Diffusion Policy:
Visuomotor Policy Learning via Action Diffusion

Cheng Chi[1], Siyuan Feng[2], Yilun Du[3], Zhenjia Xu[1], Eric Cousineau[2], Benjamin Burchfiel[2], Shuran Song[1]
[1] Columbia University   [2] Toyota Research Institute   [3] MIT
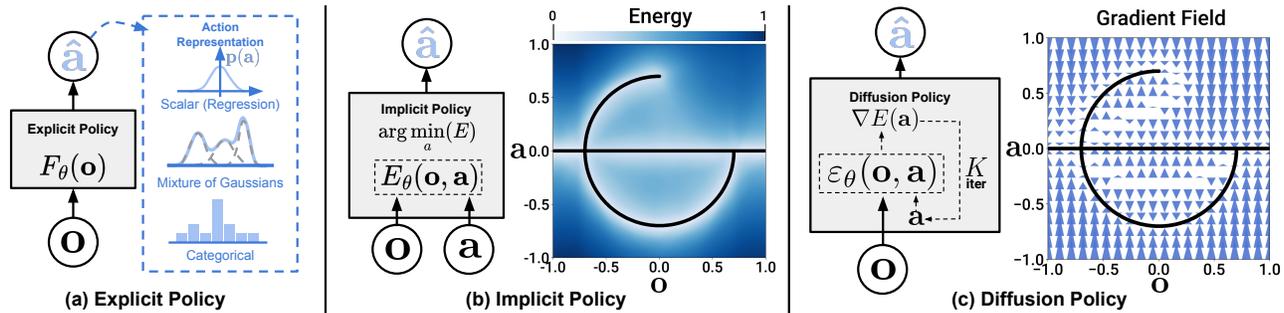https://diffusion-policy.cs.columbia.edu

Fig. 1: **Policy Representations.** a) Explicit policy with different types of action representations. b) Implicit policy learns an energy function conditioned on both action and observation and optimizes for actions that minimize the energy landscape c) Diffusion policy refines noise into actions via a learned gradient field. This formulation provides stable training, allows the learned policy to accurately model multimodal action distributions, and accommodates high-dimensional action sequences.

Abstract—This paper introduces Diffusion Policy, a new way of generating robot behavior by representing a robot's visuomotor policy as a conditional denoising diffusion process. We benchmark Diffusion Policy across 12 different tasks from 4 different robot manipulation benchmarks and find that it consistently outperforms existing state-of-the-art robot learning methods with an average improvement of 46.9%. Diffusion Policy learns the gradient of the action-distribution score function and iteratively optimizes with respect to this gradient field during inference via a series of stochastic Langevin dynamics steps. We find that the diffusion formulation yields powerful advantages when used for robot policies, including gracefully handling multimodal action distributions, being suitable for high-dimensional action spaces, and exhibiting impressive training stability. To fully unlock the potential of diffusion models for visuomotor policy learning on physical robots, this paper presents a set of key technical contributions including the incorporation of receding horizon control, visual conditioning, and the time-series diffusion transformer. We hope this work will help motivate a new generation of policy learning techniques that are able to leverage the powerful generative modeling capabilities of diffusion models. Code, data, and training details will be publicly available.

I. INTRODUCTION

Policy learning from demonstration, in its simplest form, can be formulated as the supervised regression task of learning to map observations to actions. In practice however, the unique nature of predicting robot actions — such as the existence of multimodal distributions, sequential correlation, and the requirement of high precision — makes this task distinct and challenging compared to other supervised learning problems.

Prior work attempts to address this challenge by exploring different *action representations* (Fig 1 a) – using mixtures of Gaussians [29], categorical representations of quantized actions [42], or by switching the *the policy representation* (Fig

1 b) – from explicit to implicit to better capture multi-modal distributions [12, 56].

In this work, we seek to address this challenge by introducing a new form of robot visuomotor policy that generates behavior via a "conditional denoising diffusion process [18] on robot action space", **Diffusion Policy**. In this formulation, instead of directly outputting an action, the policy infers the action-score gradient, conditioned on visual observations, for K denoising iterations (Fig. 1 c). This formulation allows robot policies to inherit several key properties from diffusion models – significantly improving performance.

- **Expressing multimodal action distributions.** By learning the gradient of the action score function [46] and performing Stochastic Langevin Dynamics sampling on this gradient field, Diffusion policy can express arbitrary normalizable distributions [32], which includes multimodal action distributions, a well-known challenge for policy learning.

- **High-dimensional output space.** As demonstrated by their impressive image generation results, diffusion models have shown excellent scalability to high-dimension output spaces. This property allows the policy to jointly infer a *sequence* of future actions instead of *single-step* actions, which is critical for encouraging temporal action consistency and avoiding myopic planning.

- **Stable training.** Training energy-based policies often requires negative sampling to estimate an intractable normalization constant, which is known to cause training instability [11, 12]. Diffusion Policy bypasses this requirement by learning the gradient of the energy function and thereby achieves stable training while maintaining

distributional expressivity.

Our **primary contribution** is to bring the above advantages to the field of robotics and demonstrate their effectiveness on complex real-world robot manipulation tasks. To successfully employ diffusion models for visuomotor policy learning, we present the following technical contributions that enhance the performance of Diffusion Policy and unlock its full potential on physical robots:

- **Closed-loop action sequences.** We combine the policy's capability to predict high-dimensional action sequences with *receding-horizon control* to achieve robust execution. This design allows the policy to continuously re-plan its action in a closed-loop manner while maintaining temporal action consistency – achieving a balance between long-horizon planning and responsiveness.
- **Visual conditioning.** We introduce a vision-conditioned diffusion policy, where the visual observations are treated as conditioning instead of a part of the joint data distribution. In this formulation, the policy extracts the visual representation once regardless of the denoising iterations, which drastically reduces the computation and enables real-time action inference.
- **Time-series diffusion transformer.** We propose a new transformer-based diffusion network that minimizes the over-smoothing effects of typical CNN-based models and achieves state-of-the-art performance on tasks that require high-frequency action changes and velocity control.

We systematically evaluate Diffusion Policy across **12** tasks from **4** different benchmarks [12, 15, 29, 42] under the behavior cloning formulation. The evaluation includes both simulated and real-world environments, 2DoF to 6DoF actions, single- and multi-task benchmarks, and fully- and under-actuated systems, with rigid and fluid objects, using demonstration data collected by single and multiple users.

Empirically, we find **consistent** performance boost across all benchmarks with an average improvement of 46.9%, providing strong evidence of the effectiveness of Diffusion Policy. We also provide detailed analysis to carefully examine the characteristics of the proposed algorithm and the impacts of the key design decisions. The code, data, and training details will be publicly available for reproducing our results. ***Please refer to the supplementary material for the robot videos.***

II. DIFFUSION POLICY FORMULATION

We formulate visuomotor robot policies as Denoising Diffusion Probabilistic Models (DDPMs) [18]. Crucially, Diffusion policies are able to express complex multimodal action distributions and possess stable training behavior – requiring little task-specific hyperparameter tuning. The following sections describe DDPMs in more detail and explain how they may be adapted to represent visuomotor policies.

A. Denoising Diffusion Probabilistic Models

DDPMs are a class of generative model where the output generation is modeled as a denoising process, often called Stochastic Langevin Dynamics [55].
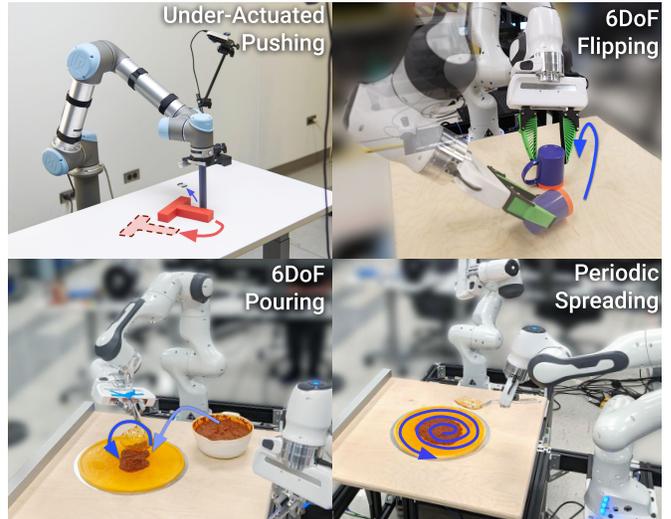

Fig. 2: **Realworld Benchmarks.** We deployed Diffusion Policy on two different robot platforms (UR5 and Franka) for 4 challenging tasks: under-actuate precise pushing (Push-T), 6DoF mug flipping, 6DoF sauce pouring, and periodic sauce spreading. Please check out the supplementary material for the resulting robot videos.

Starting from $\mathbf{x}^K$ sampled from Gaussian noise, the DDPM performs K iterations of denoising to produce a series of intermediate actions with decreasing levels of noise, $\mathbf{x}^k, \mathbf{x}^{k-1}...\mathbf{x}^0$, until a desired noise-free output $\mathbf{x}^0$ is formed. The process follows the equation

$$\mathbf{x}^{k-1} = \alpha(\mathbf{x}^k - \gamma\varepsilon_\theta(\mathbf{x}^k, k) + \mathcal{N}(0, \sigma^2 I)), \quad (1)$$

where ε_θ is the noise prediction network with parameters θ that will be optimized through learning and $\mathcal{N}(0, \sigma^2 I)$ is Gaussian noise added at each iteration.

The above equation 1 may also be interpreted as a single noisy gradient descent step:

$$\mathbf{x}' = \mathbf{x} - \gamma\nabla E(\mathbf{x}), \quad (2)$$

where the noise prediction network $\varepsilon_\theta(\mathbf{x}, k)$ effectively predicts the gradient field $\nabla E(\mathbf{x})$, and γ is the learning rate.

The choice of α, γ, σ as functions of iteration step k, also called noise schedule, can be interpreted as learning rate scheduling in gradient decent process. An α slightly smaller than 1 has been shown to improve stability [18]. Details about noise schedule will be discussed in Sec III-C.

