MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

LEAP-VLA: Latent-Enhanced Action Prototyping via Continuous Residual Latent Spaces for Vision-Language-Action Models

Yu, Bo-Yun; Peng, Kuan-Chuan; Hsieh, Jun-Wei

TR2026-127 September 09, 2026

Abstract

Vision-Language-Action (VLA) models have advanced rapidly, yet most gains still come from larger backbones, larger embodied datasets, or expensive iterative decoders, while action representation remains under-explored. We argue that action space design is a primary bottleneck and present LEAP (Latent-Enhanced Action Prototyping)-VLA, a two-stage framework that learns a structured action latent space from demonstrations via Multi-level Soft Residual Quantization (MSRQ), then trains a lightweight VLM-external aligner to predict actions by prototype-aware soft selection in that space. This design replaces discrete code assignment and multi-step denoising with fully differentiable single-pass latent prediction, preserving pretrained visionlanguage alignment without modifying the VLM. Empirically, LEAPVLA outperforms the state-of-the-art methods with substantially fewer trainable parameters and no embodied pretraining, showing that a wellstructured continuous action latent space can offset model scale and data requirements.

European Conference on Computer Vision (ECCV) 2026

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

LEAP-VLA: Latent-Enhanced Action Prototyping via Continuous Residual Latent Spaces for Vision-Language-Action Models

Bo-Yun Yu[1], Kuan-Chuan Peng[2], and Jun-Wei Hsieh[1*]

[1] College of Artificial Intelligence, National Yang Ming Chiao Tung University, Tainan, Taiwan
[2] Mitsubishi Electric Research Laboratories, Cambridge, MA, USA
*Corresponding author: jwhsieh@nycu.edu.tw

Abstract. Vision-Language-Action (VLA) models have advanced rapidly, yet most gains still come from larger backbones, larger embodied datasets, or expensive iterative decoders, while action representation remains under-explored. We argue that action space design is a primary bottleneck and present **LEAP (Latent-Enhanced Action Prototyping)-VLA**, a two-stage framework that learns a structured action latent space from demonstrations via Multi-level Soft Residual Quantization (MSRQ), then trains a lightweight VLM-external aligner to predict actions by prototype-aware soft selection in that space. This design replaces discrete code assignment and multi-step denoising with fully differentiable single-pass latent prediction, preserving pretrained vision-language alignment without modifying the VLM. Empirically, LEAP-VLA outperforms the state-of-the-art methods with substantially fewer trainable parameters and no embodied pretraining, showing that a well-structured continuous action latent space can offset model scale and data requirements.

Keywords: Vision-Language-Action Model · Robotic Manipulation · Imitation Learning · Vector Quantization

1 Introduction

Current Vision-Language-Action models often depend on large-scale embodied data (*e.g.*, Open X-Embodiment and DROID [17, 25]) and large model capacity. Since RT-2 and OpenVLA [19, 47], follow-up methods have mainly expanded decoding paradigms including autoregressive action tokenization [6, 20, 33, 44], diffusion and flow-matching heads [2, 3, 12], and other approaches [10, 28, 34, 35]. However, most still treat action representation as a secondary design choice: discrete tokenization is non-differentiable and prone to codebook collapse, raw-action regression lacks structural constraints, and diffusion requires iterative inference. As a result, the latest decoupled methods such as UniVLA [6] and UniACT [44] still continue to rely on discrete codebooks and large-scale pre-training. This motivates a key question: **How to learn a structured action**

latent space for VLAs so that vision-language-to-action becomes explicit alignment to meaningful action primitives, rather than implicit alignment achieved mainly through scale?

