OAT: Ordered Action Tokenization

Chaoqi Liu[1] Xiaoshen Han[1] Jiawei Gao[1] Yue Zhao[2] Haonan Chen[1] Yilun Du[1]

[1]Harvard University [2]Stanford University

ordered-action-tokenization.github.io

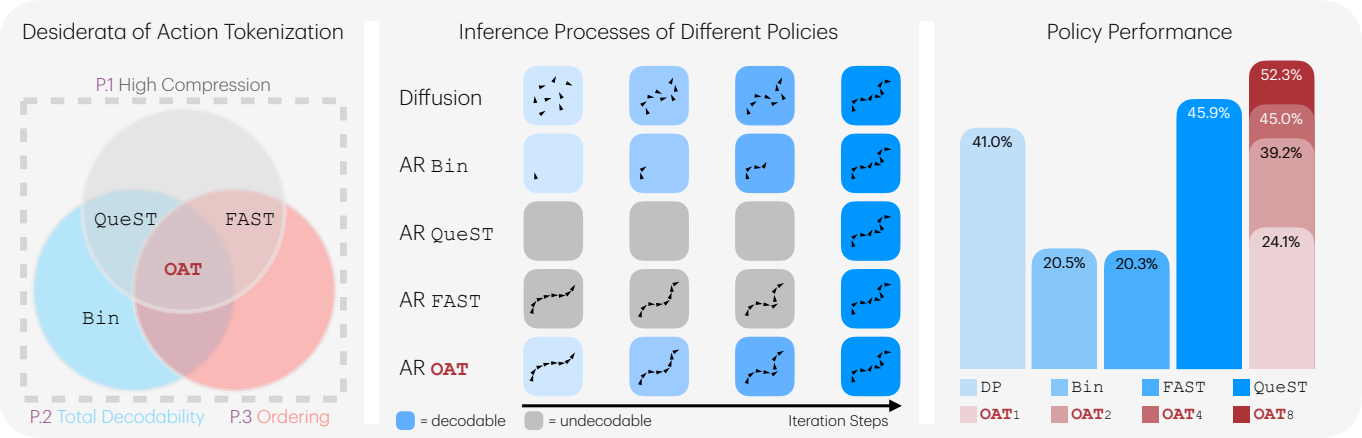


Fig. 1: **Left:** Comparison of action tokenization schemes with respect to three desiderata: high compression (P.1), total decodability (P.2), and left-to-right causally ordered token structure (P.3). Existing methods satisfy only subsets of these properties, while **OAT** uniquely satisfies all three. **Middle:** Behavior of different policy classes as inference progresses. Diffusion and flow policies refine actions through iterative sampling, while autoregressive policies generate discrete tokens step-by-step. Due to its ordered token space, **OAT** enables prefix-based detokenization: early tokens produce coarse action chunks, and additional autoregressive steps progressively refine actions, enabling flexible, anytime action generation. **Right:** Overall policy performance aggregated over 20+ tasks.

***Abstract*—Autoregressive policies offer a compelling foundation for scalable robot learning by enabling discrete abstraction, token-level reasoning, and flexible inference. However, applying autoregressive modeling to continuous robot actions requires an effective action tokenization scheme. Existing approaches either rely on analytical discretization methods that produce prohibitively long token sequences or learned latent tokenizers that lack structure, limiting their compatibility with next-token prediction. In this work, we identify three desiderata for action tokenization — high compression, total decodability, and a left-to-right causally ordered token space — and introduce *Ordered Action Tokenization* (OAT), a learned action tokenizer that satisfies all three. OAT discretizes action chunks into an ordered sequence of tokens using a transformer with registers, finite scalar quantization, and ordering-inducing training mechanisms. The resulting token space aligns naturally with autoregressive generation and enables prefix-based detokenization, yielding an anytime trade-off between inference cost and action fidelity. Across more than 20 tasks spanning four simulation benchmarks and real-world settings, autoregressive policies equipped with OAT consistently outperform prior tokenization schemes and diffusion-based baselines, while offering significantly greater flexibility at inference time.**

I. Introduction

Autoregressive sequence models have emerged as a powerful foundation for modern robot learning. In particular, large transformer-based policies have demonstrated strong generalization when trained directly on robotic data [7, 48] or adapted from pre-trained vision-language backbones [8, 28, 63]. A critical but often under-examined component underlying these successes is how continuous robot actions are represented as discrete symbols suitable for autoregressive generation.

This representation problem is known as *action tokenization*: the process of mapping continuous control signals into a sequence of discrete tokens. Experience from natural language processing and computer vision has shown that tokenization is far more than an implementation detail — it fundamentally shapes learning dynamics, model capacity utilization, scalability, and downstream performance [3, 17, 54, 66]. Despite its centrality, action tokenization for robot control remains significantly less understood than its counterparts in language and vision.

The dominant approach in existing autoregressive robot policies relies on naive discretization via per-dimension binning [7, 8, 28]. While conceptually simple, this strategy yields extremely long token sequences whose lengths scale linearly with action dimensionality and prediction horizon, leading to substantial inefficiencies in training and inference. To alleviate this issue, recent work has explored learned latent tokenizers [4, 32, 45] and analytical compression methods such as frequency-domain compression [50]. However, these alternatives introduce their own limitations: learned tokenizers often produce unstructured latent spaces that are poorly aligned with next-token prediction, while existing frequency-domain approaches may sacrifice decodability. Across these approaches, we identify a persistent and fundamental limitation: existing action tokenization strategies face an inherent

trade-off between compression rate, modelability[1] under autoregressive learning, and decodability. Improving one aspect typically degrades another, resulting in token spaces that are either too long to model efficiently, insufficiently structured for stable generation, or partially decodable at inference time.

In this work, we argue that an effective action tokenizer for autoregressive policies must simultaneously satisfy three key properties (Fig. 1 left): **(P.1) High Compression**, reducing the effective prediction horizon to enable efficient long-context modeling; **(P.2) Total Decodability**, meaning the decoder is a total function in which every token sequence maps to a valid action chunk, with no undefined or invalid outputs; and **(P.3) Causal Ordering**, imposing a left-to-right structure over tokens that aligns with the inductive bias of next-token prediction. While prior methods satisfy subsets of these desiderata, none achieve all three simultaneously.

To bridge this gap, we introduce *Ordered Action Tokenization* (**OAT**), a learned action tokenizer that discretizes continuous action chunks into highly compressed and *causally ordered* token sequences. **OAT** employs transformer-based register tokens to aggregate temporal information, finite scalar quantization (FSQ) to construct a discrete bottleneck, and nested dropout to explicitly induce ordering that aligns the latent space with autoregressive generation. The resulting tokenization ensures that any token prefix corresponds to a plausible action chunk. Beyond improved modelability, the ordered structure learned by **OAT** enables a key capability absent from prior approaches: *prefix-based decoding*. Autoregressive policies may terminate generation early and still produce valid actions, yielding a natural trade-off between computation and action fidelity. As additional tokens are generated, decoded actions are progressively refined.

In summary, this paper makes three contributions: **(i)** we formalize a set of necessary desiderata for action tokenization in autoregressive robot policies, exposing a fundamental trade-off faced by existing methods; **(ii)** we propose **OAT**, a novel tokenizer that uniquely satisfies compression, total decodability, and causal ordering simultaneously; and **(iii)** we demonstrate that *ordering* is the critical ingredient for stable and scalable autoregressive learning, enabling superior performance and flexible, prefix-based decoding across 20+ simulation and real-world manipulation tasks.

II. Related Work on Generative Policies

We focus on policies of the form $\pi(a_{1:H_a} \mid o_{1:H_o})$ that predict a *chunk* of actions conditioned on a history of observations. Predicting multi-step action sequences has been shown to improve temporal consistency, reduce compounding error, and stabilize long-horizon behavior compared to single-step prediction [13, 71, 74]. Action chunking also amortizes inference cost over multiple time steps and has become a standard design choice in modern robot policies.

Diffusion and flow-based policies [10–13, 20, 22, 24, 25, 39, 41, 52, 57, 64, 65, 68, 72] have proven highly effective for continuous action generation and imitation learning, and are widely used as standalone robot policies. More recently, in VLA systems, diffusion and flow models are increasingly employed as *action experts* or continuous decoding heads that translate higher-level representations into executable actions [5, 24, 40, 47, 63]. In this role, they complement discrete reasoning and planning components by providing expressive, high-fidelity action synthesis.

Autoregressive policies model the distribution of action sequences by factorizing it into a product of conditional distributions, generating one element at a time [61]. Autoregressive models have demonstrated remarkable scalability and generalization in language, image, and video modeling [35, 51, 58, 62, 67]. This success has motivated their adoption in robotics, particularly within VLA systems [7, 8, 21, 28, 47–50, 63].

Despite their success, the effectiveness of autoregressive policies in robotics depends critically on the choice of tokenization. In this work, we systematically study the key desiderata of action tokenization for autoregressive policies and propose a principled tokenizer that addresses these requirements. We formalize these properties and introduce **OAT** in the following sections.

III. Action Tokenization Preliminaries

Robot actions, however, are inherently continuous and high-dimensional. To enable autoregressive modeling, continuous action chunks must first be discretized into a sequence of tokens. This process, referred to as *action tokenization*, defines a mapping

$$\mathcal{T} : a_{1:H_a} \;\mapsto\; T_{1:H_l},$$

which maps a continuous action chunk of horizon H_a and dimensionality D_a to a sequence of H_l discrete tokens drawn from a vocabulary $\mathcal{V}$. A corresponding *detokenization* mapping

$$\mathcal{T}^{-1} : T_{1:H_l} \;\mapsto\; a_{1:H_a}$$

maps token sequences back into continuous action space, producing executable action chunks. Autoregressive policies operate entirely in the discrete token space defined by $\mathcal{T}$, while control execution relies on $\mathcal{T}^{-1}$ to convert generated token sequences into continuous actions.

