CARP: Visuomotor Policy Learning via Coarse-to-Fine Autoregressive Prediction

Zhefei Gong[1] Pengxiang Ding[12†] Shangke Lyu[1] Siteng Huang[12]
Mingyang Sun[12] Wei Zhao[1] Zhaoxin Fan[3] Donglin Wang[1✉]
[1]Westlake University [2]Zhejiang University
[3]Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing
{gongzhefei, dingpengxiang, lyushangke, huangsiteng, wangdonglin}@westlake.edu.cn
https://carp-robot.github.io

Abstract

*In robotic visuomotor policy learning, diffusion-based models have achieved significant success in improving the accuracy of action trajectory generation compared to traditional autoregressive models. However, they suffer from inefficiency due to multiple denoising steps and limited flexibility from complex constraints. In this paper, we introduce **C**oarse-to-Fine **A**uto**R**egressive **P**olicy (**CARP**), a novel paradigm for visuomotor policy learning that redefines the autoregressive action generation process as a coarse-to-fine, next-scale approach. CARP decouples action generation into two stages: first, an action autoencoder learns multi-scale representations of the entire action sequence; then, a GPT-style transformer refines the sequence prediction through a coarse-to-fine autoregressive process. This straightforward and intuitive approach produces highly accurate and smooth actions, matching or even surpassing the performance of diffusion-based policies while maintaining efficiency on par with autoregressive policies. We conduct extensive evaluations across diverse settings, including single-task and multi-task scenarios on state-based and image-based simulation benchmarks, as well as real-world tasks. CARP achieves competitive success rates, with up to a 10% improvement, and delivers **10×** faster inference compared to state-of-the-art policies, establishing a high-performance, efficient, and flexible paradigm for action generation in robotic tasks.*

1. Introduction

Policy learning from demonstrations, formulated as the supervised regression task of mapping observations to actions, has proven highly effective across a wide range of robotic tasks, even in its simplest form. Replacing the traditional policies with generative models, particularly those stemming from the vision community, has opened new avenues for improving performance, enabling robots to achieve the high precision required for complex tasks in robotic scenarios.

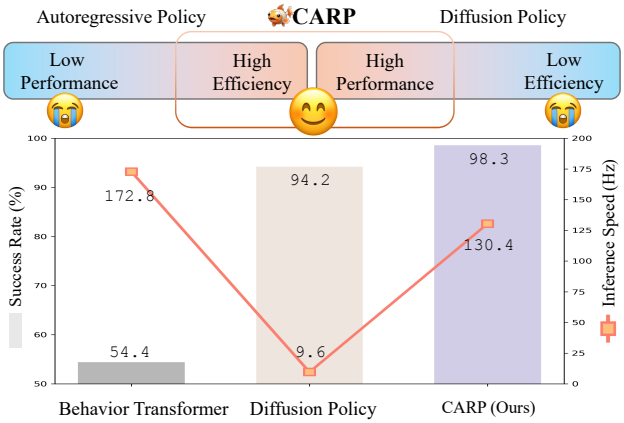
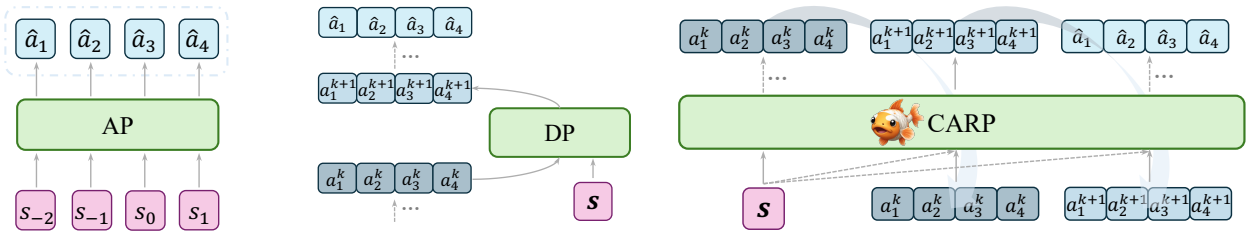


Figure 1. **Policy Comparison.** The representative performance among Behavior Transformer [54] served as an autoregressive policy, Diffusion Policy [12], and our approach in the state-based Robomimic Square task experiment. CARP demonstrates an effective balance between task performance and inference efficiency.

Existing approaches have explored different generative modeling techniques to address challenges in visuomotor policy learning. Autoregressive Modeling (AM) [13, 32, 54, 72] provides a straightforward and efficient out-of-the-box solution, benefitting from its scalability, flexibility, and mature exploration at lower computation requirements. However, AM's next-token prediction paradigm often fails to capture long-range dependencies, global structure, and temporal coherence [28], leading to poor performance, which is essential for many robotic tasks. Recently, Diffusion Modeling (DM) [12, 53, 63] has emerged as a promising alternative, bridging the precision gap in AM by modeling the gradient of the action score function to learn multimodal distributions. Nevertheless, DM requires multiple steps of sequential denoising, making it computationally prohibitive for robotic tasks, espe-

† Project lead. ✉ Corresponding author.



(a) Autoregressive Policy (b) Diffusion Policy (c) CARP (Ours)

Figure 2. **Structure of Current Policies.** $\hat{\boldsymbol{a}}$ is the predicted action, $\boldsymbol{a}^k$ denotes the refining action at step k, $\boldsymbol{s}$ is the historical condition. a) Autoregressive Policy predicts the action step-by-step in the next-token paradigm. b) Diffusion Policy models the noise process used to refine the action sequence. c) CARP refines action sequence predictions autoregressively from coarse to fine granularity.

cially for robotic tasks requiring efficient real-time inference in on-board compute-constrained environments. Additionally, DDPM [21]'s rigid generative process lacks flexibility and adaptability for tasks with long-term dependencies, often leading to cumulative errors and reduced robustness over extended time spans, limiting its use in dynamic settings.

Both AM and DM have their respective advantages and limitations, which are often orthogonal and difficult to balance in practical applications. In this work, we aim to resolve this trade-off by introducing a novel generative paradigm for robot visuomotor policy learning that predicts entire action sequences from a *coarse-to-fine* granularity in a *next-scale* prediction framework. This approach allows our model to achieve performance levels comparable to DM while retaining AM's inference efficiency and implementation flexibility.

Our primary contribution is the introduction of a **C**oarse-to-Fine **A**uto**R**egressive **P**olicy (**CARP**), a hybrid framework that combines AM's efficiency with DM's high performance to meet the demands of real-world robotic manipulation. Specifically, our main contributions are as follows:

- **Multi-Scale action tokenization:** We propose a multi-scale tokenization method for action sequences that captures the global structure and maintains temporal locality, effectively addressing AM's myopic limitations.
- **Coarse-to-Fine autoregressive prediction:** This mechanism refines action sequences in the latent space using Cross-Entropy loss with relaxed Markovian assumptions during iterations, achieving DM-like performance with high efficiency and comparable multi-modal behavior.
- **Comprehensive sim & real experiments:** Extensive experiments demonstrate CARP's effectiveness in both simulated and real-world robotic manipulation tasks.

In summary, we present CARP, a novel visuomotor policy framework that synergizes the strengths of AM and DM, offering high performance, efficiency, and flexibility.

2. Background

We start by providing the background of our approach, focusing on three key components: problem formulation, conventional autoregressive policies, and diffusion-based policies.

2.1. Problem Formulation

Problem formulation will consider a task $\mathcal{T}$, where there are N expert demonstrations $\{\tau_i\}_{i=1}^{N}$. Each demonstration τ_i is a sequence of state-action pairs. We formulate robot imitation learning as an action sequence prediction problem [12, 48, 63], training a model to minimize the error between the predicted future actions conditioned on historical states and the ground truth actions. Specifically, imitation learning minimize the behavior cloning loss $\mathcal{L}_{bc}$ formulated as

$$\mathcal{L}_{bc} = \mathbb{E}_{\boldsymbol{s},\boldsymbol{a}\sim\mathcal{T}}\left[\sum_{t=0}^{T}\mathcal{L}\left(\pi_{\boldsymbol{\theta}}(\hat{\boldsymbol{a}}_H|\boldsymbol{s}_O),\boldsymbol{a}_H\right)\right], \tag{1}$$

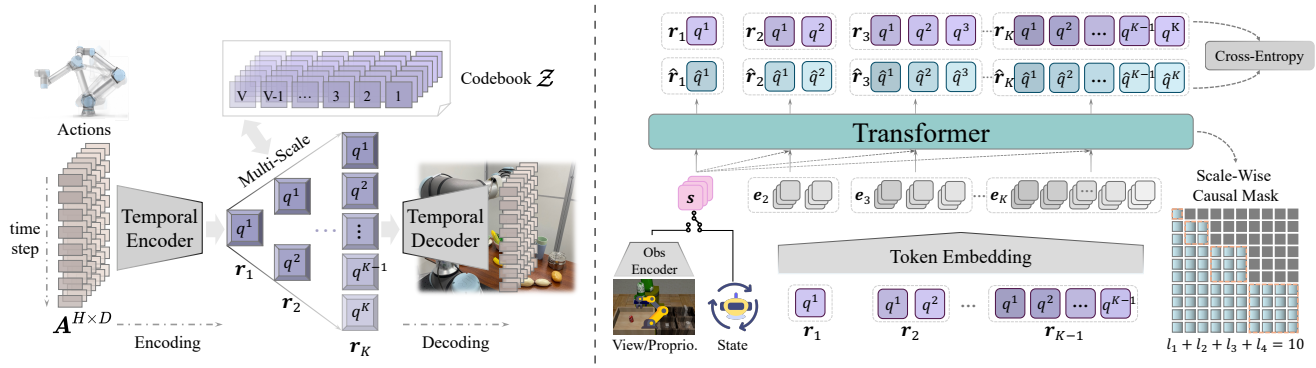
where $\boldsymbol{a}$ represents the action, $\boldsymbol{s}$ denotes the state or observation according to the specific task description, t is the current time step, H is the prediction horizon, and O is the historical horizon. For notational simplicity, we denote the action sequence $\boldsymbol{a}_{t:t+H-1}$ as $\boldsymbol{a}_H$ and the state sequence $\boldsymbol{s}_{t-O+1:t}$ as $\boldsymbol{s}_O$. Here, $\mathcal{L}_{bc}$ represents a supervised action prediction loss, such as mean squared error or negative log-likelihood, T is the length of the demonstration, and θ represents the learnable parameters of the policy network $\pi_{\boldsymbol{\theta}}$.

2.2. Autoregressive Policy

Autoregressive policies leverage the efficiency and flexibility of autoregressive models (GPT-style Decoders). Recent advancements, such as action chunking [54, 72], redefine this paradigm as a multi-token, one-pass prediction method (see Suppl. C for further details). We refer to this approach as Autoregressive Policy (AP). In this approach, the *next-token* posits the probability of observing the current action $\boldsymbol{a}_t$ depends solely on its previous states $\boldsymbol{s}_O$, which allows for the factorization of the likelihood of sequence with length H as

$$p\left(\boldsymbol{a}_t,\boldsymbol{a}_{t+1},...,\boldsymbol{a}_{t+H-1}\right) = \prod_{k=t}^{t+H-1} p\left(\boldsymbol{a}_k|\boldsymbol{s}_O\right). \tag{2}$$

However, it introduces several issues that hinder its performance. The linear, step-by-step unidirectional dependency of action prediction may overlook the global structure [28], making it challenging to capture long-range dependencies and holistic coherence or temporal locality [24] in complex scenes or sequences, which restricts their generalizability in



(a) Multi-Scale Action Tokenization

(b) Coarse-to-Fine Autoregressive Prediction

Figure 3. **Overview of the Two Stages of CARP.** a) A multi-scale action autoencoder extracts token maps $\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_K$ to represent the action sequence at different scales, trained using the standard VQVAE loss. b) The autoregressive prediction is reformulated as a *coarse-to-fine*, *next-scale* paradigm. The sequence is progressively refined from coarse token map $\boldsymbol{r}_1$ to finer granularity token map $\boldsymbol{r}_K$, where each $\boldsymbol{r}_k$ contains l_k tokens. An attention mask ensures that each $\boldsymbol{r}_k$ attends only to the preceding $\boldsymbol{r}_{1:k-1}$ during training. A standard Cross-Entropy loss is used for training. During inference, the token maps $\boldsymbol{r}_{1:K}$ are collectively decoded into continuous actions for execution.

tasks requiring bidirectional reasoning, as shown in Fig. 2a. For example, it cannot predict the former actions given the near-terminal state without backward reasoning.

