$\mathcal{M}^2$Tok: Multi-head Multi-codebook Discrete Action Tokenization for Vision-Language-Action Models

Chunpu Xu[1,3*], Zhixuan Liang[3,5†], Yuhao Zhang[2], Chi-Min Chan[4], Jiashuo Wang[1], Yang Xiao[1], Mengkang Hu[5], Xiaokang Yang[2], and Yao Mu[2,3†]

[1] The Hong Kong Polytechnic University, HongKong SAR, China
[2] Shanghai Jiao Tong University, Shanghai, China
[3] Shanghai AI Laboratory, Shanghai, China
[4] Hong Kong University of Science and Technology, HongKong SAR, China
[5] The University of Hong Kong, HongKong SAR, China
chun-pu.xu@connect.polyu.hk, zxliang@cs.hku.hk, muyao@sjtu.edu.cn

Abstract. Recent advancements have successfully adapted autoregressive language models to process multimodal signals, such as images and actions. Since raw action signals are continuous, effective tokenization is essential to map high-dimensional inputs into compact discrete tokens for autoregressive processing. However, existing discrete action tokenizers often suffer from high reconstruction loss, failing to preserve the fine-grained dynamics required for precise control. This "discretization bottleneck" significantly limits the performance ceiling of downstream Vision-Language-Action (VLA) models. To address this, we propose $\mathcal{M}^2$Tok, a Multi-head Multi-codebook Action Tokenizer designed to minimize reconstruction error and enhance policy performance. Our approach introduces two key structural innovations: (1) we decompose the latent action features into multiple heads, enabling the model to implicitly align specific heads with distinct action dimensions; (2) we assign independent codebooks to each head for quantization. By leveraging the combinatorial nature of multiple codebooks, we significantly expand the representational expressivity of the tokenizer, leading to substantially lower reconstruction loss compared to previous methods. We evaluate the $\mathcal{M}^2$Tok-based VLA on the RoboTwin, Simpler-Env, and 3 zero-shot real-world tasks. Experimental results demonstrate our method not only achieves superior reconstruction fidelity but also significantly boosts the success rate of VLA models. Comprehensive ablation studies further confirm the effectiveness of the multi-head and multi-codebook mechanisms. Code is available at https://github.com/cpaaax/M2Tok.

Keywords: Vision-Language-Action Model · Action Tokenizer

[*] Work done during internship at Shanghai AI LAB.
[†] Corresponding Authors.

1 Introduction

Autoregressive language models [1, 9, 18] have emerged as a unifying paradigm for multimodal generation, demonstrating remarkable capabilities in visual synthesis [22, 27, 29], audio generation [12, 37], and robotic control [3, 14, 15, 25, 39]. By leveraging specialized tokenizers to map continuous signals into compact discrete symbols, Vision-Language-Action (VLA) models can inherit the scalable architectures and training recipes of Large Language Models (LLMs). However, unlike text or image patches, robot actions possess unique characteristics. They are high-frequency, continuous, and exhibit strong inter-step correlations. Designing an action tokenizer that effectively captures these fine-grained dynamics remains a critical bottleneck in current VLA research.

Early approaches, such as RT-1 [3] and OpenVLA [14], adopted a naive binning strategy, discretizing each action dimension independently into fixed bins. While simple, this per-step tokenization ignores temporal correlations and struggles with high-frequency control, often leading to jerky motions. To address this, recent works have embraced "action chunking" [7, 38], predicting sequences of future actions to ensure temporal coherence and mitigate compounding errors. This shift necessitates tokenizers capable of compressing entire action trajectories rather than single steps.

Consequently, two primary paradigms for trajectory tokenization have emerged: frequency-based compression and latent vector quantization. FAST [25] treats action sequences as signals, utilizing the Discrete Cosine Transform (DCT) combined with Byte-Pair Encoding (BPE) [8]. While FAST effectively reduces sequence length, it fundamentally clashes with the topology of robotic data that BPE is designed for discrete text, and applying it to continuous signals requires a harsh rounding of DCT coefficients. This introduces the reconstruction loss and results in variable-length tokens, which complicates efficient parallel decoding [13]. Alternatively, Vector Quantization (VQ) methods, such as VQ-BET [15] and VQ-VLA [33], learn a fixed-length discrete latent representation via VQ-VAE. However, we identify a critical "discretization bottleneck" in these designs, stemming from two fundamental limitations. First, they suffer from semantic entanglement. Existing tokenizers typically treat the robot action vector as a monolithic entity, projecting heterogeneous signals, such as 6-DoF poses and binary gripper states, into a shared latent space. This approach fails to explicitly model the varying semantics across different action dimensions, forcing the tokenizer to compromise between the high-precision dynamics of arm trajectories and the discrete modes of gripper actuation. Second, and perhaps more importantly, these methods exhibit limited representational expressivity. By relying on residual vector quantization (RVQ), their capacity to represent diverse behaviors is constrained by the size of the codebook. This structure lacks combinatorial density required to cover the vast, high-frequency action space of manipulation, often resulting in coarse approximations that lose fine-grained control details.

To overcome this bottleneck, we propose $\mathcal{M}^2$Tok, a Multi-head Multi-codebook action tokenizer to maximize representational expressivity and reconstruction fidelity. Our key insight is that high-dimensional action spaces are best represented

not by a residual vector quantization, but by decomposing the latent space into orthogonal subspaces. Specifically, $\mathcal{M}^2$Tok introduces two structural innovations that (1) subspace decomposition: we split the latent action features into multiple heads, allowing the model to implicitly align specific heads with distinct underlying dynamics (e.g., end-effector pose vs. gripper status); (2) combinatorial quantization: we assign an independent codebook to each head. Consider a single codebook containing V entries, its representational capacity is consequently V. If, however, we partition the available capacity into h equal-sized codebooks, each consisting of $\frac{V}{h}$ entries (such that the total number of code vectors remains V), the number of unique action representations becomes $\left(\frac{V}{h}\right)^h$. By utilizing h codebooks of equal size, we exponentially enhance the action feature space's expressivity while maintaining a constant total codebook size. This strategic partitioning dramatically increases the diversity and richness of possible representations, thereby expanding the potential efficacy of the action tokenizer. As shown in Table 1, this architecture allows $\mathcal{M}^2$Tok to achieve reconstruction fidelity significantly superior to existing state-of-the-art tokenizers, breaking the discretization bottleneck. Additionally, building upon the subspace decomposition and combinatorial quantization of $\mathcal{M}^2$Tok, we design a simple yet effective conversion module to bridge the gap between multi-codebook tokens and downstream VLA models. This module ensures that the distinct dynamics captured by each head are preserved as coherent semantic units. This design prevents information loss during autoregressive decoding, allowing the VLA model to fully utilize the diverse and rich representation space facilitated by our multi-head multi-codebook architecture.