B. DDPM Training

The training process starts by randomly drawing unmodified examples, $\mathbf{x}^0$, from the dataset. For each sample, we randomly select a denoising iteration k and then sample a random noise ε^k with appropriate variance for iteration k. The noise prediction network is asked to predict the noise from the data sample with noise added.

$$\mathcal{L} = MSE(\varepsilon^k, \varepsilon_\theta(\mathbf{x}^0 + \varepsilon^k, k)) \quad (3)$$

As shown in [18], minimizing the loss function in Eq 3 also minimizes the variational lower bound of the KL-divergence between the data distribution $p(\mathbf{x}^0)$ and the distribution of samples drawn from the DDPM $q(\mathbf{x}^0)$ using Eq 1.
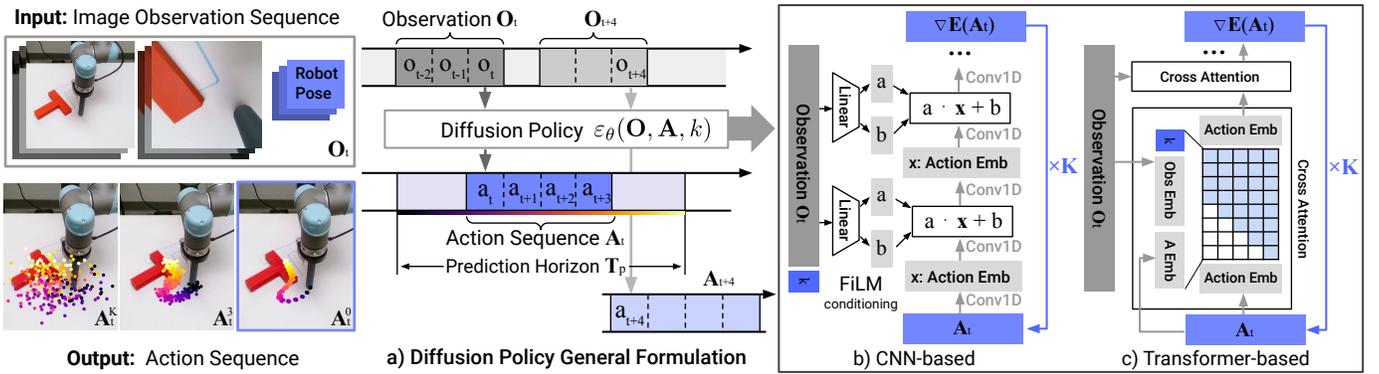
Fig. 3: **Diffusion Policy Overview** a) General formulation. At time step t, the policy takes the latest T_o steps of observation data O_t as input and outputs T_a steps of actions A_t. b) In the CNN-based Diffusion Policy, FiLM (Feature-wise Linear Modulation) [35] conditioning of the observation feature O_t is applied to every convolution layer, channel-wise. Starting from $\mathbf{A}_t^K$ drawn from Gaussian noise, the output of noise-prediction network ε_θ is subtracted, repeating K times to get $\mathbf{A}_t^0$, the denoised action sequence. c) In the Transformer-based [52] Diffusion Policy, the embedding of observation $\mathbf{O}_t$ is passed into a multi-head cross-attention layer of each transformer decoder block. Each action embedding is constrained to only attend to itself and previous action embeddings (causal attention) using the attention mask illustrated.

C. Diffusion for Visuomotor Policy Learning

While DDPMs are typically used for image generation ($\mathbf{x}$ is an image), we use a DDPM to learn robot visuomotor policies. This requires two major modifications in the formulation: 1. changing the output $\mathbf{x}$ to represent robot actions. 2. making the denoising processes *conditioned* on input observation $\mathbf{O}_t$. The following paragraphs discuss each of the modifications, and Fig. 3 shows an overview.

Closed-loop action-sequence prediction: An effective action formulation should encourage temporal consistency and smoothness in long-horizon planning while allowing prompt reactions to unexpected observations. To accomplish this goal, we integrate the action-sequence prediction produced by a diffusion model with receding horizon control [30] to achieve robust action execution. Concretely, at time step t the policy takes the latest T_o steps of observation data $\mathbf{O}_t$ as input and predicts T_p steps of actions, of which T_a steps of actions are executed on the robot without re-planning. Here, we define T_o as the observation horizon, T_p as the action prediction horizon and T_a as the action execution horizon. This encourages temporal action consistency while remaining responsive. More details about the effects of T_a are discussed in Sec IV-C.

Visual observation conditioning: We use a DDPM to approximate the conditional distribution $p(\mathbf{A}_t|\mathbf{O}_t)$ instead of the joint distribution $p(\mathbf{A}_t, \mathbf{O}_t)$ used in Janner et al. [20] for planning. This formulation allows the model to predict actions conditioned on observations without the cost of inferring future states, speeding up the diffusion process and improving the accuracy of generated actions. To capture the conditional distribution $p(\mathbf{A}_t|\mathbf{O}_t)$, we modify Eq 1 to:

$$\mathbf{A}_t^{k-1} = \alpha(\mathbf{A}_t^k - \gamma\varepsilon_\theta(\mathbf{O}_t, \mathbf{A}_t^k, k) + \mathcal{N}(0, \sigma^2 I)) \quad (4)$$

The training loss is modified from Eq 3 to:

$$\mathscr{L} = MSE(\varepsilon^k, \varepsilon_\theta(\mathbf{O}_t, \mathbf{A}_t^0 + \varepsilon^k, k)) \quad (5)$$

The exclusion of observation features $\mathbf{O}_t$ from the output of the denoising process significantly improves inference speed and better accommodates real-time control. It also helps to make **end-to-end** training of the vision encoder feasible. Details about the visual encoder are described in Sec. III-B.

III. KEY DESIGN DECISIONS

In this section, we describe key design decisions for Diffusion Policy as well as its concrete implementation of ε_θ with neural network architectures.

A. Network Architecture Options

The first design decision is the choice of neural network architectures for ε_θ. In this work, we examine two common network architecture types, convolutional neural networks (CNNs) [40] and Transformers [52], and compare their performance and training characteristics. Note that the choice of noise prediction network ε_θ is independent of visual encoders, which will be described in Sec. III-B.

CNN-based Diffusion Policy We adopt the 1D temporal CNN from Janner et al. [21] with a few modifications: First, we only model the conditional distribution $p(\mathbf{A}_t|\mathbf{O}_t)$ by conditioning the action generation process on observation features $\mathbf{O}_t$ with Feature-wise Linear Modulation (FiLM) [35] as well as denoising iteration k, shown in Fig 3 (b). Second, we only predict the action trajectory instead of the concatenated observation action trajectory. Third, we removed inpainting-based goal state conditioning due to incompatibility with our framework utilizing a receding prediction horizon. However, goal conditioning is still possible with the same FiLM conditioning method used for observations.

In practice, we found the CNN-based backbone to work well on most tasks out of the box without the need for much hyperparameter tuning. However, it performs poorly when the desired action sequence changes quickly and sharply through time (such as velocity command action space), likely due to the inductive bias of temporal convolutions to prefer low-frequency signals [49].

Time-series diffusion transformer To reduce the over-smoothing effect in CNN models [49], we introduce a novel transformer-based DDPM which adopts the transformer architecture from minGPT [42] for action prediction. Actions with

noise A_t^k are passed in as input tokens for the transformer decoder blocks, with the sinusoidal embedding for diffusion iteration k prepended as the first token. The observation $\mathbf{O}_t$ is transformed into observation embedding sequence by a shared MLP, which is then passed into the transformer decoder stack as input features. The "gradient" $\varepsilon_\theta(\mathbf{O_t}, \mathbf{A_t}^k, k)$ is predicted by each corresponding output token of the decoder stack.

In our state-based experiments, most of the best-performing policies are achieved with the transformer backbone, especially when the task complexity and rate of action change are high. However, we found the transformer to be more sensitive to hyperparameters. The difficulty of transformer training [25] is not unique to Diffusion Policy and could potentially be resolved in the future with improved transformer training techniques or increased data scale.

Recommendations. In general, we recommend starting with the CNN-based diffusion policy implementation as the first attempt at a new task. If performance is low due to task complexity or high-rate action changes, then the Time-series Diffusion Transformer formulation can be used to potentially improve performance at the cost of additional tuning.

B. Visual Encoder

The visual encoder maps the raw image sequence into a latent embedding O_t and is trained end-to-end with the diffusion policy. Different camera views use separate encoders, and images in each timestep are encoded independently and then concatenated to form O_t. We used a standard ResNet-18 (without pretraining) as the encoder with the following modifications: 1) Replace the global average pooling with a spatial softmax pooling to maintain spatial information [29]. 2) Replace BatchNorm with GroupNorm [57] for stable training. This is important when the normalization layer is used in conjunction with Exponential Moving Average [17] (commonly used in DDPMs).

C. Noise Schedule

The noise schedule, defined by σ, α, γ and the additive Gaussian Noise ε^k as functions of k, has been actively studied [18, 33]. The underlying noise schedule controls the extent to which diffusion policy captures high and low-frequency characteristics of action signals. In our control tasks, we empirically found that the Square Cosine Schedule proposed in iDDPM [33] works best for our tasks.

D. Accelerating Inference for Real-time Control

We use the diffusion process as the policy for robots; hence, it is critical to have a fast inference speed for closed-loop real-time control. The Denoising Diffusion Implicit Models (DDIM) approach [45] decouples the number of denoising iterations in training and inference, thereby allowing the algorithm to use fewer iterations for inference to speed up the process. In our real-world experiments, using DDIM with 100 training iterations and 10 inference iterations enables 0.1s inference latency on an Nvidia 3080 GPU.

IV. INTRIGUING PROPERTIES OF DIFFUSION POLICY

In this section, we provide some insights and intuitions about diffusion policy and its advantages over other forms of policy representations.

A. Model Multi-Modal Action Distributions

The challenge of modeling multi-modal distribution in human demonstrations has been widely discussed in behavior cloning literature [12, 42, 29]. Diffusion Policy's ability to express multimodal distributions naturally and precisely is one of its key advantages.