2.3. Diffusion Policy

Diffusion-based policies [12] utilize Denoising Diffusion Probabilistic Models [21] to approximate the conditional distribution $p(\boldsymbol{a}_H|\boldsymbol{s}_O)$ instead of the joint distribution [24] $p(\boldsymbol{a}_H, \boldsymbol{s}_O)$, through modeling the noise during the denoising process from Gaussian noise to noise-free output as

$$\boldsymbol{a}_H^{k+1} = \alpha(\boldsymbol{a}_H^k - \gamma\epsilon_\theta(\boldsymbol{s}_O, \boldsymbol{a}_H^k, k) + \mathcal{N}), \tag{3}$$

where ϵ_θ is a learnable noise network, k is the current denoising step, $\mathcal{N}$ is Gaussian noise, and α, γ are hyper-parameters.

Diffusion-based policies show impressive performance in robotic manipulation tasks due to their action generation which gradually refines from random samples which we can abstract as a *coarse-to-fine* process. This kind of process alike human movement or natural thinking flows in line with human intuition. However, they suffer from poor convergence due to their design. To address this, multiple denoising steps are required, constrained by the Markovian assumption, where $\boldsymbol{a}_H^{k+1}$ depends only on the previous step $\boldsymbol{a}_H^k$ (see Fig. 2b) , which leads to significant runtime inefficiency. Moreover, they lack the ability to leverage generation context effectively, hindering scalability to complex scenes [17].

3. Method

To address the limitations of existing methods, we propose **C**oarse-to-Fine **A**uto**R**egressive **P**olicy (**CARP**), a novel visuomotor policy framework that combines the high performance of recent diffusion-based policies with the inference efficiency and flexibility of traditional autoregressive policies. CARP achieves these advantages through a redesigned autoregressive modeling strategy. Specifically, we shift from conventional *next-token* prediction to a *coarse-to-fine*, *next-scale* prediction approach, using *multi-scale* action representations. The training process of CARP consists of two stages: *multi-scale* action tokenization and *coarse-to-fine* autoregressive prediction, as detailed in Fig. 3.

In this section, we first explain the construction of *multi-scale* action token maps, followed by the *coarse-to-fine* autoregressive prediction approach for action generation. Key implementation details are provided at the end.

3.1. Multi-Scale Action Tokenization

Instead of focusing on individual action steps, we extract representations at multiple scales across the entire action sequence. We propose a novel *multi-scale* action quantization autoencoder that encodes a sequence of actions into K discrete token maps, $\boldsymbol{R} = (\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_K)$, which are used for training and inference. Our approach builds on the VQ-VAE architecture [62], incorporating a modified *multi-scale* quantization layer [60] to enable hierarchical encoding.

Encoder and Decoder. As illustrated in Fig. 3a, the actions are first organized into an action sequence $\boldsymbol{A}^{H\times D}$, where H denotes the prediction horizon and D represents the dimensionality of the action $\boldsymbol{a}$. Given that each dimension in the action space is orthogonal to the others, and that there exists a natural temporal dependency within an action sequence, we employ a 1D temporal convolutional network (1D-CNN) along the time dimension, similar to the architecture used in [24], for both the encoder $\mathcal{E}(\cdot)$ and decoder $\mathcal{D}(\cdot)$. Let $\boldsymbol{F} = \mathcal{E}(\boldsymbol{A}) \in \mathbb{R}^{L\times C}$, where L denotes the compressed length of the temporal dimension with $L \leq H$, and C represents the dimensionality of the feature map $\boldsymbol{F}$.

Quantization. We introduce a quantizer with a learnable codebook $\mathcal{Z} \in \mathbb{R}^{V\times C}$, containing V code vectors, each of dimension C. The quantization process (lines 4-11 in Algorithm 1) generates the action sequence representation

iteratively across multiple scales. At scale k, it produces action token map $\boldsymbol{r}_k$ to represent the sequence, which consists of l_k tokens q. In our implementation, we set $l_k = k$. The function $\text{Lookup}(\mathcal{Z}, v)$ retrieves the v-th code vector from $\mathcal{Z}$. The quantization function is defined by $\boldsymbol{r} = \mathcal{Q}(\boldsymbol{F})$ as

$$q = \arg\min_{v \in [V]} \text{dist}\left(\text{Lookup}(\mathcal{Z}, v), \boldsymbol{f}\right) \in [V], \tag{4}$$

where token q represents the nearest vector in $\mathcal{Z}$ for a given feature vector $\boldsymbol{f} \in \mathbb{R}^{1\times C}$ in the feature map $\boldsymbol{F}$, according to a distance function $\text{dist}(\cdot)$, such as Euclidean, cosine, etc.

Specifically, we adopt a residual-style design [31, 60] for feature maps $\boldsymbol{F}$ and $\hat{\boldsymbol{F}}$, as shown in lines 4-11 of Algorithm 1. This design ensures that each finer-scale representation $\boldsymbol{r}_k$ depends only on its coarser-scale predecessors $(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_{k-1})$, facilitating a *multi-scale* representation. A shared codebook $\mathcal{Z}$ is used across all scales, ensuring that *multi-scale* token maps $\boldsymbol{R} = (\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_K)$, where K is the number of scales, are drawn from a consistent vocabulary $[V]$. To preserve information during upsampling, we employ K additional 1D convolutional layers $\{\phi_k\}_{k=1}^K$ [60], as illustrated in lines 9-10 of Algorithm 1.

Loss. The final approximation $\hat{\boldsymbol{F}}$ of the original feature map $\boldsymbol{F}$ is composed residually of the *multi-scale* representations derived from the codebook $\mathcal{Z}$, based on all token maps $\{\boldsymbol{r}_k\}_{k=1}^K$. The reconstructed action sequence is then obtained through $\hat{\boldsymbol{A}} = \mathcal{D}(\hat{\boldsymbol{F}})$. To train the quantized autoencoder, a typical VQVAE [62] loss $\mathcal{L}$ is minimized as

$$\mathcal{L} = \underbrace{\|\boldsymbol{A} - \hat{\boldsymbol{A}}\|_2}_{\mathcal{L}_{\text{recon}}} + \underbrace{\|\text{sg}(\boldsymbol{F}) - \hat{\boldsymbol{F}}\|_2}_{\mathcal{L}_{\text{quant}}} + \underbrace{\|\boldsymbol{F} - \text{sg}(\hat{\boldsymbol{F}})\|_2}_{\mathcal{L}_{\text{commit}}}, \tag{5}$$

where $\mathcal{L}_{\text{recon}}$ minimizes the difference between the original action sequence $\boldsymbol{A}$ and the reconstruction $\hat{\boldsymbol{A}}$. $\text{sg}(\cdot)$ denotes stop-gradient. $\mathcal{L}_{\text{quant}}$ aligns the quantized feature map $\hat{\boldsymbol{F}}$ with the original $\boldsymbol{F}$, and $\mathcal{L}_{\text{commit}}$ encourages the encoder to commit to codebook entries, preventing codebook collapse. For $\mathcal{L}_{\text{quant}}$ and $\mathcal{L}_{\text{commit}}$, we calculate every residual calculating moment of each scale $\boldsymbol{r}_k$ as in Algorithm 1. After the training process, the autoencoder $\{\mathcal{E}, \mathcal{Q}, \mathcal{D}\}$ tokenizes actions for subsequent *coarse-to-fine* autoregressive modeling.

Discussion. The tokenization strategy described above allows the *multi-scale* tokens $\boldsymbol{R}$, extracted from the action sequence $\boldsymbol{A} \in \mathbb{R}^{H\times D}$ via temporal 1D convolutions, to inherently preserve temporal locality [24]. Additionally, the hierarchical extraction captures the global structure, enabling the model to treat the action sequence as a unified entity.

Unlike traditional autoregressive policies that predict each action token independently (see Eq. (2) and Fig. 2a), CARP leverages dual capabilities to capture both local temporal dependencies and global features across the entire action sequence. This approach produces smoother transitions and more stable action sequences. Through *multi-scale* encoding, CARP overcomes the short-sighted limitations of conventional autoregressive models, yielding more robust, coherent, and precise behaviors over extended time horizons.

Algorithm 1 Multi-Scale Action VQVAE

1: **Inputs:** Action sequence $\boldsymbol{A}$
2: **Hyperparameters:** Number of scales K, length of each scale $(l_k)_{k=1}^K$, length of feature map's temporal dimension L
3: **Initialize:** $\boldsymbol{F} \leftarrow \mathcal{E}(\boldsymbol{A})$, $\hat{\boldsymbol{F}} \leftarrow 0$, $R \leftarrow []$
4: **for** $k = 1$ **to** K **do**
5: $\quad \boldsymbol{r}_k \leftarrow \mathcal{Q}(\text{Interpolate}(\boldsymbol{F}, l_k))$
6: $\quad \boldsymbol{R} \leftarrow \boldsymbol{R} \cup \{\boldsymbol{r}_k\}$
7: $\quad \boldsymbol{Z}_k \leftarrow \text{Lookup}(\mathcal{Z}, \boldsymbol{r}_k)$
8: $\quad \boldsymbol{Z}_k \leftarrow \text{Interpolate}(\boldsymbol{Z}_k, L)$
9: $\quad \boldsymbol{F} \leftarrow \boldsymbol{F} - \phi_k(\boldsymbol{Z}_k)$
10: $\quad \hat{\boldsymbol{F}} \leftarrow \hat{\boldsymbol{F}} + \phi_k(\boldsymbol{Z}_k)$
11: **end for**
12: $\hat{\boldsymbol{A}} \leftarrow \mathcal{D}(\hat{\boldsymbol{F}})$
13: **Return:** Multi-scale token maps $\boldsymbol{R}$, reconstructed action sequence $\hat{\boldsymbol{A}}$

3.2. Coarse-to-Fine Autoregressive Prediction

Using *multi-scale* action sequence representations, we shift from traditional *next-token* prediction to a *next-scale* prediction approach, progressing from coarse to fine granularity.

Prediction. Given the *multi-scale* representation token maps $(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_K)$ produced by the trained autoencoder, where each $\boldsymbol{r}_k$ encodes the same action sequence $\boldsymbol{A}$ at a different scale, the autoregressive likelihood is formulated as

$$p(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_K) = \prod_{k=1}^{K} p(\boldsymbol{r}_k \mid \boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_{k-1}; \boldsymbol{s}_O), \tag{6}$$

where each autoregressive unit $\boldsymbol{r}_k \in [V]^{l_k}$ is the token map at scale k containing l_k tokens q. The predix sequence $(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_{k-1})$ is served as the condition for $\boldsymbol{r}_k$, accompanying with the historical state sequence $\boldsymbol{s}_O$. This kind of *next-scale* prediction methodology is what we define as *coarse-to-fine* autoregressive prediction in CARP.

Due to the residual-style quantization, during autoregressive prediction, we first embed the previous scale token map $\boldsymbol{r}_{k-1}$ and then reconstruct the next scale feature map $\boldsymbol{e}_k$, as shown in Fig. 3b. This feature map $\boldsymbol{e}_k$ is then used as input for predicting the next scale token map $\boldsymbol{r}_k$. The token maps $\boldsymbol{r}_{1:K}$ are decoded by the *multi-scale* autoencoder into continuous actions, following the residual-style process.

Loss. During the k-th autoregressive step, all distributions over the l_k tokens in $\boldsymbol{r}_k$ will be generated in parallel, with the *coarse-to-fine* dependency ensured by a block-wise causal attention mask [60]. To optimize the autoregressive model, we utilize the standard Cross-Entropy loss to capture the difference between the predicted token map $\hat{\boldsymbol{r}}$ and the token map $\boldsymbol{r}$ from the ground truth action sequence as

$$\mathcal{L}_{\text{Cross-Entropy}} = \sum_{k=1}^{K} \sum_{i=1}^{l_k} \left[-\sum_{v=1}^{V} \boldsymbol{r}_k^{i,v} \log \hat{\boldsymbol{r}}_k^{i,v} \right], \tag{7}$$

where l_k is the length of each scale, V is the size of the action dictionary $\mathcal{Z}$, and K is the number of scales.