Tokenizer	L1 loss
Fast	0.0055
VQ-BET	0.0044
VQ-VLA	0.0032
$\mathcal{M}^2$Tok	0.0024

Table 1: Reconstruction L1 loss of different tokenizers on RoboTwin data.

In summary, our contributions are threefold:

- We present $\mathcal{M}^2$Tok, a transformer-based multi-head multi-codebook discrete action tokenizer designed for Vision-Language-Action models. The multi-codebook architecture enhances $\mathcal{M}^2$Tok's ability to represent a wide variety of action sequences.
- We design a lightweight conversion module that enables VLA models to effectively interpret disentangled semantics learned by multi-head multi-codebook architecture, significantly improving downstream VLA performance.
- We demonstrate the efficacy of $\mathcal{M}^2$Tok-based VLAs through extensive evaluation on the RoboTwin, Simpler-Env, and three zero-shot real-world tasks, showing that superior reconstruction fidelity directly translates to higher success rates in complex manipulation tasks.

2 Related Work

2.1 Vision-Language-Action Models

Vision-Language-Action models (VLAs) [2, 3, 5, 14, 15, 17, 23, 25, 26, 28, 32, 34, 39] have become a unified framework for robotic manipulation, combining visual

perception, language processing, and the generation of robotic actions. These models extend the foundational capabilities of Large Language Models (LLMs) and Vision-Language Models (VLMs) by incorporating action prediction to facilitate seamless interaction with the physical world. Early works, such as CLIPort [26] and PerAct [23], focused on aligning visual cues with language-driven action policies. The RT series [2,3,39] further advanced this field by introducing action tokenization, which enabled scalable transfers from web data to robotic applications. Recent works include Octo [28], which developed a multi-robot dataset to enhance multitask learning, and OpenVLA [14], showcasing significant generalization in executing household tasks.

2.2 Action Tokenization

Bridging continuous control with discrete LLMs hinges on effective action tokenization. While early methods like RT-1 [3] and OpenVLA [14] employed naive per-step binning, they neglect temporal correlations and high-frequency dynamics. Consequently, recent research has shifted towards trajectory tokenization to encode action chunks. Frequency-based approaches, such as FAST [25], utilize DCT combined with BPE. However, applying text-centric BPE to continuous signals forces a topological mismatch, introducing reconstruction errors and yielding variable-length tokens that complicate parallel decoding [13]. Alternatively, VQ-based methods [15, 33] map trajectories to discrete latents but suffer from a "discretization bottleneck": their reliance on monolithic codebooks entangles heterogeneous semantics (e.g., pose vs. gripper) and constrains expressivity to the size of the vocabulary. Unlike these approaches, our $\mathcal{M}^2$Tok utilizes a multi-head, multi-codebook architecture to achieve combinatorial expressivity, explicitly disentangling action semantics while maintaining a fixed token length for streamlined autoregressive generation. Alternatively, recent works have bypassed tokenization by integrating continuous diffusion heads directly into transformer architectures [4, 10, 11, 16, 19, 20, 24]. These approaches require modifying the standard LLM architecture and loss functions (e.g., incorporating denoising objectives). In contrast, our approach retains the pure discrete autoregressive interface, allowing $\mathcal{M}^2$Tok to seamlessly inherit the pre-training recipes and inference optimizations of standard LLMs without architectural deviation.

3 Method

The overview of the $\mathcal{M}^2$Tok-based VLA is demonstrated in Figure 1. The VLA training process involves two key steps. Initially, we train the $\mathcal{M}^2$Tok to function as an action tokenizer. Subsequently, we integrate the $\mathcal{M}^2$Tok with the VLA to facilitate multitask learning.

3.1 Problem Formulation

We consider the problem of learning a general-purpose robotic policy from a diverse multi-task datase $D = \{\{(o_1, s_1, a_1), ..., (o_{T_i}, s_{T_i}, a_{T_i}), L_{m,i}\}_{i=1}^{N_m}\}_{m=1}^{M}$. Here,

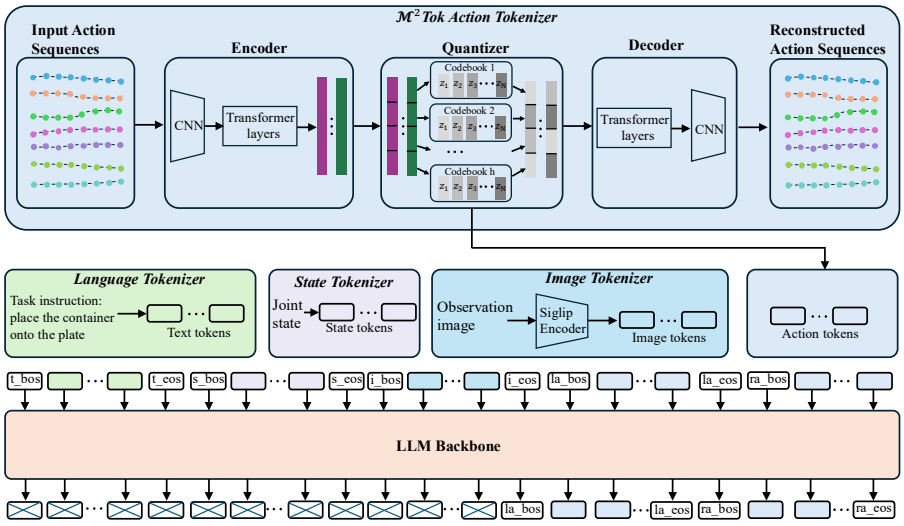

Fig. 1: Overview of $\mathcal{M}^2$Tok action tokenizer (top) and the $\mathcal{M}^2$Tok-based VLA model (bottom).

M denotes the number of tasks, N_m is the number of trajectories for the m-th task, and $L_{m,i}$ provides the natural language description of the i-th trajectory in the m-th task. Each trajectory consists of a sequence of observations o_t (RGB images), proprioceptive states s_{t_m} and continuous bimanual actions $a_t \in \mathbb{R}^{14}$, where each arm possesses 7 Degrees of Freedom (DoF). Our objective is to train a policy $\pi_\theta(a_{t:t+k}|o_t, s_t, L)$, which predicts a coherent sequence of future actions $a_{t:t+k}$ (an action chunk of length k) given the current observation and instruction. However, modeling the high-dimensional continuous distribution of $a_{t:t+k}$ directly is intractable for standard language models. Therefore, we introduce $\mathcal{M}^2$Tok to discretize the continuous action space into compact tokens, enabling the policy to generate complex behaviors via standard categorical cross-entropy optimization.