We argue that an efficient and effective action tokenizer, that balances *rate-distortion-modelability trade-off* [6, 16, 55, 59, 75], should satisfy the following three properties:

P.1 $\mathcal{T}$ **achieves a high compression rate.**
P.2 $\mathcal{T}^{-1}$ **is a well-defined total function.**
P.3 $T_{1:H_l}$ **has a left-to-right causal ordering.**

P.1: The token horizon H_l should be sufficiently small to enable efficient autoregressive modeling, while retaining enough capacity to preserve necessary information from the original action chunk. P.2: The decoder $\mathcal{T}^{-1}$ must be a well-defined *total function*: for every token sequence $T_{1:H_l}$ in the discrete token space, $\mathcal{T}^{-1}(T_{1:H_l})$ produces a valid action chunk $a_{1:H_a}$. This property is critical in autoregressive settings, where policies may generate arbitrary token sequences at inference

[1] *Modelability* characterizes how challenging it is for generative models to capture the distribution of the representation [16, 27, 29].

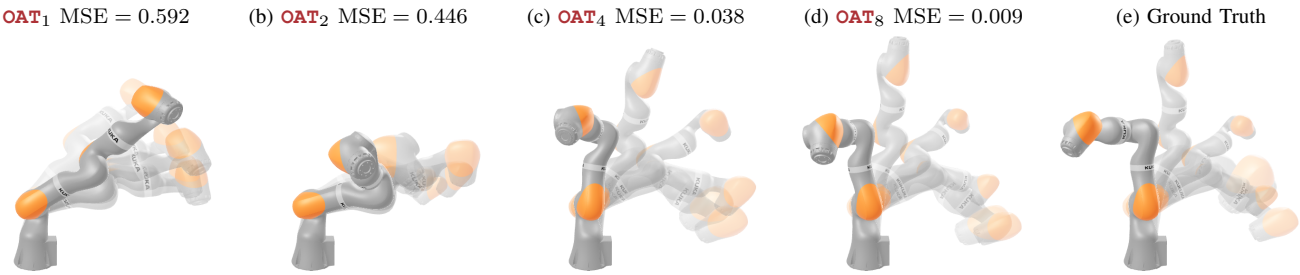


Fig. 2: **Coarse-to-fine action chunk reconstruction.** Visualization of reconstructed action chunks using increasing numbers of decoded tokens. Panels (a–d) show **OAT** reconstructions using $K \in \{1, 2, 4, 8\}$ tokens, respectively, while (e) shows the ground-truth action chunk. Earlier tokens capture the coarse, global structure of the motion, while additional tokens progressively refine fine-grained details, yielding trajectories that increasingly match the ground truth. Ghosted poses indicate temporal progression within each reconstructed action chunk. Interactive visualization on project website: ordered-action-tokenization.github.io.

time. If $\mathcal{T}^{-1}$ is only partially defined, invalid or non-decodable token sequences can lead to undefined behavior and catastrophic failures during execution. P.3: The token sequence $T_{1:H_l}$ should admit a meaningful left-to-right causal ordering aligned with causal, next-token prediction. Such a structure is essential for stable autoregressive generation: early tokens should capture coarse, globally salient aspects of the action chunk, while later tokens refine finer details. An ordered token space improves controllability, robustness, and compatibility with prefix-based generation, and we revisit this property throughout the paper both conceptually and empirically.

A. Binning

The most commonly used action tokenization approach is per-dimension binning (`Bin`) [7, 8, 28]. For each action dimension, the range of values observed in the dataset is normalized to $[-1, 1]$, then divided into N uniform bins, and each continuous value is mapped to its corresponding bin index. Given an action chunk of shape $H_a \times D_a$, binning produces a token sequence

$$\mathcal{T}(a_{1:H_a}) = [T_{1,1}, ..., T_{1,D_a}, T_{2,1}, ..., T_{H_a,D_a}], \quad T_{i,j} \in [N].$$

While `Bin` is conceptually simple and yields a well-defined, totally decodable mapping (P.2), it does not provide the left-to-right ordering we seek (P.3): the token order is a serialization over dimensions and time rather than a hierarchy aligned with causal next-token prediction. Moreover, `Bin` scales poorly — long horizons and high-dimensional actions can produce hundreds of tokens per chunk — severely slowing training and inference and introducing substantial latency. Therefore, `Bin` fails to satisfy P.1 and P.3, despite meeting P.2.

B. Frequency-domain Transform

An alternative line of work explores frequency-domain compression, for instance *Frequency-space Action Sequence Tokenization* (`FAST`) [50], which employs the Discrete Cosine Transform (DCT) to decompose action chunks into frequency coefficients, followed by Byte Pair Encoding (BPE) [18]. `FAST` achieves high information density (P.1), and crucially, its low-frequency components first then high-frequency components ordering (P.3) improves downstream autoregressive policies: early token predictions capture the overall trajectory shape, stabilizing rollout before finer details are generated.

However, `FAST` detokenization $\mathcal{T}^{-1}$ is a partial function that violates P.2. Because BPE produces variable-length sequences, there is no guarantee that an arbitrary token sequence generated by the policy will decode into a valid frequency matrix of fixed dimensions. This structural mismatch renders the decoding function partially undefined for invalid token counts, leading to potential runtime failures. We refer readers to Appendix B and the discussion on Hugging Face[2] for further details.

C. Quantized Latents

Another line of work explores learned compression via encoder-decoder architectures with vector quantization [4, 32, 45]. These methods map action chunks into a latent sequence of shape $H_l \times D_l$, which is quantized [44, 60] into tokens. The latent horizon H_l and dimension D_l are hyperparameters, often chosen relative to H_a and D_a. Such approaches can achieve extremely high compression; for example, mapping action chunks of horizon $H_a = 32$ into latent sequences with $H_l = 8$ tokens, satisfying P.1. Because $\mathcal{T}$ and $\mathcal{T}^{-1}$ are approximated by encoder and decoder neural networks respectively, $\mathcal{T}^{-1}$ are always total (P.2).

However, existing learned tokenizers typically produce unstructured token spaces. The tokens lack a consistent ordering or hierarchical abstraction, making them poorly suited for autoregressive generation. As a result, while existing learned tokenizers satisfy P.1 and P.2, they fail to meet P.3.

IV. OAT: Ordered Action Tokenization

Our objective is to learn an action tokenizer that satisfies three desiderata introduced in Sec. III: high compression (P.1), total decodability (P.2), and a structured ordering over tokens (P.3). While prior learned tokenizers achieve compact and decodable representations, they lack an explicit ordering over latent tokens [4, 32, 45], which limits their compatibility with autoregressive policies. We introduce **OAT**, a learned autoencoder framework that discretizes action chunks into an ordered sequence of tokens. **OAT** encodes actions using transformer-based register tokens, discretizes the resulting latents with

[2] https://huggingface.co/physical-intelligence/fast/discussions/4

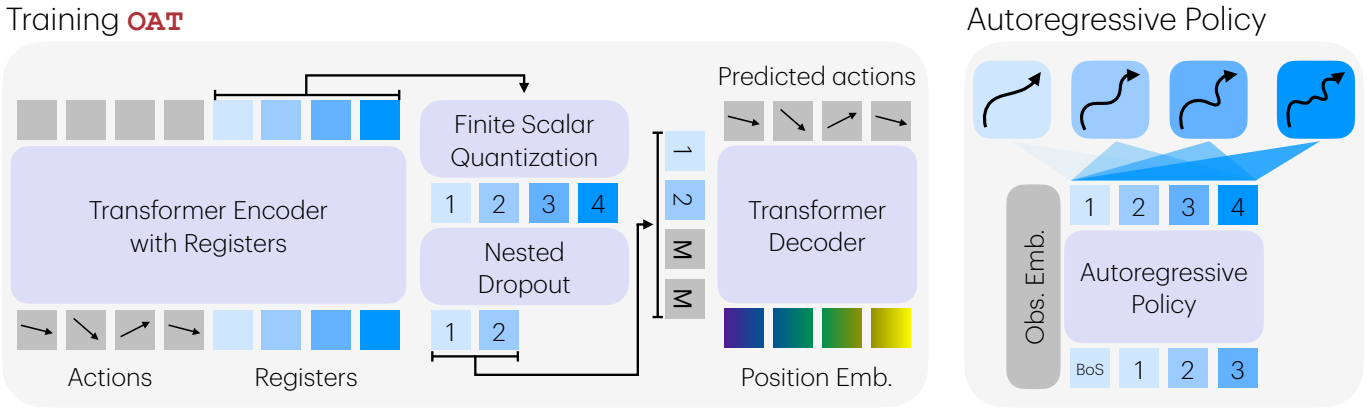


Fig. 3: **OAT overview. Left:** OAT maps a chunk of continuous actions into an ordered sequence of discrete tokens using a transformer encoder with register tokens, FSQ, and nested dropout to induce token ordering. The resulting tokens form a compact action representation, which is decoded to reconstruct action chunks for downstream autoregressive policies. **Right:** During OAT policy inference, tokens are generated autoregressively and can be detokenized from any prefix. As more autoregressive steps are taken, additional tokens progressively refine the decoded action chunk, producing actions with increasing temporal and spatial detail. OAT enables flexible, anytime action generation.

