Chain-of-Action: Trajectory Autoregressive Modeling for Robotic Manipulation

Wenbo Zhang[1,2] **Tianrun Hu**[3] **Hanbo Zhang**[3] **Yanyuan Qiao**[2] **Yuchu Qin**[4]
Yang Li[5] **Jiajun Liu**[5,6] **Tao Kong**[1] **Lingqiao Liu**[2,†] **Xiao Ma**[1,†]

[1]ByteDance Seed [2]The University of Adelaide [3]National University of Singapore
[4]Chinese Academy of Sciences [5]CSIRO [6]The University of Queensland

wenbo.zhang01@adelaide.edu.au, xiaoma@bytedance.com, lingqiao.liu@adelaide.edu.au

†Corresponding authors

Project page: https://chain-of-action.github.io/

Code: https://github.com/ByteDance-Seed/Chain-of-Action

Abstract

We present **Chain-of-Action (CoA)**, a novel visuomotor policy paradigm built upon *Trajectory Autoregressive Modeling*. Unlike conventional approaches that predict next step action(s) *forward*, CoA generates an entire trajectory by explicit *backward* reasoning with task-specific goals through an action-level Chain-of-Thought (CoT) process. This process is unified within a single autoregressive structure: (1) the first token corresponds to a stable keyframe action that encodes the task-specific goals; and (2) subsequent action tokens are generated autoregressively, conditioned on the initial keyframe and previously predicted actions. This backward action reasoning enforces a global-to-local structure, allowing each local action to be tightly constrained by the final goal. To further realize the action reasoning structure, CoA incorporates four complementary designs: continuous action token representation; dynamic stopping for variable-length trajectory generation; reverse temporal ensemble; and multi-token prediction to balance action chunk modeling with global structure. As a result, CoA gives strong spatial generalization capabilities while preserving the flexibility and simplicity of a visuomotor policy. Empirically, we observe that CoA outperforms representative imitation learning algorithms such as ACT and Diffusion Policy across 60 RLBench tasks and 8 real-world tasks.

1 Introduction

visuomotor policies have made substantial progress in enabling robots to perform complex manipulation tasks from raw sensory observations. With the rise of large-scale demonstrations [5, 19, 37] and powerful neural architectures [36, 11], recent methods have increasingly focused on end-to-end learning from visual inputs to low-level control[15, 2].

To better model multi-modal action distributions and mitigate compounding errors, various modeling paradigms have been proposed [3, 45]. For instance, ACT [45] employs a conditional variational autoencoder to learn action distributions and introduces action chunking to reduce compounding errors. Diffusion Policy [3] formulates action generation as a denoising process, capturing complex, multi-modal behaviors more effectively. Many subsequent developments have explored enhancements in multiple directions, including enriched sensory inputs [41, 40], improved network architecture [4, 24], expanded datasets[5], and scaled model capacity, represented by trend of VLA (vision-language-action) model [21, 28, 25, 1].

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Despite a wide range of these improvements, most of methods still follow a forward prediction paradigm, as illustrated in Figure 1. While this formulation is intuitive and widely adopted, it suffers from a critical limitation: the accumulation of *compounding errors* [32, 18, 22, 30] during execution. The root cause lies in the training objective: these models are optimized to predict the next-step action based on current observation, rather than to ensure successful completion of tasks with long-horizon [32]. While techniques such as action chunking and image goal conditioned behavioral cloning [28, 37] have been introduced to alleviate compounding errors, they primarily address the symptoms rather than the root cause, which lies in the inherently myopic nature of forward prediction.

We approach the problem from the *opposite end*, both conceptually and practically, by reversing the action generation process. While the change in direction may appear simple, it reflects a fundamental shift in how we conceptualize action generation. Instead of predicting actions in a forward, step-wise manner, we propose to construct action sequences in reverse, forming a *chain of actions* that starts from the a keyframe action [13, 34, 9, 8], and backward towards the initial state. Our insight is that the keyframe action encodes the task-specific goal, which provides a strong structural prior to guide the entire action sequence. By explicitly generating actions from the goal backward, our method enforces a global-to-local consistency [26, 39] that significantly mitigates compounding errors and enhances generalization under distribution shifts.

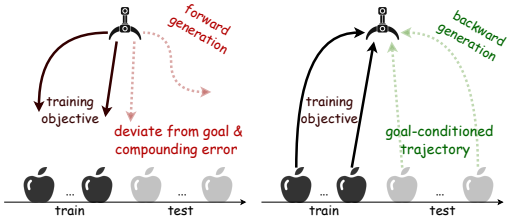


Figure 1: **Comparison between a conventional visuomotor policy (left) and our proposed Chain-of-Action (right).** The former is optimized to predict step-wise actions based on current observations, rather than long-term goals, often leading misaligned behaviors during execution. In contrast, Chain-of-Action adopts a backward generation paradigm, producing goal-conditioned trajectories that reliably execute toward the intended target.

To realize this backward reasoning paradigm while maintaining scalability potential [16, 35] for end-to-end training, we unify the entire reverse generation process into a single autoregressive framework. While the formulation is theoretically effective, its practical viability depends on four extra specific designs. These are not optional improvements, but necessary for stable training and reliable closed-loop execution. (1) Continuous action representation: Discretizing actions into finite bins introduces resolution loss [20, 31, 23], which becomes particularly problematic in long-horizon autoregressive generation. In our backward generation setup, even small quantization errors can accumulate from the goal backward, leading to significant deviations in earlier steps. To preserve fine-grained structure and trajectory fidelity, we adopt a continuous action representation. (2) Local action modeling: While the backward autoregressive structure effectively propagates high-level intent from the goal, it does not explicitly model local action dependencies [20, 45, 3] within a sub-trajectory. To address this, we adopt a multi-token prediction strategy [7, 43] during training, which encourages the model to jointly predict short action chunks. This enhances local coherence and stabilizes training. (3) Dynamic stop: Closed-loop execution [27] requires our generation stop at right point. However, in continuous action spaces, there is no discrete end-of-sequence (EOS) token to indicate termination [43]. We thus design a distance-based stop mechanism that enables the model to determine when to stop based on proximity to the goal, reducing over-generation and improving execution efficiency. (4) Reverse temporal ensemble: Original ensemble strategies [45], used in ACT, are designed under forward temporal assumptions and are not directly applicable to our backward generation setting. To address this, we develop a reverse-compatible variant that ensembles multiple backward rollouts, mitigating temporal misalignment and reducing variance during closed-loop execution.

Chain-of-Action (CoA), which integrates these four essential components into a single autoregressive framework, achieves strong performance in both simulation and real-world settings. CoA outperforms ACT by 16% and Diffusion Policy by 23% across 60 RLBench tasks, the most comprehensive evaluation conducted on this benchmark to date, and surpasses ACT by 15% in real-world robotic manipulation. Crucially, CoA adopts comparable architectures and training setups to ACT, underscoring that the performance gains stem from a principled shift in the modeling paradigm. These results position our trajectory autoregressive modeling as a competitive alternative for visuomotor policy learning.