Discussion. CARP models the entire trajectory holistically, progressively refining actions from high-level intentions to fine-grained details (see Fig. 2c). This hierarchical paradigm more closely resembles natural human behavior, where movements are guided by overarching goals and gradually refined, rather than being planned step by step.

Instead of the commonly used MSE loss [6, 61, 72], CARP employs Cross-Entropy loss (Eq. (7)) to preserve multimodality naturally, as MSE tends to enforce a unimodal distribution that can be detrimental to robotic tasks [54]. CARP's iterative refinement process resembles the denoising steps in diffusion models to achieve high accuracy.

Furthermore, our approach models actions directly rather than modeling noise, facilitating faster prediction convergence in low-dimensional manifolds [38, 66]. By operating in the latent space with action sequence tokens, CARP mitigates trajectory anomalies that can disrupt prediction, unlike methods that work on raw actions [12]. This latent-space representation allows CARP to focus on the essential components of actions, resulting in smoother and more efficient predictions. In contrast, traditional diffusion models rely on Markovian processes, which compress all information into progressively noisier inputs from previous levels, often hindering efficient learning and requiring more inference steps [17] (see Eq. (3) and Fig. 2b). CARP relaxes this constraint by allowing each scale $\boldsymbol{r}_k$ to depend on all prior scales $\boldsymbol{r}_{1:k-1}$ instead of the only previous scale $\boldsymbol{r}_{k-1}$. This structure enables CARP to generate high-quality trajectories with significantly fewer steps, demonstrating superior efficiency.

3.3. Implementation Details

Here, we present the key implementation details of CARP.

Tokenization. Due to the disentangling of the action prediction, imprecise *coarse-to-fine* action representations can decrease the upper limit of model performance. Considering the discontinuity of the most representation for rotation space like Euler angle or quaternion will increase the unstable training process, we utilize rotation6d [73] to get stable *multi-scale* tokens. And for the distance function $\mathrm{dist}(\cdot)$, we use cosine similarity rather than Euclidean distance which is the cause of unstable training based on our observations. In practice, due to the orthogonality between dimensions, we use a separate VQVAE [60] for each dimension of the action space, to gain stable training. All convolutions we used in CARP are 1D to capture the time-dimension features.

Autoregressive. We adopt the architecture of standard decoder-only transformers akin to GPT-2 [10, 60]. The state $\boldsymbol{s}_O$ is used to generate the initial coarse-scale token map and then is utilized as adaptive normalization [42] for the subsequent predictions. During training, we observe that incorporating Exponential Moving Average (EMA) [19] enhances both training stability and performance, yielding a 4-5 % improvement, consistent with findings in [12]. During the inference, kv-caching can be used and no mask is needed (for further inference details, refer to Suppl. B).

4. Experiment

In this section, we evaluate CARP on diverse robotics tasks, including state-based and image-based benchmarks in single-task and multi-task settings. CARP's performance is assessed in terms of task success rate, inference speed, and model parameter scale. Additionally, we validate CARP's practical effectiveness by deploying it on real-world tasks using UR5e and Franka robotic arms, comparing its performance against state-of-the-art diffusion-based policies. Our experiments are structured to address the following key research questions (see Suppl. H for experimental implementation details):

- **RQ1**: Can CARP achieve accuracy and robustness comparable to current state-of-the-art diffusion-based policies?
- **RQ2**: Does CARP maintain high inference efficiency, characteristic of autoregressive models?
- **RQ3**: Does CARP leverage the flexibility benefits of a GPT-style architecture?

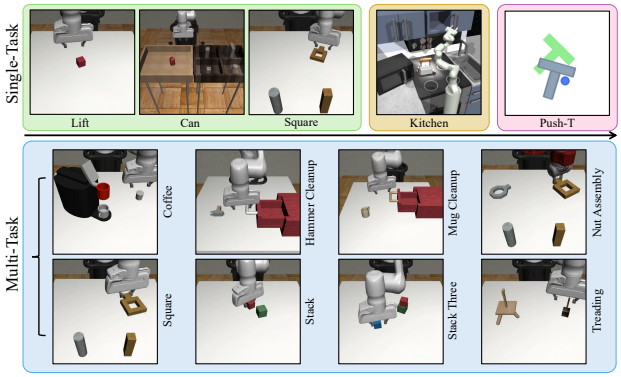


Figure 4. **Simulation Tasks Visualization.** In the single-task setting, we evaluate Lift, Can, and Square, ordered by increasing difficulty, along with the Kitchen task for long-horizon evaluation and Push-T task for assessing multi-modal behavior. In the multi-task setting, we consider eight tasks: Coffee, Hammer Cleanup, Mug Cleanup, Nut Assembly, Square, Stack, Stack Three, and Threading, arranged from left to right and top to bottom.

4.1. Evaluation on Single-Task Simulations

We begin by evaluating CARP on a set of standard simulated tasks commonly used to benchmark diffusion-based policies, aiming to assess whether it can achieve comparable performance while significantly improving inference efficiency.

Experimental Setup. We use the Robomimic [40] benchmark suite, which is widely adopted for evaluating diffusion-based policies [12, 44, 63]. We evaluate CARP on both state-based and image-based datasets collected from experts, with each task containing 200 demonstrations. The selected tasks are standard in single-task evaluations, as shown in Fig. 4, with each task predict max 400 actions to execute the

Policy	Lift	Can	Square	Params/M	Speed/s
BET [54]	0.96	0.88	0.54	**0.27**	**2.12**
DP-C [12]	**1.00**	0.94	**0.94**	65.88	35.21
DP-T [12]	**1.00**	**1.00**	0.88	8.97	37.83
CARP (Ours)	**1.00**	**1.00**	**0.98**	**0.65**	**3.07**

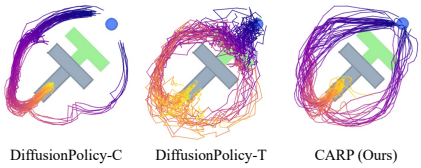
Table 1. **State-Based Simulation Results (State Policy).** We report the average success rate of the top 3 checkpoints, along with model parameter scales and inference time for generating 400 actions. CARP significantly outperforms BET and achieves competitive performance with state-of-the-art diffusion models, while also surpassing DP in terms of model size and inference speed.

Policy	Lift	Can	Square	Params/M	Speed/s
IBC [16]	0.72	0.02	0.00	**3.44**	32.35
DP-C [12]	**1.00**	0.97	**0.92**	255.61	47.37
DP-T [12]	**1.00**	**0.98**	0.86	9.01	45.12
CARP (Ours)	**1.00**	**0.98**	**0.88**	**7.58**	**4.83**

Table 2. **Image-Based Simulation Results (Visual Policy).** Results show that CARP consistently balances high performance and high inference efficiency. We highlight our results in light-blue.

Policy	p1	p2	p3	p4	Params	Speed
BET [54]	0.96	0.84	0.60	0.20	**0.30**	**1.95**
DP-C [12]	**1.00**	**1.00**	**1.00**	0.96	66.94	56.14
DP-T [12]	**1.00**	0.98	0.98	0.96	80.42	56.32
CARP (Ours)	**1.00**	**1.00**	0.98	**0.98**	**3.88**	**2.01**

Table 3. **Multi-Stage Task Results (State Policy).** In the Kitchen, px represents the success rate of interacting with x or more objects. CARP outperforms BET, especially on challenging metrics like p4, and achieves competitive performance compared to DP, with fewer parameters and faster inference speed.



(a) First 40 steps from 50 multimodal rollouts.

	DP-C	DP-T	CARP
Success Rate	0.87	0.84	**0.88**
Inference Time (s)	32.52	29.56	**2.66**
Params (M)	65.25	8.96	**0.71**

* Time measured over a rollout of 300 actions.

(b) State-based results.

Figure 5. **Multi-Modal Behavior Results (State Policy).** On the Push-T task featuring multi-modal path options, CARP generates diverse, consistent predictions from the same initialization. Notably, it outperforms baselines in both task success and inference speed.

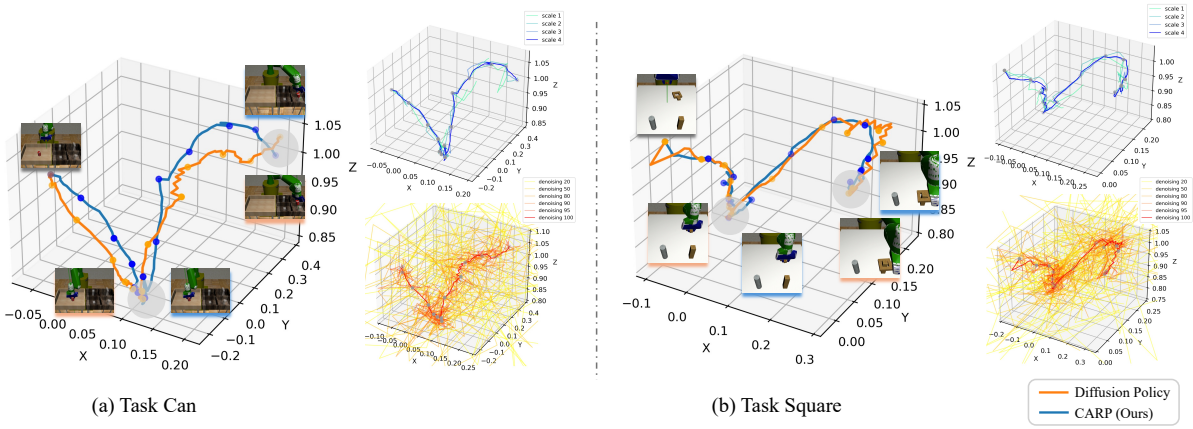
task. For evaluating the ability to learn multiple long-horizon tasks, we utilize the Franka Kitchen environment [18], which contains 7 objects for interaction and comes with a human demonstration dataset containing 566 trajectories, each completing 4 tasks in arbitrary order (see Suppl. M for more visualization). To assess multi-modal behavior, we use the Push-T task from IBC [16], which involves contact-rich dynamics to push a T-shaped block along multiple paths. The state-based observation includes 9×2D keypoints from the block's ground-truth pose and proprioception of the end-effector. We train with 200 expert demonstrations.

Baselines. We compare CARP with previous autoregressive policies and recent diffusion policies. Behavior Transformer (BET) [54] is an autoregressive policy with action discretization and correction mechanisms, similar to offset-based prediction. We further extend BET with action chunking to improve performance, as shown in Fig. 2a. Implicit Behavior Cloning (IBC) [16] utilizes energy-based models for supervised robotic behavior learning. Diffusion Policy (DP) [12] combines a denoising process with action prediction, implemented in two variants: CNN-based (DP-C) and Transformer-based (DP-T). Both follow the official implementation, employing DDPM with 100 denoising steps.

Metrics. For each task, we evaluate policies by conducting 50 trials with random initializations, computing the success rate, and reporting the average over the top three checkpoints. In the Kitchen task, success rates are cumulative, requiring completion of previous levels to achieve the next (e.g., p2 requires completing p1 first). Inference speed is measured on an A100 GPU by averaging the time taken for predicting 400 actions (280 for Kitchen, 300 for Push-T) across five runs to ensure robustness. We also record each policy's parameter count using the same PyTorch interface, excluding the visual encoder in the image-based setting.

Results. As demonstrated in Tables 1, 2, and 3, CARP consistently outperforms autoregressive policies and demonstrates comparable performance to diffusion policies across both state-based and image-based tasks, answering **RQ1**. Notably, CARP significantly outperforms diffusion policies in terms of inference speed, being approximately 10 times faster, with only 1-5% of the parameters required by diffusion models, thereby strongly supporting **RQ2** regarding CARP's efficiency. See Suppl. G for extra failure analysis. As shown in Fig. 5, CARP's use of Cross-Entropy loss at each scale during training facilitates latent-space sampling at inference, enhancing multimodality and producing smoother actions. This leads to superior performance and faster inference, further supporting **RQ1** and **RQ2**.