3.2 $\mathcal{M}^2$Tok Tokenizer

We construct $\mathcal{M}^2$Tok upon the Vector Quantized Variational Autoencoder (VQ-VAE) framework [30], specifically adapted to address the "discretization bottleneck" in robotic control. As illustrated in Figure 1, the architecture comprises three key components: a Temporal-Aware Hybrid Encoder, a Multi-Head Multi-Codebook Quantizer, and a Fidelity-Preserving Decoder.

To maximize data efficiency and enforce kinematic symmetry, we propose a Bimanual Factorization strategy. Instead of treating the 14-DoF dual-arm action as a monolithic vector, we decompose each bimanual trajectory $a_{1:T}$ into two independent single-arm sequences, a^{left}, $a^{right} \in \mathbb{R}^{T\times 7}$. These single-arm trajectories form a unified action set A', which serves as the input to our tokenizer.

This design encourages the model to learn arm-agnostic kinematic primitives, leaving inter-arm coordination to be handled by the high-level policy.

The encoder is designed to capture both high-frequency local dynamics and long-range temporal dependencies. We employ a hybrid architecture that interleaves 1D causal Convolutional layers with Transformer blocks. The convolutional layers extract local motion features and downsample the temporal dimension, while the Transformer layers integrate global context. Given an input single-arm action chunk $a' \in \mathbb{R}^{k\times 7}$ from A', the encoder produces a continuous latent sequence $\hat{Z} \in \mathbb{R}^{R\times d}$, where R is the reduced temporal resolution and d is the latent dimension.

Standard VQ-VAEs typically quantize the entire latent vector using a single codebook, limiting expressivity to the codebook size $|V|$. To overcome this limitation, we introduce a Combinatorial Quantization mechanism. We decompose the latent space $\mathbb{R}^d$ into h orthogonal subspaces (heads). For each latent vector $\hat{z}_r$ ($r \in \{1, ..., R\}$) is split into h segments $\{\hat{z}_{i,r}\}_{i=1}^{h}$, where each segment $\hat{z}_{i,r} \in \mathbb{R}^{\frac{d}{h}}$. Crucially, we maintain h independent codebooks $C = \{Z_1, ..., Z_h\}$, where each codebook $Z_i = \{z_{i,n}\}_{n=1}^{N}$ contains N learnable prototypes of dimension $\frac{d}{h}$. The quantization is performed independently within each subspace:

$$q_{i,r} = arg \min_{z_{i,n}\in Z_i} ||z_{i,n} - \hat{z}_{i,r}||_2, \tag{1}$$

where $q_{i,r}$ is the nearest neighbor lookup index from the codebook Z_i.

This process effectively expands the representational capacity from N to $(\frac{N}{h})^h$ by combining codes from different subspaces. The quantized latent vector $z_{q,r}$ is then reconstructed by concatenating the selected prototypes:

$$z_{q,r} = \text{Concat}\left(z_{1,q_{1,r}}, \ldots, z_{h,q_{h,r}}\right). \tag{2}$$

This multi-head design allows $\mathcal{M}^2$Tok to disentangle distinct semantic attributes of the action (e.g., end-effector pose vs. gripper state) into different subspaces, significantly enhancing reconstruction fidelity. The decoder mirrors the encoder's hybrid architecture. It takes the quantized sequence of latent vectors $Z_q = \{z_{q,r}\}_{r=1}^{R}$ and progressively upsamples it to reconstruct the original action $\hat{a}$.

To train the $\mathcal{M}^2$Tok tokenizer, we follow VQ-VAE [30] to employ three different loss functions for optimization. The first is the reconstruction loss $\mathcal{L}_{rec}$ to minimize the difference between the predicted action sequences $\hat{a}$ and the input action sequences a'. The second is the embedding loss $\mathcal{L}_{emb}$, which compute the distance between latent representation $\hat{z}_{i,r}$ and the nearest embedding $z_{i,q_{i,r}}$ to update the embedding space of codebooks. The third is the commitment loss $\mathcal{L}_{com}$, which specifically affects the encoder weights, motivating the encoder's output to remain near the selected codebook vector, thus reducing frequent switching between different code vectors. The total training losses are

optimized as follows:

$$\begin{aligned}\mathcal{L} &= \lambda_1 \mathcal{L}_{rec} + \lambda_2 \mathcal{L}_{emb} + \lambda_3 \mathcal{L}_{com} \\ &= \lambda_1 \|\hat{a} - a'\|_2^2 + \lambda_2 \sum_{i=1}^{h} \sum_{r=1}^{R} \|sg(\hat{z}_{i,r}) - z_{i,q_{i,r}}\|_2^2 \\ &\quad + \lambda_3 \sum_{i=1}^{h} \sum_{r=1}^{R} \|sg(z_{i,q_{i,r}}) - \hat{z}_{i,r}\|_2^2,\end{aligned} \tag{3}$$

where λ_1, λ_2 and λ_3 are the weights to each loss item, sg represents stopgradient operation.

3.3 $\mathcal{M}^2$Tok-based VLA

Our goal is to train a VLA policy $\pi_\theta(a_{t:t+k}|o_t, s_t, L)$. For the language description L, we use the base LLM's native tokenizer to obtain text tokens. Regarding the joint states s_t, we adopt the approach outlined in FAST [25] to discretize each joint dimension into 256 bins, yielding a sequence of discrete state tokens. For processing the observation image o_t, we resize each image to a resolution of 224×224 and then apply siglip-so400m-patch14-224 [36] as the image tokenizer. This procedure produces a 16×16 grid of patch embeddings, i.e., 256 continuous image tokens per frame. For the input action sequence, we use the proposed $\mathcal{M}^2$Tok to transform the continuous action sequences a' into discrete action tokens $\tilde{a}$. To delineate modality boundaries in the input tokens, we introduce special functional tokens: *t_ bos*/*t_ eos* for text, *s_ bos*/*s_ eos* for state, *i_ bos*/*i_ eos* for image, and *la_ bos*/*la_ eos* and *ra_ bos*/*ra_ eos* for left-arm and *right-arm actions*, respectively. During training, we serialize the context and targets into a single sequence, for example: [*t_ bos*, *t_ tokens*, *t_ eos*, *s_ bos*, *s_ tokens*, *s_ eos*, *i_ bos*, *i_ tokens*, *i_ eos*, *la_ bos*, *left-action_ tokens*, *la_ eos*, *ra_ bos*, *right-action_ tokens*, *ra_ eos*]. This design yields a consistent tokenized interface across modalities, enabling straightforward autoregressive learning of the action sequence conditioned on language, state, and vision.

Our $\mathcal{M}^2$Tok converts continuous actions into fixed-length sequences of discrete tokens. We explore the effectiveness of the proposed action tokenizer by employing the autoregressive decoding.