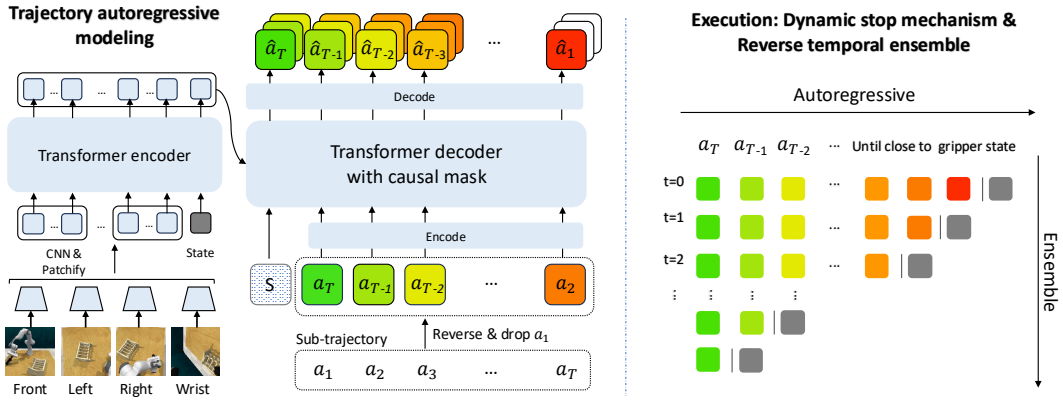


Figure 2: **Chain-of-Action built on trajectory autoregressive modeling.** The left part illustrates the network architecture where notation is for the training stage, and the right part illustrates the execution process. The model encodes visual and proprioceptive observations and generates actions in reverse order from a predicted keyframe action by an autoregressive decoder. *For clarity, the keyframe action a_T is shown in green, and subsequent steps are visualized with a gradual color transition.*

2 Related work

Hierarchical modeling in robotic manipulation A widely adopted strategy in robotic manipulation is to first identify high-level keyframes, and then rely on predefined controllers to handle the low-level execution. This paradigm is exemplified by C2F-ARM [13] and extended by methods such as PerAct [34], RVT [9], RVT-2 [8]. Recent works like ChainedDiffuser [39] and HDP [26] propose neural planners to replace traditional optimization-based planners. Despite these advances, such methods still operate in an open-loop manner [39, 26] between keyframes and struggle to adapt to dynamic environments. Our method also builds on the notion of keyframes, but differs fundamentally in its formulation. By unifying keyframe detection and trajectory generation within a single autoregressive framework, it enables efficient environment-aware action prediction and closed-loop execution, where the model can continuously refines its actions based on feedback. As a result, our method no longer relies on high-fidelity 3D inputs for one-shot accurate predictions, which are commonly required by those hierarchical approaches.

CoT-style methods in robotic manipulation A separate line of research explores CoT-style VLA agents [44, 6, 38, 46], which introduce intermediate semantic representations—such as imagined image goal, visual trace, bounding boxes, or gripper pose, as guidance for subsequent action generation. Orthogonal to these directions, our work focuses on modeling the reasoning process directly between actions without relying on extra modalities as intermediate representations. This design makes our method broadly compatible with different sensory inputs and policy architectures.

3 Chain-of-Action for robotic manipulation

Formulation The core idea of Chain-of-Action is to model trajectory generation in reverse: starting from a task-specific goal and predicting actions backward in an autoregressive manner. This reverse formulation imposes a global-to-local structure, anchoring the rollout to the final intent and mitigating compounding errors. An overview of the CoA pipeline is shown on the left of Figure 2. We adopt the definition of keyframe originally from C2F-ARM [13], where a keyframe is identified as a time step at which the gripper state changes or the joint velocities approach zero. This simple yet effective heuristic captures transitions between semantically meaningful phases, such as grasp completion or object placement, and can be interpreted as a task-specific goal. Representing the goal as an action allows it to share the same embedding space with all other actions, enabling seamless backward generation. For each training sample, CoA learns to model the action sequence in reverse order using an autoregressive decoder. *This formulation enforces a reverse causal dependency among actions, yielding a goal-conditioned reasoning chain. Such backward chaining lies at the heart of the our framework, which models the trajectory distribution as:*

$$p(a_{1:T} \mid O) = \underbrace{p(a_T \mid O)}_{\text{Keyframe Action}} \cdot \underbrace{p(a_{T-1} \mid a_T, O) \ldots p(a_2 \mid a_{3:T}, O)}_{\text{Reverse Reasoning Actions}} \cdot \underbrace{p(a_1 \mid a_{2:T}, O)}_{\text{Executed Action}} \tag{1}$$

where $\mathbf{a}_T$ denotes the keyframe action, and $\mathbf{O}$ denotes the observation context, including visual input $\mathbf{I}$ and proprioceptive state $\mathbf{S}$. To make the meaning of $\mathbf{a}_{1:T}$ explicit, we clarify how each training sample is constructed. A sub-trajectory is sampled from an expert demonstration by selecting a segment that starts at a random time step and ends at the next first keyframe action. The observation $\mathbf{O}$ is taken from the initial step, and $\mathbf{a}_{1:T}$ denotes the sequence of actions from the current step up to (and including) the keyframe. Each pair $(\mathbf{O}, \mathbf{a}_{1:T})$ forms an independent training example.

Algorithm 1: Training Phase

Inputs: dataset $\mathcal{D}$
Modules:
- Action encoder f_{enc}: $a_t \mapsto x_t$
- Action decoder f_{dec}: $x_t \mapsto a_t$
- Transformer F_θ: encoder-decoder model

Parameters: learned token x_{SOS}, loss weight λ
for *iteration* $n = 1, 2, \ldots$ **do**
  Sample $(\mathbf{I}, \mathbf{S}, \tau = (a_1, \ldots, a_T))$ from $\mathcal{D}$ based on keyframe heuristic
  $x_{1:T} \leftarrow \text{REVERSE}(f_{\text{enc}}(a_{1:T}))$
  $H \leftarrow \text{CONCAT}(x_{\text{SOS}}, x_{1:T-1})$
  $\hat{x}_{1:T} \leftarrow F_\theta(H \mid \mathbf{I}, \mathbf{S})$
  $\hat{a}_{1:T} \leftarrow \text{REVERSE}(f_{\text{dec}}(\hat{x}_{1:T}))$
  $\mathcal{L}_{\text{reg}} \leftarrow \sum_{t=1}^{T} \mathcal{L}_{\text{action}}(\hat{a}_t, a_t)$
  $\mathcal{L}_{\text{latent}} \leftarrow \sum_{t=1}^{T} \mathcal{L}_{\text{latent}}(\hat{x}_t, f_{\text{enc}}(a_t))$
  $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{reg}} + \lambda \cdot \mathcal{L}_{\text{latent}}$
  Update θ, x_{SOS} via backprop on $\mathcal{L}_{\text{total}}$

Algorithm 2: Inference Phase

Inputs: image $\mathbf{I}$, proprioceptive state $\mathbf{S}$
Modules:
- Action encoder f_{enc}: $a_t \mapsto x_t$
- Action decoder f_{dec}: $x_t \mapsto a_t$
- Transformer F_θ: encoder-decoder model

Parameters: learned token x_{SOS}, max length $T_{\max}$
Initialize $H \leftarrow [x_{\text{SOS}}]$
for $t = 1$ *to* $T_{\max}$ **do**
  $\hat{x}_t \leftarrow F_\theta(H \mid \mathbf{I}, \mathbf{S})$
  Append $\hat{x}_t$ to H
  if $\text{STOP}(f_{dec}(\hat{x}_t), \mathbf{S})$ **then**
    break
Remove x_{SOS}: $H' \leftarrow H[1:]$
$\hat{a}_{1:T} \leftarrow \text{REVERSE}(f_{\text{dec}}(H'))$
Return: action sequence $\hat{a}_{1:T}$

Continuous action token representation CoA adopts continuous action token representation. However, directly training with continuous latent tokens introduces its own challenge. Unlike discrete token embeddings [21] that are fixed indices supervised by a softmax classifier, our latent actions are generated through a learned encoder. In this setting, imposing loss directly on the action space fails to constrain the latent space to exhibit temporal consistency during autoregressive decoding. As a result, the latent space lacks meaningful regularization, allowing encoding errors to propagate and amplify through the autoregressive process. To address this, we introduce a latent consistency loss to regularize latent action space: $\mathcal{L}_{\text{consistency}} = \|\hat{x}_t - f_{\text{enc}}(\mathbf{a}_t)\|, \quad \text{where } f_{\text{enc}}(\mathbf{a}_t) = W_{\text{enc}}\mathbf{a}_t + b_{\text{enc}}$. Here, $\hat{x}_t$ denotes the predicted latent from the previous timestep, and $f_{\text{enc}}(\mathbf{a}_t)$ is the encoded latent of the ground-truth action. This loss acts as an inductive bias to align the latent space with temporal dynamics, improving the quality of autoregressive generation.

Locality modeling Multi token prediction (MTP) [7] can serve as a regularization for action locality modeling. We assign the last K layers of the transformer decoder to produce predictions for different future steps. Concretely, layer k predicts token $\hat{x}_{t+k}$, where $k = 1, ..., K$, making the model aware of the mutual dependencies across the next K steps in a single forward pass. This design introduces temporal locality into the decoding process, enhancing stability in long-horizon generation while maintaining our global-to-local chain-like structure. Importantly, this regularization is applied only during training and is removed at inference.