To address these limitations, we present **LEAP-VLA** (Fig. 1), a two-stage framework that performs action generation through latent-space prototype prediction. Similar to latent diffusion moving image generation from pixels to latent variables [29], LEAP-VLA moves action prediction from raw action space to a pre-learned latent space, providing high-fidelity reconstruction through a frozen decoder, a compact zero-centered regression target, and temporal coherence from trajectory-level encoding. Concretely, Stage 1 learns a structured latent space from demonstrations with Multi-level Soft Residual Quantization (MSRQ), and Stage 2 trains a lightweight VLM-external aligner to predict in that frozen space via single-pass soft prototype selection. With strong results, LEAP-VLA reaches 97.1% mean success rate on LIBERO and 4.28 mean sequence length on CALVIN with only a 1B backbone and no embodied pretraining. Our contributions are:

- **Continuous structured action latent space.** We introduce a decoupled VLA that learns a continuous coarse-to-fine latent action space with Multi-level Soft Residual Quantization (MSRQ), instead of discrete code assignment. In quantization experiments, MSRQ achieves the best reconstruction-quality/utilization trade-off and near-uniform codebook usage across levels.
- **Latent-space prototype prediction.** We formulate action generation as direct prediction over frozen latent prototypes using a lightweight VLM-external aligner and a Prototype-Aware Output Head, which produces full action chunks in one forward pass, removing iterative denoising at inference.
- **Evidence that action structure is a primary lever.** We show that improving action representation yields larger gains than scaling backbone size or pretraining data. On LIBERO-Spatial, moving from raw actions to latent targets gives +2.2% and adding prototype-aware prediction gives another +3.8%; overall, LEAP-VLA with a 1B backbone outperforms methods with 3–8.5× larger models.

2 Related Work

Vision-Language-Action Models. VLAs leverage pretrained vision-language models (VLMs) to produce robot actions conditioned on visual observations and language instructions. Existing approaches differ in how actions are represented and generated: (i) Discrete action tokenization methods such as OpenVLA [19], VQ-VLA [33], UniVLA [6], and UniACT [44] discretize actions into codebook indices, predicted either autoregressively or via discrete selection such as Gumbel-Softmax. (ii) Diffusion and flow-matching approaches such as π_0 [3], π_0-FAST [27], GR00T N1 [2], and OpenHelix [12] attach iterative denoising heads to VLMs. (iii) Video-latent methods such as LAPA [35] and Moto [10] learn action representations from video frame deltas, enabling pretraining from

internet-scale videos without action annotations. (iv) Extensions include action-query adapters such as VLA-Adapter [34], explicit spatial modeling such as SpatialVLA [28], and reasoning/memory augmentation such as CoT-VLA [41] and MemoryVLA [31]. LEAP-VLA instead predicts in a continuous pre-learned action latent space via prototype-based soft selection, without VLM modification or iterative generation.

Action Representation Learning. How actions are represented fundamentally shapes a VLA's expressiveness and training efficiency. Discrete tokenization methods [19, 20, 33, 44, 47] map actions to codebook indices via hard assignment, but this is non-differentiable and requires the straight-through estimator [1], is prone to codebook collapse, and discards inter-prototype information. SoftVQ-VAE [8] replaces hard assignment with soft attention in image tokenization, but uses cosine similarity that discards action magnitude and is limited to single-level quantization. Direct regression methods [34, 42] predict raw action vectors without explicit structure, offering simplicity but lacking constraints on the output space. Diffusion-based methods [3, 11] model the action distribution via iterative denoising, achieving high expressiveness at the cost of multi-step inference; recent extensions apply diffusion in learned latent spaces [5, 13] or distill flow-matching into single-step generators [23]. ACT [42] uses a CVAE with an unstructured Gaussian prior, creating a train-test mismatch since $z{=}0$ is used at inference. Our MSRQ differs from all the above: it builds a continuous yet structured latent space via multi-level soft residual prototypes—fully differentiable, single-pass, and organized in a coarse-to-fine hierarchy—without requiring iterative denoising or distillation.