Intuitively, multi-modality in action generation for diffusion policy arises from two sources – an underlying stochastic sampling procedure and a stochastic initialization. In Stochastic Langevin Dynamics, an initial sample $\mathbf{A}_t^K$ is drawn from standard Gaussian at the beginning of each sampling process, which helps specify different possible convergence basins for the final action prediction $\mathbf{A}_t^0$. This action is then further stochastically optimized, with added Gaussian perturbations across a large number of iterations, which enables individual action samples to converge and move between different multi-modal action basins. Fig. 4, shows an example of the Diffusion Policy's multimodal behavior in a planar pushing task (Push T, introduced below) without explicit demonstration for the tested scenario.

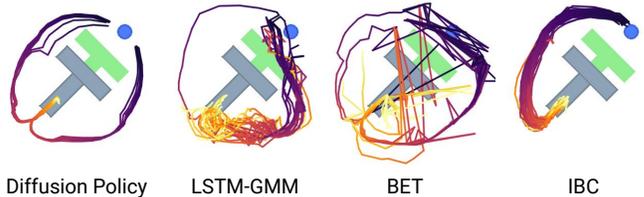Diffusion Policy   LSTM-GMM   BET   IBC

Fig. 4: **Multimodal behavior.** At the given state, the end-effector (blue) can either go left or right to push the block. **Diffusion Policy** learns both modes and commits to only one mode within each rollout. In contrast, both **LSTM-GMM** [29] and **IBC** [12] are biased toward one mode, while **BET** [42] fails to commit to a single mode due to its lack of temporal action consistency. Actions generated by rolling out 40 steps for the best-performing checkpoint.

B. Synergy with Position Control

We find that Diffusion Policy with a position-control action space consistently outperforms Diffusion Policy with velocity control, as shown in Fig 5. This surprising result stands in contrast to the majority of recent behavior cloning work that generally relies on velocity control [29, 42, 60, 13, 28, 27]. We speculate that there are two primary reasons for this discrepancy: First, action multimodality is more pronounced in position-control mode than it is when using velocity control. Because Diffusion Policy better expresses action multimodality than existing approaches, we speculate that it is inherently less affected by this drawback than existing methods. Furthermore, position control suffers less than velocity control from compounding error effects and is thus more suitable for action-sequence prediction (as discussed in the following section). As a result, Diffusion Policy is both less affected by the primary

drawbacks of position control and is better able to exploit position control's advantages.
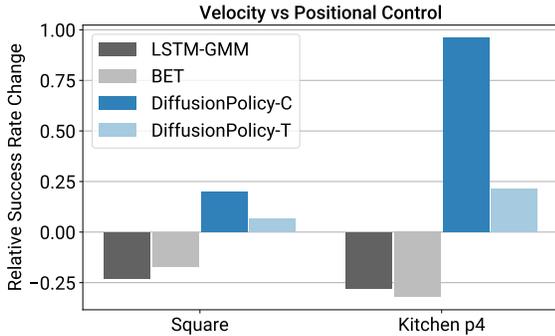

Fig. 5: **Velocity v.s. Position Control.** The performance difference when switching from velocity to position control. While both BCRNN and BET performance decrease, Diffusion Policy is able to leverage the advantage of position and improve its performance.

C. Benefits of Action-Sequence Prediction

Sequence prediction is often avoided in most policy learning methods due to the difficulties in effectively sampling from high-dimensional output spaces. For example, IBC would struggle in effectively sampling high-dimensional action space with a non-smooth energy landscape. Similarly, BC-RNN and BET would have difficulty specifying the number of modes that exist in the action distribution (needed for GMM or k-means steps).

In contrast, DDPM scales well with output dimensions without sacrificing the expressiveness of the model, as demonstrated in many image generation applications. Leveraging this capability, Diffusion Policy represents action in the form of a high-dimensional action sequence, which naturally addresses the following issues:

- **Temporal action consistency**: Take Fig 4 as an example. To push the T block into the target from the bottom, the policy can go around the T block from either left or right. However, suppose each action in the sequence is predicted as independent multimodal distributions (as done in BC-RNN and BET). In that case, consecutive actions could be drawn from different modes, resulting in jittery actions that alternate between the two valid trajectories.

- **Robustness to idle actions**: Idle actions occur when a demonstration is paused and results in sequences of identical positional actions or near-zero velocity actions. It is common during teleoperation and is sometimes required for tasks like liquid pouring. However, single-step policies can easily overfit to this pausing behavior. For example, BC-RNN and IBC often get stuck in real-world experiments when the idle actions are not explicitly removed from training.

D. Training Stability

While IBC, in theory, should possess similar advantages as diffusion policies. However, achieving reliable and high-performance results from IBC in practice is challenging due to IBC's inherent training instability [48]. Fig 7 shows training
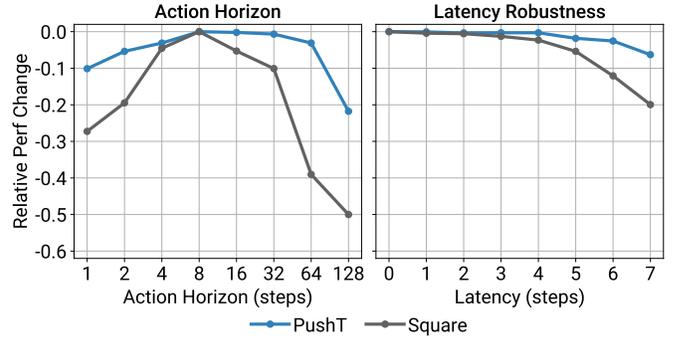

Fig. 6: **Diffusion Policy Ablation Study.** Change (difference) in success rate relative to the maximum for each task is shown on the Y-axis. **Left**: trade-off between temporal consistency and responsiveness when selecting the action horizon. **Right**: Diffusion Policy with position control is robust against latency. Latency is defined as the number of steps between the last frame of observations to the first action that can be executed.

error spikes and unstable evaluation performance throughout the training process, making hyperparameter turning critical and checkpoint selection difficult. As a result, Florence et al. [12] evaluate every checkpoint and report results for the best-performing checkpoint. In a real-world setting, this workflow necessitates the evaluation of many policies on hardware to select a final policy. Here, we discuss why Diffusion Policy appears significantly more stable to train.

An implicit policy represents the action distribution using an Energy-Based Model (EBM):

$$p_\theta(\mathbf{a}|\mathbf{o}) = \frac{e^{-E_\theta(\mathbf{o},\mathbf{a})}}{Z(\mathbf{o},\theta)} \quad (6)$$

where $Z(\mathbf{o},\theta)$ is an intractable normalization constant (with respect to $\mathbf{a}$).

To train the EBM for implicit policy, an InfoNCE-style loss function is used, which equates to the negative log-likelihood of Eq 6:
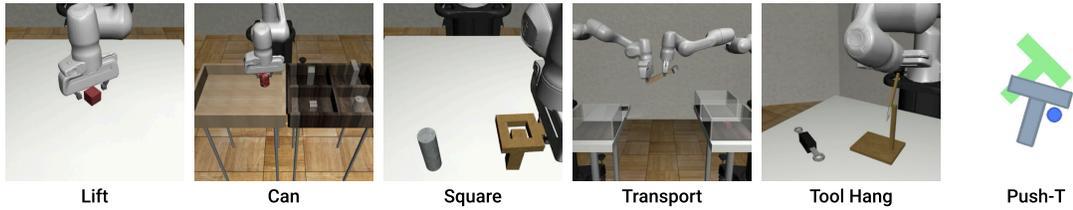
$$\mathcal{L}_{infoNCE} = -\log\left(\frac{e^{-E_\theta(\mathbf{o},\mathbf{a})}}{e^{-E_\theta(\mathbf{o},\mathbf{a})} + \sum_{j=1}^{N_{neg}} e^{-E_\theta(\mathbf{o},\widetilde{\mathbf{a}}^j)}}\right) \quad (7)$$

where a set of negative samples $\{\widetilde{\mathbf{a}}^j\}_{j=1}^{N_{neg}}$ are used to estimate the intractable normalization constant $Z(\mathbf{o},\theta)$. In practice, the inaccuracy of negative sampling is known to cause training instability for EBMs [11, 48].

Diffusion Policy and DDPMs sidestep the issue of estimating $Z(\mathbf{a},\theta)$ altogether by modeling the **score function** [46] of the same action distribution in Eq 6:

$$\nabla_\mathbf{a} \log p(\mathbf{a}|\mathbf{o}) = -\nabla_\mathbf{a} E_\theta(\mathbf{a},\mathbf{o}) - \underbrace{\nabla_\mathbf{a} \log Z(\mathbf{o},\theta)}_{=0} \approx -\varepsilon_\theta(\mathbf{a},\mathbf{o})$$
$$(8)$$

where the noise-prediction network $\varepsilon_\theta(\mathbf{a},\mathbf{o})$ is approximating the negative of the score function $\nabla_\mathbf{a} \log p(\mathbf{a}|\mathbf{o})$ [26], which is independent of the normalization constant $Z(\mathbf{o},\theta)$. As a result, neither the inference (Eq 4) nor training (Eq 5) process of Diffusion Policy involves evaluating $Z(\mathbf{o},\theta)$, thus making Diffusion Policy training more stable.

| | Lift | | Can | | Square | | Transport | | ToolHang | Push-T |
	ph	mh	ph	mh	ph	mh	ph	mh	ph	ph
LSTM-GMM [29]	**1.00**/0.96	**1.00**/0.93	**1.00**/0.91	**1.00**/0.81	0.95/0.73	0.86/0.59	0.76/0.47	0.62/0.20	0.67/0.31	0.67/0.61
IBC [12]	0.79/0.41	0.15/0.02	0.00/0.00	0.01/0.01	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.90/0.84
BET [42]	**1.00**/0.96	**1.00**/0.99	**1.00**/0.89	**1.00**/0.90	0.76/0.52	0.68/0.43	0.38/0.14	0.21/0.06	0.58/0.20	0.79/0.70
DiffusionPolicy-C	**1.00**/0.98	**1.00**/0.97	**1.00**/0.96	**1.00**/0.96	**1.00**/0.93	**0.97/0.82**	0.94/0.82	**0.68/0.46**	0.50/0.30	0.95/**0.91**
DiffusionPolicy-T	**1.00**/1.00	**1.00**/1.00	**1.00**/1.00	**1.00**/0.94	**1.00**/0.89	0.95/0.81	**1.00**/0.84	0.62/0.35	**1.00**/0.87	0.95/0.79