Dynamic stop To enable flexible-length trajectory generation in continuous action space, we introduce a distance-based stop criterion. The core idea is to terminate decoding once the predicted action sufficiently approximates the current execution state, indicating that the backward-generated trajectory has successfully reached the present, as shown in bottom-right in Figure 2. This stop mechanism is agnostic to the specific action representation and can be readily applied to delta actions or joint-space control by adjusting the reference point accordingly.

Reverse temporal ensemble We introduce a reverse temporal ensembling strategy tailored for CoA. As shown in the bottom-right corner of Figure 2, our approach aligns multiple reversed sub-trajectories by their predicted keyframe action a_T, which serves as the anchor point for autoregressive decoding. This design offers a unique advantage in CoA: since each trajectory is decoded in reverse from the keyframe, the compounding error is inherently constrained by the accuracy of the keyframe

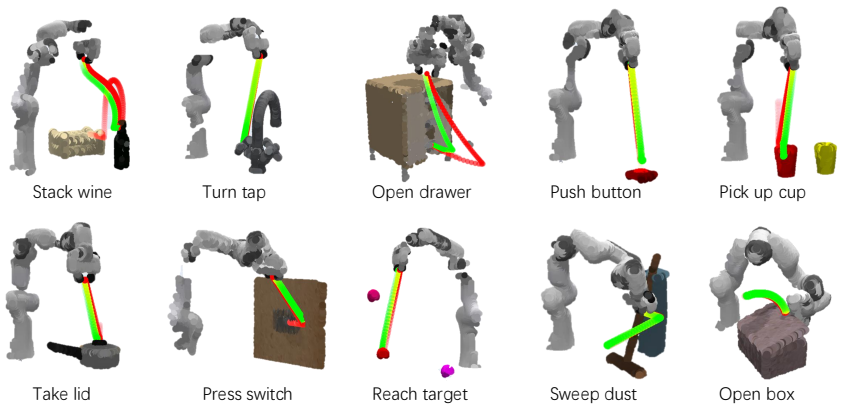


Figure 3: Visualization of predicted sub-trajectories across 10 widely used tasks. Detail refers to Table 1. Red waypoints represent ground-truth trajectories, and green waypoints denote model predictions. Each predicted trajectory is generated backward from a keyframe action to the current gripper state, enabling consistent goal-conditioned trajectory generation. The model successfully handles both straight and curved motion patterns.

action. By further improving the accuracy of the keyframe action through ensembling, we tighten this constraint even more.

4 Implementation details

Network architecture Our network follows a similar overall architecture to ACT [45], consisting of a 4-layer Transformer encoder and a 7-layer Transformer decoder. However, unlike ACT, our model does not include the conditional variational autoencoder (CVAE) module. The final decoder layer contains multiple parallel heads for multi-token prediction (MTP), which are only used during training. The observations consist of multi-view RGB images and corresponding robot states, which are encoded as follows: each image view is processed by a ResNet-18 vision encoder to extract visual tokens. The gripper state is projected via a learnable linear layer into a token representation. All tokens are concatenated and passed through the Transformer encoder to produce contextual features for decoding. Autoregressive action generation is performed by the Transformer decoder, which is initialized with a learnable start-of-sequence (SOS) token. This token serves as a query for the first prediction, corresponding to the keyframe action. The decoder iteratively predicts previous actions until the generated action becomes sufficiently close to the current gripper state, where the dynamic stopping criterion is applied. Actions are encoded and decoded into a shared latent embedding space via linear projection layers, which are regularized by the latent consistency loss as described earlier. Additionally, sinusoidal positional embeddings are added to the action tokens to preserve temporal ordering cues.

Training For each training sample, we apply two loss terms: a regression loss in the action space and a consistency loss in the latent space. Both are computed with the MTP regularization, where the model predicts a chunk of K actions at each decoding step. The total loss is defined as:

$$\mathcal{L}_{\text{total}} = \sum_{t=1}^{T} \sum_{k=1}^{K} \left\| \hat{\mathbf{a}}^k_{t+k-1} - \mathbf{a}_{t+k-1} \right\| + \lambda_1 \left\| \hat{x}^k_{t+k-1} - f_{\text{enc}}(\mathbf{a}_{t+k-1}) \right\| , \tag{2}$$

where $\hat{\mathbf{a}}^k_{t+k-1}$ and $\hat{x}^k_{t+k-1}$ are the predicted action and its latent embedding from k-th head of MTP layer at step t, and $f_{\text{enc}}(\cdot)$ is the action encoder network. Note that for decoding steps where $t+k-1 > T$, the corresponding terms are masked out and do not contribute to the loss. This ensures that predictions beyond the trajectory horizon are excluded from supervision. For parallel training with a batch of samples, we set T_{max} as the maximum sub-trajectory length (practically the longest in the dataset), and zero-pad all shorter sequences accordingly. The loss for padded steps is masked out to avoid affecting gradient updates.

Execution For each inference, CoA generates an entire trajectory segment, which can be executed in either open-loop or closed-loop mode. We generally adopt closed-loop control, as it allows

reverse temporal ensembling to continuously refine the predicted actions during execution. Under the dynamic stopping setting, we compute the Euclidean distance between the predicted action and the current end-effector pose. This termination criterion is well-suited for our continuous end-effector pose action space.

5 Experiments

In Sec. 5.1, we introduce our experiment settings, including simulation environment, train, evaluation settings and metrics. Then we show detailed results of the overall comparison in Section 5.2. To dive into the spatial generalization and obtain better understanding of how CoA work, more specific evaluation is shown in Section 5.3. Ablation studies of each components in CoA are shown in Section 5.4. Finally, the real-world robot evaluations are shown in Section 5.5

Table 1: Success rate across 10 widely-used tasks in RLBench.

Task	CoA	ACT	DP	Octo
Stack Wine	**0.80**	0.56	0.56	0.52
Turn Tap	**0.56**	0.36	0.32	0.28
Open Drawer	**0.88**	0.52	0.44	0.84
Push Button	**0.76**	0.08	0.12	**0.76**
Pick Up Cup	**0.80**	0.20	0.00	0.44
Take Lid	**0.80**	0.40	0.60	0.76
Press Switch	0.44	0.52	**0.56**	0.44
Reach Target	0.84	**0.88**	0.08	0.60
Sweep Dust	0.92	**1.00**	**1.00**	0.80
Open Box	0.76	0.36	0.48	**0.96**
Avg.	**0.756**	0.488	0.416	0.644

5.1 Simulation experiment settings

Simulation setup We conduct simulation experiments using RLBench [14], a widely-used benchmark built on CoppeliaSim and interfaced via PyRep. The robot is a 7-DoF Franka Emika Panda mounted behind a tabletop workspace. Observations are collected from four RGB cameras (front, left shoulder, right shoulder, and wrist). Images are rendered at a resolution of 128×128.

Baseline We compare our method against representative approaches from three categories: (1) training visuomotor policies from scratch, including ACT and Diffusion Policy (DP); (2) finetuned generalist robotic policies, represented by Octo [28]; (3) 3D-based hierarchical methods, including PerAct [34], 3D Diffuser Actor [17], and RVT-2 [8]. We note 3D-based hierarchical methods fundamentally differ from our approach by relying on 3D inputs and motion planners to generate trajectories. We provide additional discussion on these differences in Appendix A.

Training and evaluation protocol To ensure broad and representative evaluation, our main benchmark is conducted on a tailored set of 60 RLBench tasks, where CoA is compared with ACT and DP, Each method is trained on 100 demonstrations and evaluated on 25 demonstrations per task. For ACT and DP, we follow the RLBench training protocol introduced in [33], which is detailed in Appendix D. To better demonstrate the effectiveness of our modeling paradigm, we align our base architecture with ACT and introduce modifications primarily in the transformer decoder, as detailed in Section 4. The strong performance of these baselines—such as perfect success rates on tasks like *Sweep Dust* and competitive results on others (Table 1)—confirms that all reference models are properly trained. For comparison with Octo, we adopt the evaluation subset RLBench-10 proposed in [42] and use the reported results. This subset is also used for our ablation studies for convenience. To facilitate comparison with 3D-based hierarchical methods, we evaluate on the RLBench-18 split [34], using reported results from prior work [8]. Note that our RLBench-18 setup is not strictly identical to the PerAct protocol: we train single-task policies and use only variation 0 for both training and evaluation. Hence, the RLBench-18 results are reported as contextual reference rather than a fully protocol-matched comparison.