Autoregressive Decoding. In the autoregressive setting, the VLA generates the action sequence via next-token prediction. At each position of generation, the model outputs a distribution over the action vocabulary conditioned on the input context (o_t, s_t, L) and all previously generated action tokens. The training objective is to minimize the negative log-likelihood of the predicted action tokens, given the sequence of preceding tokens:

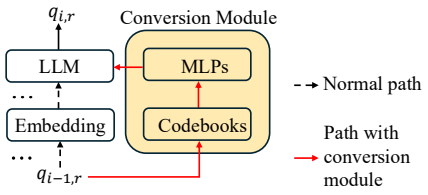

Fig. 2: $\mathcal{M}^2$Tok conversion module.

Table 2: The average number of steps and description for each task of RoboTwin.

Task Name	Avg. Step	Task Example Description	Task Name	Avg. Step	Task Example Description
Beat Block Hammer	67	Grab the silver hammer and use it to hit.	Pick Diverse Bottles	75	Grab the plastic bottle, pick up the yellow body bottle.
Click Bell	52	Press the center top of the blue bell.	Place Mouse Pad	88	Grab the gray mouse and drop it on the black mat.
Handover Mic	134	Grasp the blue microphone and pass it across.	Shake Bottle	133	Shake the orange bottle after lifting it.
Move Can Pot	90	Lift the smooth sauce can, place it by the kitchen pot.	Place Phone Stand	82	Carry the flat phone to the green phone rack.
Move Pillbottle Pad	88	Hold the white bottle and position it on the pad.	Place Burger Fries	141	Move the hamburg and fries box to the tray.
Move Playingcard Away	70	Slide the box for playing cards off the table outward.	Place Container Plate	92	Move the cup and drop the container onto the plate.

$$\mathcal{L}_{AD} = -\sum_{i=1}^{h}\sum_{r=1}^{R}\log P(q_{i,r}|q_{<}, o_t, s_t, L), \tag{4}$$

where $q_{<}$ represent previous action tokens.

In the standard setup, action tokens are handled like text tokens: the previous discrete action token $q_{i-1,r}$ is first mapped to a vector by the token-embedding layer, and the LLM then uses this representation to predict the next token $q_{i,r}$. To better leverage the action semantics captured by $\mathcal{M}^2$Tok, we replace this generic embedding with a lightweight action-token conversion module. As shown in Figure 2, this module maps each discrete action token to a continuous representation that preserves the semantics learned by $\mathcal{M}^2$Tok, providing the VLA with more informative action features while keeping the rest of the generation pipeline unchanged. Concretely, for each previously generated action token $q_{i-1,r}$, we retrieve its corresponding quantized embedding $z_{i-1,q_{i-1,r}}$ from the $\mathcal{M}^2$Tok multi-codebooks. We then pass the quantized embedding through an MLP projector that maps it into the same semantic space as the LLM's text embeddings.

4 Experiments

In Section 4.1, we describe the implementation details of our experiments. Section 4.2 evaluates the manipulation performance of the $\mathcal{M}^2$-based VLA in the RoboTwin and Simpler-Env simulation, comparing it against prior methods. In Section 4.3, we present ablations that quantify the contribution of each component. Section 4.4 analyzes how the number of codebooks influences VLAs' performance. In Section 4.5, we analyze the latency of VLAs based on different tokenizer. Finally, Section 4.6 reports both quantitative and qualitative results on real-world manipulation tasks.

4.1 Implementation Details

RoboTwin Simulation. We train the VLAs on a synthetic dataset generated with the RoboTwin simulator [6], which enables large-scale, automated collection

Table 3: Settings for Data Collection in RoboTwin Simulator.

Parameter	Value	Parameter	Value
Save Frequency	15	Random Head Camera Distance	0.03
Embodiment	Piper	Random Table Height	0.03
Random Background	True	Random Light	True
Cluttered Table	False	Crazy Random Light Rate	0.02
Clean Background Rate	0.02	Head Camera Type	D435

of diverse and realistic bimanual manipulation trajectories. The dataset spans 12 tasks, each with 1,600 expert demonstrations (19,200 trajectories in total). To increase diversity and robustness, we apply domain randomization during data collection; the specific settings are listed in Table 3. The details of each task are shown in Table 2. Observations consist of RGB images from head camera.

Simpler-Env Simulation. SimplerEnv bridges the control and visual disparities inherent in simulation, facilitating policy evaluations that accurately predict real-world performance. In this study, we utilize the WidowX robot to demonstrate this capability across four manipulation tasks: Put Carrot on Plate, Put Spoon on Towel, Stack Green on Yellow, and Put Eggplant in Basket. Following prior works, we employ the BridgeV2 dataset [31] as the training data, which is a large-scale dataset featuring 24 distinct environments (e.g., kitchens, sinks, tabletops), over 100 objects, and diverse manipulation tasks.

Training setup. To ensure a fair comparison across action tokenizers, we integrate each tokenizer into the same VLA architecture, as shown at the bottom of Figure 1. We use Qwen2.5-0.5B [35] as the LLM backbone. For $\mathcal{M}^2$Tok, the total codebook size is 2048, partitioned into 8 codebooks ($h = 8$) with 256 entries each, and a codebook embedding dimension of 256. The per-arm action dimension is $H = 7$, yielding 14 dimensions for the RoboTwin bimanual setting and Simpler-Env unimanual setting. $\mathcal{M}^2$Tok is trained with AdamW [21] at a learning rate of 5×10^{-5}, using a total batch size of 2048×4 on four 4090 GPUs.

Subsequent to pre-training the $\mathcal{M}^2$Tok tokenizer, we leverage it to discretize continuous action trajectories into latent tokens for VLA training. During this phase, the parameters of both the $\mathcal{M}^2$Tok and the visual encoder (SigLIP) are kept frozen, exclusively updating the weights of the LLM backbone and the projection MLPs. We employ the AdamW optimizer with a base learning rate of 1×10^{-4}, modulated by a cosine decay scheduler. The models are trained for 10 epochs on the RoboTwin benchmark and 20 epochs on Simpler-Env to ensure convergence. Additionally, a warm-up ratio of 0.03 is applied to learning rate.

Baselines. In order to assess the effectiveness of the proposed $\mathcal{M}^2$Tok, we compare it against four baseline tokenizers. Binning tokenizer [14]: this tokenizer converts the action at each timestep across each action dimension into 256 discrete bins. FAST tokenizer [25]: the tokenizer employs DCT to convert action sequences into the frequency domain. It then uses BPE to encode and compress the action tokens. VQ-BET tokenizer [15]: VQ-BET uses the designed encoder to encode input action chunks, followed by a two-layer RVQ process to quantize the processed features. VQ-VLA Tokenizer [33]: The VQ-VLA utilizes 2D

Table 4: Success rates of different VLAs across 12 RoboTwin tasks.