Algorithm 1 OAT Tokenizer Training

Require: Dataset $\mathcal{D}$ of action chunks $\{a_{1:H_a}\}$; encoder $E_\phi(\cdot)$; learnable register tokens $r_{1:H_l}$; quantizer $\mathrm{FSQ}(\cdot)$; decoder $D_\theta(\cdot)$; learnable mask token $\texttt{MASK}$; nested-dropout distribution $p(\cdot)$.

while not converged **do**
  Sample action chunk $a_{1:H_a} \sim \mathcal{D}$
  Encoding: $\tilde{a}_{1:H_a} \oplus z_{1:H_l} \leftarrow E_\phi(a_{1:H_a} \oplus r_{1:H_l})$
  Quantization: $\hat{z}_{1:H_l} \leftarrow \mathrm{FSQ}(z_{1:H_l})$
  Tail dropout: $\hat{z}_{1:H_l} \leftarrow \hat{z}_{1:K} \oplus \langle\texttt{MASK}\rangle_{K+1:H_l}, K \sim p(\cdot)$
  Decoding: $\hat{a}_{1:H_a} \leftarrow D_\theta(\hat{z}_{1:H_l})$
  Reconstruction loss: $\mathcal{L} \leftarrow \|\hat{a}_{1:H_a} - a_{1:H_a}\|_2^2$
  Optimization: $\{\phi, r, \theta, \texttt{MASK}\} \leftarrow \{\phi, r, \theta, \texttt{MASK}\} - \eta\nabla\mathcal{L}$
end while
$\mathcal{T}(\cdot) \leftarrow \{E_\phi, r_{1:H_l}, \mathrm{FSQ}\}$, $\mathcal{T}^{-1}(\cdot) \leftarrow \{D_\theta, \texttt{MASK}\}$
return $\mathcal{T}(\cdot), \mathcal{T}^{-1}(\cdot)$

FSQ [44], and reconstructs actions via a conditional decoder. To induce ordering in the token space, we combine causal attention over register tokens with nested dropout during training. Together, these design choices encourage an ordered latent representation in which earlier tokens capture coarse, global structure and later tokens refine details (Fig. 2). As a result, OAT supports decoding from any prefix of the token sequence, enabling variable-length and anytime reconstruction of action chunks. We demonstrate OAT training pipeline (left) and autoregressive policies operate on OAT (right) in Fig. 3.

A. Tokenization $\mathcal{T}$ and Detokenization $\mathcal{T}^{-1}$

The objective of the tokenizer $\mathcal{T}$ is to compress a continuous action chunk of shape $H_a \times D_a$ into a compact discrete representation of shape $H_l \times D_l$. To this end, we concatenate the input action sequence with a fixed set of learnable *register tokens*, $r_{1:H_l}$, which act as a compact read–write memory for summarizing the temporal structure of the input [15, 69]. A transformer encoder jointly processes the action chunk and register tokens, allowing information from the action sequence to be aggregated into the registers. After encoding, the register tokens form the bottleneck representation [16] of the autoencoder, while the encoded action tokens are discarded.

The register latents $z_{1:H_l}$ are discretized using FSQ, yielding a sequence of H_l discrete tokens $T_{1:H_l}$. These tokens constitute the action representation used both for reconstruction during tokenizer training and as the action space for downstream autoregressive policies.

The decoder implements the detokenization mapping $\mathcal{T}^{-1}$, generating a continuous action chunk conditioned on the discrete token sequence. The OAT framework imposes no restrictions on the specific decoder architecture or training objective. In this work, we employ a single-pass transformer decoder similar to [73] (see Fig. 3), which we find provides a favorable trade-off between reconstruction quality, stability, and computational efficiency. We provide more details on decoding in Appendix C. The tokenizer $\mathcal{T}$ and detokenizer $\mathcal{T}^{-1}$ are trained jointly end-to-end using a reconstruction objective. Pseudocode for OAT training is provided in Algo. 1, also see Fig. 3 for the pipeline.

B. Inducing Token Ordering For Modelability

Prior work has highlighted the importance of left-to-right causal ordering for effective autoregressive modeling [23, 26, 27, 29]. To align the learned token space with autoregressive generation, we explicitly induce a left-to-right ordering over tokens $T_{1:H_l}$ that naturally aligns with next-token prediction. Our goal is to ensure that earlier tokens capture coarse, globally salient aspects of an action chunk, while later tokens refine finer details. We introduce two complementary mechanisms to impose an ordering and support variable-length token sequences.

1) Nested Dropout: We train OAT to produce an ordered representation by applying nested dropout to the register tokens during training [9, 30, 53]. Given register tokens of length H_l, we randomly sample the number of tokens to retain, $K \in [H_l]$, and mask out the remaining $H_l - K$ tail tokens. Under this training regime, the encoder is encouraged to pack information into the register tokens in a prioritized,

ordered manner, while the decoder learns to reconstruct action chunks from variably sized token prefixes. As a result, the first few tokens capture the most important aspects of the action sequence, while additional tokens progressively refine the reconstruction. Simple action chunks can therefore be faithfully represented with few tokens, whereas more complex behaviors require longer token sequences. Importantly, this ordering is not manually specified but emerges naturally from the nested dropout objective applied to the register tokens.

2) Causal Attention: Complementary to nested dropout, we impose a causal attention [61] structure over the register tokens to further reinforce ordering. Specifically, the encoded action tokens attend freely to one another but do not attend to registers. Each register token attends to all action tokens, enabling global aggregation, but register-register attention is causally masked such that the i-th register token only attends to the j-th register token if $i \geq j$. This causal dependency structure enforces a left-to-right information flow among registers, aligning the learned token sequence with autoregressive prediction and stabilizing generation from partial prefixes.

C. Information-Theoretic Interpretation of Token Ordering

The ordering induced by **OAT** admits a natural interpretation from information theory. Classical results by Shannon show that the optimal code length for representing an event scales with the negative logarithm of its probability, i.e., $-\log p$ [55]: frequent patterns require fewer bits to encode, while rare or atypical events demand greater representational capacity. In our setting, action chunks $a_{1:H_a}$ are drawn from a data distribution with highly non-uniform structure — most trajectories share common coarse patterns, while fine-grained deviations occur less frequently.

Under this lens, the ordered token sequence $T_{1:H_l}$ learned by **OAT** can be viewed as an implicit progressive coding of action information. Early tokens are encouraged to capture the dominant motion pattern shared across many trajectories. Later tokens then progressively correct residual errors and local details. This behavior emerges naturally from nested dropout: since prefixes must reconstruct actions under partial information, the tokenizer learns to allocate information in decreasing order of frequency and importance. This interpretation explains both the monotonic improvement in reconstruction quality with increasing prefix length and the strong alignment between token order and autoregressive next-token prediction. Importantly, the ordering is not imposed heuristically but arises from optimizing reconstruction under variable information budgets.

D. Autoregressive OAT Policies

We use **OAT** as the discrete action representation for autoregressive policy learning. Given an observation history $o_{1:H_o}$, the policy models a distribution over action tokens by factorizing

$$p(T_{1:H_l} \mid o_{1:H_o}) = \prod_{i=1}^{H_l} p(T_i \mid T_{<i}, o_{1:H_o}),$$

Algorithm 2 Autoregressive **OAT** Policy Inference

Require: Observation history $o_{1:H_o}$; autoregressive policy $\pi(\cdot)$; detokenizer $\mathcal{T}^{-1} = \{D(\cdot), \texttt{MASK}\}$; prefix length $K \leq H_l$.

```
Initialize empty token prefix T_{1:K} ← ∅
for i ← 1 to K do
    Next-token sampling: T_i ~ π( · | T_{<i}, o_{1:H_o} )
    T_{1:K} ← T_{1:K} ⊕ T_i
end for
Pad tail tokens: T_{1:H_l} ← T_{1:K} ⊕ ⟨MASK⟩_{K+1:H_l}
Detokenize to action chunk: â_{1:H_a} ← T^{-1}(T_{1:H_l})
return â_{1:H_a}
```

and generates tokens sequentially. The resulting token sequence is detokenized via $\mathcal{T}^{-1}$ to produce a continuous action chunk for execution.

The ordered token space (P.3) induced by **OAT** is essential for effective autoregressive modeling. Earlier tokens encode the coarse, global structure of the action chunk, while later tokens progressively refine finer details, aligning next-token prediction with the semantics of action generation. As a result, prefixes of the token sequence correspond to valid, increasingly detailed action chunks rather than arbitrary partial reconstructions.

Crucially, autoregressive generation need not proceed to completion. Because any prefix $T_{1:K}$ can be detokenized into a valid action chunk, **OAT** supports prefix-based execution and enables an anytime trade-off between computation and performance. Short prefixes yield fast but coarse predictions, while longer prefixes produce more refined actions at higher computational cost. This flexibility arises naturally from the ordered tokenization and requires no changes to the policy architecture or training objective, distinguishing **OAT** from prior tokenizers that rely on fixed-length detokenization. The pseudocode for **OAT** policy inference is provided in Algo. 2.

V. Experiments

We evaluate **OAT** by comparing autoregressive policies equipped with different action tokenization schemes, as well as non-autoregressive diffusion-based policies. Our experiments assess both downstream policy performance and the impact of key design choices through controlled ablations.

A. Experimental Setup

Unless otherwise specified, all policies, tokenizers, and evaluation protocols follow the setup described below. We provide more details in Appendix A.

1) Policy Implementation: All policies are trained to predict an action chunk of horizon $H_a = 32$ conditioned on the past $H_o = 2$ observations. During execution, we only execute the first $\frac{1}{2}H_a = 16$ actions from each chunk before re-inferring, following standard practice in action chunking.