TABLE I: **Behavior Cloning Benchmark (State Policy)** We present success rates with different checkpoint selection methods in the format of (max performance) / (average of last 10 checkpoints), with each averaged across 3 training seeds and 50 different environment initial conditions (150 in total). LSTM-GMM corresponds to BC-RNN in RoboMimic[29], which we reproduced and obtained slightly better results than the original paper. Our results show that Diffusion Policy significantly improves state-of-the-art performance across the board.

| | Lift | | Can | | Square | | Transport | | ToolHang | Push-T |
	ph	mh	ph	mh	ph	mh	ph	mh	ph	ph
LSTM-GMM [29]	**1.00**/0.96	**1.00**/0.95	**1.00**/0.88	0.98/0.90	0.82/0.59	0.64/0.38	0.88/0.62	0.44/0.24	0.68/0.49	0.69/0.54
IBC [12]	0.94/0.73	0.39/0.05	0.08/0.01	0.00/0.00	0.03/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.00/0.00	0.75/0.64
DiffusionPolicy-C	**1.00**/1.00	**1.00**/1.00	**1.00**/0.97	**1.00**/0.96	0.98/**0.92**	**0.98/0.84**	**1.00**/0.93	**0.89/0.69**	**0.95/0.73**	**0.91/0.84**
DiffusionPolicy-T	**1.00**/1.00	**1.00**/0.99	**1.00**/0.98	**1.00**/0.98	**1.00**/0.90	0.94/0.80	0.98/0.81	0.73/0.50	0.76/0.47	0.78/0.66

TABLE II: **Behavior Cloning Benchmark (Visual Policy)** Performance are reported in the same format as in Tab I. LSTM-GMM numbers were reproduced to get a complete evaluation in addition to the best checkpoint performance reported. Diffusion Policy shows consistent performance improvement, especially for complex tasks like Transport and ToolHang.
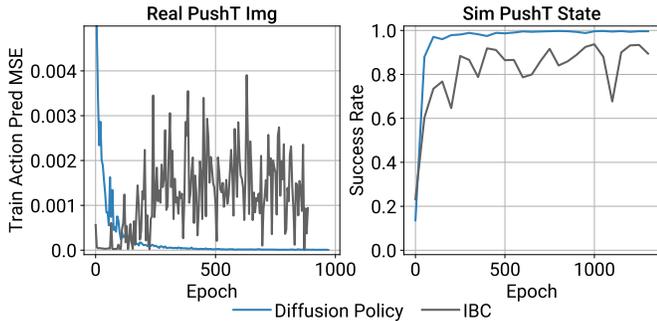

Fig. 7: **Training Stability.** Left: IBC fails to infer training actions with increasing accuracy despite smoothly decreasing training loss for energy function. Right: IBC's evaluation success rate oscillates, making checkpoint selection difficult (evaluated using policy rollouts in simulation).

Task	# Rob	# Obj	ActD	#PH	#MH	Steps	Img?	HiPrec
	Simulation Benchmark							
Lift	1	1	7	200	300	400	Yes	No
Can	1	1	7	200	300	400	Yes	No
Square	1	1	7	200	300	400	Yes	Yes
Transport	2	3	14	200	300	700	Yes	No
ToolHang	1	2	7	200	0	700	Yes	Yes
Push-T	1	1	2	200	0	300	Yes	Yes
BlockPush	1	2	2	0	0	350	No	No
Kitchen	1	7	9	656	0	280	No	No
	Realworld Benchmark							
Push-T	1	1	2	136	0	600	Yes	Yes
6DoF Pour	1	liquid	6	90	0	600	Yes	No
Periodic Spread	1	liquid	6	90	0	600	Yes	No
Mug Flip	1	1	7	250	0	600	Yes	No

TABLE III: **Tasks Summary.** # Rob: number of robots, #Obj: number of objects, ActD: action dimension, PH: proficient-human demonstration, MH: multi-human demonstration, Steps: max number of rollout steps, HiPrec: whether the task has a high precision requirement. BlockPush uses 1000 episodes of scripted demonstrations.

V. EVALUATION

We systematically evaluate Diffusion Policy on 12 tasks from 4 benchmarks [12, 15, 29, 42]. This evaluation suite includes both simulated and real environments, single and multiple task benchmarks, fully actuated and under-actuated systems, and rigid and fluid objects. We found Diffusion Policy to consistently outperform the prior state-of-the-art on all of the tested benchmarks, with an average success-rate improvement of 46.9%. In the following sections, we provide an overview of each task, our evaluation methodology on that task, and our key takeaways.

### A. Simulation Environments and datasets

1) Robomimic: [29] is a large-scale robotic manipulation benchmark designed to study imitation learning and offline RL. The benchmark consists of 5 tasks with a proficient human (PH) teleoperated demonstration dataset for each and mixed proficient/non-proficient human (MH) demonstration datasets for 4 of the tasks (9 variants in total). For each variant, we report results for both state- and image-based observations. Properties for each task are summarized in Tab III.

2) Push-T: adapted from IBC [12], requires pushing a T-shaped block (gray) to a fixed target (red) with a circular
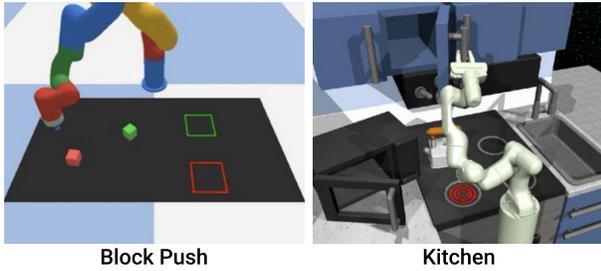
Block Push   Kitchen

	BlockPush		Kitchen			
	p1	p2	p1	p2	p3	p4
LSTM-GMM [29]	0.03	0.01	**1.00**	0.90	0.74	0.34
IBC [12]	0.01	0.00	0.99	0.87	0.61	0.24
BET [42]	0.96	0.71	0.99	0.93	0.71	0.44
DiffusionPolicy-C	0.36	0.11	**1.00**	**1.00**	**1.00**	**0.99**
DiffusionPolicy-T	**0.99**	**0.94**	**1.00**	0.99	0.99	0.96

TABLE IV: **Multi-Stage Tasks (State Observation)**. For PushBlock, px is the frequency of pushing x blocks into the targets. For Kitchen, px is the frequency of interacting with x or more objects (e.g. bottom burner). Diffusion Policy performs better, especially for difficult metrics such as $p2$ for Block Pushing and $p4$ for Kitchen, as demonstrated by our results.

end-effector (blue)s. Variation is added by random initial conditions for T block and end-effector. The task requires exploiting complex and contact-rich object dynamics to push the T block precisely, using point contacts. There are two variants: one with RGB image observations and another with 9 2D keypoints obtained from the ground-truth pose of the T block, both with proprioception for end-effector location.

3) Multimodal Block Pushing: adapted from BET [42], this task tests the policy's ability to model multimodal action distributions by pushing two blocks into two squares in any order. The demonstration data is generated by a scripted oracle with access to groundtruth state info. This oracle randomly selects an initial block to push and moves it to a randomly selected square. The remaining block is then pushed into the remaining square. This task contains **long-horizon** multimodality that can not be modeled by a single function mapping from observation to action.

4) Franka Kitchen: is a popular environment for evaluating the ability of IL and Offline-RL methods to learn multiple long-horizon tasks. Proposed in Relay Policy Learning [15], the Franka Kitchen environment contains 7 objects for interaction and comes with a human demonstration dataset of 566 demonstrations, each completing 4 tasks in arbitrary order. The goal is to execute as many demonstrated tasks as possible, regardless of order, showcasing both short-horizon and long-horizon multimodality.

B. Evaluation Methodology

We present the **best-performing for each baseline method** on each benchmark from all possible sources – our reproduced result (LSTM-GMM) or original number reported in the paper (BET, IBC). We report results from the average of the last 10 checkpoints (saved every 50 epochs) across **3** training seeds and **50** environment initializations (an average of **1500** experiments in total). The metric for most tasks is success rate,

except for the Push-T task, which uses target area coverage. In addition, we report the average of best-performing checkpoints for robomimic and Push-T tasks to be consistent with the evaluation methodology of their respective original papers [29, 12]. All state-based tasks are trained for 4500 epochs, and image-based tasks for 3000 epochs. Each method is evaluated with its best-performing action space: position control for Diffusion Policy and velocity control for baselines (the effect of action space will be discussed in detail in Sec V-C). The results from these simulation benchmarks are summarized in Table I and Table II.

C. Key Findings

Diffusion Policy outperforms alternative methods on all tasks and variants, with both state and vision observations, in our simulation benchmark study (Tabs I, II and IV) with an average improvement of 46.9%. Following paragraphs summarize the key takeaways.
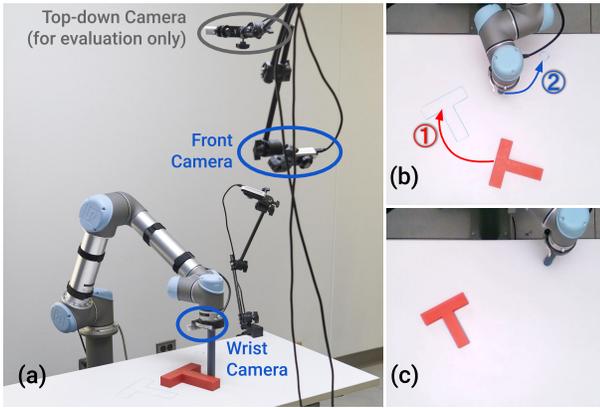
Diffusion Policy can express short-horizon multimodality. We define short-horizon action multimodality as multiple ways of achieving **the same immediate goal**, which is prevalent in human demonstration data [29]. In Fig 4, we present a case study of this type of short-horizon multimodality in the Push-T task. Diffusion Policy learns to approach the contact point equally likely from left or right, while LSTM-GMM [29] and IBC [12] exhibit bias toward one side and BET [42] cannot commit to one mode.