Tokenizer	Beat Block Hammer	Move Playingcard Away	Pick Diverse Bottles	Move Can Pot	Move Pillbottle Pad	Click Bell	Handover Mic
Binning	0.11	0.01	**0.27**	0.02	0.00	0.67	0.85
FAST	0.00	0.18	0.06	0.08	0.01	0.68	0.17
VQ-BET	0.10	0.27	0.12	0.13	0.06	0.64	0.59
VQ-VLA	0.14	0.46	0.24	0.30	0.20	0.59	0.93
$\mathcal{M}^2$**Tok**	**0.20**	**0.48**	0.22	**0.58**	**0.33**	**0.71**	**0.94**

Tokenizer	Place Mouse Pad	Place Container Plate	Place Phone Stand	Place Burger Fries	Shake Bottle	Average Success
Binning	0.00	0.00	0.02	0.01	0.92	0.24
FAST	0.01	0.07	0.01	0.00	0.76	0.17
VQ-BET	0.00	0.54	0.04	0.15	0.87	0.29
VQ-VLA	0.08	0.79	**0.23**	0.52	**0.96**	0.45
$\mathcal{M}^2$**Tok**	**0.10**	**0.83**	0.20	**0.64**	0.87	**0.51**

temporal convolutional layers, combined with time and action-type embeddings, to process input actions. These are subsequently passed through RVQ layers to generate discrete action tokens.

To ensure a fair comparison, we retrain the FAST, VQ-BET, and VQ-VLA tokenizers using the same action sequence training dataset employed by $\mathcal{M}^2$Tok. All VLAs are trained in the multi-task setting.

4.2 Simulation Results

RoboTwin. We evaluate all VLAs on the 12 tasks of RoboTwin outlined in Section 4.1. We perform 100 rollouts for each task. Each rollout is labeled as a success (1) or failure (0), and we report the success rate, the fraction of successful rollouts, as the evaluation metric. Quantitative comparisons across 12 diverse manipulation tasks are presented in Table 4. Our proposed $\mathcal{M}^2$Tok demonstrates superior efficacy, achieving a state-of-the-art average success rate of 51%, surpassing the strongest baseline VQ-VLA (45%) by a significant relative margin of 13%. Notably, in the Move Can Pot task, $\mathcal{M}^2$Tok achieves a success rate of 0.58, nearly doubling the performance of VQ-VLA (0.30). Similarly, in Place Burger Fries, our method reaches 0.64, outperforming VQ-VLA (0.52) and dominating VQ-BET (0.12). We attribute this to the subspace decomposition mechanism, which disentangles distinct kinematic features (e.g., separating gripper actuation from arm trajectory) into orthogonal heads. This ensures that the VLA model can attend to and predict fine-grained action details necessary for delicate manipulation. Figure 3 presents visualization examples of three tasks within the RoboTwin simulator.

Simpler-Env. We also conduct experiments on the four tasks of the Simpler-Env benchmark. As summarized in Table 5, $\mathcal{M}^2$Tok demonstrates superior generalization capabilities, achieving the highest average success rate of 28%, outperforming the strongest VQ-VLA baseline (21%) by a relative margin of 33%.

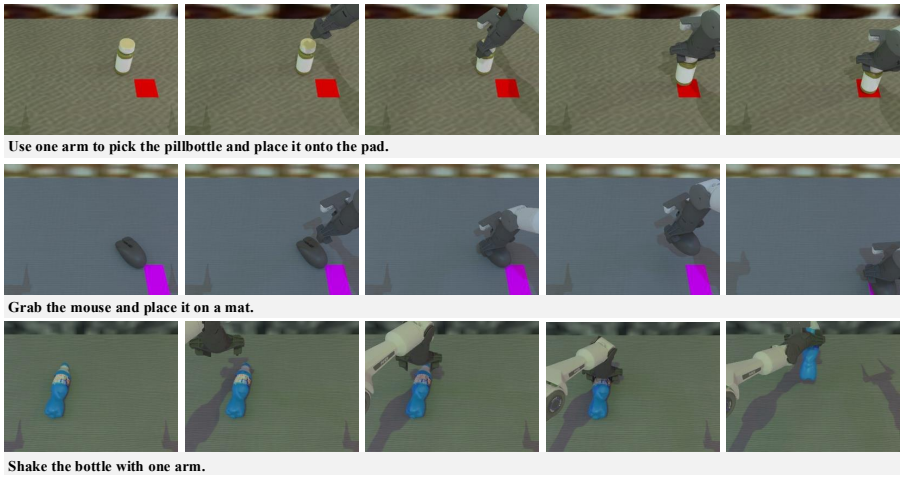

Fig. 3: RoboTwin visualizations for "Move Pillbottle Pad", "Place Mouse Pad", and "Shake Bottle" tasks.

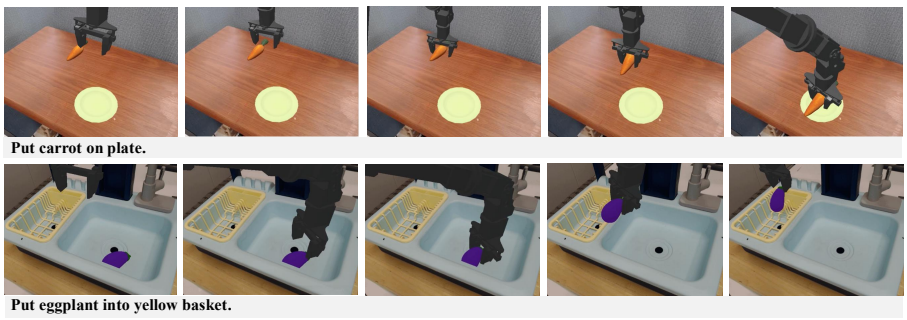

Fig. 4: Simpler-Env visualizations for "Put Carrot" and "Put Eggplant" tasks.

The most significant performance gain is observed in the Put Eggplant in Basket task. While VQ-VLA and other baselines completely failed to solve this task, $\mathcal{M}^2$Tok achieved a remarkable 33% success rate. This task requires precise grasping of an irregular object (eggplant) and accurate placement. The success of $\mathcal{M}^2$Tok suggests that our Combinatorial Quantization strategy enables the policy to generate high-fidelity action tokens that can effectively manipulate objects with complex geometries, whereas RVQ codebooks (like VQ-BET) struggle to capture the necessary fine-grained control primitives. Figure 4 illustrates the visualization results of two tasks in the Simpler-Env simulator.

Table 5: Success rates of different VLAs across 4 Simpler-Env tasks.

Tokenizer	Put Spoon on Towel	Put Carrot on Plate	Stack Green on Yellow	Put Eggplant in Basket	Avg.
Bining	0.08	0.00	0.00	0.04	0.03
FAST	0.21	0.17	0.00	0.08	0.12
VQ-BET	0.04	0.04	0.00	0.00	0.02
VQ-VLA	0.29	0.33	**0.21**	0.00	0.21
$\mathcal{M}^2$Tok	**0.33**	**0.38**	0.08	**0.33**	**0.28**

Table 6: Results of Ablation study across 12 RoboTwin tasks.

