Mini Diffuser: Accelerating Diffusion Policy Optimization via Two-Level Minibatching

Authors

***Abstract*—Diffusion policies has been established as a dominant paradigm for robotic Imitation learning (IL). However, the high computational cost of training these models remains a bottleneck for scaling to multi-task regimes and performing efficient online adaptation via reinforcement learning (RL). We present Mini-Diffuser, a method that reduces the time and memory required to train vision-language robotic diffusion policies by an order of magnitude.**

Our approach exploits a fundamental asymmetry in action diffusion: while image diffusion targets high-dimensional outputs, action generation targets a comparatively low-dimensional space where only the visual condition is high-dimensional. By introducing two-level minibatching, Mini-Diffuser pairs multiple noised action samples with a single vision-language condition. This decoupling allows for significantly more efficient gradient steps during both supervised imitation learning and iterative RL optimization.

In PushT Benchmark environment, a Level-2 batch equipped Diffusion Policy can reach the same performance using 60% less training time. In RLBench simulations, Mini-Diffuser achieves 95% of the performance of state-of-the-art multi-task policies while using only 5% of the training time and 7% of the memory. By drastically lowering the resource requirements for diffusion-based gradients, Mini-Diffuser provides a practical path for the pretraining, continuous fine-tuning, and real-world adaptation of foundation models at the intersection of IL and RL.

I. INTRODUCTION

Diffusion models [1] have emerged as powerful generative tools due to their ability to model complex, multimodal distributions using iterative denoising processes. Initially popularized in image generation tasks [2], diffusion models have recently demonstrated significant potential in decision-making areas like robotic control [3], [4], [5], [6], [7], [8], showing competitive performance in both simulated benchmarks and real-world applications.

Despite their success, using diffusion models for action generation have a major limitation: they inherently require multiple denoising steps with condition-dependent predictions, leading to high computational costs during training and inference. Recent methods, such as DDIM [9], consistency models [10], and flow-matching [11], have successfully reduced inference complexity by collapsing or skipping denoising steps. However, *training* still requires sampling all noise levels thoroughly, posing a significant challenge for generalist agents. Such agents need to efficiently scale to diverse tasks, instructions, and observation modalities.

Compared to task-specific diffusion policies, generalist agents typically need much larger models and training datasets with much more training steps, increasing training costs considerably. This challenge has been clearly shown in recent works such as Pi-Zero [12], and 3D Diffuser Actor [7], where training can take multiple days on clusters with multiple GPUs—similar to general-purpose image generators like Stable Diffusion [2].

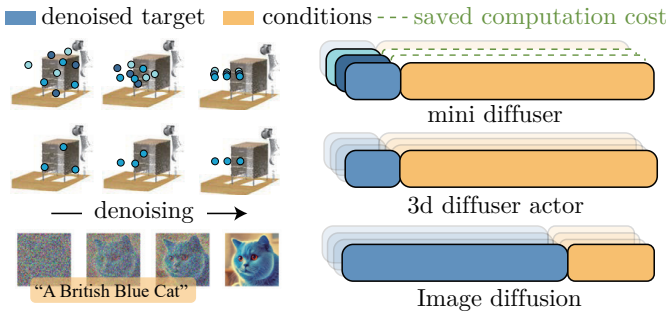

Fig. 1. **Difference between image diffusion (bottom), state-of-art action diffusion [7] (middle) and our mini-diffuser (top).** In image diffusion, a semantically meaningful image is denoised from random pixels. At token level, the denoised target (the image) dominates token space. By contrast, in action diffusion, structured and meaningful actions are denoised from random samples, and the denoised target lie in a low-dimensional vector space relative to the conditioning vision and language data. In our Mini Diffuser (top), we re-use the same condition for learning-phase denoising of multiple action samples, yielding a per-sample computation and memory cost that is significantly lower than 3D diffuser actor.

We identify a critical but often overlooked asymmetry between robotic policy learning and image generation. In image generation, the condition (e.g., a text prompt) is typically smaller and simpler than the output (high-dimensional pixels). In contrast, robotic action generation usually has conditions (rich multimodal robot states including visual features, proprioception, and language instructions) that are much larger and more complex than the relatively low-dimensional action outputs.

This imbalance offers a unique opportunity to improve training efficiency. Specifically, during training, conditions remain the same across multiple noise-level predictions within a given context. Leveraging this, we propose *two-level batching*, a novel yet simple sampling strategy that pairs multiple action samples with each condition, to enhance sample efficiency significantly. However, applying this strategy directly would cause redundant computations, as traditional network architectures would repeatedly process the same condition for each prediction.

To address this, we introduce a non-invasive mini-diffuser architecture, which employs masked global attention, sample-wise adaptive normalization, and local kernel-based feature fusion. These carefully selected modules avoid inter-sample dependencies, enabling the processing of large flattened two-level batches without additional memory usage or computational overhead. Consequently, we significantly scale effective

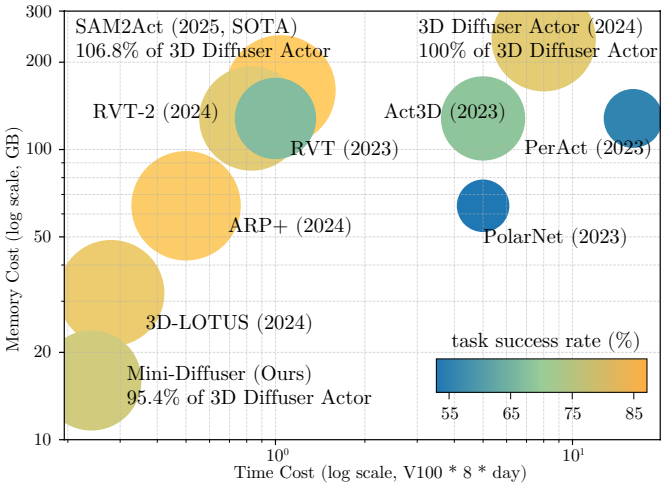

Fig. 2. **Comparison with state of the arts in RLbench Peract-18 benchmark.** Our method by far takes least the time and memory to train, while maintaining 95% of the performance of current SOTA diffusion-based models.

batch sizes and reduce the number of gradient updates necessary for convergence.