5.2 Overall comparisons

The overall results are presented in Figure 4, with task-wise averages summarized in the accompanying wrapped table. To better assess the effectiveness of our method, we report task-level improvements over both ACT and DP. Compared to ACT, our method achieves higher success rates on 81.7% of the tasks, with an average improvement of 16.3%. Relative to DP, our method improves performance on 80.0% of the tasks, with an average gain of 23.2%. These improvements are especially pronounced in tasks involving significant variation in object position and pose, indicating stronger spatial generalization. As ACT and CoA share a consistent Transformer encoder-decoder architecture and are trained on the same setting, the observed gains highlight the effectiveness of our modeling paradigm.

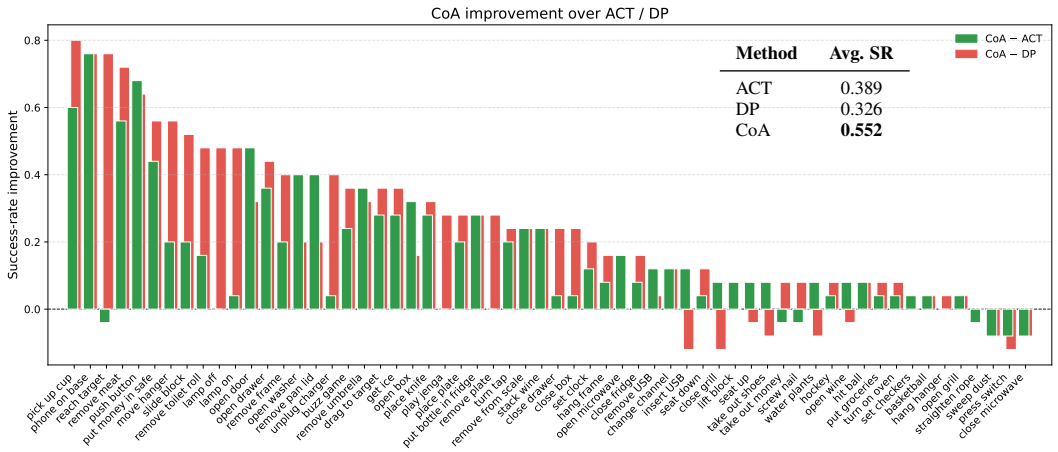


Figure 4: Success rate improvement on RLBench-60, sorted by improvement from high to low. The average success rate over all tasks is shown in the inset on the right.

Table 2: Comparison on the RLBench-18. 3D-based hierarchical methods use 3D point clouds and motion planners, while image-based visuomotor policies operate directly on RGB inputs.

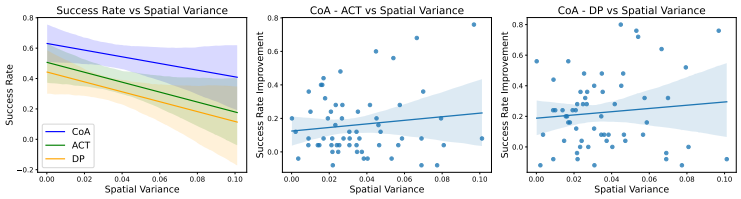
	3D-based hierarchical methods			Image-based visuomotor policies				
Task	PerAct	3D Diffuser Actor	RVT-2	Image-BC (CNN)	Image-BC (ViT)	DP	ACT	CoA
Close Jar	55.2 ± 4.7	96.0 ± 2.5	100.0 ± 0.0	0	0	0	0	0
Drag Stick	89.6 ± 4.1	100.0 ± 0.0	99.0 ± 1.7	0	0	0	0	0
Insert Peg	5.6 ± 4.1	65.6 ± 4.1	40.0 ± 0.0	0	0	0	0	0
Meat off Grill	70.4 ± 2.0	96.8 ± 1.6	99.0 ± 1.7	0	0	16	32	88
Open Drawer	88.0 ± 5.7	89.6 ± 4.1	74.0 ± 11.8	4	0	44	52	88
Place Cups	2.4 ± 3.2	24.0 ± 7.6	38.0 ± 4.5	0	0	0	0	0
Place Wine	44.8 ± 7.8	93.6 ± 4.8	95.0 ± 3.3	0	0	56	56	80
Push Buttons	92.8 ± 3.0	98.4 ± 2.0	100.0 ± 0.0	0	0	0	32	28
Put in Cupboard	28.0 ± 4.4	85.6 ± 4.1	66.0 ± 4.5	0	0	0	0	8
Put in Drawer	51.2 ± 4.7	96.0 ± 3.6	96.0 ± 0.0	8	0	40	60	88
Put in Safe	84.0 ± 3.6	97.6 ± 2.0	96.0 ± 2.8	4	0	24	36	80
Screw Bulb	17.6 ± 2.0	82.4 ± 2.0	88.0 ± 4.9	0	0	0	0	0
Slide Block	74.0 ± 13.0	97.6 ± 3.2	92.0 ± 2.8	0	0	0	36	64
Sort Shape	16.8 ± 4.7	44.0 ± 4.4	35.0 ± 7.1	0	0	0	0	0
Stack Blocks	26.4 ± 3.2	68.3 ± 3.3	80.0 ± 2.8	0	0	0	0	0
Stack Cups	2.4 ± 2.0	47.2 ± 8.5	69.0 ± 5.9	0	0	0	0	0
Sweep to Dustpan	52.0 ± 0.0	84.0 ± 4.4	100.0 ± 0.0	0	0	100	100	92
Turn Tap	88.0 ± 4.4	99.2 ± 1.6	99.0 ± 1.7	8	16	32	36	56
Average	48.7	81.3	81.4	1.33	0.89	17.33	24.44	37.33

The results suggest that a principled change in how action sequences are represented and generated can lead to substantially better performance under distribution shifts. The detailed per-task results are in Appendix B. In addition, the comparison with Octo and detailed results with ACT and DP over 10 selected tasks are shown in Table 1. Results on the RLBench-18 benchmark are reported in Tab 2. We observe that, in these settings, CoA outperforms the finetuned generalist robot policy Octo, while a substantial performance gap remains compared to the 3D-based hierarchical methods.

5.3 Dive into spatial generalization

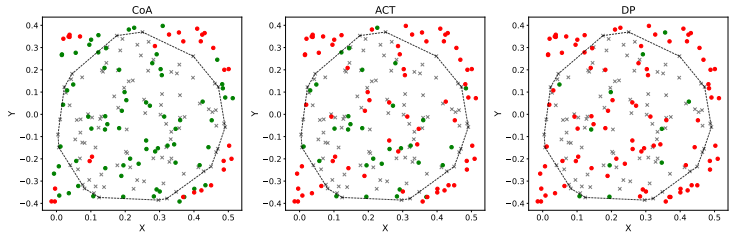
Although CoA significantly outperforms ACT and DP on the overall benchmark, it remains crucial to understand *why* such improvements emerge. To this end, we investigate the spatial generalization behavior of our model from three complementary perspectives.

First, in the *Interpolation vs. Extrapolation case study*, we analyze CoA's performance under controlled spatial distributions within a single representative task. This study reveals that CoA not only achieves higher success rates under in-distribution (interpolation) configurations, but also demonstrates a substantially larger advantage in out-of-distribution (extrapolation) settings, indicating stronger spatial generalization.



Method/Gap	Pearson (r)
CoA	-0.1679
ACT	-0.2471
DP	-0.2455
CoA - ACT	0.1311
CoA - DP	0.1028

Figure 5: **Correlation between success rate and spatial variance.** Left image: Overall success rate decreases as object spatial variance increases. Middle and right image: CoA consistently outperforms ACT and DP across varying spatial generalization levels, with larger advantages in more challenging (higher variance) settings. Table: Pearson correlations highlight CoA's robustness to spatial perturbations.