Tokenizer	Beat Block Hammer	Move Playingcard Away	Pick Diverse Bottles	Move Can Pot	Move Pillbottle Pad	Click Bell	Handover Mic
$\mathcal{M}^2$Tok	**0.20**	**0.48**	**0.22**	**0.58**	**0.33**	**0.71**	**0.94**
w/o Multi-head	0.10	0.25	0.03	0.12	0.00	0.32	0.59
w/o Multi-codebook	0.14	0.34	0.18	0.34	0.09	0.55	0.65
w/o Conversion	0.13	0.35	0.20	0.35	0.15	0.53	0.81

Tokenizer	Place Mouse Pad	Place Container Plate	Place Phone Stand	Place Burger Fries	Shake Bottle	Average Success
$\mathcal{M}^2$Tok	**0.10**	**0.83**	**0.20**	0.64	0.87	**0.51**
w/o Multi-head	0.00	0.36	0.01	0.12	0.79	0.22
w/o Multi-codebook	0.01	0.46	0.07	0.46	0.84	0.34
w/o Conversion	0.07	0.79	0.09	**0.67**	**0.88**	0.42

Fig. 5: Real-world manipulation visualizations for "Place Container Plate" and "Pick Diverse Bottles" tasks.

4.3 Ablation Study

To further investigate the effectiveness of the multi-head architecture, multi-codebook quantization, and the conversion module in $\mathcal{M}^2$Tok, we conducted ablation experiments by individually removing each component while maintaining the others. The experimental setups are as follows: (1) w/o Multi-head: replacing the multi-head mechanism with a single head to evaluate the importance of subspace decomposition. (2) w/o Multi-codebook: utilizing a single codebook instead of multiple quantization to assess the impact of combinatorial expressivity. (3) w/o Conversion: removing the conversion module and directly using the embedding layer to process discrete tokens.

The results are summarized in Table 6. We observe that the full $\mathcal{M}^2$Tok model achieves the best performance (51%), and removing any component leads to a degradation in the overall success rate. Removing the multi-head mechanism causes the most catastrophic performance drop, plummeting the success rate from 51% to 22%. This degradation is particularly severe in precision-dependent tasks like Move Pillbottle Pad (0.33 to 0.00) and Place Mouse Pad (0.10 to 0.00). This result highlights the critical role of kinematic disentanglement. Without

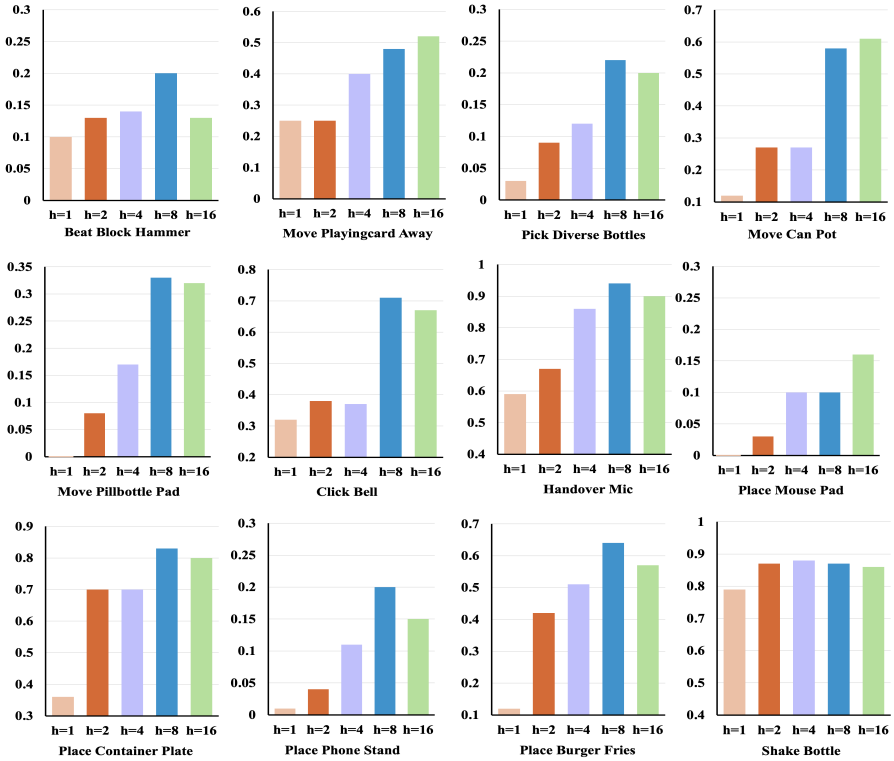

Fig. 6: Effect of varying number h across 12 RoboTwin tasks.

multiple heads to independently model different action subspaces (e.g., separating gripper actuation from arm translation), the tokenizer fails to capture the fine-grained dynamics required for complex manipulation, leading to a "smearing" of action details. Replacing the diverse codebooks with a single shared codebook results in a significant decline to 34%. This indicates that the expressive capacity of the tokenizer is heavily reliant on the combinatorial nature of our design. A single codebook creates a bottleneck, limiting the diversity of representable action primitives. Removing the conversion module leads to a moderate decrease in performance (0.51 to 0.42). While the model retains some capability, the drop indicates that the conversion module plays a vital role in aligning the discrete latent space of the VQ-VAE with the continuous embedding space of the LLM. It acts as a bridge that smoothes the transition from quantized tokens to autoregressive prediction, thereby enhancing the overall stability and accuracy of the VLA policy.

4.4 Analysis on the number of Multi-head Multi-codebook

Recall that the latent features from the $\mathcal{M}^2$Tok encoder are split into h segments, each of which is then quantized by a independant codebook. To quantitatively

Table 7: Latency and Frequency in the RoboTwin Environment.

Method	Bining	FAST	VQ-BET	VQVLA	$\mathcal{M}^2$Tok	$\mathcal{M}^2$Tok(vLLM)
Latency(ms)	485.3	42.7	30.7	81.2	121.6	17.8
Speed(Hz)	2.0	23.4	32.6	12.3	8.2	56.2

explore this, we evaluate VLA performance across varying the number of h, as shown in Figure 6. The results align with our theoretical analysis: increasing h significantly boosts performance, confirming that the expanded subspace decomposition and combinatorial diversity allow the tokenizer to capture more nuanced manipulation behaviors without increasing the total vocabulary size. We observe that performance improves consistently for most tasks as h increases, peaking at $h = 8$. Further increasing h yields diminishing returns, likely due to the over-fragmentation of the latent space.