In summary, our contributions are:

- **Two-level Batch Sampling for Condition-Element Asymmetry:** We formalize the asymmetry in diffusion-based policy training and introduce a two-level batching method that significantly speeds up training by exploiting shared conditions.
- **Non-invasive Mini-Diffuser Architecture:** We design a compact diffusion policy architecture composed of non-invasive, condition-invariant layers, enabling efficient processing of two-level batches. This approach maintains most of the expressiveness of full-scale 3D diffusion policies while dramatically reducing training time and computational resources.

Thanks to these improvements, we achieve by far the lowest training cost for high-capacity multitask diffusion policies while sacrificing only about 5% of performance compared with current SOTA [7]. Training efficiency comparisons using a unified time standard are highlighted in Figure 2. Notably, our model can be trained end-to-end on a consumer-level GPU, such as an RTX 4090, in just 13 hours—while existing diffusion and non-diffusion methods typically require multiple GPUs and days of training. Real-world experiments further confirm that our approach preserves the robust multimodal action-generation capabilities that diffusion models are known for, ensuring reliable performance across diverse perceptual inputs.

II. Related Works

A. Robot Learning from demonstration

Earlier works on learning from demonstrations train deterministic policies with behavior cloning [13], mapping observations directly to actions. To better capture multimodal action distributions, later approaches discretize the action space and apply cross-entropy losses [14], [15], [16], or leverage generative models such as VAEs [17], and diffusion models [4], [6]. Autoregressive training [18], [19] and pretrained foundition models [20] are also used for better capture spatial and temporal features.

The other line of work is trying to broaden a single model's capability. Multi-task policies aim to generalize across variations of the same set if tasks or appearance changes in the environment. By incorporating multi-view perception together with language instructions, C2F-ARM [21] and PerAct [22] voxelize the workspace to localize target keyposes, while Act3D [23] avoids voxelization by sampling 3D points and applying cross-attention to physical scene features. Robotic View Transformer (RVT) series [24], [25] improve 3D scene understanding by projecting RGB-D inputs into multiple views and lifting them into 3D space. To even go beyond training task sets and environment, Generalist policies such as RT-X series [26], [27], [28], Octo [29] and OpenVLA [30] adopt large transformer-based architectures to directly predict low-level actions from raw visual input. These models show strong scalability and task coverage, but often rely on massive datasets and train time to implicitly learn state and action representation together. While a single task policy can be trained within hours [15], [4], multi-task policies normally take days [22], and generalist policies take weeks on a cluster of cards [30], [12].

B. Diffusion policies and their extensions

Recently, diffusion models have been increasingly explored in the field of embodied AI, showing promising progress in tasks that require nuanced decision-making and adaptive control. Early works [3], [31], [4] demonstrated the effectiveness of diffusion models in low-dimensional control settings. Building on this, more recent efforts have extended diffusion-based approaches to complex 3D robotic manipulation tasks [6], [7], achieving performance that surpasses traditional architectures.

Despite their success, applying diffusion models in 3D robotic domains presents significant challenges. These tasks involve intricate spatial representations and demand high-frequency decision-making, which conflicts with the inherently iterative and computationally intensive nature of diffusion-based training and denoising processes. Several methods propose skipping inference steps via hierarchical sampling [32], [33], or try replacing diffusion models with Consistency Models [34], [11]. Though these method mitigate inference time efficiency. The training cost remains high, especially in the multi-task training setting.

III. Diffusion Model Formulation and Training

A multi-task robotic manipulation policy aims to predict an action vector $\boldsymbol{a}$ conditioned on the current state $\boldsymbol{s}$. To train such a policy, we use expert demonstrations in the form of temporally ordered state-action sequences $\{(\boldsymbol{s}_0, \boldsymbol{a}_0), \ldots, (\boldsymbol{s}_t, \boldsymbol{a}_t)\}$, consistent with prior work in multimodal imitation learning.

Each state $\boldsymbol{s}$ consists of a combination of modalities, including RGB-D images with known camera poses, proprioceptive signals such as joint angles and end-effector velocities, and task-specific language instruction. These components may be

sampled from a single timestep or a short history temporal window.

Each action $\boldsymbol{a}$ defines a low-level end-effector command or a short sequence of future commands. It is represented as a tuple:

$$\boldsymbol{a} = (\boldsymbol{a}_{\text{pos}}, \boldsymbol{a}_{\text{rot}}, a_{\text{open}}) \in \mathbb{R}^3 \times \mathbb{SO}(3) \times \{0, 1\}, \tag{1}$$

where $\boldsymbol{a}_{\text{pos}}$ is the 3D position, $\boldsymbol{a}_{\text{rot}}$ is the 3D rotation, and a_{open} is the gripper open/close flag.