Analysis. To further examine CARP's stability and efficiency, we visualize spatial trajectories along the xyz axes for the Can and Square tasks in Fig. 6. In each task's left panel, CARP consistently reaches specific regions (light grey area) to perform task-related actions, such as positioning for object grasping or placement. Compared to diffusion-based policies, CARP's trajectories are smoother and more consistent, underscoring its stability and accuracy (supporting **RQ1**). The right panels of Fig. 6 compare CARP's *coarse-to-fine* action predictions with 6 selected denoising steps of the diffusion model. CARP achieves accurate predictions within just 4 *coarse-to-fine* steps, whereas the diffusion model requires numerous denoising iterations, with many early steps introducing redundant computations. This analysis further



(a) Task Can (b) Task Square

Figure 6. **Visualization of the Trajectory and Refining Process.** The left panel shows the final predicted trajectories for each task, with CARP producing smoother and more consistent paths than Diffusion Policy (DP). The right panel visualizes intermediate trajectories during the refinement process for CARP (top-right) and DP (bottom-right). DP displays considerable redundancy, resulting in slower processing and unstable action prediction, as illustrated by 6 selected steps among 100 denoising steps. In contrast, CARP achieves efficient trajectory refinement across all 4 scales, with each step contributing meaningful updates.

Policy	Prams/M	Speed/s	Coffee	Hammer	Mug	Nut	Square	Stack	Stack three	Threading	Avg.
TCD [36]	156.11	107.15	0.77	0.92	0.53	0.44	0.63	0.95	0.62	0.56	0.68
SDP [63]	159.85	112.39	0.82	**1.00**	0.62	0.54	0.82	0.96	0.80	**0.70**	0.78
CARP (Ours)	**16.08**	**6.92**	**0.86**	0.98	**0.74**	**0.78**	**0.90**	**1.00**	**0.82**	**0.70**	**0.85**

Table 4. **Multi-Task Simulation Results (Visual Policy).** Success rates are averaged across the top three checkpoints for each task, as well as the overall average across all tasks. We also report parameter count and inference time for generating 400 actions. CARP outperforms diffusion-based policies by 9%-25% in average performance, with significantly fewer parameters and over 10× faster inference.

supports **RQ2** and highlights CARP's efficiency and fast inference convergence advantage over diffusion policies in generating accurate actions with fewer refinement steps, as detailed in Sec. 3.2. See Suppl. D, E, and F for more analysis.

4.2. Evaluation on Multi-Task Simulations

We further evaluate CARP on the MimicGen [41] multi-task simulation benchmark, widely used by the state-of-the-art Sparse Diffusion Policy (SDP) [63], to demonstrate CARP's flexibility, a result of its GPT-style autoregressive design.

Experimental Setup. MimicGen extends benchmark Robomimic [40] by including 1K–10K human demonstrations per task, with diverse initial state distributions for enhanced generalization in multi-task evaluation. We select 8 robosuite [74] tasks for evaluation, each with 1K training trajectories, following the settings in SDP [63], as shown in Fig. 4. For CARP, we simply extend the single-task implementation by incorporating a task embedding as an additional condition alongside the observation sequence $\boldsymbol{s}$.

Baselines. We compare CARP with two baselines: task-conditioned diffusion (TCD) [2, 36]: A basic diffusion-based multi-task policy and sparse diffusion policy (SDP) [63]: A transformer-based diffusion policy that leverages mixture of experts (MoE) [55]. Both baselines are trained with visual inputs following their official implementation and settings.

Metrics. Success rates are reported for each task as the average of the best three checkpoints. For Nut Assembly, partial success (e.g., placing one block inside the cylinder) is assigned a score of 0.5, while full success is scored as 1. For all other tasks, a score of 1 is awarded only when strict success criteria are fully satisfied. Additionally, we calculate the average success rate across all tasks. To evaluate model efficiency, we report both parameter counts (excluding the identically configured visual encoder) and the inference time required to predict 400 actions on a single A100 GPU.

Results. As shown in Tab. 4, CARP achieves up to a 25% average improvement in success rates compared to state-of-the-art diffusion-based policies, highlighting its strong performance. Additionally, as shown in Tab. 4, CARP achieves over 10× faster inference speed and uses only 10% of the parameters compared to SDP. Leveraging its GPT-style autoregressive design [8, 49], CARP can seamlessly transition from single-task to multi-task settings with minimal structural modifications [63], demonstrating the flexibility of this architecture, which strongly supports **RQ3**. Considering fine-grained manipulation, such as Nut and Threading, CARP outperforms diffusion methods by up to 20%, demonstrating its strong applicability in these tasks (see Suppl. L for further analysis). These results strongly demonstrate CARP's flexibility, solidifying its role as a high-performance and high-efficiency approach for visuomotor robotic policies.

4.3. Evaluation on Real-World

In this section, we evaluate our approach, CARP, on real-world tasks under compute-constrained conditions, comparing its performance and efficiency against baseline methods.

Experimental Setup. To validate CARP's real-world applicability, we design two manipulation tasks: 1) *Cup*: The robot must locate a cup on the table, pick it up, move to the right area of the table, and put it down steadily. 2) *Bowl*: The robot needs to identify a smaller bowl and a larger pot on the table, pick up the bowl, and place it inside the pot. We use a UR5e robotic arm with a Robotiq-2f-85 gripper, equipped with two RGB cameras: one mounted on the wrist and one in a third-person perspective (left panel, Fig. 7). The robot is controlled through 6D end-effector positioning, with inverse kinematics for joint angle calculation. For teleoperation, we collected 50 human demonstration trajectories for each task using a 3D Connexion space mouse. We reproduce the image-based CNN-based Diffusion Policy as a baseline, as it has been shown to outperform current autoregressive policies, adapting the model's input size to match our observational setup. CARP maintains the design with $K = 4$, consistent with previous experiments (Suppl. J for ablation studies).

Metrics. For each trained policy, we report the average success rate across 20 trials per task, with the initial positions randomized. We also measure inference speed on an NVIDIA GeForce RTX 2060 GPU, reporting action prediction frequency in Hertz (Suppl. M for more visualization).

Results. As shown in the top-right table of Fig. 7, CARP achieves comparable or superior performance, with up to a 10% improvement in success rate over the Diffusion Policy across all real-world tasks, supporting **RQ1**. Additionally, CARP achieves approximately $8\times$ faster inference than the baseline on limited computational resources, demonstrating its suitability for real-time robotic applications, thus supporting **RQ2** (Suppl. K for additional real-world experiments).

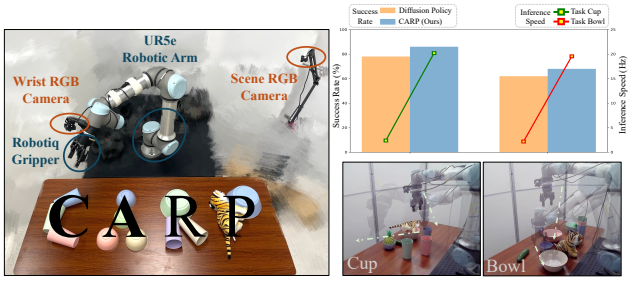


Figure 7. **Real-World Study.** The left panel illustrates the environment used for both experimentation and demonstration collection. The bottom-right panel visualizes trajectories from the Cup and Bowl datasets. In the top-right panel, we present the average success rate over 20 trials alongside inference speed, measured as action prediction frequency. CARP achieves competitive success rates while significantly outperforming DP in inference speed.

5. Related Work

Visual Generation. Advances in visual generative models have strongly influenced the robotics community. Autoregressive models generate images using discrete tokens from image tokenizers [15, 22, 62]. GPT-2-style transformers [4, 31, 50, 65] demonstrate strong performance by generating tokens sequentially. Recent work has scaled these models to achieve impressive text-to-image synthesis results [37, 71] and robotic action generation [14, 56]. VAR [60] introduces a new next-scale autoregressive paradigm that shifts image representation from patches to scales. This framework [60], has been applied across tasks [17, 34, 35, 39, 45, 68, 69]. Studies [58, 60] show that autoregressive models can surpass diffusion models in achieving compatible performance, serving as a key inspiration for the design of our approach.

Visuomotor Policy Learning. Behavior cloning [9] provides an effective approach, especially in autonomous driving [43] and manipulation [70], offering a simpler alternative to complex reinforcement learning. Explicit policies map observations to actions efficiently [70], but struggle with complex tasks. Solutions like action discretization [67] and mixture density networks [13, 54] mitigate this, yet suffer from action space explosion and hyperparameter sensitivity. Implicit policies, such as energy-based models [16, 25], offer greater flexibility but are harder to train due to optimization challenges. Diffusion models have proven effective for policy learning [2, 12, 52], but suffer from high computational cost due to multi-step denoising. Recent work aims to improve generalization [64], support 3D environments [63, 66], and enhance modularity via mixture of experts [63]. Consistency models [38, 57] accelerate inference but compromise action prediction accuracy, along with inflexible model design.

6. Conclusion

In this work, we introduce Coarse-to-Fine Autoregressive Policy (CARP), a novel paradigm for robotic visuomotor policy learning that combines the efficiency of autoregressive modeling (AM) with the high performance of diffusion modeling (DM). CARP incorporates: 1) *multi-scale* action tokenization to capture global structure and temporal locality, addressing AM's limitations in long-term dependency; 2) *coarse-to-fine* autoregressive prediction that refines actions from high-level intentions to detailed execution, achieving DM-like performance with AM-level efficiency through latent space prediction and relaxed Markovian constraints. The comprehensive evaluations from single- to multi-task, simulation to real-world, demonstrate CARP's effectiveness in balancing high performance, efficiency, and flexibility.

We hope this work inspires further exploration of next-generation GPT-style autoregressive models for policy learning, fostering a unified perspective on current generative modeling techniques (see Suppl. A for further discussion).

Acknowledgement

This work was supported by the National Science and Technology Innovation 2030 - Major Project (Grant No. 2022ZD0208800)

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 12

[2] 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. 7, 8

[3] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022. 12

[4] Dosovitskiy Alexey. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv: 2010.11929*, 2020. 8

[5] Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement–residual rl for precise assembly. *arXiv preprint arXiv:2407.16677*, 2024. 17

[6] Mariusz Bojarski. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 5

[7] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. 12

[8] Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 7, 12

[9] Harish chaandar Ravichandar, Athanasios S. Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annu. Rev. Control. Robotics Auton. Syst.*, 3:297–330, 2020. 8

[10] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, et al. Pixart-alpha: Fast training of diffusion transformer for photorealistic text-to-image synthesis. *arXiv preprint arXiv:2310.00426*, 2023. 5

[11] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021. 12

[12] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023. 1, 2, 3, 5, 6, 8, 12, 14, 16, 17, 18

[13] Zichen Jeff Cui, Yibin Wang, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022. 1, 8, 12

[14] Pengxiang Ding, Han Zhao, Wenjie Zhang, Wenxuan Song, Min Zhang, Siteng Huang, Ningxi Yang, and Donglin Wang. Quar-vla: Vision-language-action model for quadruped robots. In *European Conference on Computer Vision*, pages 352–367. Springer, 2025. 8

[15] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021. 8

[16] 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 *Proceedings of the 5th Conference on Robot Learning*, pages 158–168. PMLR, 2022. 6, 8

[17] Jiatao Gu, Yuyang Wang, Yizhe Zhang, Qihang Zhang, Dinghuai Zhang, Navdeep Jaitly, Josh Susskind, and Shuangfei Zhai. Dart: Denoising autoregressive transformer for scalable text-to-image generation. *arXiv preprint arXiv:2410.08159*, 2024. 3, 5, 8

[18] 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. 6

[19] David Haynes, Steven Corns, and Ganesh Kumar Venayagamoorthy. An exponential moving average algorithm. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2012. 5

[20] Minho Heo, Youngwoon Lee, Doohyun Lee, and Joseph J Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. *The International Journal of Robotics Research*, page 02783649241304789, 2023. 17

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arxiv:2006.11239*, 2020. 2, 3, 12

[22] Mengqi Huang, Zhendong Mao, Zhuowei Chen, and Yongdong Zhang. Towards accurate image coding: Improved autoregressive image generation with dynamic vector quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22596–22605, 2023. 8

[23] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021. 12

[24] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022. 2, 3, 4

[25] 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. 8

[26] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2(3):6, 2022. 12

[27] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 12

[28] Maciej Kilian, Varun Jampani, and Luke Zettlemoyer. Computational tradeoffs in image synthesis: Diffusion, masked-token, and next-token prediction. *arXiv preprint arXiv:2405.13218*, 2024. 1, 2