We evaluate multiple action tokenization schemes within an autoregressive policy framework. We consider per-dimension binning (`Bin`) and frequency-domain tokenization (`FAST`). We set `Bin` vocabulary size to $|\mathcal{V}| = N = 256$, and we use

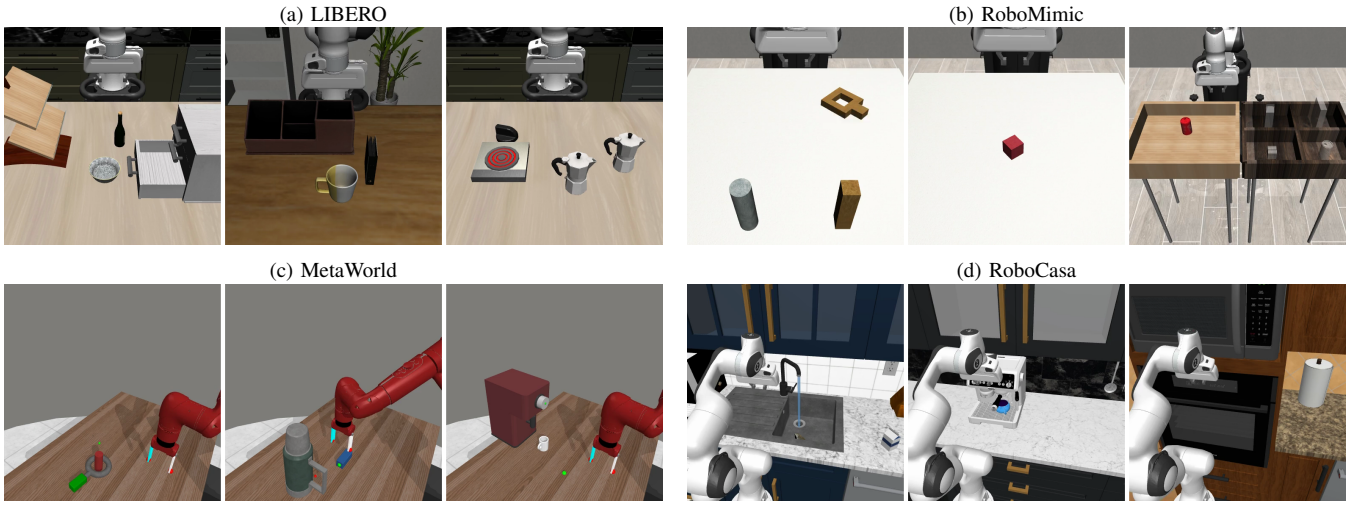


Fig. 4: **Simulation setups.** We evaluate **OAT** across four widely used robotic manipulation benchmarks spanning diverse task structures and dynamics. These environments cover a range of skills, including object manipulation, tool use, and multi-stage interactions.

$|\mathcal{V}| = 1024$ for FAST, which are common configurations in prior work. We additionally compare against *Quantized Skill Transformer* (QueST) [45], a representative learned latent tokenizer. QueST compresses action sequences using a temporal convolution followed by a causal transformer encoder, reducing the temporal horizon from H_a to H_l with a downsampling factor of 4 (i.e., $H_l = \frac{1}{4}H_a$). QueST and **OAT** use the same decoder architecture. **OAT** adopts the same hyperparameters as QueST: a 2-layer transformer encoder with model dimension 256 and head dimension 64, a 4-layer transformer decoder with the same dimensions, latent horizon $H_l = 8$, latent dimension $D_l = 4$, and FSQ levels $[8, 5, 5, 5]$, corresponding to an implicit codebook size $|\mathcal{V}| = 1000$. In addition to autoregressive policies, we include a non-autoregressive baseline based on diffusion policy (DP) [13] with a transformer backbone. To isolate the effects of action representation and tokenization, we use the same policy backbone architecture for all methods.

2) Evaluation Tasks: We conduct comprehensive ablations and analyses, comparing **OAT** against Bin, FAST, QueST, and DP across 20+ tasks drawn from 4 widely used simulation benchmarks (Fig. 4). Specifically, we evaluate on LIBERO [38], RoboMimic [43], MetaWorld [70], and RoboCasa [46]. For simulation experiments, we evaluate each task across 5 random seeds, with 50 evaluation rollouts per seed, resulting in a total of 250 rollouts per task. We report the mean success rate along with its standard error across rollouts.

We additionally validate **OAT** on real-world tabletop manipulation using a fixed-base ARX-5 robotic arm with a single Logitech Webcam for visual observations. We consider two tasks: *Pick & Place Ball* and *Stack Cups* (Fig. 6). For each task, we collect 200 human teleoperation demonstrations. The action space is 7D, consisting of end-effector position, orientation, and gripper control. During evaluation, each task is executed for 20 independent trials, and we report task success rates.

Policy	LIBERO	RoboMimic	MetaWorld	RoboCasa
DP	36.6 ± 0.2	67.1 ± 1.3	19.3 ± 1.6	54.0 ± 1.6
Bin	14.4 ± 0.6	39.5 ± 1.2	14.5 ± 0.7	27.7 ± 0.9
FAST	23.0 ± 0.5	24.0 ± 1.5	7.1 ± 0.7	13.2 ± 1.1
QueST	48.2 ± 0.6	66.9 ± 0.8	17.9 ± 0.9	52.3 ± 1.9
OAT$_1$	11.7 ± 0.7	50.8 ± 1.4	11.3 ± 0.4	47.7 ± 1.3
OAT$_2$	39.8 ± 0.5	52.5 ± 1.2	16.4 ± 0.3	50.3 ± 0.8
OAT$_4$	46.4 ± 0.6	65.3 ± 0.9	19.5 ± 0.8	51.7 ± 1.0
OAT$_8$	56.3 ± 1.0	73.1 ± 0.5	24.4 ± 0.3	54.6 ± 1.1

TABLE I: **Simulation benchmarking** across four manipulation benchmarks. **OAT** consistently outperforms prior action tokenization schemes and exhibits monotonic performance improvements as the number of decoded tokens increases. **OAT**$_K$ denotes detokenization using the first K tokens. Results report mean success rates with standard error across 5 seeds and 50 evaluation rollouts per seed per task. Complete results in Appendix D.

B. Simulation Benchmarking

Table I reports performance across four simulation benchmarks. Bin performs poorly across all benchmarks, as it produces excessively long token sequences and thus violates P.1. FAST achieves compact representations but suffers from invalid or non-decodable token sequences, violating P.2 and leading to unstable policy behavior. Notably, both methods exhibit high reconstruction fidelity, confirming that reconstruction error alone is not predictive of downstream policy performance. QueST provides a substantially stronger baseline by leveraging quantized latent actions. However, its latent token space lacks an ordering, violating P.3, hence its autoregressive modeling does not benefit from inductive biases from causal token ordering aligned with next-token prediction.

OAT consistently outperforms prior action tokenization schemes and matches or exceeds the strongest baselines, while additionally enabling prefix-based decoding that is unavailable to existing methods. We denote **OAT**$_K$ as executing action chunks reconstructed from the first K tokens, i.e., detokenizing the prefix $T_{1:K}$ with $K \leq H_l$. **OAT** exhibits a clear and consistent monotonic performance trend as the number of autoregressive steps increases. As additional tokens

Policy	LIBERO #Tok.	LIBERO Lat.	RoboMimic #Tok.	RoboMimic Lat.	MetaWorld #Tok.	MetaWorld Lat.	RoboCasa #Tok.	RoboCasa Lat.
DP	×	42.0	×	38.1	×	37.7	×	35.3
Bin	224	517.2	224	509.5	128	306.6	384	888.3
FAST	44.2	114.4	53.1	142.0	49.8	129.7	69.7	166.1
QueST	8	27.1	8	29.6	8	31.4	8	30.2
OAT$_1$	1	10.5	1	11.3	1	15.5	1	13.5
OAT$_2$	2	13.2	2	15.3	2	17.9	2	15.8
OAT$_4$	4	17.4	4	18.4	4	22.1	4	19.8
OAT$_8$	8	27.4	8	29.9	8	31.3	8	30.0

TABLE II: **Token count and inference latency.** Comparison of action token counts (#Tok.↓) and policy inference times (Lat.↓) across various benchmarks. For FAST, which generates variable-length sequences, we report the mean token count. **OAT**$_K$ denotes detokenization using the first K tokens. Policy latency is measured in milliseconds (ms) per inference on one NVIDIA A100.

Policy	LIBERO	RoboMimic	MetaWorld	RoboCasa
QueST	48.2 ± 0.6	66.9 ± 0.8	17.9 ± 0.9	52.3 ± 1.9
OAT$_1$	11.7 ± 0.7	50.8 ± 1.4	11.3 ± 0.4	47.7 ± 1.3
OAT$_2$	39.8 ± 0.5	52.5 ± 1.2	16.4 ± 0.3	50.3 ± 0.8
OAT$_4$	46.4 ± 0.6	65.3 ± 0.9	19.5 ± 0.8	51.7 ± 1.0
OAT$_8$	56.3 ± 1.0	73.1 ± 0.5	24.4 ± 0.3	54.6 ± 1.1
OAT$_\times$	35.2 ± 0.7	61.1 ± 1.2	17.6 ± 0.5	48.5 ± 1.6

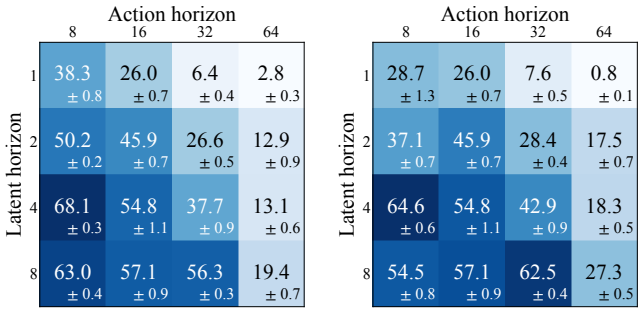
TABLE III: **OAT without ordering underperforms.** Simulation benchmarking across four manipulation benchmarks. **OAT**$_K$ denotes detokenization using the first K tokens, while **OAT**$_\times$ denotes tokenizer training without nested dropout. Results report mean success rates with standard error across 5 seeds and 50 evaluation rollouts per seed per task.

are generated, performance improves steadily: **OAT**$_4$ closes much of the gap to QueST and DP, while **OAT**$_8$ achieves the best performance across all benchmarks. This enables an anytime trade-off between computation and performance: policies may terminate autoregressive generation early when latency constraints are tight, or generate longer sequences for improved performance.