A. Conditional Diffusion Model Formulation

For simplicity and generality, we omit the real-world time index t of $\boldsymbol{a}$ and $\boldsymbol{s}$ to avoid confusion with the denoising step index k used in diffusion.

We aim to model the conditional probability distribution $p(\boldsymbol{a}|\boldsymbol{s})$ via a diffusion model. Given action-state pairs $(\boldsymbol{a}_0, \boldsymbol{s}) \sim q(\boldsymbol{a}, \boldsymbol{s})$, the forward diffusion process is defined as:

$$q(\boldsymbol{a}_k|\boldsymbol{a}_0, \boldsymbol{s}) = \mathcal{N}(\boldsymbol{a}_k; \sqrt{\bar{\alpha}_k}\boldsymbol{a}_0, (1 - \bar{\alpha}_k)\boldsymbol{I}), \tag{2}$$

where $k \in \{1, \ldots, K\}$, $\alpha_k = 1 - \beta_k$, $\bar{\alpha}_k = \prod_{j=1}^{k} \alpha_k$, and $\{\beta_j\}$ is a noise schedule defined by a pre-specified function [1] with their correspond terms σ_j used for denoising. Although $\boldsymbol{s}$ does not affect the forward process directly, we include it for clarity, since our goal is to learn the conditional distribution $p(\boldsymbol{a}|\boldsymbol{s})$. As $k \to K$, the distribution $q(\boldsymbol{a}_K|\boldsymbol{a}_0)$ approaches a standard Gaussian $\mathcal{N}(0, \mathbf{I})$, ensuring that we can begin the reverse generation from pure white noise, and progressively apply the reverse diffusion steps conditioned on $\boldsymbol{s}$:

$$\boldsymbol{a}_{k-1} = \frac{1}{\sqrt{\alpha_k}} \left(\boldsymbol{a}_k - \frac{1 - \alpha_k}{\sqrt{1 - \bar{\alpha}_k}} \epsilon_\theta(\boldsymbol{a}_k, k, \boldsymbol{s}) \right) + \sigma_k z, \tag{3}$$

where $z \sim \mathcal{N}(0, \mathbf{I})$, $k = K, ..., 1$, and $\epsilon_\theta(\cdot)$ is a neural network parameterized by θ. This iterative reverse process yields the final predicted action $\boldsymbol{a}_0$ conditioned on the state $\boldsymbol{s}$.

We train our model by minimizing the conditional denoising objective:

$$L(\theta) = \mathbb{E}_{(\boldsymbol{a}_0, \boldsymbol{s}), k, \epsilon} \left[\|\epsilon - \epsilon_\theta(\boldsymbol{a}_k, k, \boldsymbol{s})\|^2 \right]. \tag{4}$$

This objective teaches the model to predict the noise ϵ that was added to $\boldsymbol{a}_0$ during the forward process, thereby enabling accurate recovery of $\boldsymbol{a}_0$ during the reverse process.

In practice, training is performed over mini-batches. For a mini-batch of N samples $\{(\boldsymbol{a}_0^{(i)}, \boldsymbol{s}^{(i)}, k^{(i)}, \epsilon^{(i)})\}_{i=1}^{N}$, the empirical loss becomes:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left\| \epsilon^{(i)} - \epsilon_\theta(\boldsymbol{a}_k^{(i)}, k^{(i)}, \boldsymbol{s}^{(i)}) \right\|^2. \tag{5}$$

B. Two-level Mini-batch Sampling

As mentioned earlier, a unique characteristic of robotic policy learning is the discrepancy in dimensionality between actions and states: $\dim(\boldsymbol{a}) \ll \dim(\boldsymbol{s})$. This motivates a specialized sampling strategy, which we call Two-level Batching, where multiple noise-level predictions are computed under shared state conditions. Our mini-batch is organized into two nested levels:

Level-1: We first sample B independent state–action pairs:

$$\{(\boldsymbol{s}^{(i)}, \boldsymbol{a}_0^{(i)})\}_{i=1}^{B}, \quad (\boldsymbol{s}^{(i)}, \boldsymbol{a}_0^{(i)}) \sim q(\boldsymbol{a}, \boldsymbol{s}). \tag{6}$$

Level-2: For each of those B pairs (indexed by i), we independently draw M noise-and-step pairs,

$$\{(k^{(i,j)}, \epsilon^{(i,j)})\}_{j=1}^{M}, k^{(i,j)} \sim \mathcal{U}(1, K), \epsilon^{(i,j)} \sim \mathcal{N}(0, \mathbf{I}). \tag{7}$$

This means that for each state $\boldsymbol{s}_i$, we generate M noised action samples. Therefore once Level-1 and Level-2 samplings are complete, we have $B \cdot M$ total noisy action samples. For each (i, j):

$$\boldsymbol{a}_{k^{(i,j)}} = \sqrt{\bar{\alpha}_{k^{(i,j)}}}\boldsymbol{a}_0^{(i)} + \sqrt{1 - \bar{\alpha}_{k^{(i,j)}}}\epsilon^{(i,j)}, \tag{8}$$

where $\bar{\alpha}_k$ is the cumulative noise schedule we mentioned earlier.