[29] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 12

[30] Lucy Lai, Ann Zixiang Huang, and Samuel J Gershman. Action chunking as policy compression. *PsyArXiv*, 2022. 12

[31] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11523–11532, 2022. 4, 8, 16

[32] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024. 1, 12, 15

[33] Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, et al. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*, 2023. 12

[34] Xiang Li, Hao Chen, Kai Qiu, Jason Kuen, Jiuxiang Gu, Bhiksha Raj, and Zhe Lin. Imagefolder: Autoregressive image generation with folded tokens. *arXiv preprint arXiv:2410.01756*, 2024. 8

[35] Xiang Li, Kai Qiu, Hao Chen, Jason Kuen, Zhe Lin, Rita Singh, and Bhiksha Raj. Controlvar: Exploring controllable visual autoregressive modeling. *arXiv preprint arXiv:2406.09750*, 2024. 8

[36] Zhixuan Liang, Yao Mu, Hengbo Ma, Masayoshi Tomizuka, Mingyu Ding, and Ping Luo. Skilldiffuser: Interpretable hierarchical planning via skill abstractions in diffusion-based task execution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16467–16476, 2024. 7

[37] Yang Liu, Pengxiang Ding, Siteng Huang, Min Zhang, Han Zhao, and Donglin Wang. Pite: Pixel-temporal alignment for large video-language model. In *European Conference on Computer Vision*, pages 160–176. Springer, 2025. 8

[38] Guanxing Lu, Zifeng Gao, Tianxing Chen, Wenxun Dai, Ziwei Wang, and Yansong Tang. Manicm: Real-time 3d diffusion policy via consistency model for robotic manipulation. *arXiv preprint arXiv:2406.01586*, 2024. 5, 8

[39] Xiaoxiao Ma, Mohan Zhou, Tao Liang, Yalong Bai, Tiejun Zhao, Huaian Chen, and Yi Jin. Star: Scale-wise text-to-image generation via auto-regressive representations. *arXiv preprint arXiv:2406.10797*, 2024. 8

[40] 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 *arXiv preprint arXiv:2108.03298*, 2021. 5, 7

[41] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *7th Annual Conference on Robot Learning*, 2023. 7, 18

[42] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2337–2346, 2019. 5

[43] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988. 8

[44] Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. *arXiv preprint arXiv:2405.07503*, 2024. 5, 15

[45] Kai Qiu, Xiang Li, Hao Chen, Jie Sun, Jinglu Wang, Zhe Lin, Marios Savvides, and Bhiksha Raj. Efficient autoregressive audio modeling via next-scale prediction. *arXiv preprint arXiv:2408.09027*, 2024. 8

[46] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 12

[47] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 12

[48] Ilija Radosavovic, Baifeng Shi, Letian Fu, Ken Goldberg, Trevor Darrell, and Jitendra Malik. Robot learning with sensorimotor pre-training. *arXiv:2306.10007*, 2023. 2

[49] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020. 7

[50] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019. 8

[51] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022. 12

[52] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023. 8

[53] Moritz Reuss, Ömer Erdinç Yağmurlu, Fabian Wenzel, and Rudolf Lioutikov. Multimodal diffusion transformer: Learning versatile behavior from multimodal goals. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*, 2024. 1

[54] Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022. 1, 2, 5, 6, 8, 12, 15

[55] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017. 7

[56] Wenxuan Song, Han Zhao, Pengxiang Ding, Can Cui, Shangke Lyu, Yaning Fan, and Donglin Wang. Germ: A generalist robotic model with mixture-of-experts for quadruped robot. *arXiv preprint arXiv:2403.13358*, 2024. 8

[57] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023. 8

[58] Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024. 8

[59] Garrett Thomas, Ching-An Cheng, Ricky Loynd, Felipe Vieira Frujeri, Vibhav Vineet, Mihai Jalobeanu, and Andrey Kolobov. Plex: Making the most of the available data for robotic manipulation pretraining. In *Conference on Robot Learning*, pages 2624–2641. PMLR, 2023. 12

[60] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *arXiv preprint arXiv:2404.02905*, 2024. 3, 4, 5, 8, 16

[61] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018. 5

[62] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 3, 4, 8

[63] Yixiao Wang, Yifei Zhang, Mingxiao Huo, Ran Tian, Xiang Zhang, Yichen Xie, Chenfeng Xu, Pengliang Ji, Wei Zhan, Mingyu Ding, et al. Sparse diffusion policy: A sparse, reusable, and flexible policy for robot learning. *arXiv preprint arXiv:2407.01531*, 2024. 1, 2, 5, 7, 8, 16, 18

[64] Jingyun Yang, Zi-ang Cao, Congyue Deng, Rika Antonova, Shuran Song, and Jeannette Bohg. Equibot: Sim (3)-equivariant diffusion policy for generalizable and data efficient learning. *arXiv preprint arXiv:2407.01479*, 2024. 8

[65] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved vqgan. *arXiv preprint arXiv:2110.04627*, 2021. 8

[66] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *ICRA 2024 Workshop on 3D Visual Representations for Robot Manipulation*, 2024. 5, 8

[67] 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. 8

[68] Jinzhi Zhang, Feng Xiong, and Mu Xu. G3pt: Unleash the power of autoregressive modeling in 3d generation via cross-scale querying transformer. *arXiv preprint arXiv:2409.06322*, 2024. 8

[69] Qian Zhang, Xiangzi Dai, Ninghua Yang, Xiang An, Ziyong Feng, and Xingyu Ren. Var-clip: Text-to-image generator with visual auto-regressive modeling. *arXiv preprint arXiv:2408.01181*, 2024. 8

[70] 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. 8

[71] Han Zhao, Min Zhang, Wei Zhao, Pengxiang Ding, Siteng Huang, and Donglin Wang. Cobra: Extending mamba to multi-modal large language model for efficient inference. *arXiv preprint arXiv:2403.14520*, 2024. 8

[72] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023. 1, 2, 5, 12, 14

[73] 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. 5

[74] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, Yifeng Zhu, and Kevin Lin. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020. 7

A. Limitations and Future Work

In this work, we propose CARP, a next-generation paradigm for robotic visuomotor policy learning, which effectively balances the long-standing trade-off between high performance and high inference efficiency seen in previous autoregressive modeling (AM) and diffusion modeling (DM) approaches. Despite these advancements, there remain several limitations and opportunities for improvement in near-future research.

First, the architectural design of CARP can be further optimized for simplicity. Currently, CARP employs a two-stage design, where the first stage utilizes separate *multi-scale* action VQVAE modules for each action dimension to address their orthogonality. A promising direction for future work could focus on developing a unified one-stage method that integrates *multi-scale* tokenization with the *coarse-to-fine* prediction process, resulting in a more efficient and streamlined framework without compromising performance.

Second, CARP's multimodal capacity has not yet been fully explored or leveraged. To address the inherent unimodality of conventional autoregressive policies trained with MSE loss, CARP employs a Cross-Entropy objective that preserves the potential for multi-modal predictions. Compared to the Diffusion Policy [12]'s ability to model multimodality via DDPM's integration over stochastic differential equations [21], CARP adopts a more direct yet effective alternative that achieves comparable multimodal expressiveness. Nevertheless, the role of multi-modality in visuomotor policy learning remains underexplored. Many current benchmark tasks either do not require diverse output distributions or tend to induce overfitting to a single prediction path. Future research should investigate the necessity of multi-modal reasoning in robotic decision-making and further harness CARP's capacity to model action diversity.

Third, CARP's adoption of the GPT-style paradigm opens up promising yet unexplored possibilities. Beyond the flexibility already demonstrated, the contextual understanding capabilities inherent in GPT-style architectures [46] suggest that CARP could be extended to support multi-modal inputs [3, 47] like tactile and auditory information and address robotic tasks requiring long-term dependency reasoning [1]. Moreover, its inherent capacity for in-context learning suggests strong potential for generalization under few-shot and zero-shot learning settings [8], making it a compelling foundation for more adaptive and versatile visuomotor policies.

Finally, but not exhaustively, the scaling potential of CARP presents a promising avenue for future exploration. The scaling laws established in existing GPT-style models [27] could be seamlessly applied to CARP, suggesting that increasing model capacity and leveraging larger pre-training datasets could lead to substantial performance gains. Furthermore, recent advances in Vision-Language-Action (VLA) [7, 29, 33] models present a promising opportunity to integrate CARP into such frameworks. Such integration could further demonstrate CARP's scalability and its potential for general-purpose embodied intelligence.

B. Coarse-to-Fine Inference

Unlike the training process, the inference process predicts token maps of the action sequence across different scales in an autoregressive *next-scale*, *coarse-to-fine* manner without teacher forcing, as illustrated in Fig. 8. Additionally, kv-caching is employed to eliminate redundant computations.

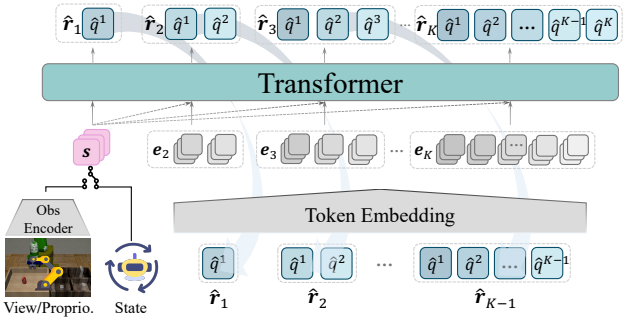


Figure 8. **Coarse-to-Fine Autoregressive Inference.** During inference, we leverage kv-caching to enable *coarse-to-fine* prediction without the need for causal masks. The full set of token maps, $\boldsymbol{r}_{1:K}$, is collectively decoded by the action *multi-scale* VQVAE into executable actions for the robotic arm.

C. Definition of Autoregressive Policy

Autoregressive policies naturally capitalize on the efficiency and flexibility of autoregressive models. Initial works from a reinforcement learning perspective applied models like Transformers to predict the next action using states or rewards as inputs [11, 23], as shown in Fig. 9 and formalized as

$$p\left(\boldsymbol{a}_t, \boldsymbol{a}_{t+1}, ..., \boldsymbol{a}_{t+H-1}\right) = \prod_{k=t}^{t+H-1} p\left(\boldsymbol{a}_k | \boldsymbol{a}_{k-H:k-1}, \boldsymbol{s}_{k-H:k}\right), \tag{8}$$

where $\boldsymbol{s}_{k-H:k-1}$ represents the states or observations corresponding to the previous actions $\boldsymbol{a}_{k-H:k-1}$, and $\boldsymbol{s}_k$ is the current state or observation. Following this paradigm, several subsequent works [26, 51, 59] employ autoregressive models to predict one action at a time during inference.

More recently, the concept of action chunking [30], derived from neuroscience, has demonstrated notable benefits for imitation learning [13, 32, 54, 72]. In action chunking, individual actions are grouped and executed as cohesive units, leading to improved efficiency in storage and execution, as depicted in Fig. 2a of the main paper. This paradigm extends the capabilities of GPT-style decoders by modifying them to generate chunks of actions in one forward pass, replacing the

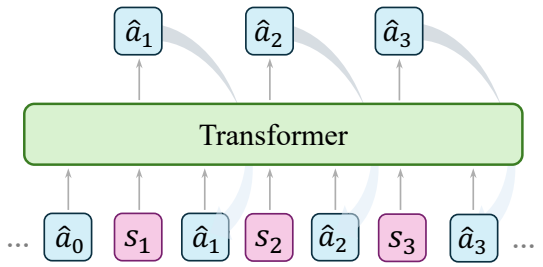


Figure 9. **Conventional Autoregressive Policy.** In reinforcement learning, conventional autoregressive policies generate action tokens sequentially, where each token is predicted based on the previously generated tokens. This differs from the action chunking prediction (see Sec. 2.2 of the main paper).

traditional single-step autoregressive operation with a multi-token, pseudo-autoregressive process. The action generation process for this paradigm is described as

$$p\left(\boldsymbol{a}_t, \boldsymbol{a}_{t+1}, \ldots, \boldsymbol{a}_{t+H-1}\right) = \prod_{k=t}^{t+H-1} p\left(\boldsymbol{a}_k | \boldsymbol{s}_O\right), \quad (9)$$

where O is the historical horizon. The model predicts the entire action sequence in one forward pass without strictly adhering to step-by-step autoregressive operations.