4.5 Latency Analysis

We evaluate the inference frequency and latency of VLAs based on different tokenizers on a single RTX 4090 in the bimanual RoboTwin environment, with results shown in Table 7. Bining-based VLA runs at 2 Hz, while our method increases the frequency to over 8 Hz—a 4.1× improvement. Furthermore, our VLA supports the vLLM inference engine. By leveraging vLLM, our VLA achieves a throughput of 56.2 Hz. This significantly exceeds the control frequency required for most real-time manipulation tasks.

4.6 Real-World Results

Table 8: The results of real-world experiments.

Tokenizer	Click Bell	Place Container Plate	Pick Diverse Bottles	Average Success
Binning	6/20	0/20	0/20	0.10
FAST	4/20	1/20	0/20	0.08
VQ-BET	7/20	2/20	0/20	0.15
VQ-VLA	10/20	7/20	0/20	0.28
$\mathcal{M}^2$**Tok**	11/20	7/20	2/20	0.33

In our real-world experiments, we utilize the AgileX Cobot Magic, a mobile platform configured with an Aloha setup that includes four robotic arms. Each arm is an AgileX Piper featuring six degrees of freedom and is equipped with a one-DoF parallel gripper. The platform is also outfitted with a RealSense D435 RGB camera, which captures real-time RGB images at a resolution of 640 × 480 pixels and a frame rate of approximately 30 Hz. To assess the zero-shot sim-to-real transfer capability of our VLA models, we directly applied the VLAs, trained using RGB images in the RoboTwin simulator, to three real-world manipulation tasks: Click Bell, Place Container Plate, and Pick Diverse Bottles. Each task is tested with 20 trials. Table 8 displays the evaluation results, where $\mathcal{M}^2$Tok attained an average success rate of 0.33 across all these tasks, consistently outperforming the baselines. This demonstrates that our $\mathcal{M}^2$Tok tokenizer can effectively adapt to real-world scenarios. Additionally, Figure 5 showcases two examples of manipulation processes performed by $\mathcal{M}^2$Tok. The top and bottom images illustrate the execution of the "Place Container Plate" and "Pick Diverse Bottles" tasks, respectively. These

cases highlight the model's ability to accurately identify the spatial positions of objects and execute precise grasping operations to complete tasks.

5 Conclusion

In this work, we have presented $\mathcal{M}^2$Tok, a novel action tokenization framework that fundamentally addresses the "discretization bottleneck" hindering current Vision-Language-Action (VLA) models. By departing from monolithic quantization strategies, our approach decomposes high-dimensional action signals into orthogonal subspaces and leverages the combinatorial density of multi-head codebooks. This architectural shift enables $\mathcal{M}^2$Tok to maximize representational expressivity, achieving a level of reconstruction fidelity previously unattainable by standard vector quantization or frequency-based methods. Across the RoboTwin and Simpler-Env benchmarks, as well as three challenging zero-shot real-world tasks, $\mathcal{M}^2$Tok-based VLA demonstrates significant improvements in task success rates compared to existing methods. Furthermore, comprehensive ablation studies confirmed that our structural innovations, specifically subspace decomposition and combinatorial codebook assignment, are the critical drivers of these gains.

Acknowledgement. This work was supported in part by the Shanghai Magnolia Talent Program Pujiang Project under Grant No. 25PJA076.

References

1. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
2. Belkhale, S., Ding, T., Xiao, T., Sermanet, P., Vuong, Q., Tompson, J., Chebotar, Y., Dwibedi, D., Sadigh, D.: Rt-h: Action hierarchies using language. ArXiv **abs/2403.01823** (2024)
3. Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., et al.: Rt-1: Robotics transformer for real-world control at scale. arXiv preprint arXiv:2212.06817 (2022)
4. Bu, Q., Yang, Y., Cai, J., Gao, S., Ren, G., Yao, M., Luo, P., Li, H.: Univla: Learning to act anywhere with task-centric latent actions. ArXiv **abs/2505.06111** (2025)
5. Cheang, C.L., Chen, G., Jing, Y., Kong, T., Li, H., Li, Y., Liu, Y., Wu, H., Xu, J., Yang, Y., Zhang, H., Zhu, M.: Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation. ArXiv **abs/2410.06158** (2024)
6. Chen, T., Chen, Z., Chen, B., Cai, Z., Liu, Y., Liang, Q., Li, Z., Lin, X., Ge, Y., Gu, Z., Deng, W., Guo, Y., Nian, T., Xie, X., Chen, Q., Su, K., Xu, T., Liu, G., Hu, M., ang Gao, H., Wang, K., Liang, Z., Qin, Y., Yang, X., Luo, P., Mu, Y.: Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. ArXiv **abs/2506.18088** (2025)
7. Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B., Song, S.: Diffusion policy: Visuomotor policy learning via action diffusion. In: Robotics: Science and Systems (2023)

8. Gage, P.: A new algorithm for data compression. C Users Journal **12**(2), 23–38 (1994)
9. Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al.: The llama 3 herd of models. arXiv preprint arXiv:2407.21783 (2024)
10. Hu, Y., Guo, Y., Wang, P., Chen, X., Wang, Y.J., Zhang, J., Sreenath, K., Lu, C., Chen, J.: Video prediction policy: A generalist robot policy with predictive visual representations. ArXiv **abs/2412.14803** (2024)
11. Intelligence, P., Black, K., Brown, N., Darpinian, J., Dhabalia, K., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Galliker, M.Y., Ghosh, D., Groom, L., Hausman, K., Ichter, B., Jakubczak, S., Jones, T., Ke, L., LeBlanc, D., Levine, S., Li-Bell, A., Mothukuri, M., Nair, S., Pertsch, K., Ren, A.Z., Shi, L.X., Smith, L., Springenberg, J.T., Stachowicz, K., Tanner, J., Vuong, Q., Walke, H.R., Walling, A., Wang, H., Yu, L., Zhilinsky, U.: π0.5: a vision-language-action model with open-world generalization. ArXiv **abs/2504.16054** (2025)
12. Ji, S., Jiang, Z., Wang, W., Chen, Y., Fang, M., Zuo, J., Yang, Q., Cheng, X., Wang, Z., Li, R., Zhang, Z., Yang, X., Huang, R., Jiang, Y., Chen, Q., Zheng, S., Zhao, Z.: Wavtokenizer: an efficient acoustic discrete codec tokenizer for audio language modeling. In: The Thirteenth International Conference on Learning Representations (2025)
13. Kim, M.J., Finn, C., Liang, P.: Fine-tuning vision-language-action models: Optimizing speed and success (2025)
14. Kim, M.J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E., Lam, G., Sanketi, P.R., Vuong, Q., Kollar, T., Burchfiel, B., Tedrake, R., Sadigh, D., Levine, S., Liang, P., Finn, C.: Openvla: An open-source vision-language-action model. ArXiv **abs/2406.09246** (2024)
15. Lee, S., Wang, Y., Etukuru, H., Kim, H.J., Shafiullah, N.M.M., Pinto, L.: Behavior generation with latent actions. In: ICML (2024)
16. Li, S., Gao, Y., Sadigh, D., Song, S.: Unified video action model. ArXiv **abs/2503.00200** (2025)
17. Liang, Z., Li, Y., Yang, T., Wu, C., Mao, S., Pei, L., Yang, X., Pang, J., Mu, Y., Luo, P.: Discrete diffusion vla: Bringing discrete diffusion to action decoding in vision-language-action policies. arXiv preprint arXiv:2508.20072 (2025)
18. Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., et al.: Deepseek-v3 technical report. arXiv preprint arXiv:2412.19437 (2024)
19. Liu, J., Chen, H., An, P., Liu, Z., Zhang, R., Gu, C., Li, X., Guo, Z., Chen, S., Liu, M., Hou, C., Zhao, M., alex Zhou, K., Heng, P.A., Zhang, S.: Hybridvla: Collaborative diffusion and autoregression in a unified vision-language-action model. ArXiv **abs/2503.10631** (2025)
20. Liu, S., Wu, L., Li, B., Tan, H., Chen, H., Wang, Z., Xu, K., Su, H., Zhu, J.: Rdt-1b: a diffusion foundation model for bimanual manipulation. ArXiv **abs/2410.07864** (2024)
21. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2017)
22. Ma, C., Jiang, Y., Wu, J., Yang, J., Yu, X., Yuan, Z., Peng, B., Qi, X.: Unitok: a unified tokenizer for visual generation and understanding. In: Belgrave, D., Zhang, C., Montoya, L.N., Lin, H., Pascanu, R., Koniusz, P., Ghassemi, M., Chen, N., Ruíz, I.V.M., Loaiza-Bonilla, A. (eds.) Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025,