The final training loss becomes:

$$L(\theta) = \frac{1}{B \cdot M} \sum_{i=1}^{B} \sum_{j=1}^{M} \left\| \epsilon^{(i,j)} - \epsilon_\theta(\boldsymbol{a}_{k^{(i,j)}}, k^{(i,j)}, \boldsymbol{s}^{(i)}) \right\|^2, \tag{9}$$

where all level-2 samples $\boldsymbol{a}_{k^{(i,j)}}$ within the same level-1 batch index i share the same state condition $\boldsymbol{s}^{(i)}$. The key objective is to approximate the learning effect of having $N = B \cdot M$ independently sampled $\{(\boldsymbol{a}_{k^{[n]}}, \boldsymbol{s}^{[n]})\}_{n=1}^{N}$ from $q(\boldsymbol{a}, \boldsymbol{s})$ with random noise, while incurring only the cost of processing B unique state conditions. In the next section, we describe the network architecture designed to support this efficient reuse of condition encoding.

IV. Model Architecture Design

To support two-level batching, our model processes all noised action samples and shared condition information in a single forward pass. After project them into feature space of dimension d, we concatenate per-sample action tokens and condition tokens into a flattened sequence:

$$\boldsymbol{Z} = [\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \ldots, \boldsymbol{z}^{(M)}, \boldsymbol{h}_{\text{vis}}^{(\text{share})}, \boldsymbol{h}_{\text{ctx}}^{(\text{share})}]. \tag{10}$$

Each $\boldsymbol{z}^{(m)} \in \mathbb{R}^{L \times d}$ is the token sequence of the m-th noised action sample, with a sequence length of L. The shared visual condition tokens $\boldsymbol{h}_{\text{vis}}^{(\text{share})} \in \mathbb{R}^{N \times d}$ are projected from RGB-D pixels lifted into 3D space, and $\boldsymbol{h}_{\text{ctx}}^{(\text{share})} \in \mathbb{R}^{C \times d}$ represent non-spatial features such as language and proprioception.

We use a Transformer-style architecture in which the entire sequence $\boldsymbol{Z}$ is linearly projected into queries $\boldsymbol{Q}$, keys $\boldsymbol{K}$, and values $\boldsymbol{V}$. For spatial tokens (actions and 3D visual points), we apply 3D rotary positional encoding (RoPE)[35][23] to capture relative spatial relationships. For context tokens, we add a learned modality-specific embedding.

A standard Transformer architecture uses multi-layer self-attention, which enables global information sharing but introduces a risk of information leakage across independently sampled action sequences. This is especially problematic under two-level batching, where each sample must remain isolated. To address this, we replace standard attention layers with specialized non-invasive modules that allow efficient condition querying while preserving isolation between noised samples.

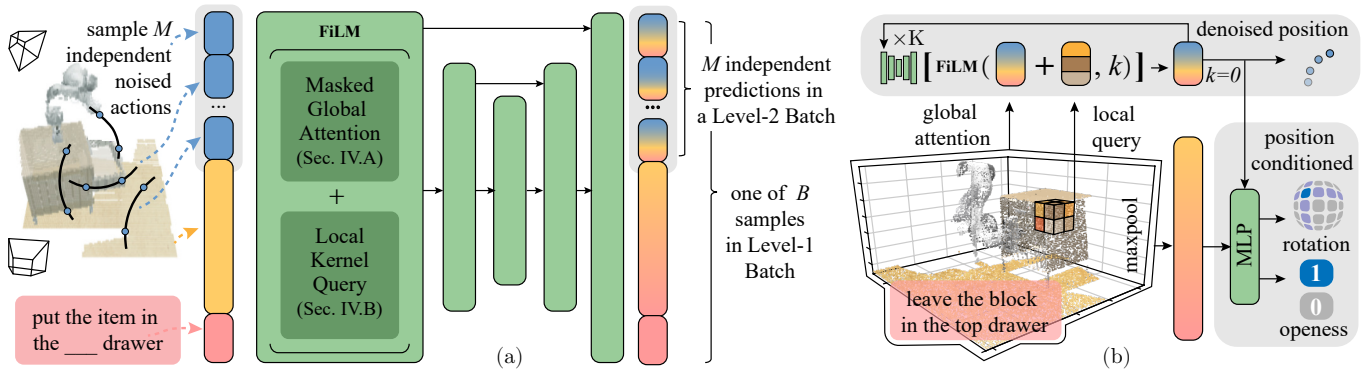

Fig. 3. **Mini-diffuser model structure.** (a) During training, we first construct a level-1 batch from B samples of the state, then assemble a level-2 batch by sampling M noised actions independently under each state condition. Tokens are then flattened and fed into a multi-layer model that contains a Masked attention module, a local query module, and FiLM layers. (b) During inference, we apply denoising only to the end-effector position. Rotation and gripper openness are predicted separately via classification heads conditioned on the final denoised position.

A. Masked Global Attention

We apply a combination of self- and cross-attention using masked attention, which enables selective communication and avoids information leakage. The masked attention is defined as:

$$\begin{aligned} \boldsymbol{Z}' &= \text{MaskedAttn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) \\ &= \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top - \text{Inf} \cdot (1 - \boldsymbol{M}_{\text{QK}})}{\sqrt{d}}\right)\boldsymbol{V}, \end{aligned} \tag{11}$$

where $\boldsymbol{M}_{\text{QK}}$ is a binary mask matrix that defines which tokens may attend to which others, and Inf is a large constant used to suppress masked entries in the attention logits.