Diffusion Policy can express long-horizon multimodality. Long-horizon multimodality is the completion of **different sub-goals** in inconsistent order. For example, the order of pushing a particular block in the Block Push task or the order of interacting with 7 possible objects in the Kitchen task are arbitrary. We find that Diffusion Policy copes well with this type of multimodality; it outperforms baselines on both tasks by a large margin: 32% improvement on Block Push's p2 metric and 213% improvement on Kitchen's p4 metric.

Diffusion Policy can better leverage position control. Our ablation study (Fig. 5) shows that selecting position control as the diffusion-policy action space significantly outperformed velocity control. The baseline methods we evaluate, however, work best with velocity control (and this is reflected in the literature where most existing work reports using velocity-control action spaces [29, 42, 60, 13, 28, 27]).

The tradeoff in action horizon. As discussed in Sec IV-C, having an action horizon greater than 1 helps the policy predict consistent actions and compensate for idle portions of the demonstration, but too long a horizon reduces performance due to slow reaction time. Our experiment confirms this trade-off (Fig. 6 left) and found the action horizon of 8 steps to be optimal for most tasks that we tested.

Robustness against latency. Diffusion Policy employs receding horizon position control to predict a sequence of actions into the future. This design helps address the latency gap caused by image processing, policy inference, and network delay. Our ablation study with simulated latency showed Diffusion Policy is able to maintain peak performance with

TABLE V: **Realworld Push-T Experiment.** a) Hardware setup. b) Illustration of the task. The robot needs to ① precisely push the T-shaped block into the target region, **and** ② move the end-effector to the end-zone. c) The ground truth end state used to calculate IoU metrics used in this table. Table: Success is defined by the end-state IoU greater than the minimum IoU in the demonstration dataset. Average episode duration presented in seconds. T-E2E stands for end-to-end trained Transformer-based Diffusion Policy

	Human Demo	IBC pos	vel	LSTM-GMM pos	vel	Diffusion Policy T-E2E	ImgNet	R3M	E2E
IoU	0.84	0.14	0.19	0.24	0.25	0.53	0.24	0.66	**0.80**
Succ %	1.00	0.00	0.00	0.20	0.10	0.65	0.15	0.80	**0.95**
Duration	20.3	56.3	41.6	47.3	51.7	57.5	55.8	31.7	**22.9**

latency up to 4 steps (Fig 6). We also find that velocity control is more affected by latency than position control, likely due to compounding error effects.

Diffusion Policy is stable to train. We found that the optimal hyperparameters for Diffusion Policy are mostly consistent across tasks. In contrast, IBC [12] is prone to training instability. This property is discussed in Sec IV-D.

VI. REALWORLD EVALUATION

We evaluated Diffusion Policy in the realworld performance on 4 tasks across 2 hardware setups – with training data from different demonstrators for each setup. On the realworld Push-T task, we perform ablations examining Diffusion Policy on 2 architecture options and 3 visual encoder options; we also benchmarked against 2 baseline methods with both position-control and velocity-control action spaces. On all tasks, Diffusion Policy variants with both CNN backbones and end-to-end-trained visual encoders yielded the best performance. More details about the task setup and parameters may be found in supplemental materials.

A. Realworld Push-T Task

Real-world Push-T is significantly harder than the simulated version due to 3 modifications: 1. The real-world Push-T task is **multi-stage**. It requires the robot to ① push the T block into the target and then ② move its end-effector into a designated end-zone to avoid occlusion. 2. The policy needs to make fine adjustments to make sure the T is fully in the goal region before heading to the end-zone, creating additional short-term multimodality. 3. The IoU metric is measured at

the **last step** instead of taking the maximum over all steps. We threshold success rate by the minimum achieved IoU metric from the human demonstration dataset. Our UR5-based experiment setup is shown in Fig V. Diffusion Policy predicts robot commands at 10 Hz and these commands then linearly interpolated to 125 Hz for robot execution.

Result Analysis. Diffusion Policy performed close to human level with 95% success rate and 0.8 v.s. 0.84 average IoU, compared with the 0% and 20% success rate of best-performing IBC and LSTM-GMM variants. Fig 8 qualitatively illustrates the behavior for each method starting from the same initial condition. We observed that poor performance during the transition between stages is the most common failure case for the baseline method due to high multimodality during those sections and an ambiguous decision boundary. LSTM-GMM got stuck near the T block in 8 out of 20 evaluations (3rd row), while IBC prematurely left the T block in 6 out of 20 evaluations (4th row). We did not follow the common practice of removing **idle actions** from training data due to task requirements, which also contributed to LSTM and IBC's tendency to overfit on small actions and get stuck in this task. The results are best appreciated with videos in supplemental materials.

End-to-end v.s. pre-trained vision encoders We tested Diffusion Policy with pre-trained vision encoders (ImageNet [9] and R3M[31]), as seen in Tab. V. Diffusion Policy with R3M achieves an 80% success rate but predicts jittery actions and is more likely to get stuck compared to the end-to-end trained version. Diffusion Policy with ImageNet showed less promising results with abrupt actions and poor performance. We found that end-to-end training is still the most effective way to incorporate visual observation into Diffusion Policy, and our best-performing models were all end-to-end trained.

Robustness against perturbation Diffusion Policy's robustness against visual and physical perturbations was evaluated in a separate episode from experiments in Tab V. As shown in Fig 9, three types of perturbations are applied. 1) The front camera was blocked for 3 secs by a waving hand (left column), but the diffusion policy, despite exhibiting some jitter, remained on-course and pushed the T block into position. 2) We shifted the T block while Diffusion Policy was making fine adjustments to the T block's position. Diffusion policy immediately re-planned to push from the opposite direction, negating the impact of perturbation. 3) We moved the T block while the robot was en route to the end-zone after the first stage's completion. The Diffusion Policy immediately changed course to adjust the T block back to its target and then continued to the end-zone. This experiment indicates that Diffusion Policy may be able to **synthesize novel behavior** in response to unseen observations.

B. Mug Flipping Task

The mug flipping task is designed to test Diffusion Policy's ability to handle complex **3D rotations** while operating close to the hardware's kinematic limits. The goal is to reorient a randomly placed mug to have ① the lip facing down ② the
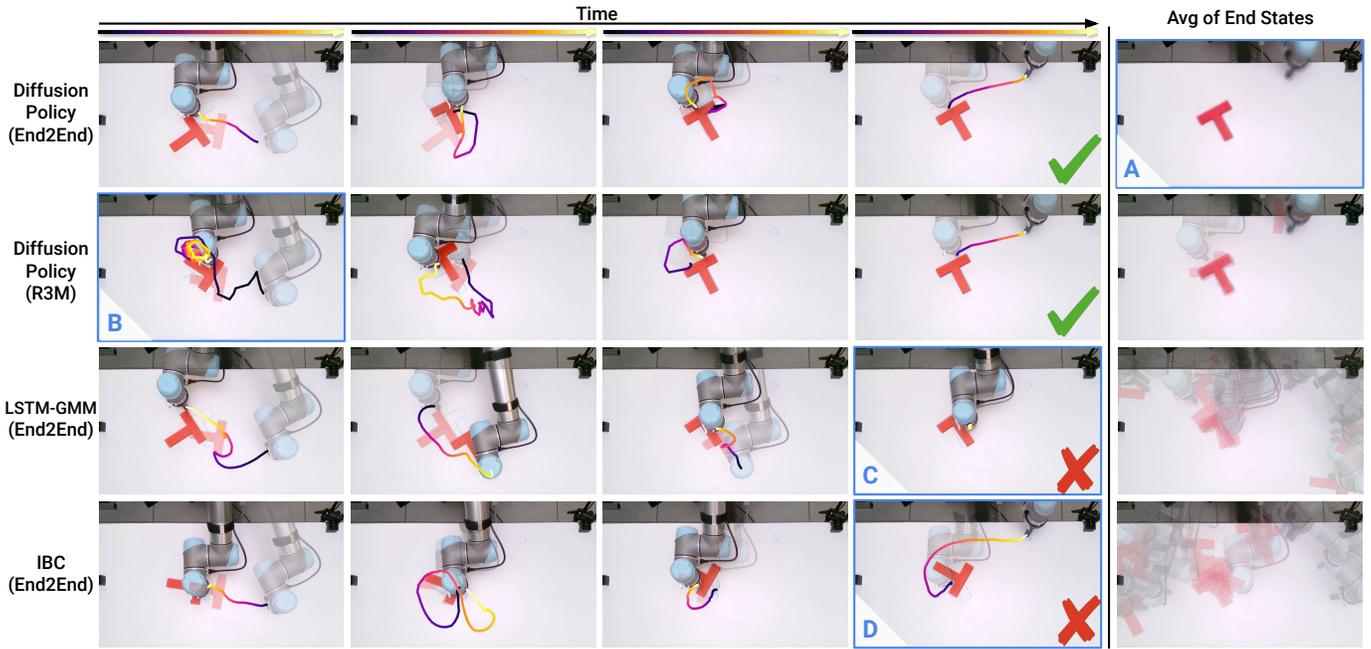
Fig. 8: **Realworld Push-T Comparisons.** Columns 1-4 show action trajectories based on key events. The last column shows averaged images of the end state. **A**: Diffusion policy (End2End) achieves more accurate and consistent end states. **B**: Diffusion Policy (R3M) gets stuck initially but later recovers and finishes the task. **C**: LSTM-GMM fails to reach the end zone while adjusting the T block, blocking the eval camera view. **D**: IBC prematurely ends the pushing stage.
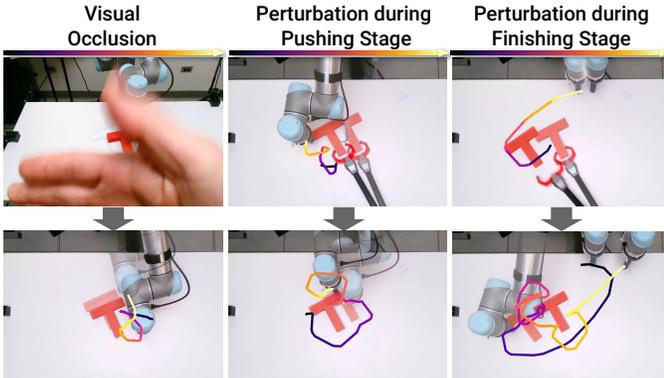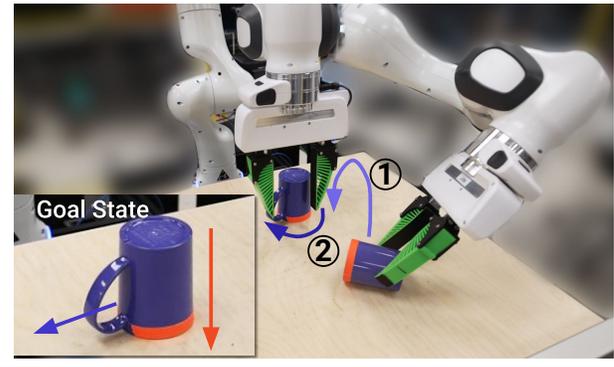