VLM-to-Action Architectures. A key design choice in VLAs is which VLM features are used for action generation. Most methods [6, 19, 33, 44] rely solely on last-layer features, discarding rich intermediate representations. GR00T N1 [2] found that using intermediate-layer embeddings yields faster inference and higher success rates than last-layer features. π_0 [3] feeds all VLM hidden states to its flow-matching head via cross-attention. VLA-Adapter [34] systematically studied this question and demonstrated that leveraging multi-layer features significantly outperforms using the last layer alone. LEAP-VLA builds on this insight: the Aligner extracts multi-layer visual and textual features from the VLM and fuses them outside the VLM via Aligner Attention. Unlike methods that inject action tokens or queries into the VLM, the VLM architecture remains entirely untouched with only lightweight LoRA adapters applied, preserving pretrained vision-language alignment.

3 Methodology

3.1 Problem Formulation

Action space. We represent a continuous action sequence as $\mathbf{a} \in \mathbb{R}^{T \times D_a}$, where T is the **action horizon** and D_a is the action dimensionality. We adopt end-effector (EEF) pose control with relative commands.

Observation space. At each decision step, the model receives a context tuple $\mathcal{X} = (\mathbf{I}, \mathbf{s}, \mathbf{w})$: (1) V time-synchronized RGB views $\mathbf{I} \in \mathbb{R}^{V\times 3\times H\times W}$ with image height H and width W, (2) the proprioceptive state $\mathbf{s} \in \mathbb{R}^{D_p}$ encoding the current EEF pose and gripper state, where D_p is the proprioception dimension, and (3) a natural language instruction $\mathbf{w} = (w_1, \ldots, w_{N_w})$ of N_w tokens specifying the task goal.

Problem setup. Given a dataset $\mathcal{D} = \{(\mathcal{X}_i, \mathbf{a}_i)\}_{i=1}^{M}$ of M expert demonstrations, we learn a policy π_θ that predicts action chunks:

$$\hat{\mathbf{a}} = \pi_\theta(\mathcal{X}) \in \mathbb{R}^{T\times D_a}. \tag{1}$$

Rather than regressing $\hat{\mathbf{a}}$ directly from raw perceptual inputs, LEAP-VLA first learns an action latent space and then predicts actions by navigating within it, as described next.

3.2 The LEAP-VLA Paradigm: Decoupling Action Structure Learning from Multimodal Grounding

Current VLA paradigms often suffer from **unstructured action learning**: whether actions are produced via direct regression [34, 42], tokenization [6, 33], or diffusion-based generation [2,3], models are typically forced to resolve complex action distributions in a high-dimensional space without an explicit, physically meaningful motion structure. LEAP-VLA addresses this limitation by **decoupling action structure learning from multimodal grounding** (Fig. 1):

- **Stage 1: Action structure learning (Sec. 3.3).** We distill expert motions into a structured action latent space using Multi-level Soft Residual Quantization (MSRQ). Learnable prototypes organize the latent space into a coarse-to-fine structure. Training uses only action data—no vision or language—capturing physical regularities and valid motion variations.
- **Stage 2: Aligning VLM representations to the action latent space (Sec. 3.4).** With the action latent space frozen, we train a lightweight Latent Action Aligner that extracts multi-layer visual and textual features from a frozen VLM, fuses them with proprioceptive state $\mathbf{s}$ via Aligner Attention, and predicts the action latent $\hat{\mathbf{z}}_q$ through a Prototype-Aware Output Head. The entire Stage 2 module operates **without embodied pretraining** and has the **fewest trainable parameters** among comparable methods, as it is external to the VLM with only LoRA adapters.