The masking matrix $\boldsymbol{M}_{\text{QK}}$ is constructed to satisfy: (i) An action sample attends to itself and shared conditions, but not to other action samples, and (ii) shared conditions do not attend back to action samples. The structure of the mask is:

$$\boldsymbol{M}_{\text{QK}} = \begin{bmatrix} \boldsymbol{I}_{M\times M} \otimes \boldsymbol{J}_{L\times L} & \boldsymbol{J}_{(M\cdot L)\times(N+C)} \\ \boldsymbol{0}_{(N+C)\times(M\cdot L)} & \boldsymbol{J}_{(N+C)\times(N+C)} \end{bmatrix}, \tag{12}$$

where $\boldsymbol{J}$ is an all-ones matrix, $\boldsymbol{I}_{M\times M} \otimes \boldsymbol{J}_{L\times L}$ constructs M diagonal intra-sample blocks of size $L\times L$ using the Kronecker product $\otimes$. The top-right block enables all action tokens to perform cross-attention to condition tokens, while the bottom-left zero block prevents condition tokens from attending back to samples.

This masked attention mechanism is central to enabling efficient two-level batching, allowing all samples to share condition processing while maintaining proper sample-wise independence during training.

B. Local Kernel-Based Query

While masked attention captures the global structure, we also introduce a local feature aggregation module to reinforce spatial grounding by leveraging nearby 3D geometry. This module further guides the denoising process toward target regions based on fine-grained scene details. Although local aggregation can be regarded as a subset of the global attention weights that the model may eventually approximate, explicitly introducing this inductive bias at an early stage promotes training stability and accelerates convergence.

This operation is non-invasive: it retrieves spatial context from the environment without modifying shared condition features, making it fully compatible with two-level batching. During training, each noised sample gathers local cues around its position. During inference, these query locations gradually shift toward the target end-effector position as the denoising process progresses.

C. Per-Sample Modulation Conditioned on Noise Step

To allow the model to adapt to different stages of the denoising process, we apply per-sample modulation using Feature-wise Linear Modulation (FiLM) [36]. For each sample at a diffusion timestep k, we embed the timestep index and transform it into a pair of scale $\boldsymbol{\gamma}$ and shift vectors $\boldsymbol{\beta}$ using a lightweight MLP. These vectors are then applied to intermediate features within the network using affine transformation. Formally, the FiLM layer modulates a feature vector $\boldsymbol{z}$ as:

$$\text{FiLM}(\boldsymbol{z}; \boldsymbol{\gamma}, \boldsymbol{\beta}) = \boldsymbol{\gamma} \odot \boldsymbol{z} + \boldsymbol{\beta}. \tag{13}$$

This simple yet effective mechanism allows the network to dynamically adjust its behavior for different noise levels—handling coarse predictions at early timesteps and refining details at later ones. Crucially, FiLM is applied independently to each sample, ensuring that no information is shared within a level-2 batch.

D. Design Choices

We explore some variants that integrate our three non-invasive building blocks—masked attention, local kernel–based query, and FiLM modulation—into transformer-style diffusion models. The first variant serves as a lightweight modified baseline which we used for ablation, while the second, which we refer to as the *Mini-Diffuser*, represents our fully optimized architecture.

1) Minimal Modifications to Diffusion Policy: For the standard Diffusion Policy, which employs a 1D convolutional U-Net, the required modifications are remarkably minimal. Because the original network inherently utilizes sample-independent Group Normalization and element-wise FiLM

modulation, its residual blocks naturally prevent information leakage among independent action samples. To enable two-level batching, we simply broadcast the extracted visual state condition across the M independent noise samples prior to the U-Net forward pass. This straightforward adjustment allows the lightweight 1D action decoder to process the expanded mini-batch in parallel, result in a 60% reduced training time remaining the same performance of original Diffusion Policy.

2) Minimal Modifications to 3D Diffuse Actor: In the simplest setup, we replace the self-attention modules in each transformer layer of 3D Diffuse Actor [7] with our non-invasive counterparts. This drop-in replacement preserves the original layer-wise architecture and requires no additional structural changes. Importantly, this modification alone enables two-level batching during training, allowing us to isolate and evaluate the resulting memory and computational savings.

3) Mini-Diffuser: To further improve parameter training efficiency, we adopt the U-Net-style architecture of Point Transformer v3 (PTv3) [37]. PTv3 uses transformer layers within a U-Net framework [38], combining downsampling and upsampling stages to compute compact 3D-aware latent features. This hierarchical structure reduces memory usage and computation in the deeper middle layers. Additionally, we can cache the point indices used during down-sampling, allowing us to accelerate the local kernel query among point neighborhoods described in Sec. IV-B. Despite inheriting the PTv3 backbone, we replace all internal transformer blocks with non-invasive counterparts. As a result, we cannot directly use pretrained PTv3 weights and the training time saving does not come from pretrained weights.

We also adopt a decoupled action head. Since 3D RoPE applies only to spatial position coordinates, end-effector rotation and gripper states are not spatially aligned with the point-based token structure and may introduce noise if fused too early. Unlike 3D Diffuser Actor, which embeds all action components into a single denoised token, we adopt a decoupled design following prior works [25], [3]: Denoising is applied only to the end-effector position. Rotation and gripper state are predicted separately via classification heads (with cross-entropy loss), conditioned on the final denoised position. This design preserves the multimodal nature of action generation, as the model can flexibly associate different discrete rotations or gripper commands with the different predicted position.

V. Experiments

We evaluate our Mini-Diffuser for multi-task robotic manipulation from demonstrations in both simulation and real-world settings. Our primary simulated benchmark is RLBench [39], a widely adopted platform for vision-language manipulation tasks. Our experiments aim to answer the following questions: (1) How does Mini-Diffuser perform compared to state-of-the-art methods? (2) How do our proposed architectural design choices contribute to training acceleration and sample efficiency? (3) Can Mini-Diffuser maintain competitive task performance despite significantly reduced training time and resources?