Given the significant performance improvements enabled by action chunking, we adopt this multi-token, one-forward-pass framework throughout the article and experiments when referring to Autoregressive Policy (AP).

D. Efficiency Concerns

Efficiency, as discussed throughout this paper, specifically refers to inference efficiency—the ability of CARP to generate actions significantly faster than diffusion-based policies during deployment. While efficiency can be examined from various perspectives, we focus on three key aspects relevant to CARP: inference efficiency, training efficiency, and data efficiency, each of which is analyzed in detail below.

Task	*CARP*		*DP*
	Predict	Decode	
Can	2.279 s	0.639 s	34.79 s
Square	2.597 s	0.679 s	35.62 s

Table 5. **Inference Efficiency Comparison.** We report the time consumption for the *coarse-to-fine* prediction phase and the subsequent action decoding phase over 400 timesteps of action generation. The results indicate that in CARP, the majority of inference time is allocated to the prediction step, whereas the decoding process is completed within a short duration.

Inference Efficiency. We analyze the inference efficiency of CARP's two-stage process. During inference, CARP first predicts action tokens in a *coarse-to-fine* manner, followed by a single forward pass to decode all token maps into executable actions. Since token maps are collected during the prediction phase, and decoding requires only a single forward computation, the majority of the computational cost is incurred during prediction, while action decoding remains relatively lightweight. This is empirically validated by the results in Tab. 5. Compared to DP, CARP achieves significantly faster inference by eliminating the iterative denoising steps required by diffusion-based policies, instead directly predicting actions as a low-dimensional generation problem.

Training Efficiency. We analyze training efficiency through convergence behavior and time consumption.

While training convergence depends on task complexity and hyperparameter configurations, both CARP and DP exhibit stable learning dynamics under our respective settings. To accommodate the architectural differences of CARP, we employ slightly different training configurations from those used for DP. As shown in Fig. 10, under our experimental settings, both DP and CARP achieve good convergence within the same number of training epochs. While convergence speed may differ due to structural and training differences, it does not inherently indicate superiority in model design. Instead, both CARP and DP demonstrate reliable training behavior under their respective settings.

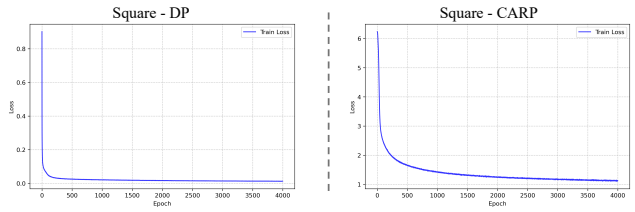


Figure 10. **Training Efficiency on Convergence Analysis.** With different training configurations, both DP and CARP converge effectively within 4000 epochs in the state-based Square task.

	GPU hours	**GPU Memory**	**CPU Memory**
DP-C	12.17 h	3.06 GB	18.13 GB
DP-T	10.83 h	1.56 GB	15.51 GB
CARP-VQ	4.67 h (28min×10)	0.69 GB	2.55 GB
CARP-TF	13.85 h	2.91 GB	19.86 GB

* A distinct VQ is trained for each dimension (10 total in our setting).
* Test on state-based Square with 200 trajectories using a V100 GPU.
* Similar results across simulation and real-world dataset.

Figure 11. **Training Efficiency on Time Consumption.** Comparison of training time on the Square task. For CARP, we separately report the tokenizer training time (with 10 VQ-VAEs, one per action dimension) and policy learning time.

In terms of wall-clock training time, a comparison is pro-

vided in Fig. 11. The total training cost of CARP is comparable to that of DP when considering the policy learning stage (`CARP-TF`) alone. Although CARP introduces an additional tokenizer pretraining phase (`CARP-VQ`)—where separate VQ-VAEs are trained for each action dimension—the cost is amortized across tasks and environments. In particular, when trained on sufficiently diverse data (e.g., multi-task settings in both simulation and the real world), the tokenizers become reusable, substantially reducing the overall training burden in practical deployments.

Data Efficiency. We further assess the data efficiency of CARP by evaluating its performance under varying amounts of training data. Specifically, we investigate whether CARP can maintain strong performance when trained with limited trajectories, indicating robustness to data scarcity. As shown in Fig. 12, we compare CARP with baseline policies (following implementations from [12]) on the state-based Square task, using training datasets ranging from 200 to 30 trajectories. CARP consistently outperforms the baselines across all data regimes, demonstrating its superior data efficiency and reduced reliance on large-scale datasets.

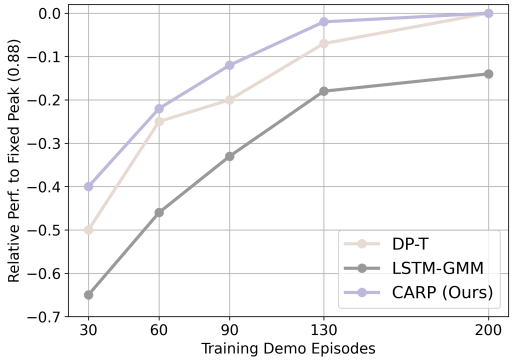


Figure 12. **Data Efficiency Analysis.** Performance comparison under varying training dataset sizes on the Square task. Each policy is trained following its official best-practice settings. CARP consistently outperforms the baselines at all data scales.

E. Comparative Analysis of CARP and AR.

CARP outperforms traditional autoregressive (AR) models in terms of success rate while maintaining high computational efficiency. CARP adopts a straightforward action tokenizer and leverages a GPT-style transformer for prediction, similar to standard AR policies. However, instead of conventional *next-token* prediction, CARP introduces a paradigm shift towards *next-scale* prediction. Despite these seemingly minor modifications, CARP achieves substantial performance gains. In this section, we analyze the key factors contributing to this improvement.

Temporal Locality. CARP encodes action sequences into a latent space in its first stage. Specifically, it employs 1D convolution to explicitly capture the local correlations within actions, facilitating a more effective learning of temporal dependencies—something that step-by-step action modeling struggles to achieve. As depicted in the magnified region of Fig. 14, encoding actions into a latent space enhances the smoothness of predictions while simultaneously denoising raw actions. This encoding process enables the model to capture similarities and overarching trends across contiguous actions, leveraging temporal locality to its advantage. While recent work on action chunking [72] has highlighted the significance of temporal locality, existing *next-token* prediction models still suffer from weakened action dependencies due to their traditional independent action output mechanisms.

Global Structure. CARP represents action sequences across multiple scales and predicts actions in a *coarse-to-fine* manner. The coarser scales compress the sequence using fewer tokens, promoting the learning of global action patterns. This hierarchical representation explicitly models the overall structure of action sequences—an aspect that traditional unidirectional *next-token* prediction struggles to capture. By progressively refining actions from high-level to low-level representations, CARP enhances sequence stability and mitigates the risk of producing erratic, inconsistent motions, leading to more precise execution.

Action Scalability. Similar to the approach used in Diffusion Policy, encoding action sequences improves the scalability of action generation. By encoding action sequences, CARP enables flexible adjustments to sequence length with minimal modifications to the model architecture, offering greater adaptability across different tasks and environments.

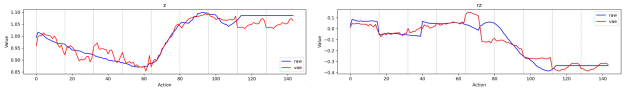


Figure 13. **Raw vs. Reconstructed Actions (Jointly Trained VQ-VAE).** Flattening and jointly encoding all action dimensions into a single VQ-VAE leads to poor reconstruction, as shown by the large discrepancy between raw (blue) and reconstructed (red) trajectories. This highlights the limitation of naive joint encoding.

Precision in Encoding. Adapting the *multi-scale* VQ-VAE to action space necessitates careful architectural design. Rather than flattening the action trajectory into an image-like structure and training a single joint VQ-VAE—which leads to unstable and less interpretable tokens (Fig. 13)—we instead encode each action dimension independently, using a dedicated VQ-VAE per dimension. As illustrated in Fig. 14, the *multi-scale* tokenization process ensures that generated action sequences closely match the raw inputs, exhibiting nearly identical trajectory lines while yielding smoother motions. This demonstrates that our action tokenization approach effectively preserves the fidelity of original action sequences. Moreover, the enhanced success rates observed in

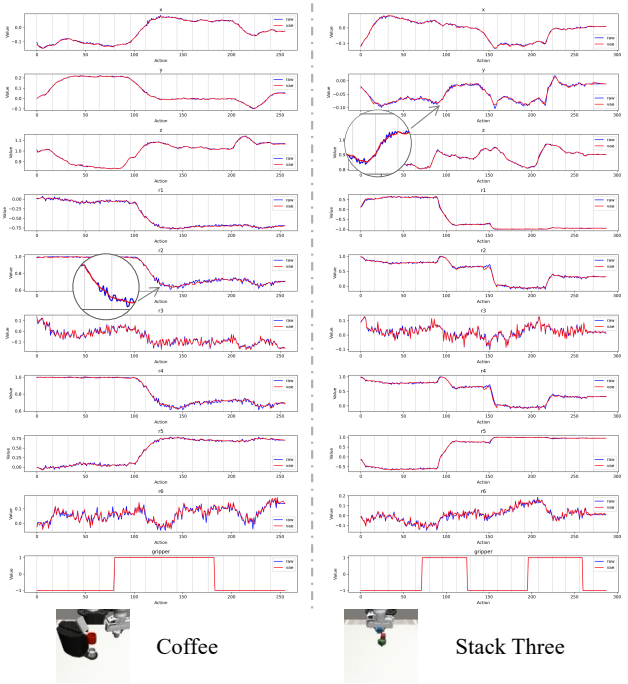


Figure 14. **Comparison of Raw and Reconstructed Actions.** Comparison across 10 action dimensions in the Coffee and Stack Three tasks. Reconstructed actions (red) closely align with raw signals (blue), preserving structural patterns while smoothing the sequences and filtering out noise (see magnified region), highlighting the effectiveness of our *multi-scale* tokenization.

the experiments presented in the main paper further validate the accuracy and effectiveness of CARP's design.

F. Additional Baselines Comparison

In addition to the baseline policies discussed in the main paper, we further compare CARP with enhanced versions of each category: VQ-BET [32], an improved variant of BeT [54], and Consistency Policy [44], which reduces sampling steps to improve inference efficiency. Both are implemented using their official codebases with recommended settings. As shown in Tab. 6, CARP achieves consistently higher success rates across all tasks, while maintaining competitive inference time. These results highlight CARP as a promising design that combines strong task performance with high inference efficiency.

G. Failure Analysis

In this section, we analyze common failure cases observed during experiments with both the diffusion-based policies and our proposed CARP.

Accident Recovery. A notable failure mode is the inability to recover from disturbances, as illustrated in Fig. 15. When tools are accidentally knocked over due to suboptimal

Policy	p1	p2	p3	p4	Inf.T(s)	Push-T	Inf.T(s)
ConsisP	0.99	0.96	0.95	0.93	2.31	0.80	2.93
VQ-BeT	0.96	0.92	0.87	0.71	1.48	0.72	1.70
CARP	**1.00**	**1.00**	**0.98**	**0.98**	2.01	**0.88**	2.66

Table 6. **State-Based Kitchen and Push-T Results.** We compare CARP with VQ-BET [32] and Consistency Policy [44] under identical settings. CARP consistently outperforms both baselines in success rate, while offering competitive inference times.

action trajectories, the model struggles to generate appropriate recovery behaviors. This limitation arises because the policy is trained purely by imitating expert demonstrations, which do not account for such out-of-distribution failure scenarios. Addressing this issue requires further incorporation mechanisms for failure detection and recovery.

Hesitant Movements. Another common failure case involves the generation of jerky or oscillatory movements when the robot encounters two similar situations with only slight visual differences across consecutive timesteps, as shown in the first row of Fig. 16. This issue arises because the policy conditions its predictions on only the previous one or two observations, potentially overlooking long-term historical context. When faced with multiple plausible action choices under limited observations, the policy may produce ambiguous actions, leading to hesitation. Consequently, these hesitant movements can prevent the robot from meeting the success criteria, as shown in the bottom-right of Fig. 16.