Method	Inter. / Extra.
CoA	0.94 / 0.48
ACT	0.54 / 0.08
DP	0.18 / 0.04

Figure 6: **Interpolate vs. extrapolate performance.** Success rate comparison on interpolation (in-distribution) and extrapolation (out-of-distribution) subsets for the *Push Button* task. CoA maintains stronger performance across both regimes, with a notably smaller degradation under extrapolation. The gray dashed line denotes the 2D convex hull computed from the training samples. Cross markers represent training data, while circular markers denote evaluation samples, where green circles indicate successful executions and red circles indicate failures.

Second, in *Correlation with spatial distribution*, we quantitatively examine how task performance correlates with spatial variation difficulty across the 60 RLBench tasks. The results show that CoA consistently improves over ACT and DP across all spatial variance levels, and that the performance gap widens as spatial generalization becomes more challenging.

Finally, in *Attention-based analysis of action chain*, we visualize the attention maps between action tokens in the Transformer decoder. The attention patterns clearly reveal structured dependencies along the predicted action sequence, supporting the hypothesis that CoA performs chain-like global-to-local reasoning throughout the trajectory generation process.

Interpolation vs. Extrapolation case study We conduct qualitative analyses on *Push button* task to contrast model behavior under interpolated (in-distribution) versus extrapolated (out-of-distribution) spatial configurations. For this analysis, we choose the *Push Button* task due to its large spatial variation and its frequent use in prior works. Unlike the standard benchmark setting, we randomly sample 200 demonstrations from the full dataset and project the button target positions onto the (x, y) workspace plane. We then compute the centroid of all sampled positions and select the 150 samples closest to this centroid based on Euclidean distance, which are used to form a 2D convex hull. Within this convex hull, we randomly assign 100 samples for training and 50 samples for *interpolation* testing, while the remaining 50 samples lying outside the convex hull are used as *extrapolation* testing data. This protocol ensures a controlled and reproducible evaluation of spatial generalization, where extrapolation explicitly corresponds to goal configurations beyond the spatial support of the training distribution.

Correlation with spatial distribution We examine the relationship between success rate and the spatial distribution of objects in the evaluation set, aiming to quantify each model's spatial generalization ability. We use the variance of object coordinates to measure how widely objects are spread in the workspace. As shown in the left plot of Figure 5, all methods exhibit a clear trend: success rate decreases as spatial variance increases. This indicates that there spatial generalization becomes more difficult when object placement is more diverse. The improvement plots in the Figure 5 reveal more details. Compared to ACT and DP, our method consistently outperforms across all levels of spatial variance, and its advantage becomes more pronounced as task difficulty increases. This trend is further supported by quantitative Pearson correlation.

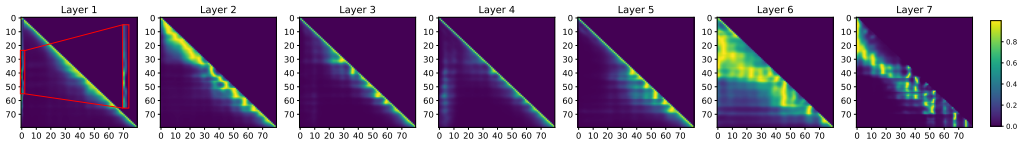


Figure 7: **Attention-based analysis of action chains.** Self-attention maps reveal two key patterns: (1) chain-like dependencies, where each action token attends to recent predecessors, and (2) long-range dependencies (highlighted in the red box in Layer 1), where some tokens directly attend to the initial keyframe action.

As shown in Figure 6, our method outperforms both ACT and DP under both interpolation and extrapolation conditions. Interestingly, while the success rate of CoA in extrapolated settings is about half of that in interpolation, ACT and DP suffer from significantly steeper drops. This highlights the particular difficulty of spatial extrapolation for forward modeling approaches, and suggests that the reverse autoregressive modeling in CoA provides more robust generalization under spatial distribution shifts.

Attention-based analysis of action chain Figure 7 presents the self-attention maps among action tokens across all decoder layers in our model. The horizontal and vertical indices correspond to the autoregressive decoding order of action tokens, where index 0 denotes keyframe action. We observe two distinct attention patterns: (1) a dominant local chain-like structure, where each action token primarily attends to a recent window of preceding tokens, directly reflecting modeling of CoA; and (2) occasional long-range dependencies (e.g., red box in layer 1 and most of tokens in layer 6), where later tokens exhibit strong attention to initial tokens. This behavior suggests the model leverages the goal-conditioned actions to anchor and guide the full trajectory generation.

5.4 Ablation on architectural components

We summarize how each architectural component contributes to performance across 10 representative RLBench tasks (selected consistently with Table 1). The average success rate of each variants are provided in Table 3.

Table 3: Ablation study on individual components by replacing them with alternative settings. The **bold** indicates the best setting adopted by our final model.

Components	Setting	Avg. SR
Modeling Paradigm	Reverse	**0.756**
	Forward	0.668
	Hybrid	0.600
Embedding Loss	Action consistency	0.212
	Latent consistency	**0.756**
Execution	Non-ensemble	0.66
	Reverse ensemble	**0.756**
Num. of MTP head	1	0.710
	2	0.704
	4	0.720
	5	**0.756**
	8	0.672
	10	0.660

Modeling paradigm. CoA's modeling incorporates two core designs: (1) chain-style autoregressive generation, and (2) goal anchoring via a keyframe action. To assess the necessity of each component, we compare *Reverse* ordering of CoA against two ablated variants:

Forward ordering retains the autoregressive structure but removes goal anchoring, starting from the current state and predicting actions forward. Compared to CoA, its lower success rate (0.668 vs. 0.756) highlights the importance of reverse ordering, the core of our proposed modeling. On the other hand, it significantly outperforms ACT (0.668 vs. 0.488), which also uses a autoregressive architecture but predicts fixed-length action chunks. This contrast underscores the advantage of modeling the joint distribution over the entire trajectory, rather than treating it as separated chunks.

Hybrid ordering retains goal anchoring but drops chain-style reasoning. It initializes from the keyframe action but switches to forward action generation, removing backward generation process between actions. As a result, the local continuity of autoregressive is lost, and performance drops greatly to 0.600.

These results confirm that trajectory autoregressive modeling is essential for effective robotic manipulation. Furthermore, reverse autoregressive ordering further enhances performance by anchoring the generation process to the a task-specific goal, providing global guidance throughout the rollout.

Figure 8: Real-world experiments on 8 kitchen tasks.

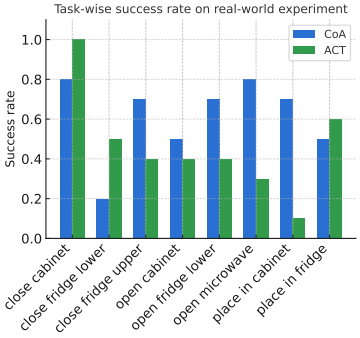


Figure 9: Real-world experimental results.

Number of MTP heads Multi-token prediction regularization enables the model to capture local action chunks while preserving global causality. Allocating too few heads underutilizes this local context, whereas allocating too many heads disrupts the causal structure. A moderate configuration of 5 heads strikes an effective balance, achieving the highest overall score 0.752.

Latent consistency loss We ablate the latent consistency loss by replacing it with a direct action reconstruction loss, which supervises the action encoder and action decoder to reproduce the input action. This substitution leads to a significant performance drop from 0.752 to 0.212, and results in unstable trajectories with unnatural curling. In contrast, enforcing latent consistency yields a well-structured representation and substantially improves task success.

Reverse temporal ensemble We evaluate the impact of reverse temporal ensemble by comparing it with a non-ensemble baseline. Without ensembling, the model achieves 0.660. Applying our reverse-compatible ensemble strategy improves performance to 0.756, highlighting the benefit of aggregating multiple backward rollouts during inference.