A. Simulation Benchmark

1) Datasets: We evaluate Mini-Diffuser on the multi-task RLBench benchmark proposed by PerAct [22], consisting of 18 tasks and 249 task variations. These tasks require generalization across diverse goal configurations including object types, colors, shapes, and spatial arrangements.

2) Results: Table I summarizes the results across all tasks. Mini-Diffuser is trained using Level-1 batch size $B = 100$ and Level-2 batch size $M = 64$, and achieves a strong average success rate while drastically reducing computational cost. Specifically, it reaches **95.6%** of the average task performance of 3D Diffuse Actor using only **4.8%** of its training time and **6.6%** of its memory consumption.

B. Ablation Study

We assess the impact of Mini-Diffuser's core components through ablation experiments, summarized in Table II. Unless otherwise specified, all models are trained with a level-1 batch size of $B = 100$ and a level-2 batch size of $M = 64$, on a subset of RLBench: *Stack Block*, *Slide Color*, and *Turn Tap*.

1) Effect of Two-level Batching.: The primary innovation of Mini-Diffuser is the two-level batching strategy, which increases effective sample coverage without proportional increases in memory or compute. We evaluate performance under varying M. When $M = 1$, two-level batching is disabled and training defaults to conventional sampling.

Fig. 4 illustrates learning curves for different M values. A larger M accelerates convergence. This highlights that even without architectural changes, our two-level batching strategy alone yields substantial efficiency gains, though the benefit saturates around $M = 128$. We attribute this to two factors: (i) Too large batches reduce gradient variance and diminish the stochasticity that benefits generalization; (ii) Level-2 batches reuse the same condition across samples, limiting diversity relative to fully independent samples.

Table II further compares per-step memory and compute costs under different batching configurations. First, in rows 1 and 2 of the table, we quantify the effect of a 64-fold difference in level-2 batch size M, keeping the level-1 size at $B = 100$. A 64-fold increase of M yields 64 times more training samples being processed per step. It leads to a steep rise in success rate (44% to 78%), with a comparably modest 2% increase in memory usage and 6% increase in compute time. Achieving a batch size of 6400 through level-1 alone is not possible.

Next, with rows 2 and 3 of Table II, we quantify the effect of doubling the level-1 batch size B, keeping the level-1 size M at 1. A two-fold increase of B nearly yields a doubling of both memory and computation costs — an expected result of scaling real batch size. The effect on success rate is comparatively small.

VI. Discussion and Conclusion

Mini-Diffuser revisits diffusion-based policy learning with a focus on efficiency and practicality. Contrary to the common belief that diffusion models are slower than non-diffusion counterparts, our results show that with the right architectural

TABLE I

MULTI-TASK RLBENCH BENCHMARK RESULTS. METRICS INCLUDE TASK SUCCESS RATE (%), NORMALIZED TRAINING TIME (V100×8×DAYS), AND MEMORY USAGE (GB). NUMBERS IN GREY FACILITATE THE COMPARISON OF MINI DIFFUSER AND 3D DIFFUSE ACTOR: MINI-DIFFUSER ACHIEVES 95.4% OF 3D DIFFUSE ACTOR'S SUCCESS RATE, WITH A SUBSTANTIALLY LOWER COMPUTE COST.

Method	Avg. Suc. (%)	Norm. Time	Memory (GB)	Close Jar	Drag Stick	Insert Peg	Meat Grill	Open Drawer	Place Cups	Place Wine	Push Buttons	Put in Cup
PerAct	49.4	128	128	55.2 ± 4.7	89.6 ± 4.1	5.6 ± 4.1	70.4 ± 2.0	88.0 ± 5.7	2.4 ± 3.2	44.8 ± 7.8	92.8 ± 3.0	28.0 ± 4.4
RVT	62.9	8	128	52.0 ± 2.5	92.2 ± 1.6	11.0 ± 4.0	88.0 ± 2.5	71.2 ± 6.9	4.0 ± 2.5	91.0 ± 5.2	100.0 ± 0.0	49.6 ± 3.2
Act3D	63.2	40	128	96.8 ± 3.2	80.8 ± 6.4	24.0 ± 8.4	95.2 ± 1.6	78.4 ± 11.2	3.2 ± 3.2	59.2 ± 9.8	92.8 ± 3.0	67.2 ± 3.9
RVT-2	81.4	6.6	128	100.0 ± 3.6	97.2 ± 1.6	4.2 ± 1.2	99.0 ± 1.7	74.0 ± 6.9	14.0 ± 2.8	95.0 ± 3.2	100.0 ± 0.0	66.0 ± 4.5
3D-Dif-Actor	81.3(100%)	39(100%)	240(100%)	96.0 ± 2.5	100.0 ± 0.0	65.6 ± 4.1	96.8 ± 1.6	89.6 ± 4.1	24.0 ± 7.6	93.6 ± 4.8	98.4 ± 2.0	85.0 ± 4.1
SAM2Act	86.8	8.3	160	99.0 ± 2.0	99.0 ± 2.0	84.0 ± 5.7	98.0 ± 2.3	83.0 ± 6.0	47.0 ± 6.0	93.0 ± 3.8	100.0 ± 0.0	75.0 ± 3.8
Mini-diffuser	77.6(95.4%)	**1.9**(4.8%)	**16**(6.6%)	**98.7 ± 0.5**	97.3 ± 0.5	**68.0 ± 1.5**	**100.0 ± 0.0**	85.3 ± 3.7	16.0 ± 1.6	93.3 ± 2.1	**100.0 ± 0.0**	73.3 ± 5.4