Figure 15. **Accident Recovery in the Square Task.** When tools are accidentally knocked over, the robot struggles to recover due to its reliance on imitation learning from expert demonstrations, which lack exposure to such out-of-distribution failure cases.

H. Experiment Implementation Details

Here, we provide implementation details for the main experiments presented in the paper.

Single-Task Simulation Experiment. For baseline models, we follow the same implementation and training configurations provided by Diffusion Policy. For all state-based experiments, including the Kitchen and Push-T tasks, we uniformly set the observation horizon $O = 2$ and the prediction horizon $H = 16$ across all models. For image-based experiments, we set $O = 1$ and $H = 16$ for better transferability to real-world scenarios. As per the benchmark, only the first 8 actions in the prediction horizon are executed, starting from the current step (see Suppl. I for further discus-

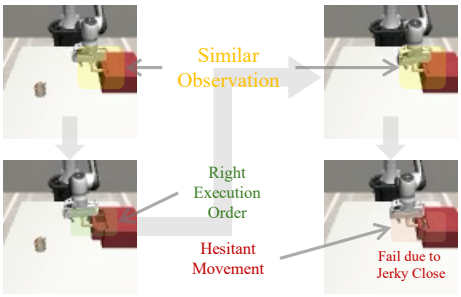


Figure 16. **Hesitant Movements in the Mug Task.** During task execution, the robot may encounter visually similar observations at different timesteps. Without leveraging long-term historical context, the policy may misinterpret these similarities and generate ambiguous actions, resulting in hesitant (jerky) movements. In this failure case, after successfully closing the drawer, the policy perceives the scene as similar to the initial step and erroneously attempts to reopen it. This cycle of opening and closing continues indefinitely, leading to task failure.

sion). For CARP, we first train an action VQVAE model (see Sec. 3.1 of the main paper) following [31], using $V = 512$, $C = 8$, a batch size of 256, and 300 epochs per task. Given a horizon $H = 16$, we design *multi-scale* representations with scales of $1, 2, 3$ and 4 to capture *coarse-to-fine* information across the action sequence. We then train an autoregressive GPT-2 style, decoder-only transformer (see Sec. 3.2 of the main paper) , based on [60], using the same training settings as the benchmark, with a batch size of 256 for state-based experiments (4000 epochs) and a batch size of 64 for image-based experiments (3000 epochs). We use Cross-Entropy loss during training, which preserves the model's sampling capability. During inference, we typically select the token with the highest probability at each scale. However, to visualize multi-modal behavior in the Push-T task, we sample the top-k tokens at each scale (with k=3), allowing for diverse predictions with controlled randomness.

Multi-Task Simulation Experiment. To enable multi-task generalization, CARP augments the single-task formulation by introducing a learnable 3-dimensional task embedding for each task. These embeddings, retrieved based on task indices, are concatenated with the observation sequence $\boldsymbol{s}$ and act as additional conditional inputs to the policy. In our experiments involving 8 tasks, this corresponds to an 8×3 embedding matrix. We also use a moderately deeper decoder-only transformer in GPT-2 style. CARP is trained with a batch size of 512 for 200 epochs on an A100 GPU. Baseline models follow the same training settings as SDP [63]. This minimal modification enables CARP to adapt to multi-task learning seamlessly.

Real-World Experiment. For both baselines and CARP, the input consists of current visual observations from the wrist and scene cameras (resolution: 120×160), as well as proprioceptive data from the robotic arm. We execute 8 predicted actions out of a horizon of 16 predictions. We train the diffusion policy for 3000 epochs with a batch size of 64. For CARP, we use the same visual policy structure as in the simulation tasks, training the *multi-scale* action tokenizer for 300 epochs with a batch size of 256, and the *coarse-to-fine* transformer for 3000 epochs with a batch size of 64.

I. Consistent with Diffusion Policy

We adopt similar experimental settings with 1 or 2 observations, a prediction horizon of 16, and an executable action length of 8, following the standard setup used in Diffusion Policy (DP) [12]. It is important to note that our classical formulation introduces a minor discrepancy in the horizon definition compared to the implementation of DP. Specifically, in DP's experimental setting, the horizon H encompasses the past observed steps, meaning that the index of the current next predicted action is O, rather than 0. In contrast, as outlined in the formulation of Eq. (1) in the main paper, the horizon H does not include past observations, with the first prediction step corresponding to the next time step. While the rationale behind this design remains unclear due to limitations in the author's understanding, we retain the horizon definition introduced by Diffusion Policy (DP) [12] to ensure consistency in our experimental comparisons.

J. Ablation Study on the Number of Scales

To maintain consistency with Diffusion Policy, we set the action prediction horizon H to 16 across all tasks. Given $H = 16$, we adopt $K = 4$ for all experiments. To further investigate the impact of K, we conduct an ablation study by varying K from 1 to 6 on three representative tasks: Can, Square, and Kitchen (which requires executing four consecutive subtasks, thus we report success rate based on the final subtask, denoted as p4). All other experimental settings remain unchanged.

For each chosen number of scales K, the token map sizes at each scale level are defined as $1, \ldots, K$. Notably, when $K = 5$ and $K = 6$, the scales slightly exceed the default feature map size. To ensure fair evaluation, we appropriately expand the feature map size to accommodate these settings.

As shown in Fig. 17, using fewer scales leads to insufficient action tokenization, resulting in less precise predictions. When $K = 4$, the policy effectively meets task requirements. Further increasing K results in stable performance with negligible fluctuations, indicating that the policy has likely reached its peak performance for the given tasks, with additional scaling providing little to no further benefit.

K. Additional Real-World Experiment

We provide further visualization of the real-world experiments presented in the main paper. As shown in Fig. 18,

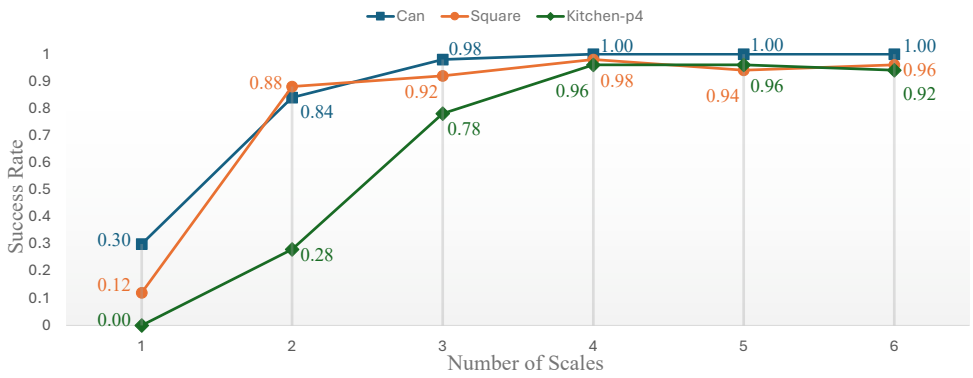


Figure 17. **Ablation Study on K.** We evaluate the performance of CARP across three tasks using six different scale configurations. Results indicate that when the number of scales exceeds 4, the model achieves optimal performance. Considering both model efficiency and performance, we set $K = 4$ in all experiments throughout the paper.

CARP generates smooth and successful trajectories for the *Cup* and *Bowl* tasks, with temporal progression illustrated from left to right.

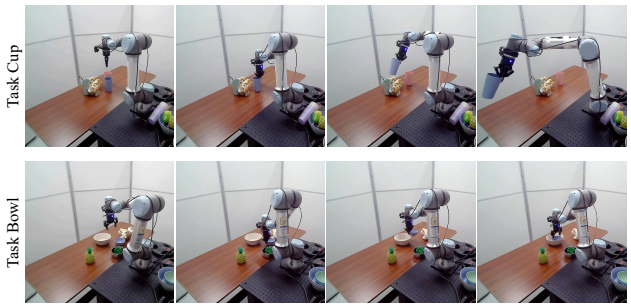


Figure 18. **Visualization of CAPR on Real-World Tasks.** CARP generates smooth and successful trajectories on the Cup and Bowl tasks, progressing from left to right.

Beyond the real-world evaluations on a UR5e robot arm, we further deploy CARP on a distinct robotic embodiment: a 7-DoF Franka Emika Panda arm. For this, we adopt the challenging *FurnitureBench* benchmark [20], which comprises long-horizon, contact-rich manipulation tasks (e.g., pick, place, insert, screw, and flip), with episodes spanning up to 1000 steps (700 steps for `One_Leg`). A corresponding standard simulator is also provided, as shown in Fig. 19.

We first evaluate CARP and Diffusion Policy (DP) in simulation on three tasks, followed by real-world deployment of the `One_Leg` task. All experiments are conducted in the state-based setting. To bridge the sim-to-real gap during real-world deployment, 6-DoF object poses are estimated using AprilTags provided by FurnitureBench [20], enabling consistent state-based policy execution.

For simulation, we use 200 trajectories per task from [5]. DP is trained using its official implementation [12] with 100 DDPM denoising steps. CARP follows the same state-based, single-task setting. We evaluate success rates using 1024 rollouts per task for statistical stability. As shown in Tab. 7, CARP achieves competitive performance while offering significantly lower inference time and fewer parameters.

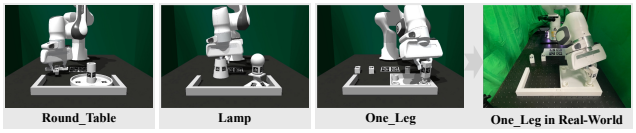


Figure 19. **FurnitureBench Tasks for Evaluation.** All three tasks are first evaluated in simulation. A corresponding real-world environment is then constructed to assess the performance in real-world.

Policy	One_Leg	Round_Table	Lamp	Inf.Time ↓	Params ↓
DP-C	39.62%	5.76%	3.91%	74.05s	66.06M
CARP	**43.75%**	**6.25%**	**4.30%**	**6.29**s	**2.54**M

Table 7. **Simulation Results on FurnitureBench (State-Based).** All policies are trained under identical single-task settings. Compared to DP-C [12] (with 100 DDPM denoising steps), CARP achieves comparable success rates while offering significantly lower inference time and parameter count.

We further evaluate CARP on the real-world `One_Leg` task using 40 expert demonstrations collected via a 3D SpaceMouse (left panel of Fig. 20). The task involves complex rotations and contact-rich interactions, as illustrated in the right panel of Fig. 20. CARP achieves higher success rates across key stages, especially in precision-critical steps such as *Insert*, demonstrating robust real-world performance.

L. Analysis on Fine-Grained Manipulation

Beyond standard pick-and-place tasks, which are relatively straightforward for robotic manipulation, tasks requiring fine-grained skills have garnered increasing attention. For

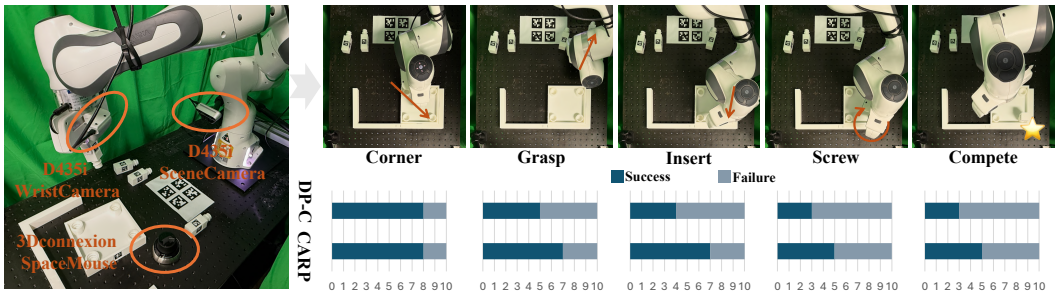


Figure 20. **Real-World Evaluation on the `One_Leg` Task.** The left panel shows the real-world setup, while the right panel illustrates the execution process and stage-wise results (left to right). Success rates at key stages are reported below. CARP produces smoother motions and outperforms baselines in precision-critical phases such as *Grasp* and *Insert*.

example, Nut-Assembly and Threading from the MimicGen [41] benchmark demand precise action generation to ensure successful task completion, as illustrated in Fig. 21. In Nut-Assembly, the robot must place a nut onto a designated peg, which is slightly larger in size. Furthermore, the Threading task requires the robot to insert a needle into a small hole on a tripod, a significantly more challenging task due to the minimal margin for error. To evaluate CARP's fine-grained manipulation capability, we compare it with the state-of-the-art Sparse Diffusion Policy (SDP) [63] in the multi-task setting, and Diffusion Policy (DP) [12] in the single-task setting. We evaluate success rates alongside the mean and variance of the distance between the ideal insertion centers of the fixed structures (peg, tripod) and the centers of the tools (nut, needle) at the moment of first contact. A lower mean distance indicates higher action precision, while a smaller variance reflects the model's ability to consistently achieve accurate and stable manipulations.