C. Ablation and Analysis

*1) **Compression Rate and Inference Latency**:* Table II compares action compression rates and inference latency across methods. Bin produces extremely long token sequences, resulting in prohibitively high inference latency, while FAST achieves only moderate compression. QueST compresses each action chunk into a fixed-length token sequence, yielding significantly lower inference latency. However, its fixed decoding length limits flexibility. **OAT** enables a smooth and controllable trade-off between compression rate, inference latency, and policy performance. With full decoding, **OAT** and QueST have the same amount of compute per inference.

*2) **How Token Space Ordering (P.3) Matters?**:* Table III studies the role of token space ordering by comparing **OAT** trained with and without ordering-inducing mechanisms, i.e., nested dropout, which enforces a left-to-right priority structure over tokens during training. The variant **OAT**$_\times$ disables nested dropout, resulting in an unordered token space, while all other architectural and training settings are kept identical.



(a) Execute $\frac{1}{2}H_a$ actions (b) Execute a fixed 8 actions

Fig. 5: **Effect of action and token horizons.** Performance of **OAT**$_{H_l}$ on LIBERO as a function of action horizon H_a (rows) and token horizon H_l (columns). Results report mean success rates with standard error across 5 seeds and 50 evaluation rollouts per seed per task.

Across all benchmarks, removing token ordering leads to a consistent performance degradation. **OAT**$_\times$'s performance is significantly worse than **OAT**$_4$ and **OAT**$_8$, and in some cases falls below QueST. This indicates that the structure of the token space plays a critical role in effective autoregressive policy learning: by aligning the token space with next-token prediction, ordering introduces a favorable inductive bias that facilitates both learning and inference.

*3) **How Action (H_a) and Latent (H_l) Horizon Matter?**:* Table 5 analyzes the interaction between action horizon H_a and latent token horizon H_l for **OAT** on LIBERO. The latent horizon H_l is a training-time hyperparameter that determines the number of register tokens. We train separate models for all combinations of $H_a \in \{8, 16, 32, 64\}$ and $H_l \in \{1, 2, 4, 8\}$. To disentangle modeling effects from execution effects, we report two execution regimes: executing $\frac{1}{2}H_a$ actions before re-inference (Table 5a), which reflects practical receding-horizon control, and executing a fixed 8 actions for all H_a (Table 5b) as a controlled diagnostic.

Under the practical execution regime (Table 5a), performance degrades monotonically with increasing H_a for a fixed H_l, reflecting the growing difficulty of long-horizon prediction under limited latent capacity. Increasing H_l consistently mitigates this effect, indicating that additional register tokens enable finer-grained temporal encoding. However, when $H_a \leq H_l$, the information bottleneck largely disappears, yielding diminishing returns; prior work suggests that moderate bottlenecks are beneficial for learning [16, 19, 34], explaining the observed saturation for short horizons such as $H_a = 8$. The fixed-execution setting (Table 5b) reveals a complementary trend. For a fixed H_l, performance becomes non-monotonic in H_a: moderate horizons improve performance by stabilizing early actions [71], while excessively long horizons degrade performance due to the difficulty of compressing long futures into a limited number of tokens.

Together, these results highlight the trade-off between temporal lookahead and latent capacity. Predicting beyond the execution horizon can improve robustness and consistency, but only when the tokenizer can faithfully represent the future. Although the fixed-step execution regime is not intended to

FSQ Levels	[8,6,5]	[8,8,8]	[8,5,5,5]	[8,8,6,5]	[7,5,5,5,5]
Induced $\lvert\mathcal{V}\rvert$	240	512	1000	1920	4375
LIBERO	29.2 ± 0.8	53.5 ± 1.2	56.3 ± 1.0	54.6 ± 1.1	46.9 ± 0.6

TABLE IV: **Effect of codebook size.** Performance of **OAT** on LIBERO under varying FSQ codebook sizes. Results are relatively insensitive to codebook size once moderate capacity is reached, while excessively large codebooks degrade downstream autoregressive learning. Results report mean success rate with standard error across 5 seeds and 50 evaluation rollouts per seed per task.

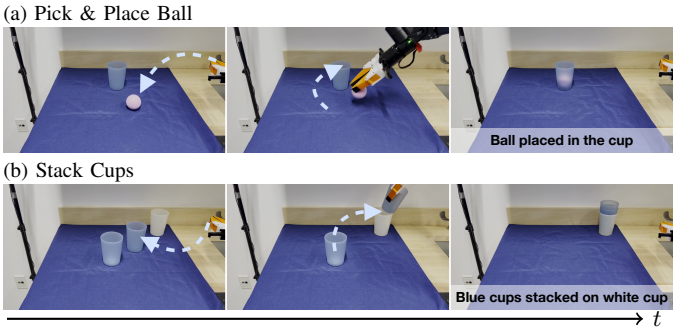


Fig. 6: **Real-world setups.** We validate **OAT** on two tabletop manipulation tasks using a fixed-base robotic arm: (a) *Pick & Place Ball* and (b) *Stack Cups*. Objects are randomly placed on the table.

reflect deployment, it provides a useful diagnostic when interpreted alongside the receding-horizon setting. These findings motivate our default choice of $H_a = 32$ and $H_l = 8$, which balances long-horizon expressivity, compression, and execution stability.

*4) **How Codebook Size Matters?**:* Table IV examines the impact of discrete codebook size $|\mathcal{V}|$, controlled via FSQ level configurations. We vary $|\mathcal{V}|$ from 2^8 to 2^{12} while keeping all other architectural and training settings fixed. Performance improves substantially as the codebook capacity increases from very small to moderate, after which it saturates. However, further enlarging the codebook leads to a clear performance drop. We attribute this degradation to reduced *modelability* for downstream autoregressive policies: larger codebooks increase token entropy and sparsity, making next-token prediction more difficult despite improved reconstruction fidelity.

D. Real-world Results

Table V reports real-world performance on two tabletop manipulation tasks. The results closely mirror trends observed in simulation, validating that the benefits of ordered, prefix-decodable action tokens transfer to real-world robotic control. `Bin` performs poorly primarily due to excessive latency induced by long token sequences, which degrades closed-loop responsiveness. `FAST`, despite its compact tokenization, fails to decode consistently and exhibits unstable, overly aggressive behavior, preventing reliable task execution. `QueST` improves over these baselines but remains limited by its unstructured latent representation.

OAT consistently achieves the highest success rates across both tasks, with performance improving monotonically as the number of decoded tokens increases. Beyond success rates, we observe clear qualitative differences in trajectory execution.

Policy	P&P Ball	Stack Cups
`DP`	14 / 20	11 / 20
`Bin`	4 / 20	8 / 20
`FAST`	8 / 20	6 / 20
`QueST`	11 / 20	8 / 20
OAT$_1$	7 / 20	3 / 20
OAT$_2$	11 / 20	9 / 20
OAT$_4$	13 / 20	12 / 20
OAT$_8$	16 / 20	16 / 20

TABLE V: **Real-world results** on two manipulation tasks. **OAT** consistently outperforms others, and performance improves as the number of decoded tokens increases. **OAT**$_K$ denotes detokenization using the first K tokens. We report mean success rates over 20 evaluation rollouts per task.

OAT produces noticeably smoother motions, with smoothness improving as more tokens are decoded. A common failure mode for **OAT**$_{<4}$ is insufficient execution precision: the robot often reaches configurations that are visually close to success but fails to complete fine-grained insertions (e.g., placing the ball fully into the cup). This behavior indicates that early tokens capture coarse, global action structure, while later tokens encode fine-grained corrective details necessary for precise manipulation, directly supporting the design intent of ordered tokenization.

VI. Discussion and Limitations

This work introduces **OAT**, an action tokenization framework for autoregressive policies that emphasizes ordered, prefix-decodable action representations. While our results demonstrate strong performance and flexibility, several broader implications and open challenges remain.

Recent VLA systems increasingly combine discrete reasoning with continuous control by integrating multiple policy components. For example, the BEHAVIOR-1K [33] winning system [31] employs `FAST` as an auxiliary discrete action representation alongside continuous flow-based experts, highlighting an emerging paradigm in which action tokenization complements rather than replaces continuous policies. In this context, **OAT** offers a principled alternative: its left-to-right ordered and prefix-decodable token space supports autoregressive reasoning over actions while remaining compatible with continuous decoders such as diffusion or flow models. This makes **OAT** a natural auxiliary supervision signal, planning interface, or intermediate abstraction for future VLA pipelines.