Method	reported hardware	Put in Drawer	Put in Safe	Screw Bulb	Slide Color	Sort Shape	Stack Blocks	Stack Cups	Sweep Dust	Turn Tap
PerAct	V100×8 × 16days	51.2 ± 4.7	84.0 ± 3.6	17.6 ± 2.0	74.0 ± 13.6	16.8 ± 4.7	26.4 ± 3.2	2.4 ± 2.0	52.0 ± 0.0	88.0 ± 4.4
RVT	V100×8 × 1day	88.0 ± 5.7	91.2 ± 3.0	48.0 ± 4.9	81.6 ± 2.8	36.0 ± 2.5	28.8 ± 3.9	26.4 ± 2.4	72.0 ± 0.0	93.6 ± 4.1
Act3D	V100×8 × 5days	91.2 ± 6.2	95.2 ± 4.0	32.8 ± 6.9	96.0 ± 2.5	29.6 ± 3.2	4.0 ± 3.6	6.3 ± 2.0	86.4 ± 6.5	94.4 ± 2.0
RVT-2	V100×8 × 20hours	92.0 ± 0.0	96.0 ± 1.8	88.0 ± 4.9	81.0 ± 4.8	35.0 ± 7.1	80.0 ± 2.8	69.0 ± 5.9	100.0 ± 0.0	99.0 ± 1.7
3D-Dif-Act	A100×6 × 6days	96.0 ± 3.6	97.6 ± 2.0	82.4 ± 4.2	97.6 ± 3.2	44.0 ± 4.4	68.3 ± 3.3	47.2 ± 8.5	84.0 ± 4.4	99.2 ± 1.6
SAM2Act	H100×8 × 12hours	99.0 ± 2.0	98.0 ± 2.3	89.0 ± 2.3	86.0 ± 4.0	64.0 ± 4.6	76.0 ± 8.6	78.0 ± 4.0	99.0 ± 2.0	96.0 ± 5.7
Mini-diffuser	4090×13hours or A100×1day	96.0 ± 4.8	94.7 ± 0.4	77.3 ± 3.7	**98.7 ± 0.1**	28.0 ± 6.0	38.7 ± 4.0	**48.0 ± 1.1**	94.7 ± 0.9	89.3 ± 1.0

TABLE II

ABLATION ON TWO-LEVEL BATCHES AND MODEL COMPONENTS.

	memory cost	time cost per gradient step	Avg. suc. after 10^5 steps
$B = 100$, $M = 64$	102.2%	106.3%	78.3
$B = 100$, $M = 1$	100%	100%	44.1
$B = 200$, $M = 1$	188.8%	176.6%	50.8
w.o. 3D ROPE	101.2%	101.3%	67.8
w.o. PTv3 backbone	147.7%	125.5%	79.1
w.o. local kernel	102.2%	106.2%	77.9

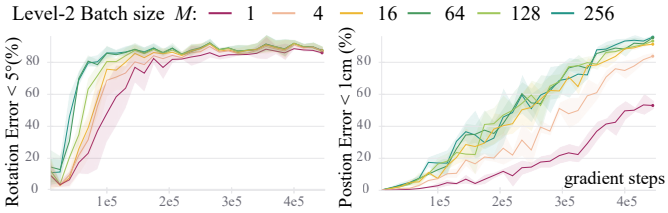

Fig. 4. **Policy Learning Efficiency.** The y-axis shows the proportion of generated actions with errors below specified thresholds (i.e., $<$ 1cm or $<5°$), indicating successful diffusion. Increasing the level-2 batch size accelerates convergence at the same number of gradient steps (x-axis).

design and batch sampling strategy, training time can be drastically reduced. While inference remains iterative, our architecture is compatible with step-skipping techniques like DDIM [9] or Flow Matching [11], which can further reduce runtime during deployment. Another potential improvement can be addressing the limitations shared by most 3D-based manipulation policies, including reliance on camera calibration and depth input, and a focus only on quasi-static tasks, by extending Mini-Diffuser to dynamic settings with velocity control. To better handle fine-grained tasks, the end-effector rotation can be predicted jointly with the position using a regression-based loss [7], rather than our current separate classification-based method. Extending the idea of Mini Diffuser further, large vision–language–action (VLA) models [12], [29] that employ denoising-based action experts could similarly adopt a two-level batching scheme, conditioning multiple action samples on the same vision–language KV cache to effectively broaden the training batch size.

Overall, Mini-Diffuser provides a fast, simple, and scalable baseline for multi-task manipulation. It can not only serve as a practical recipe for efficient policy training, but also has the potential to become a flexible platform for rapid experimentation and future research in architecture design, training strategies, and real-world robotic generalization.

REFERENCES

[1] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851.

[2] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 684–10 695.

[3] A. Ajay, Y. Du, A. Gupta, J. B. Tenenbaum, T. S. Jaakkola, and P. Agrawal, "Is conditional generative modeling all you need for decision making?" in *The Eleventh International Conference on Learning Representations*, Sep. 2022.

[4] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *Int. J. Rob. Res.*, p. 2783649241273668, Oct. 2024.

[5] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *Proceedings of the 39th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 162. PMLR, 2022, pp. 9902–9915.

[6] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3D diffusion policy: Generalizable visuomotor policy learning via simple 3D representations," in *2nd Workshop on Dexterous Manipulation: Design, Perception and Control (RSS)*, Jul. 2024.