As summarized in Tab. 8 and Tab. 9, our *coarse-to-fine* autoregressive prediction framework demonstrates strong performance in fine-grained tasks, achieving competitive results comparable to diffusion-based policies. Notably, CARP consistently achieves lower mean error and reduced variance across most tasks, regardless of whether in single-task or multi-task settings. Moreover, CARP achieves these results with an inference speed that is **10** times faster than current diffusion-based policies, highlighting its efficiency and effectiveness in fine-grained robotic manipulation.

M. Task Visualizations

In this section, we provide visualizations of the tasks used in our experiments. For the single-task experiment, the corresponding visualizations are presented in Fig. 22. For the multi-task experiment, visualizations are shown in Fig. 23. For the long-horizon, multi-stage Kitchen experiment, we provide visualizations in Fig. 24, along with the sequential execution process in Fig. 25. Finally, for real-world experiments, visualizations are included in Fig. 26.

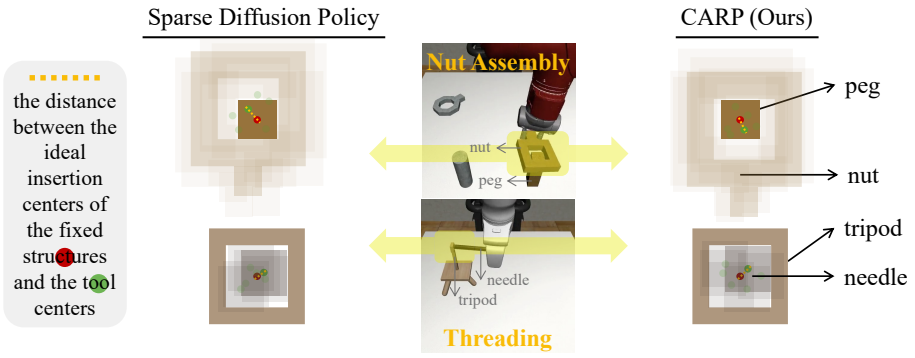


Figure 21. **Visualization of Fine-Grained Manipulation.** We evaluate the precision of generated actions by measuring the distance between the ideal and actual contact centers, represented by the dotted yellow line. Experiments are conducted on Nut-Assembly (top row) and Threading (bottom row). The visualization highlights that CARP achieves a comparable level of precision to diffusion-based policies.

Policy	Inference Speed ↑	*Nut Assembly*			*Threading*		
		Success ↑	Mean ↓	Variance ↓	Success ↑	Mean ↓	Variance ↓
SDP	8.2 hz	0.54	7.70	2.36	**0.70**	**5.20**	**1.25**
CARP	**118.5** hz	**0.66**	**7.30**	**1.68**	**0.70**	5.50	1.36

Table 8. **Fine-Grained Manipulation Study on Multi-Task Setting.** CARP demonstrates a high level of precision comparable to diffusion-based policies, as indicated by the similar mean and variance values. Additionally, CARP achieves a significant speed advantage, running over 10 times faster than diffusion-based approaches. This highlights CARP as a superior balance between performance and efficiency.

Policy	Inference Speed ↑	*Nut Assembly*			*Threading*		
		Success ↑	Mean ↓	Variance ↓	Success ↑	Mean ↓	Variance ↓
DP-C	10.13 hz	0.80	5.20	1.86	**0.88**	4.08	1.07
CARP	**119.05** hz	**0.82**	**5.12**	**1.28**	**0.88**	**3.92**	**0.94**

Table 9. **Fine-Grained Manipulation Study on Single-Task Setting.** To eliminate potential underfitting caused by multi-task training, we evaluate fine-grained tasks under single-task settings. CARP achieves success rates on par with DP-C, while offering lower variance and over 10× faster inference. These results highlight CARP's ability to maintain high precision and stability with greater inference efficiency.

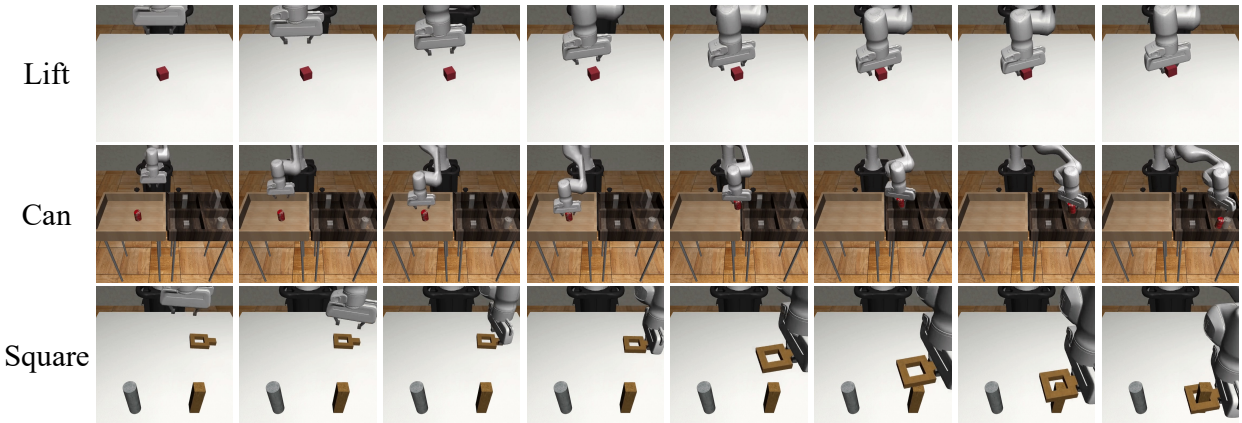


Figure 22. **Visualization of Tasks in Single-Task Experiment.**

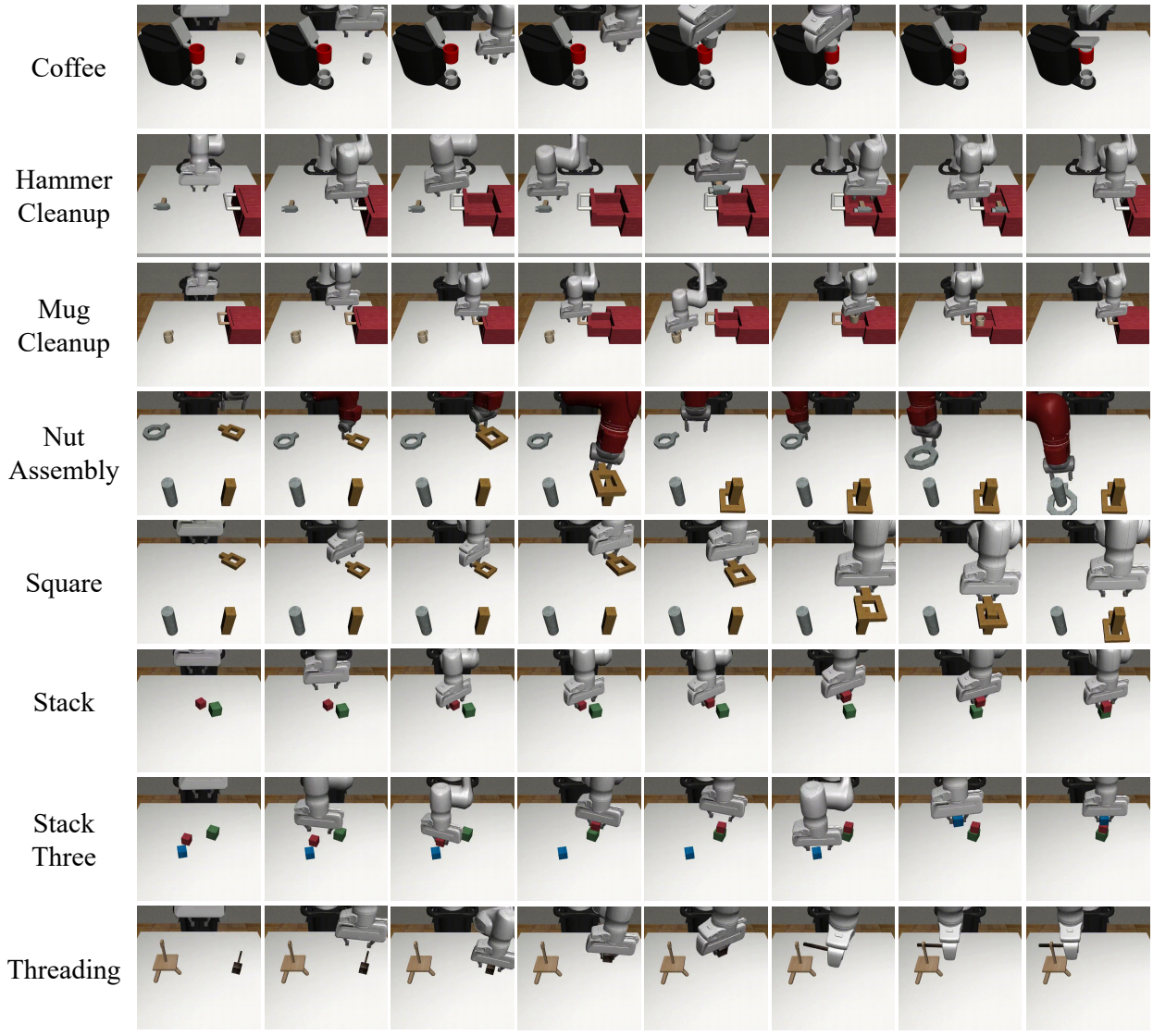


Figure 23. **Visualization of Tasks in Multi-Task Experiment.**

Figure 24. **Visualization of All Interaction Tasks in Kitchen Experiment.**

Figure 25. **Visualization of the Consecutive Execution in Kitchen Experiment.**

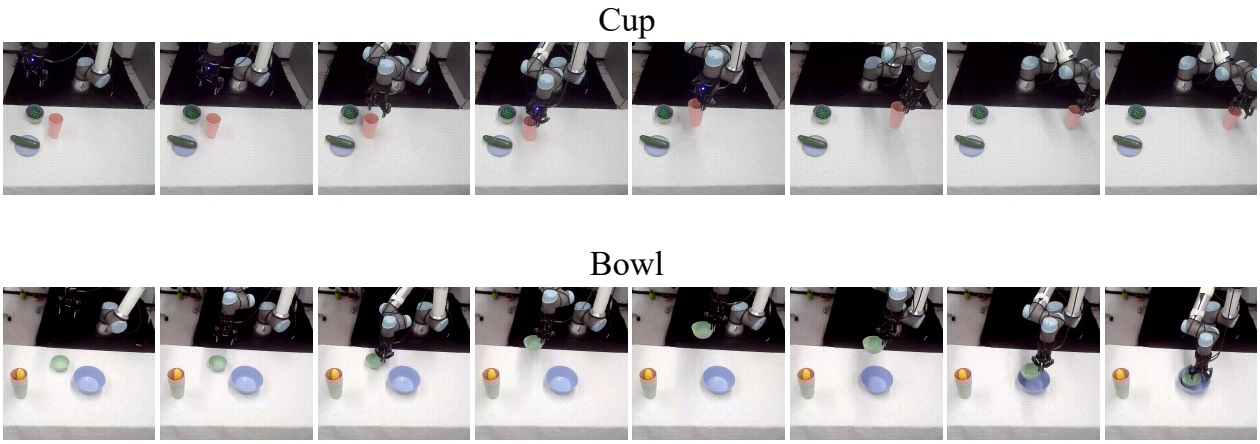


Figure 26. **Visualization of Tasks in Real-World Setup.**