).** The total completion time is around 36 seconds. The DDIM policy is occasionally stuck during the grasping and placing stage which leads to longer completion time.

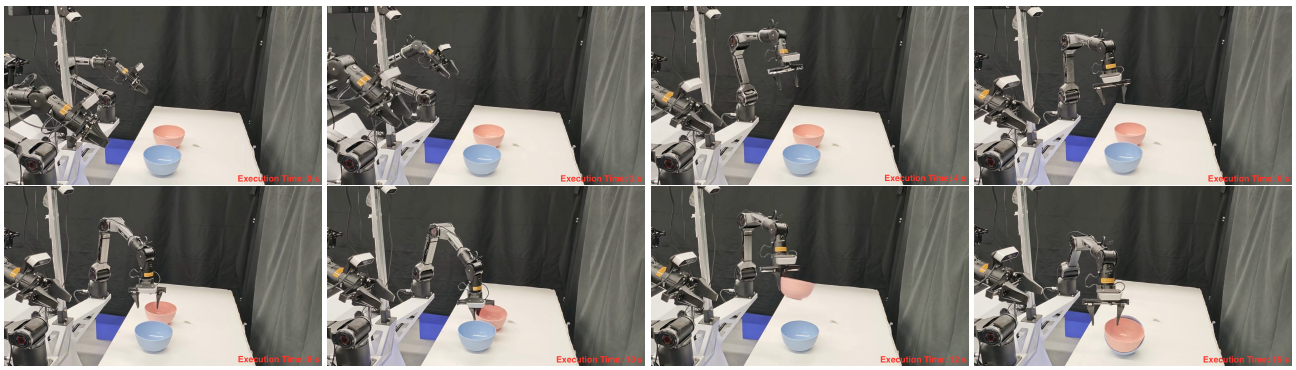


Figure 10. **A Real Execution Example of L1 Flow (Ours).** The total completion time is around 15 seconds. Due to hardware limitations, the actual end-to-end completion time yields at most a $3\times$ speed-up.

Listing 1. Training

```
noise = torch.randn((batchsize,horizon,action_dim), device=device)

# logistic+uniform
timesteps = logisticnormal_dist.sample((batchsize,))[:,0].to(device)
uni_timesteps = torch.rand_like(timesteps)
mask = torch.rand_like(uni_timesteps) < 0.01
timesteps[mask] = uni_timesteps[mask]

noisy_action = timesteps * action + (1 - timesteps) * noise

# Predict the sample
x_pred = self.model(noisy_action, timesteps, cond=cond)

# Compute the loss
loss = F.l1_loss(x_pred, action)
```

Listing 2. Inference

```
action = torch.randn(
            size=((batchsize, horizon, action_dim),
            device=device)

# set step values
dt = 0.5
t = torch.zeros(1, device=device)

# one-step integration
x_pred = model(action, t,
            cond=cond)
v_t = (x_pred - action)/(1-t)
action = action + dt*v_t
t = t + dt

# direct prediction
x_pred = model(action, t, cond=cond)
```

Figure 11. L1 Flow Code Example.