A key capability enabled by **OAT** is prefix-based detokenization, which allows actions to be decoded from variable-length token prefixes and provides an anytime trade-off between computation and action fidelity. In this work, however, the autoregressive depth is fixed at deployment time. From an information-theoretic perspective, this is suboptimal: the number of tokens required to represent an action chunk $a_{1:H_a}$ should depend on its intrinsic complexity and required precision. Simple behaviors may admit compact representations, while complex, contact-rich interactions may require deeper autoregressive refinement. Estimating action complexity online and deciding when additional tokens meaningfully reduce uncertainty remains an open problem. We view adaptive autoregressive depth as a natural and important direction for future work, enabled precisely by **OAT** 's ordered and prefix-decodable structure.

Acknowledgments

The computations in this paper were run on the FASRC cluster supported by the FAS Division of Science Research Computing Group at Harvard University. We thank the members of the Embodied Minds Lab at Harvard for insightful discussions, constructive feedback during early stages of this work, and assistance with manuscript proofreading.

References

[1] N. Ahmed, T. Natarajan, and K.R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1): 90–93, 1974. doi: 10.1109/T-C.1974.223784. 14

[2] Roman Bachmann, Jesse Allardice, David Mizrahi, Enrico Fini, Oğuzhan Fatih Kar, Elmira Amirloo, Alaaeldin El-Nouby, Amir Zamir, and Afshin Dehghan. Flextok: Resampling images into 1d token sequences of flexible length. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=DgdOkUUBzf. 15

[3] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers, 2022. URL https://arxiv.org/abs/2106.08254. 1

[4] Suneel Belkhale and Dorsa Sadigh. Minivla: A better vla with a smaller footprint, 2024. URL https://github.com/Stanford-ILIAD/openvla-mini. 1, 3

[5] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164. 2

[6] Yochai Blau and Tomer Michaeli. Rethinking lossy compression: The rate-distortion-perception tradeoff, 2019. URL https://arxiv.org/abs/1901.07821. 2

[7] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2212.06817*, 2022. 1, 2, 3, 14

[8] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alex Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *arXiv preprint arXiv:2307.15818*, 2023. 1, 2, 3, 14

[9] Mu Cai, Jianwei Yang, Jianfeng Gao, and Yong Jae Lee. Matryoshka multimodal models. In *International Conference on Representation Learning*, 2025. 4

[10] Haonan Chen, Jiaming Xu, Hongyu Chen, Kaiwen Hong, Binghao Huang, Chaoqi Liu, Jiayuan Mao, Yunzhu Li, Yilun Du, and Katherine Driggs-Campbell. Multi-modal manipulation via multi-modal policy consensus, 2025. URL https://arxiv.org/abs/2509.23468. 2

[11] Haonan Chen, Jiaming Xu, Lily Sheng, Tianchen Ji, Shuijing Liu, Yunzhu Li, and Katherine Driggs-Campbell. Learning coordinated bimanual manipulation policies using state diffusion and inverse dynamics models. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[12] Haonan Chen, Cheng Zhu, Shuijing Liu, Yunzhu Li, and Katherine Rose Driggs-Campbell. Tool-as-interface: Learning robot policies from observing human tool use. In *Proceedings of Robotics: Conference on Robot Learning (CoRL)*, 2025.

[13] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024. 2, 6, 14

[14] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965. ISSN 00255718, 10886842. URL http://www.jstor.org/stable/2003354. 14

[15] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International Conference on Representation Learning*, 2024. 4

[16] Sander Dieleman. Generative modelling in latent space, 2025. URL https://sander.ai/2025/04/15/latents.html. 2, 4, 7, 15

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An

image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929. 1

[18] Philip Gage. A new algorithm for data compression. *C Users J.*, 12(2):23–38, February 1994. ISSN 0898-9788. 3, 14

[19] Haoran He, Peilin Wu, Chenjia Bai, Hang Lai, Lingxiao Wang, Ling Pan, Xiaolin Hu, and Weinan Zhang. Bridging the sim-to-real gap from the information bottleneck perspective. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=Bq4XOaU4sV. 7

[20] Zhi Hou, Tianyi Zhang, Yuwen Xiong, Hengjun Pu, Chengyang Zhao, Ronglei Tong, Yu Qiao, Jifeng Dai, and Yuntao Chen. Diffusion transformer policy, 2025. URL https://arxiv.org/abs/2410.15959. 2

[21] Haifeng Huang, Xinyi Chen, Yilun Chen, Hao Li, Xiaoshen Han, Zehan Wang, Tai Wang, Jiangmiao Pang, and Zhou Zhao. Roboground: Robotic manipulation with grounded vision-language priors. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22540–22550, 2025. 2

[22] Sigmund Hennum Høeg, Aksel Vaaler, Chaoqi Liu, Olav Egeland, and Yilun Du. Hybrid diffusion for simultaneous symbolic and continuous planning, 2025. URL https://arxiv.org/abs/2509.21983. 2

[23] Daniel Jiwoong Im, Kevin Zhang, Nakul Verma, and Kyunghyun Cho. Deep autoregressive models as causal inference engines, 2025. URL https://arxiv.org/abs/2409.18581. 4

[24] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054. 2

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

[26] Ilyes Khemakhem, Ricardo Monti, Robert Leech, and Aapo Hyvarinen. Causal autoregressive flows. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 3520–3528. PMLR, 13–15 Apr 2021. URL https://proceedings.mlr.press/v130/khemakhem21a.html. 4

[27] Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions, 2025. URL https://arxiv.org/abs/2502.06768. 2, 4

[28] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 1, 2, 3, 14

[29] Alexander Kolesnikov, André Susano Pinto, Lucas Beyer, Xiaohua Zhai, Jeremiah Harmsen, and Neil Houlsby. Uvim: A unified modeling approach for vision with learned guiding codes. In *Advances in Neural Information Processing Systems*, volume 35, pages 26295–26308. Curran Associates, Inc., 2022. 2, 4

[30] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 30233–30249. Curran Associates, Inc., 2022. 4

[31] Ilia Larchenko, Gleb Zarin, and Akash Karnatak. Task adaptation of vision-language-action model: 1st place solution for the 2025 behavior challenge, 2025. URL https://arxiv.org/abs/2512.06951. 8

[32] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 26991–27008. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/lee24y.html. 1, 3

[33] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, Hang Yin, Michael Lingelbach, Minjune Hwang, Ayano Hiranaka, Sujay Garlanka, Arman Aydin, Sharon Lee, Jiankai Sun, Mona Anvari, Manasi Sharma, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Yunzhu Li, Silvio Savarese, Hyowon Gweon, C. Karen Liu, Jiajun Wu, and Li Fei-Fei. Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation. *arXiv preprint arXiv:2403.09227*, 2024. 8

[34] Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise, 2025. URL https://arxiv.org/abs/2511.13720. 7

[35] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization, 2024. URL https://arxiv.org/abs/2406.11838. 2

[36] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative

modeling, 2023. URL https://arxiv.org/abs/2210.02747. 15

[37] Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky T. Q. Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code, 2024. URL https://arxiv.org/abs/2412.06264. 15

[38] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL https://arxiv.org/abs/2306.03310. 6, 14

[39] Chaoqi Liu, Haonan Chen, Sigmund H. Høeg, Shaoxiong Yao, Yunzhu Li, Kris Hauser, and Yilun Du. Flexible multitask learning with factorized diffusion policy, 2025. URL https://arxiv.org/abs/2512.21898. 2

[40] Jiaming Liu, Hao Chen, Pengju An, Zhuoyang Liu, Renrui Zhang, Chenyang Gu, Xiaoqi Li, Ziyu Guo, Sixiang Chen, Mengzhen Liu, Chengkai Hou, Mengdi Zhao, KC alex Zhou, Pheng-Ann Heng, and Shanghang Zhang. Hybridvla: Collaborative diffusion and autoregression in a unified vision-language-action model, 2025. URL https://arxiv.org/abs/2503.10631. 2

[41] Minghuan Liu, Zhengbang Zhu, Xiaoshen Han, Peng Hu, Haotong Lin, Xinyao Li, Jingxiao Chen, Jiafeng Xu, Yichu Yang, Yunfeng Lin, et al. Manipulation as in simulation: Enabling accurate geometry perception in robots. *arXiv preprint arXiv:2509.02530*, 2025. 2

[42] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022. URL https://arxiv.org/abs/2209.03003. 15

[43] 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. 6, 14

[44] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. In *International Conference on Representation Learning*, volume 2024, pages 51772–51783, 2024. 3, 4

[45] Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. Quest: Self-supervised skill abstractions for learning continuous control. In *Advances in Neural Information Processing Systems*, volume 37, pages 4062–4089. Curran Associates, Inc., 2024. doi: 10.52202/079017-0133. 1, 3, 6, 14

[46] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems*, 2024. 6, 14

[47] NVIDIA, :, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734. 2

[48] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024. 1

[49] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao

Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi Jim Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick Tree Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, and Zipeng Lin. Open x-embodiment: Robotic learning datasets and rt-x models : Open x-embodiment collaboration0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903, 2024. doi: 10.1109/ICRA57147.2024.10611477.