5.5 Real-world experiments

We deploy our method on a Fetch robot featuring a 7-DoF arm and a mobile base for real-world validation. For each task, the robot navigates to a predefined location using its built-in 2D LiDAR-based localization system. Observations are captured from a single RGB camera at 640×480. resolution and resized to 224×224. for policy input. Execution is command by absolute end effector poses. To execute commands, we implement a PD controller that calculates the difference between current and desired end effector poses, projects this error into joint space via the Jacobian, and sends velocity commands to the robot. The neural policy operates at 10Hz on a laptop with a 4070 GPU, while the PD controller runs locally on the robot at 1000Hz, with communication handled through ROS for both image data and control commands. As shown in Figure 8, we evaluated CoA and ACT on 8 kitchen tasks, with the number of expert demonstrations ranging from 35 to 81 per task. Each task was evaluated over 10 trials. The results, summarized in Figure 9, show that CoA achieves an average success rate of 0.613, outperforming ACT, which achieves 0.463, by a margin of 15%. The detailed results are in Appendix C.

6 Conclusion

We present **Chain-of-Action**, an action-level reasoning model built upon trajectory autoregressive modeling. By decomposing the joint distribution of the trajectory in reverse, starting from a keyframe and progressing backward to the initial gripper state, our formulation imposes a *global-to-local* structure that enforces consistency between local actions and global task goal. To enable stable training and execution under this backward autoregressive framework, we introduce four necessary design components. Overall, our proposed visuomotor modeling paradigm significantly improves spatial generalization, and we hope it offers a compelling alternative for future visuomotor policy design. However, the current modeling paradigm relies on keyframe heuristics to split the trajectory, which may not generalize well to diverse task types. Future work can explore learning keyframe structures in an unsupervised manner.

7 Acknowledgements

We thank the Adaptive Computing Laboratory at National University of Singapore for facilitating the real-robot experiments in this project. Jiajun Liu was partially supported by the Responsible AI Research Center at the University of Adelaide.

References

[1] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale. In *Robotics: Science and Systems XIX*. Robotics: Science and Systems Foundation, July 2023.

[3] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[4] E. Chisari, N. Heppert, M. Argus, T. Welschehold, T. Brox, and A. Valada. Learning robotic manipulation policies from point clouds with conditional flow matching. *arXiv preprint arXiv:2409.07343*, 2024.

[5] O. X.-E. Collaboration, A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, A. Raffin, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Ichter, C. Lu, C. Xu, C. Finn, C. Xu, C. Chi, C. Huang, C. Chan, C. Pan, C. Fu, C. Devin, D. Driess, D. Pathak, D. Shah, D. Büchler, D. Kalashnikov, D. Sadigh, E. Johns, F. Ceola, F. Xia, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Schiavi, G. Kahn, H. Su, H.-S. Fang, H. Shi, H. B. Amor, H. I. Christensen, H. Furuta, H. Walke, H. Fang, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Peters, J. Schneider, J. Hsu, J. Bohg, J. Bingham, J. Wu, J. Wu, J. Luo, J. Gu, J. Tan, J. Oh, J. Malik, J. Tompson, J. Yang, J. J. Lim, J. Silvério, J. Han, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Zhang, K. Rana, K. Srinivasan, L. Y. Chen, L. Pinto, L. Tan, L. Ott, L. Lee, M. Tomizuka, M. Du, M. Ahn, M. Zhang, M. Ding, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Di Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, P. R. Sanketi, P. Wohlhart, P. Xu, P. Sermanet, P. Sundaresan, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Martín-Martín, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Song, S. Xu, S. Haldar, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Dasari, S. Belkhale, T. Osa, T. Harada, T. Matsushima, T. Xiao, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, V. Jain, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Wang, X. Zhu, X. Li, Y. Lu, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Xu, Y. Wang, Y. Bisk, Y. Cho, Y. Lee, Y. Cui, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Xu, and Z. J. Cui. Open X-Embodiment: Robotic Learning Datasets and RT-X Models, Oct. 2023.

[6] S. Deng, M. Yan, S. Wei, H. Ma, Y. Yang, J. Chen, Z. Zhang, T. Yang, X. Zhang, H. Cui, et al. Graspvla: a grasping foundation model pre-trained on billion-scale synthetic action data. *arXiv preprint arXiv:2505.03233*, 2025.

[7] F. Gloeckle, B. Y. Idrissi, B. Rozière, D. Lopez-Paz, and G. Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024.

[8] A. Goyal, V. Blukis, J. Xu, Y. Guo, Y.-W. Chao, and D. Fox. RVT-2: Learning Precise Manipulation from Few Demonstrations, June 2024.

[9] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox. RVT: Robotic View Transformer for 3D Object Manipulation, June 2023.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] J. Ho, A. Jain, and P. Abbeel. Denoising Diffusion Probabilistic Models, Dec. 2020.

[12] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[13] S. James and P. Abbeel. Coarse-to-fine q-attention with learned path ranking. *arXiv preprint arXiv:2204.01571*, 2022.

[14] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5:3019–3026, 2019.

[15] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.

[16] J. Kaplan, S. McCandlish, T. Henighan, et al. Scaling laws for neural language models. *arXiv:2001.08361*, 2020.

[17] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.

[18] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083, 2018.

[19] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black,

C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

[20] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[21] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. Openvla: An open-source vision-language-action model. In *Conference on Robot Learning*, volume 270, pages 2679–2713. PMLR, 2024.

[22] M. Laskey, J. Lee, R. Fox, A. D. Dragan, and K. Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on Robot Learning*, 2017.

[23] A. Liang, P. Czempin, M. Hong, Y. Zhou, E. Biyik, and S. Tu. Clam: Continuous latent action models for robot learning from unlabeled demonstrations. *arXiv preprint arXiv:2505.04999*, 2025.

[24] J. Liu, M. Liu, Z. Wang, P. An, X. Li, K. Zhou, S. Yang, R. Zhang, Y. Guo, and S. Zhang. Robomamba: Efficient vision-language-action model for robotic reasoning and manipulation. *Advances in Neural Information Processing Systems*, 37:40085–40110, 2024.

[25] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

[26] X. Ma, S. Patidar, I. Haughton, and S. James. Hierarchical diffusion policy for kinematics-aware multi-task robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18081–18090, 2024.

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

[28] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. `https://octo-models.github.io`, 2023.

[29] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

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

[31] Z. Sheebaelhamd, M. Tschannen, M. Muehlebach, and C. Vernade. Quantization-free autoregressive action transformer. *arXiv preprint arXiv:2503.14259*, 2025.

[32] L. X. Shi, A. Sharma, T. Z. Zhao, and C. Finn. Waypoint-based imitation learning for robotic manipulation. *arXiv preprint arXiv:2307.14326*, 2023.

[33] M. Shridhar, Y. L. Lo, and S. James. Generative Image as Action Models.

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

[35] K. Tian, Y. Jiang, Z. Yuan, B. Peng, and L. Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *arXiv:2404.02905*, 2024.

[36] A. Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[37] H. Walke, K. Black, A. Lee, M. J. Kim, M. Du, C. Zheng, T. Zhao, P. Hansen-Estruch, Q. Vuong, A. He, V. Myers, K. Fang, C. Finn, and S. Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023.

[38] C. Wen, X. Lin, J. So, K. Chen, Q. Dou, Y. Gao, and P. Abbeel. Any-point trajectory modeling for policy learning. *arXiv preprint arXiv:2401.00025*, 2023.

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

[40] H. Xue, J. Ren, W. Chen, G. Zhang, Y. Fang, G. Gu, H. Xu, and C. Lu. Reactive diffusion policy: Slow-fast visual-tactile policy learning for contact-rich manipulation. In *RSS*, 2025.

[41] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *Robotics: Science and Systems*, 2024.

[42] W. Zhang, Y. Li, Y. Qiao, S. Huang, J. Liu, F. Dayoub, X. Ma, and L. Liu. Effective tuning strategies for generalist robot manipulation policies. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7255–7262. IEEE, 2025.

[43] X. Zhang, Y. Liu, H. Chang, L. Schramm, and A. Boularias. Autoregressive action sequence learning for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025.