Fig. 9: **Robustness Test for Diffusion Policy. Left**: A waving hand in front of the camera for 3 seconds causes slight jitter, but the predicted actions still function as expected. **Middle**: Diffusion Policy immediately corrects shifted block position to the goal state during the pushing stage. **Right**: Policy immediately aborts heading to the end zone, returning the block to goal state upon detecting block shift. This novel behavior was never demonstrated. Please check the videos in the supplementary material.

	Human	LSTM-GMM	Diffusion Policy
Succ %	1.0	0.0	0.9

Fig. 10: **6DoF Mug Flipping Task.** The robot needs to ① Pickup a randomly placed mug and place it lip down (marked orange). ② Rotate the mug such that its handle is pointing left.

handle pointing left, as shown in Fig. 10. Depending on the mug's initial pose, the demonstrator might directly place the mug in desired orientation, or may use additional push of the handle to rotation the mug. As a result, the demonstration dataset is highly multi-modal: grasp vs push, different types of grasps (forehand vs backhand) or local grasp adjustments (rotation around mug's principle axis), and are particularly challenging for baseline approaches to capture.

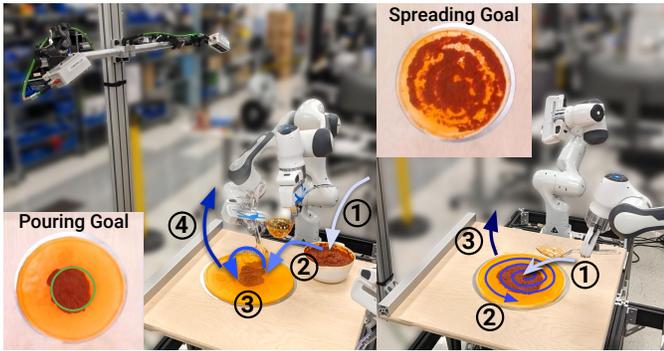**Result Analysis.** Diffusion policy is able to complete this task with 90% success rate over 20 trials. The richness of

captured behaviors is best appreciated with the video. Although never demonstrated, the policy is also able to sequence multiple pushes for handle alignment or regrasps for dropped mug when necessary. For comparison, we also train a LSTM-GMM policy trained with a subset of the same data. For 20 in-distribution initial conditions, the LSTM-GMM policy never aligns properly with respect to the mug, and fails to grasp in all trials.

C. Sauce Pouring and Spreading

The sauce pouring and spreading tasks are designed to test Diffusion Policy's ability to work with **non-rigid** objects, **6**

Fig. 11: **Realworld Sauce Manipulation.** [Left] **6DoF pouring Task.** The robot needs to ① dip the ladle to scoop sauce from the bowl, ② approach the center of the pizza dough, ③ pour sauce, and ④ lift the ladle to finish the task. [Right] **Periodic spreading Task** The robot needs to ① approach the center of the sauce with a grasped spoon, ② spread the sauce to cover pizza in a spiral pattern, and ③ lift the spoon to finish the task.

	Pour		Spread	
	IoU	Succ	Coverage	Succ
Human	0.79	1.00	0.79	1.00
LSTM-GMM	0.06	0.00	0.27	0.00
Diffusion Policy	**0.74**	**0.79**	**0.77**	**1.00**

Dof action spaces, and **periodic** actions in real-world setups. Our Franka Panda setup and tasks are shown in Fig 11. The goal for the **6DoF pouring task** is to pour one full ladle of sauce onto the center of the pizza dough, with performance measured by IoU between the poured sauce mask and a nominal circle at the center of the pizza dough (illustrated by the green circle in Fig 11). The goal for the **periodic spreading task** is to spread sauce on pizza dough, with performance measured by sauce coverage. Variations across evaluation episodes come from random locations for the dough and the sauce bowl. The success rate is computed by thresholding with minimum human performance. Results are best viewed in supplemental videos. Both tasks were trained with the same Push-T hyperparameters, and successful policies were achieved on the first attempt.

The sauce pouring task requires the robot to remain stationary for a period of time to fill the ladle with viscous tomato sauce. The resulting idle actions are known to be challenging for behavior cloning algorithms and therefore are often avoided or filtered out. Fine adjustments during pouring are necessary during sauce pouring to ensure coverage and to achieve the desired shape.

The demonstrated sauce-spreading strategy is inspired by the human chef technique, which requires both a long-horizon cyclic pattern to maximize coverage and short-horizon feedback for even distribution (since the tomato sauce used often drips out in lumps with unpredictable sizes). Periodic motions are known to be difficult to learn and therefore are often addressed by specialized action representations [58]. Both tasks require the policy to self-terminate by lifting the ladle/spoon.

Result Analysis. Diffusion policy achieves close-to-human performance on both tasks, with coverage 0.74 vs 0.79 on pouring and 0.77 vs 0.79 on spreading. Diffusion policy reacted gracefully to external perturbations such as moving the pizza dough by hand during pouring and spreading. Results are best appreciated with videos in the supplemental material.

LSTM-GMM performs poorly on both sauce pouring and spreading tasks. It failed to lift the ladle after successfully scooping sauce in 15 out of 20 of the pouring trials. When the ladle was successfully lifted, the sauce was poured off-centered. LSTM-GMM failed to self-terminate in all trials. We suspect LSTM-GMM's hidden state failed to capture sufficiently long history to distinguish between the ladle dipping and the lifting phases of the task. For sauce spreading, LSTM-GMM always lifts the spoon right after the start, and failed to make contact with the sauce in all 20 experiments.

VII. RELATED WORK

Creating capable robots without requiring explicit programming of behaviors is a longstanding challenge in the field [3, 2, 38]. While conceptually simple, behavior cloning has shown surprising promise on an array of real-world robot tasks, including manipulation [60, 13, 28, 27, 59, 37, 4] and autonomous driving [36, 6]. Current behavior cloning approaches can be categorized into two groups, depending on the policy's structure.

Explicit Policy. The simplest form of explicit policies maps from world state or observation directly to action [36, 60, 13, 41, 50, 37, 6]. They can be supervised with a direct regression loss and have efficient inference time with one forward pass. Unfortunately, this type of policy is not suitable for modeling multi-modal demonstrated behavior, and struggles with high-precision tasks [12]. A popular approach to model multimodal action distributions while maintaining the simplicity of direction action mapping is convert the regression task into classification by discretizing the action space [59, 56, 4]. However, the number of bins needed to approximate a continuous action space grows exponentially with increasing dimensionality. Another approach is to combine Categorical and Gaussian distributions to represent continuous multimodal distributions via the use of MDNs [5, 29] or clustering with offset prediction [42, 43]. Nevertheless, these models tend to be sensitive to hyperparameter tuning, exhibit mode collapse, and are still limited in their ability to express high-precision behavior [12].

Implicit Policy. Implicit policies [12, 22] define distributions over actions by using Energy-Based Models (EBMs) [24, 10, 8, 14, 11]. In this setting, each action is assigned an energy value, with action prediction corresponding to the optimization problem of finding a minimal energy action. Since different actions may be assigned low energies, implicit policies naturally represent multi-modal distributions. However, existing implicit policies [12] are unstable to train due to the necessity of drawing negative samples when computing the underlying Info-NCE loss.

Diffusion Models. Diffusion models are probabilistic generative models that iteratively refine randomly sampled noise into draws from an underlying distribution. They can also

be conceptually understood as learning the gradient field of an implicit action score and then optimizing that gradient during inference. Diffusion models [44, 18] have recently been applied to solve various different control tasks [20, 51, 1].

In particular, Janner et al. [20] and Huang et al. [19] explore how diffusion models may be used in the context of planning and infer a trajectory of actions that may be executed in a given environment. In the context of Reinforcement Learning, Wang et al. [53] use diffusion model for policy representation and regularization with state-based observations. In contrast, in this work, we explore how diffusion models may instead be effectively applied in the context of behavioral cloning for effective visuomotor control policy. To construct effective visuomotor control policies, we propose to combine DDPM's ability to predict high-dimensional action squences with closed-loop control, as well as a new transformer architecture for action diffusion and a manner to integrate visual inputs into the action diffusion model.

Wang et al. [54] explore how diffusion models learned from expert demonstrations can be used to augment classical explicit polices without directly taking advantage of diffusion models as policy representation.

Concurrent to us, Pearce et al. [34], Reuss et al. [39] and Hansen-Estruch et al. [16] has conducted a complimentary analysis of diffusion-based policies in simulated environments. While they focus more on effective sampling strategies, leveraging classifier-free guidance for goal-conditioning as well as applications in Reinforcement Learning, and we focus on effective action spaces, our empirical findings largely concur in the simulated regime. In addition, our extensive real-world experiments provide strong evidence for the importance of a receding-horizon prediction scheme, the careful choice between velocity and position control, and the necessity of optimization for real-time inference and other critical design decisions for a physical robot system.