[50] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. In *Robotics: Science and Systems*, 2025. 1, 2, 3, 14

[51] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://openai.com/blog/better-language-models/. 2

[52] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal conditioned imitation learning using score-based diffusion policies. In *Robotics: Science and Systems*, 2023. 2

[53] Oren Rippel, Michael Gelbart, and Ryan Adams. Learning ordered representations with nested dropout. In *Proceedings of the 31st International Conference on Machine Learning*, Proceedings of Machine Learning Research, Bejing, China, 22–24 Jun 2014. PMLR. URL https://proceedings.mlr.press/v32/rippel14.html. 4

[54] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 1715–1725, 2016. 1

[55] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27: 379–423, 1948. URL http://plan9.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf. 2, 5

[56] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=St1giarCHLP. 14

[57] Chenrui Tie, Yue Chen, Ruihai Wu, Boxuan Dong, Zeyi Li, Chongkai Gao, and Hao Dong. ET-SEED: EFFICIENT TRAJECTORY-LEVEL SE(3) EQUIVARIANT DIFFUSION POLICY. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=OheAR2xrtb. 2

[58] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971. 2

[59] Michael Tschannen, Olivier Bachem, and Mario Lucic. Recent advances in autoencoder-based representation learning, 2018. URL https://arxiv.org/abs/1812.05069. 2

[60] Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 3

[61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 2, 5

[62] Lirui Wang, Kevin Zhao, Chaoqi Liu, and Xinlei Chen. Learning real-world action-video dynamics with heterogeneous masked autoregression, 2025. URL https://arxiv.org/abs/2502.04296. 2

[63] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, and Jian Tang. Tinyvla: Toward fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 10(4):3988–3995, 2025. doi:

10.1109/LRA.2025.3544909. 1, 2

[64] Rosa Wolf, Yitian Shi, Sheng Liu, and Rania Rayyes. Diffusion models for robotic manipulation: A survey, 2025. URL https://arxiv.org/abs/2504.08438. 2

[65] Haoyu Xiong, Xiaomeng Xu, Jimmy Wu, Yifan Hou, Jeannette Bohg, and Shuran Song. Vision in action: Learning active perception from human demonstrations. *arXiv preprint arXiv:2506.15666*, 2025. 2

[66] Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. Byt5: Towards a token-free future with pre-trained byte-to-byte models, 2022. URL https://arxiv.org/abs/2105.13626. 1

[67] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388. 2

[68] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications, 2025. URL https://arxiv.org/abs/2209.00796. 2

[69] Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation. In *Advances in Neural Information Processing Systems*, volume 37, pages 128940–128966. Curran Associates, Inc., 2024. doi: 10.52202/079017-4096. 4

[70] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1094–1100. PMLR, 30 Oct–01 Nov 2020. URL https://proceedings.mlr.press/v100/yu20a.html. 6, 14

[71] Thomas T. Zhang, Daniel Pfrommer, Chaoyi Pan, Nikolai Matni, and Max Simchowitz. Action chunking and exploratory data collection yield exponential improvements in behavior cloning for continuous control, 2025. URL https://arxiv.org/abs/2507.09061. 2, 7

[72] Xi Zhang, Yuan Pu, Yuki Kawamura, Andrew Loza, Yoshua Bengio, Dennis L. Shung, and Alexander Tong. Trajectory flow matching with applications to clinical time series modelling. In *Advances in Neural Information Processing Systems*, volume 37, pages 107198–107224. Curran Associates, Inc., 2024. doi: 10.52202/079017-3404. 2

[73] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.016. 4, 15

[74] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705. 2

[75] Yue Zhao, Hanwen Jiang, Zhenlin Xu, Chutong Yang, Ehsan Adeli, and Philipp Krähenbühl. Spherical leech quantization for visual tokenization and generation. *arXiv preprint arXiv:2512.14697*, 2025. 2

APPENDIX

A. Implementation Details

This section provides comprehensive implementation details for all policies, tokenizers, optimization settings, and evaluation protocols referenced in the main paper.

All policies are trained to predict a contiguous action chunk of horizon $H_a = 32$, conditioned on the most recent $H_o = 2$ observations. During deployment, policies operate in a receding-horizon manner: only the first $\frac{1}{2}H_a = 16$ actions of each predicted chunk are executed before re-inference. This execution strategy balances temporal consistency and responsiveness, and is used consistently across all methods unless stated otherwise.

We compare multiple action tokenization schemes within the same autoregressive policy framework:

- `Bin` [7, 8, 28]: Each action dimension is discretized into $N = 256$ uniform bins, yielding a token sequence whose length scales with $H_a \times D_a$.
- `FAST` [50]: We use a vocabulary size of $|\mathcal{V}| = 1024$, following standard configurations in prior work.
- `QueST` [45]: Action chunks are compressed using a temporal convolution followed by a causal transformer encoder, reducing the temporal horizon from H_a to $H_l = \frac{1}{4}H_a = 8$.
- **OAT**: For fair comparison, **OAT** adopts the same decoder architecture and latent dimensionality as `QueST`. The tokenizer encoder is a 2-layer transformer with model dimension 256 and head dimension 64. The latent representation has horizon $H_l = 8$ and latent dimension $D_l = 4$, discretized using finite scalar quantization (FSQ) with levels $[8, 5, 5, 5]$, corresponding to an implicit codebook size of approximately 1000.

All autoregressive policies share the same backbone architecture: a transformer decoder with 4 layers, model dimension 256, and head dimension 64. The decoder predicts discrete action tokens autoregressively, using teacher forcing during training and fully autoregressive rollout during inference. Using a shared policy backbone isolates the effect of action representation from policy capacity. In addition to autoregressive policies, we include a diffusion-based baseline (`DP`) [13]. The diffusion policy uses exactly the same 4-layer transformer backbone as the autoregressive models, ensuring that performance differences arise from the action representation and inference paradigm rather than architectural capacity. We employ a 10-step Denoising Diffusion Implicit Models (DDIM) [56] sampling schedule for `DP`. All models are trained using AdamW with identical optimization settings: a constant learning rate of `5e-5` for tokenizers and policy networks, and `1e-5` for observation encoders, with no weight decay.

We evaluate and analyze policies on four widely used simulation benchmarks:

- LIBERO [38]: *libero10*; 50 demonstrations per task; action dimension $D_a = 7$.
- RoboMimic [43]: *lift, square, can*; 200 multi-human (mh) demonstrations per task; action dimension $D_a = 7$.
- MetaWorld [70]: *box close, coffee pull, disassemble, stick pull*; 50 demonstrations per task; action dimension $D_a = 4$.
- RoboCasa [46]: *close drawer, coffee press button, turn off microwave, turn off sink faucet*; 50 human demonstrations and 150 machine-generated demonstrations per task; action dimension $D_a = 12$.

For each task, we evaluate 5 random seeds with 50 rollouts per seed, resulting in 250 evaluation episodes per task. Performance is reported as the mean task success rate with standard error across rollouts.

B. The Structural Mismatch of `FAST`

A critical limitation of the `FAST` tokenizer arises from the fundamental structural conflict between the probabilistic, variable-length nature of Byte Pair Encoding (BPE) [18] and the strict, fixed-dimensional requirements of robotic control.

1) Mechanism of `FAST`.: `FAST` operates by applying a Discrete Cosine Transform (DCT) [1, 14] to action chunks, pruning low-magnitude high-frequency components, and flattening the remaining coefficients into a 1D integer sequence. A BPE tokenizer is then trained to compress this sequence. While this effectively separates coarse structure from fine detail, it introduces a critical dependency between the token sequence length and the action chunk topology.

2) Variable Expansion vs. Fixed Topology: In standard large language models, the decoding process is agnostic to the exact number of characters produced; a token representing `apple` (5 bytes) is structurally valid in the same context as `a` (1 byte). However, the `FAST` tokenizer maps discrete tokens to variable-length sequences of DCT coefficients. Let a generated token sequence be $T_{1:H_l} = [T_1, T_2, \ldots, T_{H_l}]$. Each token T_i expands into a sequence of integers s_i of length $|s_i|$. The total sequence of coefficients S is the concatenation of these expansions:

$$S = s_1 \oplus s_2 \oplus \cdots \oplus s_k, \quad \text{where } |S| = \sum_{i=1}^{k} |s_i|.$$

The robot controller, however, strictly requires a control chunk of dimensions $H_a \times D_a$ (time horizon $\times$ action dimension), necessitating a fixed total coefficient count $N_{\text{target}} = T \times D$.

3) The Decoding Failure: Because the policy is autoregressive and probabilistic, it generates tokens based on likelihood rather than structural constraints. There is no guarantee that the generated sequence $T_{1:H_l}$ will satisfy $|S| = N_{\text{target}}$. When $|S| \neq N_{\text{target}}$, the reshaping operation into (H_a, D_a) becomes mathematically impossible, raising the "undecodable" error (e.g., `ValueError: cannot reshape array`).

4) The "Spectral Shift" Catastrophe: Naive solutions, such as padding or truncating S to match N_{target}, are catastrophic due to the use of the discrete cosine transform (DCT) [1, 14]. The sequence S is an ordered flattening of frequency coefficients. If a token generating 3 coefficients is replaced by

a token generating 2 coefficients (a "missing" coefficient at index j), every subsequent coefficient at indices $k > j$ shifts position. In the frequency domain, this shift is semantically destructive, for example, coefficients governing joint J may drift into the slots for joint $J+1$. Consequently, the undecodable state acts as a necessary safety assertion. It is preferable to halt execution (outputting a no-op) than to reshape a corrupted coefficient sequence that would result in unpredictable and potentially dangerous physical motion.

C. OAT Detokenization $\mathcal{T}^{-1}$

Similar to [73], the single-pass decoder is implemented as a transformer decoder consisting of alternating self-attention and cross-attention layers. The decoder cross-attends from a fixed set of sinusoidal positional embeddings to the discrete action tokens produced by **OAT**. The final decoder embeddings are projected back into the continuous action space, yielding a reconstructed action chunk of shape $H_a \times D_a$. The tokenizer and decoder are trained end-to-end using a reconstruction objective, specifically mean squared error (MSE) between the original and reconstructed action chunks.