[44] Q. Zhao, Y. Lu, M. J. Kim, Z. Fu, Z. Zhang, Y. Wu, Z. Li, Q. Ma, S. Han, C. Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. *arXiv preprint arXiv:2503.22020*, 2025.

[45] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023.
[46] R. Zheng, Y. Liang, S. Huang, J. Gao, H. Daumé III, A. Kolobov, F. Huang, and J. Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*, 2024.

A Comparison on RLBench-18

The RLBench-18 subset was originally introduced by PerAct [34] and later became the standard comparison benchmark for 3D hierarchical methods. The results are detailed in Table 2. To clarify the fundamental difference between these two categories of approaches, Table 4 summarizes their methodological distinctions. 3D-based hierarchical methods typically rely on 3D perception and motion planning, whereas image-based visuomotor policies operate directly on raw RGB observations and learn end-to-end trajectory generation without explicit planners. We observe that, for RGB-only policies, several tasks in RLBench-18 are challenging and frequently result in zero success rates, which limits their discriminative power. This observation motivates our use of the proposed RLBench-60 evaluation split.

Table 4: Key differences between 3D-based hierarchical methods and image-based visuomotor policies.

Aspect	**3D-based Hierarchical Methods**	**Image-based Visuomotor Policies**
Typical Methods	PerAct, RVT, RVT-2, 3D Diffuser Actor	ACT, Diffusion Policy, CoA
Input Modality	3D point cloud / RGB-D	RGB-only
Pipeline	Two-stage: keyframe action detection + motion planning	End-to-end trajectory prediction without explicit planning
Execution Mode	Open-loop execution between keyframes	Closed-loop prediction and control

B Per-task success rates on RLBench-60

To complement the summary figure in the main paper, which visualizes the performance gap between CoA and baseline methods, we provide the full success rates on all 60 RLBench tasks in Table 5. This table lists the per-task success rate of CoA, ACT, and DP, along with the gap of baselines over CoA. Tasks are ordered by the maximum improvement CoA achieves over either baseline, highlighting where our method provides the most substantial gains.

Table 5: Detailed results of the overall comparison on RLBench. The simplified names used in Figure 4 are matched with their corresponding original task names. The success gap between ACT, DP and CoA is shown as superscripts.

Simplified name	Original name	CoA	ACT	DP
pick up cup	`pick_up_cup`	0.80	$0.20^{-0.60}$	$0.00^{-0.80}$
phone on base	`phone_on_base`	0.80	$0.04^{-0.76}$	$0.04^{-0.76}$
reach target	`reach_target`	0.84	$0.88^{+0.04}$	$0.08^{-0.76}$
remove meat	`meat_off_grill`	0.88	$0.32^{-0.56}$	$0.16^{-0.72}$
push button	`push_button`	0.76	$0.08^{-0.68}$	$0.12^{-0.64}$
put money in safe	`put_money_in_safe`	0.80	$0.36^{-0.44}$	$0.24^{-0.56}$
move hanger	`move_hanger`	0.88	$0.68^{-0.20}$	$0.32^{-0.56}$
slide block	`slide_block_to_target`	0.52	$0.32^{-0.20}$	$0.00^{-0.52}$
remove toilet roll	`take_toilet_roll_off_stand`	0.56	$0.40^{-0.16}$	$0.08^{-0.48}$
lamp off	`lamp_off`	0.68	$0.68^{-0.00}$	$0.20^{-0.48}$
lamp on	`lamp_on`	0.48	$0.44^{-0.04}$	$0.00^{-0.48}$
open door	`open_door`	0.92	$0.44^{-0.48}$	$0.60^{-0.32}$
open drawer	`open_drawer`	0.88	$0.52^{-0.36}$	$0.44^{-0.44}$
remove frame	`take_frame_off_hanger`	0.64	$0.44^{-0.20}$	$0.24^{-0.40}$

Continued on next page

Simplified name	Original name	CoA	ACT	DP
open washer	open_washing_machine	0.76	$0.44^{-0.32}$	$0.60^{-0.16}$
remove pan lid	take_lid_off_saucepan	0.80	$0.40^{-0.40}$	$0.60^{-0.20}$
unplug charger	unplug_charger	0.60	$0.56^{-0.04}$	$0.20^{-0.40}$
buzz game	beat_the_buzz	0.36	$0.12^{-0.24}$	$0.00^{-0.36}$
remove umbrella	take_umbrella_out_of_umbrella_stand	0.52	$0.16^{-0.36}$	$0.20^{-0.32}$
drag to target	reach_and_drag	0.64	$0.36^{-0.28}$	$0.28^{-0.36}$
get ice	get_ice_from_fridge	0.60	$0.32^{-0.28}$	$0.24^{-0.36}$
open box	open_box	0.32	$0.16^{-0.16}$	$0.32^{-0.00}$
place knife	place_knife_on_chopping_board	0.04	$0.04^{-0.00}$	$0.00^{-0.04}$
play jenga	play_jenga	1.00	$1.00^{-0.00}$	$0.72^{-0.28}$
place plate	put_plate_in_colored_dish_rack	0.32	$0.12^{-0.20}$	$0.04^{-0.28}$
put bottle in fridge	put_bottle_in_fridge	0.28	$0.00^{-0.28}$	$0.00^{-0.28}$
remove plate	take_plate_off_colored_dish_rack	0.40	$0.40^{-0.00}$	$0.12^{-0.28}$
turn tap	turn_tap	0.56	$0.36^{-0.20}$	$0.32^{-0.24}$
remove from scale	take_off_weighing_scales	0.84	$0.44^{-0.40}$	$0.64^{-0.20}$
stack wine	stack_wine	0.80	$0.56^{-0.24}$	$0.56^{-0.24}$
close drawer	close_drawer	1.00	$0.96^{-0.04}$	$0.76^{-0.24}$
close box	close_box	1.00	$0.96^{-0.04}$	$0.76^{-0.24}$
set clock	change_clock	0.40	$0.28^{-0.12}$	$0.20^{-0.20}$
hang frame	hang_frame_on_wall	0.16	$0.08^{-0.08}$	$0.00^{-0.16}$
open microwave	open_microwave	0.44	$0.40^{-0.04}$	$0.40^{-0.04}$
close fridge	close_fridge	0.92	$0.84^{-0.08}$	$0.76^{-0.16}$
remove USB	take_usb_out_of_computer	0.60	$0.48^{-0.12}$	$0.72^{+0.12}$
change channel	change_channel	0.12	$0.00^{-0.12}$	$0.00^{-0.12}$
insert USB	insert_usb_in_computer	0.92	$0.80^{-0.12}$	$0.88^{-0.04}$
seat down	toilet_seat_down	1.00	$0.96^{-0.04}$	$0.88^{-0.12}$
close grill	close_grill	0.56	$0.48^{-0.08}$	$0.68^{+0.12}$
lift block	lift_numbered_block	0.08	$0.00^{-0.08}$	$0.08^{-0.00}$
seat up	toilet_seat_up	0.84	$0.76^{-0.08}$	$0.88^{+0.04}$
take out shoes	take_shoes_out_of_box	0.08	$0.00^{-0.08}$	$0.16^{+0.08}$
take out money	take_money_out_safe	0.76	$0.80^{+0.04}$	$0.68^{-0.08}$
screw nail	screw_nail	0.08	$0.12^{+0.04}$	$0.00^{-0.08}$
water plants	water_plants	0.48	$0.40^{-0.08}$	$0.56^{+0.08}$
hockey	hockey	0.08	$0.04^{-0.04}$	$0.00^{-0.08}$
open wine	open_wine_bottle	0.36	$0.28^{-0.08}$	$0.40^{+0.04}$
hit ball	hit_ball_with_cue	0.08	$0.00^{-0.08}$	$0.00^{-0.08}$
put groceries	put_groceries_in_cupboard	0.08	$0.04^{-0.04}$	$0.00^{-0.08}$
turn on oven	turn_oven_on	0.36	$0.32^{-0.04}$	$0.28^{-0.08}$
set checkers	setup_checkers	0.04	$0.00^{-0.04}$	$0.04^{-0.00}$
basketball	basketball_in_hoop	0.76	$0.72^{-0.04}$	$0.72^{-0.04}$
hang hanger	place_hanger_on_rack	0.32	$0.04^{-0.28}$	$0.00^{-0.32}$
open grill	open_grill	0.24	$0.00^{-0.24}$	$0.00^{-0.24}$
straighten rope	straighten_rope	0.00	$0.04^{+0.04}$	$0.00^{-0.00}$
sweep dust	sweep_to_dustpan	0.92	$1.00^{+0.08}$	$1.00^{+0.08}$
press switch	press_switch	0.44	$0.52^{+0.08}$	$0.56^{+0.12}$
close microwave	close_microwave	0.72	$0.80^{+0.08}$	$0.80^{+0.08}$

C Supplementary real-world results

Table 6 reports the per-task success rates of CoA, ACT, and DP across 8 real-world kitchen manipulation tasks. CoA consistently achieves the highest average performance.