VIII. LIMITATIONS AND FUTURE WORK

Although we have demonstrated the effectiveness of diffusion policy in both simulation and real-world systems, there are limitations that future work can improve. First, our implementation inherits limitations from behavior cloning, such as suboptimal performance with inadequate demonstration data. Diffusion policy can be applied to other paradigms, such as reinforcement learning [54, 16], to take advantage of suboptimal and negative data. Second, diffusion policy has higher computational costs and inference latency compared to simpler methods like LSTM-GMM. Our action sequence prediction approach partially mitigates this issue, but may not suffice for tasks requiring high rate control. Future work can exploit the latest advancements in diffusion model acceleration methods to reduce the number of inference steps required, such as new noise schedules[7], inference solvers[23], and consistency models [47].

IX. CONCLUSION

In this work, we assess the feasibility of diffusion-based policies for robot behaviors. Through a comprehensive evaluation of 12 tasks in simulation and the real world, we demonstrate that diffusion-based visuomotor policies consistently and definitively outperform existing methods while also being stable and easy to train. Our results also highlight critical design factors, including receding-horizon action prediction, end-effector position control, and efficient visual conditioning, that is crucial for unlocking the full potential of diffusion-based policies. While many factors affect the ultimate quality of behavior-cloned policies — including the quality and quantity of demonstrations, the physical capabilities of the robot, the policy architecture, and the pretraining regime used — our experimental results strongly indicate that policy structure poses a significant performance bottleneck during behavior cloning. We hope that this work drives further exploration in the field into diffusion-based policies and highlights the importance of considering all aspects of the behavior cloning process beyond just the data used for policy training.

X. ACKNOWLEDGEMENT

REFERENCES

[1] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022. 11

[2] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57 (5):469–483, 2009. 10

[3] Christopher G Atkeson and Stefan Schaal. Robot learning from demonstration. In *ICML*, volume 97, pages 12–20, 1997. 10

[4] Yahav Avigal, Lars Berscheid, Tamim Asfour, Torsten Kröger, and Ken Goldberg. Speedfolding: Learning efficient bimanual folding of garments. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2022. 10

[5] Christopher M Bishop. *Mixture density networks*. Aston University, 1994. 10

[6] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 10

[7] Ting Chen. On the importance of noise scheduling for diffusion models. *arXiv preprint arXiv:2301.10972*, 2023. 11

[8] Bo Dai, Zhen Liu, Hanjun Dai, Niao He, Arthur Gretton, Le Song, and Dale Schuurmans. Exponential family estimation via adversarial dynamics embedding. *Advances in Neural Information Processing Systems*, 32, 2019. 10

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 8

[10] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019. 10

[11] Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. *arXiv preprint arXiv:2012.01316*, 2020. 1, 5, 10

[12] 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 *5th Annual Conference on Robot Learning*, 2021. 1, 2, 4, 5, 6, 7, 8, 10

[13] Peter Florence, Lucas Manuelli, and Russ Tedrake. Self-supervised correspondence in visuomotor policy learning. *IEEE Robotics and Automation Letters*, 5(2):492–499, 2019. 4, 7, 10

[14] Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, and Richard Zemel. Learning the stein discrepancy for training and evaluating energy-based models without sampling. In *International Conference on Machine Learning*, 2020. 10

[15] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019. 2, 6, 7

[16] Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023. 11

[17] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020. 4

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020. 1, 2, 4, 11

[19] Siyuan Huang, Zan Wang, Puhao Li, Baoxiong Jia, Tengyu Liu, Yixin Zhu, Wei Liang, and Song-Chun Zhu. Diffusion-based generation, optimization, and planning in 3d scenes. *arXiv preprint arXiv:2301.06015*, 2023. 11

[20] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022. 3, 11

[21] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 17–23 Jul 2022. 3

[22] Daniel Jarrett, Ioana Bica, and Mihaela van der Schaar. Strictly batch imitation learning by energy-based distribution matching. *Advances in Neural Information Processing Systems*, 33:7354–7365, 2020. 10

[23] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *arXiv preprint arXiv:2206.00364*, 2022. 11

[24] Yann LeCun, Sumit Chopra, Raia Hadsell, Fu Jie Huang, and et al. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press, 2006. 10

[25] Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. *arXiv preprint arXiv:2004.08249*, 2020. 4

[26] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. Compositional visual generation with composable diffusion models. *arXiv preprint arXiv:2206.01714*, 2022. 5

[27] Ajay Mandlekar, Fabio Ramos, Byron Boots, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Dieter Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020. 4, 7, 10

[28] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Silvio Savarese, and Li Fei-Fei. Learning to generalize across long-horizon tasks from human demonstrations. *arXiv preprint arXiv:2003.06085*, 2020. 4, 7, 10

[29] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *5th Annual Conference on Robot Learning*, 2021. 1, 2, 4, 6, 7, 10, 14

[30] David Q Mayne and Hannah Michalska. Receding horizon control of nonlinear systems. In *Proceedings of the 27th IEEE Conference on Decision and Control*, pages 464–465. IEEE, 1988. 3

[31] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. In *6th Annual Conference on Robot Learning*, 2022. 8

[32] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2011. 1

[33] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages

8162–8171. PMLR, 2021. 4, 14

[34] Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, et al. Imitating human behaviour with diffusion models. *arXiv preprint arXiv:2301.10677*, 2023. 11

[35] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 3

[36] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988. 10

[37] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3758–3765. IEEE, 2018. 10

[38] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems*, 3:297–330, 2020. 10

[39] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023. 11

[40] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 3

[41] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011. 10

[42] Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 1, 2, 3, 4, 6, 7, 10

[43] Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. In *Conference on robot learning*. PMLR, 2018. 10

[44] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2015. 11

[45] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. 4, 14, 15

[46] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. 1, 5

[47] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023. 11

[48] Duy-Nguyen Ta, Eric Cousineau, Huihua Zhao, and Siyuan Feng. Conditional energy-based models for implicit policies: The gap between theory and practice. *arXiv preprint arXiv:2207.05824*, 2022. 5

[49] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020. 3

[50] Sam Toyer, Rohin Shah, Andrew Critch, and Stuart Russell. The magical benchmark for robust imitation. *Advances in Neural Information Processing Systems*, 33: 18284–18295, 2020. 10

[51] Julen Urain, Niklas Funk, Georgia Chalvatzaki, and Jan Peters. Se (3)-diffusionfields: Learning cost functions for joint grasp and motion optimization through diffusion. *arXiv preprint arXiv:2209.03855*, 2022. 11

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3

[53] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022. 11

[54] 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. URL https://openreview.net/forum?id=AHvFDPi-FA. 11

[55] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011. 2

[56] Jimmy Wu, Xingyuan Sun, Andy Zeng, Shuran Song, Johnny Lee, Szymon Rusinkiewicz, and Thomas Funkhouser. Spatial action maps for mobile manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020. 1, 10

[57] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 4

[58] Jingyun Yang, Junwu Zhang, Connor Settle, Akshara Rai, Rika Antonova, and Jeannette Bohg. Learning periodic tasks from human demonstrations. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8658–8665. IEEE, 2022. 10

[59] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, pages 726–747. PMLR, 2021. 10

[60] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5628–5635. IEEE, 2018. 4, 7, 10

[61] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019. 14

APPENDIX

A. Normalization

Properly normalizing action data is critical to achieve best performance for Diffusion Policy. Scaling the min and max of each action dimension independently to $[-1, 1]$ works well for most tasks. Since DDPMs clip prediction to $[-1, 1]$ at each iteration to ensure stability, the common zero-mean unit-variance normalization will cause some region of the action space to be inaccessible. When the data variance is small (e.g., near constant value), shift the data to zero-mean without scaling to prevent numerical issues. We leave action dimensions corresponding to rotation representations (e.g. Quaternion) unchanged.

B. Rotation Representation

For all environments with velocity control action space, we followed the standard practice [29] to use 3D axis-angle representation for the rotation component of action. Since velocity action commands are usually close to 0, the singularity and discontinuity of the axis-angle representation don't usually cause problems. We used the 6D rotation representation proposed in Zhou et al. [61] for all environments (real-world and simulation) with positional control action space.

C. Image Augmentation

Following Mandlekar et al. [29], we employed random crop augmentation during training. The crop size for each task is indicated in Tab. VI. During inference, we take a static center crop with the same size.

D. Hyperparameters

Hyerparameters used for Diffusion Policy on both simulation and realworld benchmarks are shown in Tab. VI and Tab. VII. Since the Block Push task uses a Markovian scripted oracle policy to generate demonstration data, we found its optimal hyper parameter for observation and action horizon to be very different from other tasks with human teleop demostrations.
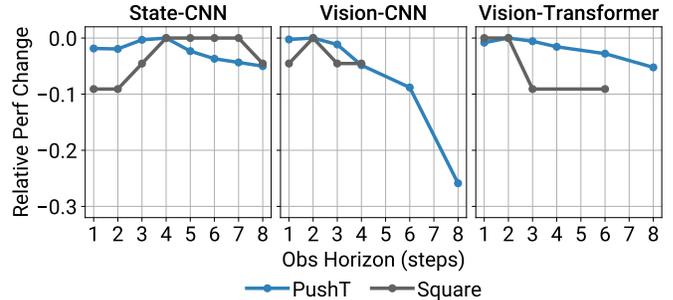

Fig. 12: **Observation Horizon Ablation Study.** State-based Diffusion Policy is not sensitive to observation horizon. Vision-based Diffusion Policy prefers low but > 1 observation horizon, with 2 being a good compromise for most tasks.
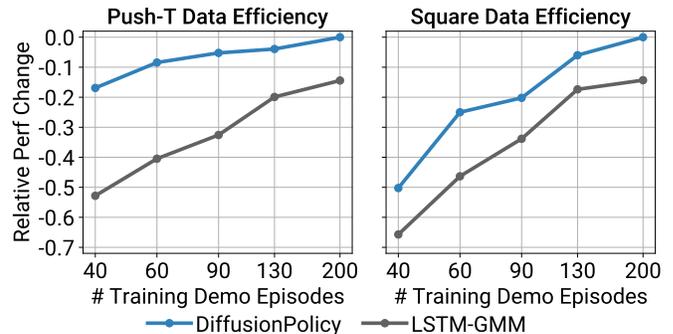

Fig. 13: **Data Efficiency Ablation Study.** Diffusion Policy outperforms LSTM-GMM [29] at every training dataset size.