When the latent bottleneck is small, training the decoder with a simple reconstruction loss can lead to degraded reconstruction quality, as the decoder must recover long-horizon action sequences from severely compressed representations [16, 36, 37]. To address this limitation, **OAT** can employ a rectified flow decoder conditioned on the quantized register latents. Concretely, we construct partially noised action sequences

$$a^t_{1:H_a} = (1-t)\, a^0_{1:H_a} + t\,\epsilon,$$

where $a^0_{1:H_a}$ denotes the clean action chunk, $t \in [0,1]$ is a randomly sampled time step, and $\epsilon \sim \mathcal{N}(0, I)$ is Gaussian noise. The flow decoder receives the concatenation of the noised actions and the quantized register tokens $\mathit{Quant}(z_{1:H_l})$ and is trained to predict the flow

$$v = \epsilon - a^0_{1:H_a}.$$

We minimize the rectified flow objective $\|\hat{v} - v\|^2$, where

$$\hat{v} = \mathit{Dec}\big(\mathit{Quant}(z_{1:H_l}) \oplus a^t_{1:H_a}\big),$$

following prior work on flow-based generative modeling [2, 42].

D. Simulation Benchmarking

We provide full results of simulation experiments in Table VI.

LIBERO

Policy	#Tok.	Inf. Lat.	Soup/Sauce Basket	Cheese/Butter Basket	Soup/Cheese Basket	Two Moka Pots	Stove & Moka	Bowl to Drawer	Mugs on Plates	Book to Caddy	Mug & Pudding	Mug to Micro	Avg.
DP	×	42.0	26.0 ± 3.0	18.8 ± 1.4	24.8 ± 1.9	52.4 ± 2.7	56.8 ± 3.4	62.8 ± 2.1	20.0 ± 1.7	18.4 ± 1.9	29.6 ± 2.6	56.0 ± 3.1	36.6 ± 0.2
Bin	224	517.2	1.6 ± 0.7	3.6 ± 0.7	4.0 ± 1.7	8.8 ± 2.0	24.0 ± 1.1	46.0 ± 3.4	2.8 ± 1.5	31.2 ± 2.5	6.8 ± 0.8	15.6 ± 2.4	14.4 ± 0.6
FAST	44.2	114.4	14.8 ± 1.6	6.4 ± 0.7	1.6 ± 0.7	33.6 ± 4.4	52.8 ± 1.4	50.4 ± 5.0	16.0 ± 1.7	28.4 ± 1.7	22.0 ± 3.5	4.4 ± 1.3	23.0 ± 0.5
QueST	8	27.1	22.4 ± 2.8	16.0 ± 2.8	31.6 ± 2.7	47.6 ± 2.0	79.6 ± 1.9	88.0 ± 1.4	20.8 ± 2.8	65.6 ± 4.8	35.6 ± 1.3	74.8 ± 3.7	48.2 ± 0.6
OAT$_1$	1	10.5	2.4 ± 0.7	1.6 ± 0.7	1.6 ± 0.7	2.8 ± 0.8	23.6 ± 1.2	26.0 ± 2.9	0.8 ± 0.5	26.8 ± 3.4	3.6 ± 1.0	28.0 ± 1.7	11.7 ± 0.7
OAT$_2$	2	13.2	15.2 ± 3.1	16.4 ± 1.3	25.2 ± 2.1	39.2 ± 1.5	59.2 ± 2.4	69.6 ± 4.3	14.0 ± 1.4	81.2 ± 1.9	13.6 ± 3.0	64.8 ± 2.9	39.8 ± 0.5
OAT$_4$	4	17.4	14.8 ± 1.4	16.4 ± 1.7	32.4 ± 2.2	57.2 ± 3.2	68.8 ± 3.1	78.4 ± 2.7	24.4 ± 1.5	86.0 ± 2.1	14.8 ± 4.8	70.8 ± 2.2	46.4 ± 0.6
OAT$_8$	8	27.4	26.8 ± 3.2	35.6 ± 2.6	51.6 ± 2.2	61.2 ± 4.3	87.6 ± 1.2	91.2 ± 1.0	31.2 ± 2.7	70.8 ± 4.5	32.0 ± 2.8	75.2 ± 3.8	56.3 ± 1.0
OAT$_\times$	8	27.4	5.6 ± 1.6	5.6 ± 0.7	21.2 ± 4.1	33.6 ± 2.4	65.6 ± 1.2	81.2 ± 3.4	6.0 ± 1.1	73.6 ± 4.0	3.2 ± 1.6	56.0 ± 2.2	35.2 ± 0.7

RoboMimic

Policy	#Tok.	Inf. Lat.	Lift	Square	Can	Avg.
DP	×	38.1	99.6 ± 0.4	24.0 ± 1.8	77.6 ± 2.6	67.1 ± 1.3
Bin	224	509.5	86.0 ± 1.3	1.2 ± 0.8	31.2 ± 2.4	39.5 ± 1.2
FAST	53.1	142.0	53.6 ± 3.0	0.4 ± 0.4	18.0 ± 3.2	24.0 ± 1.5
QueST	8	29.6	98.8 ± 0.5	29.2 ± 4.8	72.8 ± 3.0	66.9 ± 0.8
OAT$_1$	1	11.3	89.6 ± 1.5	6.4 ± 1.2	56.4 ± 3.4	50.8 ± 1.4
OAT$_2$	2	15.3	86.6 ± 1.6	11.2 ± 0.8	59.6 ± 2.8	52.5 ± 1.2
OAT$_4$	4	18.4	99.2 ± 0.5	23.6 ± 1.6	73.2 ± 2.7	65.3 ± 0.9
OAT$_8$	8	29.9	99.2 ± 0.5	39.2 ± 2.4	80.8 ± 2.3	73.1 ± 0.5
OAT$_\times$	8	29.2	96.8 ± 1.0	16.0 ± 4.2	70.4 ± 4.9	61.1 ± 1.2

MetaWorld

Policy	#Tok.	Inf. Lat.	Box Close	Coffee Pull	Disassemble	Stick Pull	Avg.
DP	×	37.7	21.2 ± 4.6	27.6 ± 1.3	23.2 ± 1.0	5.2 ± 0.5	19.3 ± 1.6
Bin	128	306.6	9.6 ± 2.6	24.4 ± 0.7	20.8 ± 1.6	3.2 ± 0.5	14.5 ± 0.7
FAST	49.8	129.7	0.0 ± 0.0	16.4 ± 2.0	10.4 ± 2.6	1.6 ± 0.7	7.1 ± 0.7
QueST	8	31.4	12.4 ± 2.0	28.4 ± 1.8	23.2 ± 1.4	7.6 ± 0.4	17.9 ± 0.9
OAT$_1$	1	15.5	20.0 ± 0.6	15.2 ± 0.4	6.4 ± 0.9	3.6 ± 1.3	11.3 ± 0.4
OAT$_2$	2	17.9	32.4 ± 0.7	19.2 ± 1.2	10.8 ± 0.4	3.2 ± 0.4	16.4 ± 0.3
OAT$_4$	4	22.1	37.2 ± 2.2	22.4 ± 1.5	14.0 ± 1.7	4.4 ± 0.7	19.5 ± 0.8
OAT$_8$	8	31.3	44.4 ± 1.2	26.4 ± 0.4	17.2 ± 0.7	9.6 ± 1.0	24.4 ± 0.3
OAT$_\times$	8	31.3	32.4 ± 0.9	19.6 ± 1.3	13.6 ± 0.9	4.8 ± 1.3	17.6 ± 0.5

RoboCasa

Policy	#Tok.	Inf. Lat.	Close Drawer	Coffee Press Button	Turn Off Microwave	Turn Off Sink Faucet	Avg.
DP	×	35.3	52.0 ± 2.8	56.8 ± 3.6	52.8 ± 3.5	54.4 ± 1.8	54.0 ± 1.6
Bin	384	888.3	20.4 ± 1.6	22.0 ± 2.3	31.2 ± 4.7	27.2 ± 3.2	27.7 ± 0.9
FAST	69.7	166.1	20.8 ± 2.0	8.4 ± 1.5	16.8 ± 1.4	6.8 ± 2.1	13.2 ± 1.1
QueST	8	30.2	54.8 ± 2.1	55.6 ± 4.5	42.0 ± 2.5	56.8 ± 1.6	52.3 ± 1.9
OAT$_1$	1	13.5	47.2 ± 3.4	49.6 ± 1.7	35.2 ± 2.6	58.8 ± 3.5	47.7 ± 1.3
OAT$_2$	2	15.8	55.2 ± 2.2	53.6 ± 1.2	34.0 ± 4.6	58.4 ± 2.5	50.3 ± 0.8
OAT$_4$	4	19.9	52.0 ± 1.4	52.8 ± 1.5	39.2 ± 1.2	62.8 ± 2.7	51.7 ± 1.0
OAT$_8$	8	30.0	53.6 ± 2.9	63.6 ± 1.9	42.8 ± 3.9	58.4 ± 4.6	54.6 ± 1.1
OAT$_\times$	8	30.0	55.6 ± 4.3	43.2 ± 1.9	36.4 ± 2.2	58.8 ± 1.7	48.5 ± 1.6

TABLE VI: **Simulation benchmarking** policy performance, tokenizer compression rate (#Tok.), and policy inference latency (Inf. Lat.) in milliseconds (ms) on one NVIDIA A100. For FAST, which generates variable-length sequences, we report the mean token count. **OAT**$_K$ denotes detokenization using the first K tokens, while **OAT**$_\times$ denotes tokenizer training without nested dropout. Results report mean success rates with standard error across 5 seeds and 50 evaluation rollouts per seed per task.