[7] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki, "3D Diffuser Actor: Policy Diffusion with 3D Scene Representations," *Conference on Robot Learning*, vol. abs/2402.10885, Feb. 2024.

[8] K. Saha, V. Mandadi, J. Reddy, A. Srikanth, A. Agarwal, B. Sen, A. Singh, and M. Krishna, "Edmp: Ensemble-of-costs-guided diffusion for motion planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 10 351–10 358.

[9] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *International Conference on Learning Representations*, Oct. 2020.

[10] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, "Consistency models," in *Proceedings of the 40th International Conference on Machine Learning*. PMLR, Jul. 2023, pp. 32 211–32 252.

[11] Q. Zhang, Z. Liu, H. Fan, G. Liu, B. Zeng, and S. 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*, vol. 39, 2025, pp. 14 754–14 762.

[12] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky, "pi0: A vision-language-action flow model for general robot control," Nov. 2024.

[13] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Adv. Neural Inf. Process. Syst.*, vol. 1, 1988.

[14] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, and V. Sindhwani, "Transporter networks: Rearranging the visual world for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2021, pp. 726–747.

[15] S. Chen, R. Garcia, C. Schmid, and I. Laptev, "PolarNet: 3D Point Clouds for Language-Guided Robotic Manipulation," *Conference on Robot Learning*, vol. abs/2309.15596, Sep. 2023.

[16] R. Garcia, S. Chen, and C. Schmid, "Towards generalizable vision-language robotic manipulation: A benchmark and LLM-guided 3D policy," *Arxiv.org*, vol. abs/2410.1345, Oct. 2024.

[17] A. Mandlekar, F. Ramos, B. Boots, S. Savarese, L. Fei-Fei, A. Garg, and D. 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, pp. 4414–4420.

[18] X. Zhang, Y. Liu, H. Chang, L. Schramm, and A. Boularias, "Autoregressive action sequence learning for robotic manipulation," *IEEE Robot. Autom. Lett.*, 2025.

[19] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," Apr. 2023.

[20] H. Fang, M. Grotz, W. Pumacay, Y. R. Wang, D. Fox, R. Krishna, and J. Duan, "SAM2Act: Integrating visual foundation model with a memory architecture for robotic manipulation," Feb. 2025.

[21] S. James, K. Wada, T. Laidlow, and A. J. Davison, "Coarse-to-fine Q-attention: Efficient learning for visual robotic manipulation via discretisation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 739–13 748.

[22] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 785–799.

[23] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki, "Act3D: 3D feature field transformers for multi-task robotic manipulation," in *Proceedings of the 7th Conference on Robot Learning*. PMLR, Dec. 2023, pp. 3949–3965.

[24] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox, "Rvt: Robotic view transformer for 3d object manipulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 694–710.

[25] A. Goyal, V. Blukis, J. Xu, Y. Guo, Y.-W. Chao, and D. Fox, "RVT-2: Learning precise manipulation from few demonstrations," in *RSS 2024 Workshop: Data Generation for Robotics*, Jul. 2024.

[26] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, "Rt-1: Robotics transformer for real-world control at scale," in *Proceedings of the Robotics: Science and Systems (RSS)*, Jul. 2023.

[27] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, Q. Vuong, V. Vanhoucke, H. T. Tran, R. Soricut, A. Singh, J. Singh, P. Sermanet, P. R. Sanketi, G. Salazar, M. S. Ryoo, K. Reymann, K. Rao, K. Pertsch, I. Mordatch, H. Michalewski, Y. Lu, S. Levine, L. Lee, T.-W. E. Lee, I. Leal, Y. Kuang, D. Kalashnikov, R. Julian, N. J. Joshi, A. Irpan, B. Ichter, J. Hsu, A. Herzog, K. Hausman, K. Gopalakrishnan, C. Fu, P. Florence, C. Finn, K. A. Dubey, D. Driess, T. Ding, K. M. Choromanski, X. Chen, Y. Chebotar, J. Carbajal, N. Brown, A. Brohan, M. G. Arenas, and K. Han, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Proceedings of the Conference on Robot Learning (CoRL)*, ser. Proceedings of Machine Learning Research, vol. 229. PMLR, Nov. 2023, pp. 2165–2183.

[28] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, and A. Jain, "Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6892–6903.

[29] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y. L. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine, "Octo: An open-source generalist robot policy," May 2024.

[30] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn, "OpenVLA: An open-source vision-language-action model," in *8th Annual Conference on Robot Learning*, Sep. 2024.

[31] M. Reuss, M. Li, X. Jia, and R. Lioutikov, "Goal-conditioned imitation learning using score-based diffusion policies," 2023.

[32] Z. Xian and N. Gkanatsios, "Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation," in *Conference on Robot Learning/Proceedings of Machine Learning Research*. Proceedings of Machine Learning Research, 2023.

[33] X. Ma, S. Patidar, I. Haughton, and S. James, "Hierarchical diffusion policy for kinematics-aware multi-task robotic manipulation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 18 081–18 090.

[34] Y. Chen, H. Li, and D. Zhao, "Boosting continuous control with consistency policy," in *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '24. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2024, pp. 335–344.

[35] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "RoFormer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, p. 127063, Feb. 2024.

[36] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[37] X. Wu, L. Jiang, P.-S. Wang, Z. Liu, X. Liu, Y. Qiao, W. Ouyang, T. He, and H. Zhao, "Point transformer v3: Simpler faster stronger," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 4840–4851.

[38] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[39] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3019–3026, 2020.