We found that the optimal hyperparameters for CNN-based Diffusion Policy are consistent across tasks. In contrast, transformer-based Diffusion Policy's optimal attention dropout rate and weight decay varies greatly across different tasks. During tuning, we found increasing the number of parameters in CNN-based Diffusion Policy always improves performance, therefore the optimal model size is limited by the available compute and memory capacity. On the other hand, increasing model size for transformer-based Diffusion Policy (in particular number of layers) hurts performance sometimes. For CNN-based Diffusion Policy, We found using FiLM conditioning to pass-in observations is better than impainting on all tasks except Push-T. Performance reported for DiffusionPolicy-C on Push-T in Tab. I used impaiting instead of FiLM.

On simulation benchmarks, we used the iDDPM algorithm [33] with the same 100 denoising diffusion iterations for both training and inference. We used DDIM [45] on realworld benchmarks to reduce the inference denoising iterations to 16 therefore reducing inference latency.

We used batch size of 256 for all state-based experiments and 64 for all image-based experiments. For learning-rate scheduling, we used cosine schedule with linear warmup. CNN-based Diffusion Policy is warmed up for 500 steps while Transformer-based Diffusion Policy is warmed up for 1000 steps.

E. Data Efficiency

We found Diffusion Policy to outperform LSTM-GMM [29] at every training dataset size, as shown in Fig. 13.

H-Param	Ctrl	To	Ta	Tp	ImgRes	CropRes	#D-Params	#V-Params	Lr	WDecay	D-Iters Train	D-Iters Eval
Lift	Pos	2	8	16	2x84x84	2x76x76	256	22	1e-4	1e-6	100	100
Can	Pos	2	8	16	2x84x84	2x76x76	256	22	1e-4	1e-6	100	100
Square	Pos	2	8	16	2x84x84	2x76x76	256	22	1e-4	1e-6	100	100
Transport	Pos	2	8	16	4x84x85	4x76x76	264	45	1e-4	1e-6	100	100
ToolHang	Pos	2	8	16	2x240x240	2x216x216	256	22	1e-4	1e-6	100	100
Push-T	Pos	2	8	16	1x96x96	1x84x84	256	22	1e-4	1e-6	100	100
Block Push	Pos	3	1	12	N/A	N/A	256	0	1e-4	1e-6	100	100
Kitchen	Pos	2	8	16	N/A	N/A	256	0	1e-4	1e-6	100	100
Real Push-T	Pos	2	6	16	2x320x240	2x288x216	67	22	1e-4	1e-6	100	16
Real Pour	Pos	2	8	16	2x320x240	2x288x216	67	22	1e-4	1e-6	100	16
Real Spread	Pos	2	8	16	2x320x240	2x288x216	67	22	1e-4	1e-6	100	16
Real Mug Flip	Pos	2	8	16	2x320x240	2x288x216	67	22	1e-4	1e-6	100	16

TABLE VI: **Hyperparameters for CNN-based Diffusion Policy** Ctrl: position or velocity control To: observation horizon Ta: action horizon Tp: action prediction horizon ImgRes: environment observation resolution (Camera views x W x H) CropRes: random crop resolution #D-Params: diffusion network number of parameters in millions #V-Params: vision encoder number of parameters in millions Lr: learining rate WDecay: weight decay D-Iters Train: number of training diffusion iterations D-Iters Eval: number of inference diffusion iterations (enabled by DDIM [45])

H-Param	Ctrl	To	Ta	Tp	#D-params	#V-params	#Layers	Emb Dim	Attn Dropout	Lr	WDecay	D-Iters Train	D-Iters Eval
Lift	Pos	2	8	10	9	22	8	256	0.3	1e-4	1e-3	100	100
Can	Pos	2	8	10	9	22	8	256	0.3	1e-4	1e-3	100	100
Square	Pos	2	8	10	9	22	8	256	0.3	1e-4	1e-3	100	100
Transport	Pos	2	8	10	9	45	8	256	0.3	1e-4	1e-3	100	100
ToolHang	Pos	2	8	10	9	22	8	256	0.3	1e-4	1e-3	100	100
Push-T	Pos	2	8	16	9	22	8	256	0.01	1e-4	1e-1	100	100
Block Push	Vel	3	1	5	9	0	8	256	0.3	1e-4	1e-3	100	100
Kitchen	Pos	4	8	16	80	0	8	768	0.1	1e-4	1e-3	100	100
Real Push-T	Pos	2	6	16	80	22	8	768	0.3	1e-4	1e-3	100	16

TABLE VII: **Hyperparameters for Transformer-based Diffusion Policy** Ctrl: position or velocity control To: observation horizon Ta: action horizon Tp: action prediction horizon #D-Params: diffusion network number of parameters in millions #V-Params: vision encoder number of parameters in millions Emb Dim: transformer token embedding dimension Attn Dropout: transformer attention dropout probability Lr: learining rate WDecay: weight decay (for transformer only) D-Iters Train: number of training diffusion iterations D-Iters Eval: number of inference diffusion iterations (enabled by DDIM [45])

F. Observation Horizon

We found state-based Diffusion Policy to be insensitive to observation horizon, as shown in Fig. 12. However, vision-based Diffusion Policy, in particular the variant with CNN backbone, see performance decrease with increasing observation horizon. In practice, we found an observation horizon of 2 is good for most of the tasks for both state and image observations.

G. Performance Improvement Calculation

For each task i (column) reported in Tab. I, Tab. II and Tab. IV (mh results ignored), we find the maximum performance for baseline methods $max\_baseline_i$ and the maximum performance for Diffusion Policy variant (CNN vs Transformer) max_ours_i. For each task, the performance improvement is calculated as $improvement_i = \frac{max\_ours_i - max\_baseline_i}{max\_baseline_i}$ (positive for all tasks). Finally, the average improvement is calculated as $avg\_improvement = \frac{1}{N} \sum_N^i improvement_i = 0.46858 \approx 46.9\%$.

H. Push-T

1) Demonstrations: 136 demonstrations are collected and used for training. The initial condition is varied by randomly pushing or tossing the T block onto the table. Prior to this data collection session, the operator has performed this task for many hours and should be considered proficient at this task.

2) Evaluation: We used a fixed training time of 12 hours for each method, and selected the last checkpoint for each, with the exception of IBC, where the checkpoint with minimum training set action prediction MSE error due to IBC's training stability issue. The difficulty of training and checkpoint selection for IBC is demonstrated in main text Fig. 7.

Each method is evaluated for 20 episodes, all starting from the same set of initial conditions. To ensure the consistency of initial conditions, we carefully adjusted the pose of the T block and the robot according to overlayed images from the top-down camera.

Each evaluation episode is terminated by either keeping the end-effector within the end-zone for more than 0.5 second, or by reaching the 60 sec time limit.

The IoU metric is directly computed in the top-down camera pixel space.

I. Sauce Pouring and Spreading

1) Demonstrations: 50 demonstrations are collected, and 90% are used for training for each task. For pouring, initial locations of the pizza dough and sauce bowl are varied. After each demonstration, sauce is poured back into the bowl, and the dough is wiped clean. For spreading, location of the

pizza dough as well as the poured sauce shape are varied. For resetting, we manually gather sauce towards the center of the dough, and wipe the remaining dough clean. The rotational components for tele-op commands are discarded during spreading and sauce transferring to avoid accidentally scooping or spilling sauce.

2) Evaluation: Both Diffusion Policy and LSTM-GMM are trained for 1000 epochs. The last checkpoint is used for evaluation.

Each method is evaluated from the same set of random initial conditions, where positions of the pizza dough and sauce bowl are varied. We use a similar protocol as in **Push-T** to set up initial conditions. We do not try to match initial shape of poured sauce for spreading. Instead, we make sure the amount of sauce is fixed during all experiments.

The evaluation episodes are terminated by moving the spoon upward (away form the dough) for 0.5 seconds, or when the operator deems the policy's behavior is unsafe.

The coverage metric is computed by first projecting the RGB image from both the left and right cameras onto the table space through homography, then computing the coverage in each projected image. The maximum coverage between the left and right cameras is reported.

3) UR5 robot station: Experiments for the **Push-T** task are performed on the UR5 robot station.

The UR5 robot accepts end-effector space positional command at 125Hz, which is linearly interpolated from the 10Hz command from either human demonstration or the policy. The interpolation controller limits the end-effector velocity to be below 0.43 m/s and its position to be within the region 1cm above the table for safety reason. Position-controlled policies directly predicts the desired end-effector pose, while velocity-controlled policies predicts the difference the current positional setpoint and the previous setpoint.

The UR5 robot station has 5 realsense D415 depth camera recording 720p RGB videos at 30fps. Only 2 of the cameras are used for policy observation, which are down-sampled to 320x240 at 10fps.

During demonstration, the operator teleoperates the robot via a 3dconnexion SpaceMouse at 10Hz.

J. Franka robot station

Experiments for **Sauce Pouring and Spreading** tasks are performed on the Franka robot station.

A custom mid-level controller is implemented to generate desired joint positions from desired end effector poses from the learned policies. At each time step, we solve a differential kinematics problem (formulated as a Quadratic Program) to compute the desired joint velocity to track the desired end effector velocity. The resulting joint velocity is Euler integrated into joint position, which is tracked by a joint level controller on the robot. This formulation allows us to impose constraints such as collision avoidance, safety region for end effector and joint limits. It also enables regulating redundant DoF in the null space of the end effector commands. This mid-level controller is particularly valuable for safeguarding the learned policy during hardware deployment.

The operator uses a SpaceMouse input device to control robot's end effector, and the gripper is controlled by a trigger button. Tele-op and learned policies run at 10Hz, and the mid-level controller runs around 1kHz. Desired end effector pose commands are interpolated by the mid-level controller. This station has 2 realsense D415 RGBD camera streaming VGA RGB images at 30fps, which are downsampled to 320x240 at 10fps as input to the learned policies.