NeurIPS 2025, San Diago, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025 (2025), http://papers.nips.cc/paper_files/paper/2025/hash/bbe04d329d531e8814f1199098bd8fb6-Abstract-Conference.html

23. Mohit, S., Lucas, M., Dieter, F.: Perceiver-actor: A multi-task transformer for robotic manipulation. ArXiv **abs/2209.05451** (2022)
24. Nvidia, Bjorck, J., Castaneda, F., Cherniadev, N., Da, X., Ding, R., LinxiJimFan, Fang, Y., Fox, D., Hu, F., Huang, S., Jang, J., Jiang, Z., Kautz, J., Kundalia, K., Lao, L., Li, Z., Lin, Z., Lin, K., Liu, G., Llontop, E., Magne, L., Mandlekar, A., Narayan, A., Nasiriany, S., Reed, S., Tan, Y.L., Wang, G., Wang, Z., Wang, J., Wang, Q., Xiang, J., Xie, Y., Xu, Y., Xu, Z.T., Ye, S., Yu, Z., Zhang, A., Zhang, H., Zhao, Y., Zheng, R., Zhu, Y.: Gr00t n1: An open foundation model for generalist humanoid robots. ArXiv **abs/2503.14734** (2025)
25. Pertsch, K., Stachowicz, K., Ichter, B., Driess, D., Nair, S., Vuong, Q., Mees, O., Finn, C., Levine, S.: Fast: Efficient action tokenization for vision-language-action models. arXiv preprint arXiv:2501.09747 (2025)
26. Shridhar, M., Manuelli, L., Fox, D.: Cliport: What and where pathways for robotic manipulation. ArXiv **abs/2109.12098** (2021)
27. Sun, P., Jiang, Y., Chen, S., Zhang, S., Peng, B., Luo, P., Yuan, Z.: Autoregressive model beats diffusion: Llama for scalable image generation. arXiv preprint arXiv:2406.06525 (2024)
28. Team, O.M., Ghosh, D., Walke, H.R., Pertsch, K., Black, K., Mees, O., Dasari, S., Hejna, J., Kreiman, T., Xu, C., Luo, J., Tan, Y.L., Sanketi, P.R., Vuong, Q., Xiao, T., Sadigh, D., Finn, C., Levine, S.: Octo: An open-source generalist robot policy. ArXiv **abs/2405.12213** (2024)
29. Tian, K., Jiang, Y., Yuan, Z., Peng, B., Wang, L.: Visual autoregressive modeling: Scalable image generation via next-scale prediction. Advances in neural information processing systems **37**, 84839–84865 (2024)
30. Van Den Oord, A., Vinyals, O., et al.: Neural discrete representation learning. Advances in neural information processing systems **30** (2017)
31. Walke, H.R., Black, K., Zhao, T.Z., Vuong, Q., Zheng, C., Hansen-Estruch, P., He, A.W., Myers, V., Kim, M.J., Du, M., et al.: Bridgedata v2: A dataset for robot learning at scale. In: Conference on Robot Learning. pp. 1723–1736. PMLR (2023)
32. Wang, L., Chen, X., Zhao, J., He, K.: Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. ArXiv **abs/2409.20537** (2024)
33. Wang, Y., Zhu, H., Liu, M., Yang, J., Fang, H.S., He, T.: Vq-vla: Improving vision-language-action models via scaling vector-quantized action tokenizers. ArXiv **abs/2507.01016** (2025), https://api.semanticscholar.org/CorpusID:280145409
34. Wu, H., Jing, Y., Cheang, C.H., Chen, G., Xu, J., Li, X., Liu, M., Li, H., Kong, T.: Unleashing large-scale video generative pre-training for visual robot manipulation. ArXiv **abs/2312.13139** (2023)
35. Yang, Q.A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Dong, G., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Su, Y., Zhang, Y.C., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., Qiu, Z., Quan, S., Wang, Z.: Qwen2.5 technical report. ArXiv **abs/2412.15115** (2024)
36. Zhai, X., Mustafa, B., Kolesnikov, A., Beyer, L.: Sigmoid loss for language image pre-training. 2023 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 11941–11952 (2023)

37. Zhang, X., Zhang, D., Li, S., Zhou, Y., Qiu, X.: Speechtokenizer: Unified speech tokenizer for speech language models. In: The Twelfth International Conference on Learning Representations (2024)
38. Zhao, T.Z., Kumar, V., Levine, S., Finn, C.: Learning fine-grained bimanual manipulation with low-cost hardware. arXiv preprint arXiv:2304.13705 (2023)
39. Zitkovich, B., Yu, T., Xu, S., Xu, P., Xiao, T., Xia, F., Wu, J., Wohlhart, P., Welker, S., Wahid, A., et al.: Rt-2: Vision-language-action models transfer web knowledge to robotic control. In: Conference on Robot Learning. pp. 2165–2183. PMLR (2023)