Table 6: Per-task success rate in real-world experiments.

Task	**CoA**	**ACT**	**DP**
close cabinet	0.80	1.00	0.90
close fridge lower	0.20	0.50	0.60
close fridge upper	0.70	0.40	0.80
open cabinet	0.50	0.40	0.10
open fridge lower	0.70	0.40	0.00
open microwave	0.80	0.30	0.50
place in cabinet	0.70	0.10	0.00
place in fridge	0.50	0.60	0.00
Avg.	**0.613**	0.463	0.363

D Hyperparameters for RLBench

We provide the training and evaluation hyperparameters for CoA and all baseline methods used in the simulation experiments. To ensure a fair comparison, the hyperparameters for ACT are largely aligned with those of CoA, allowing us to isolate and assess the impact of our proposed modeling paradigm. For DP, we observe slower convergence relative to CoA and ACT, and thus extend its training duration to 100,000 iterations. In addition, we incorporate temporal ensembling into DP following the implementation in ACT. Octo converges substantially faster, and we find that 2,000 training iterations are sufficient. Given that Octo is primarily pretrained on single-camera data, we finetune it using only the front camera, while increasing the image resolution to enhance visual fidelity. All models are trained on a single NVIDIA H100 GPU per task.

Table 7: Hyperparameters for CoA

Backbone	ImageNet-trained ResNet18 [10]
Action dimension	8 (3 position + 4 quaternion + 1 gripper)
Cameras	wrist, front, right shoulder, left shoulder
Learning rate	$1e^{-4}$
Weight decay	$1e^{-4}$
Image size	128×128
Execution horizon	1
Observation horizon	1
# encoder layers	4
# decoder layers	7 (6 + 1 multi-token prediction layer)
# heads	8
Feedforward dimension	3200
Hidden dimension	512
Dropout	0.1
Iteration	20000
Batch size	128
Temporal ensembling	true (reverse temporal ensemble)
Action normalization	$[-1, 1]$

Table 8: Hyperparameters for ACT

Backbone	ImageNet-trained ResNet18 [10]
Action dimension	8 (3 position + 4 quaternion + 1 gripper)
Cameras	wrist, front, right shoulder, left shoulder
Learning rate	$1e^{-4}$
Weight decay	$1e^{-4}$
Image size	128×128
Action sequence	20
Execution horizon	1
Observation horizon	1
# encoder layers	4
# decoder layers	7
# heads	8
Feedforward dimension	3200
Hidden dimension	512
Dropout	0.1
Iteration	20000
Batch size	128
Temporal ensembling	true
Action normalization	$[-1, 1]$

Table 9: Hyperparameters for DP

Backbone	ImageNet-trained ResNet18 [10]
Noise predictor	UNet [29]
Action dimension	8 (3 position + 4 quaternion + 1 gripper)
Cameras	wrist, front, right shoulder, left shoulder
Learning rate	$1e^{-4}$
Weight decay	$1e^{-6}$
Image size	128×128
Observation horizon	1
Action sequence	20
Execution horizon	1
Train, test diffusion steps	50, 50
Hidden dimension	512
Iteration	100000
Batch size	128
Temporal ensembling	true (following ACT's)
Scheduler	DDPM [12]
Action normalization	[-1, 1]

Table 10: Hyperparameters for Octo

Pretrained model	Octo-small [28]
Action dimension	8 (7 delta joints + 1 gripper)
Cameras	front
Learning rate	$3e^{-4}$
Weight decay	$1e^{-2}$
Image size	256×256
Observation horizon	1
Action sequence	4
Execution horizon	1
Iteration	2000
Batch size	128
Temporal ensembling	false
Action normalization	mean 0, std 1
Finetuning head	linear head
Image augmentation	`resized crop`, `brightness`, `contrast`, `saturation`, `hue`

E ACT variant with keyframe action

To further examine the impact of keyframe action on action sequence modeling, we conduct an additional ablation by modifying the ACT baseline. Specifically, we introduce a variant, ACT+KF, in which an extra keyframe action is appended to ACT's original action chunk.

As shown in Table 11, ACT+KF achieves a higher average success rate (0.516) compared to the original ACT (0.488), indicating that injecting keyframe actions yields marginal improvements. However, the overall gain remains limited.

This result suggests that while keyframe actions may provide some global guidance, they do not substantially improve the final action quality when introduced in this manner. A similar trend is observed in the poor performance of *Hybrid* (Table 3), a variant of CoA that incorporates both keyframe supervision and causal decoding but lacks trajectory continuity. The limited effectiveness of both ACT+KF and Hybrid underscores a key insight: merely injecting keyframe signals and enforcing an autoregressive structure is not sufficient. Instead, it is crucial to model the entire trajectory holistically with temporal continuity, whch is explicitly realized in our CoA formulation.

Table 11: Comparison of ACT vs. ACT+KF (with keyframe action) on 10 RLBench tasks.

Task	**ACT**	**ACT+KF**
Stack Wine	0.56	0.56
Turn Tap	0.36	0.32
Open Drawer	0.52	0.76
Push Button	0.08	0.16
Pick Up Cup	0.20	0.36
Take Lid	0.40	0.40
Press Switch	0.52	0.28
Reach Target	0.88	0.72
Sweep Dust	1.00	0.96
Open Box	0.36	0.64
Avg.	0.488	**0.516**

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the main contributions, including the trajectory autoregressive modeling and its key components. These claims are supported by experiments across 60 RLBench tasks and real-world tests.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses key limitations, including the reliance on hand-crafted keyframe heuristics for trajectory segmentation. This limitation is acknowledged in the conclusion section and help clarify the scope of the proposed approach.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [No]

Justification: Our work introduces a autoregressive modeling framework for visuomotor policies, with contributions that are primarily architectural and algorithmic. As the focus is on empirical performance and system-level design rather than formal theoretical development, no formal theorems or proofs are provided.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides detailed descriptions of the model architecture, training setting, and evaluation protocols across all experiments, including both simulation and real-robot settings.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The code will be open-sourced upon publication. The simulation data used in our experiments is based on the publicly available RLBench benchmark and can be generated by its official code base.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All training and evaluation details are provided in main paper, except for hyperparameters, which are reported in the appendix. We also include the hyperparameter settings used for all baseline methods to ensure fair comparison.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: We report success rates averaged over 25 evaluation trials per task for all 60 RLBench tasks, but we do not provide error bars or other statistical significance metrics. We acknowledge this limitation and plan to include such measures in future versions to strengthen the statistical interpretation of our results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide details of the compute resources used in appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research does not involve human subjects, personally identifiable data, or other ethically sensitive content. All experiments were conducted in simulation or on standard robotic platforms, and the work complies with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The paper does not include an explicit discussion of societal impacts. Our work focuses on improving robotic manipulation through a novel trajectory generation method and is intended primarily for academic and industrial automation research. We do not foresee direct negative societal impacts such as disinformation, surveillance, or fairness concerns, as the method is task-agnostic and does not involve human subjects or personal data.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work does not involve the release of models or datasets with a high risk for misuse. It focuses on visuomotor policy learning for robotic manipulation.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use the RLBench dataset, which is publicly available. We also build upon the ACT, Diffusion Policy, Octo codebases, both of which are properly cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release any new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve any crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The research does not involve human subjects and thus does not require IRB or equivalent approval.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core methodology does not involve large language models as any important, original, or non-standard component.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.