3.3 Stage 1: Action Structure Learning via MSRQ

Stage 1 learns a structured action latent space from expert demonstrations using only action sequences $\mathbf{a} \in \mathbb{R}^{T\times D_a}$; that is, no vision or language input is required (Fig. 2). The pipeline consists of three components: an **ActionEncoder** $\mathcal{E}$ that maps raw action trajectories to continuous latent representations $\mathbf{z}_e \in \mathbb{R}^{T\times D_c}$, where D_c is the code dimension, an **MSRQ quantizer** $\mathcal{Q}$ that transforms $\mathbf{z}_e$ into

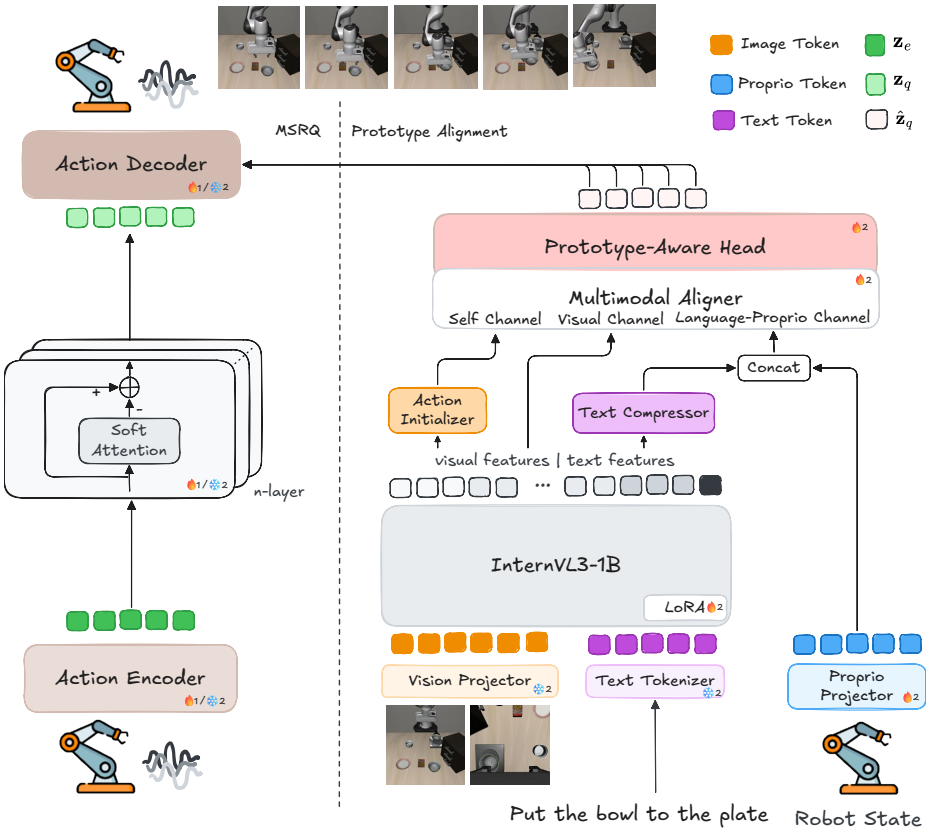

Fig. 1: Overview of the LEAP-VLA pipeline. Stage 1 learns an action latent space via MSRQ, where prototypes organize it into a coarse-to-fine structure. Stage 2 trains the Aligner to predict within the frozen latent space conditioned on vision, language, and proprioception.

structured latent codes $\mathbf{z}_q \in \mathbb{R}^{T \times D_c}$ via soft attention over learnable prototypes, and an **ActionDecoder** $\mathcal{D}$ that reconstructs actions $\hat{\mathbf{a}} = \mathcal{D}(\mathbf{z}_q)$. Both encoder and decoder use a hybrid **Bi**directional **C**NN–**T**ransformer (**BiCT**) architecture (detailed in Sec. 3.3). Only the learned prototypes and decoder are retained after Stage 1; the encoder is discarded. The quantized codes $\mathbf{z}_q$ serve as the constrained prediction target in Stage 2.

MSRQ: Multi-level Soft Residual Quantization. We propose **Multi-level Soft Residual Quantization (MSRQ)**, a fully differentiable quantization mechanism that builds a structured action latent space using soft attention over learnable prototypes. Unlike discrete action tokenizers [19, 20, 33] that rely on non-differentiable hard codebook assignment and suffer from codebook collapse, MSRQ employs continuous soft selection, enabling end-to-end gradient flow while maintaining a multi-scale residual structure.

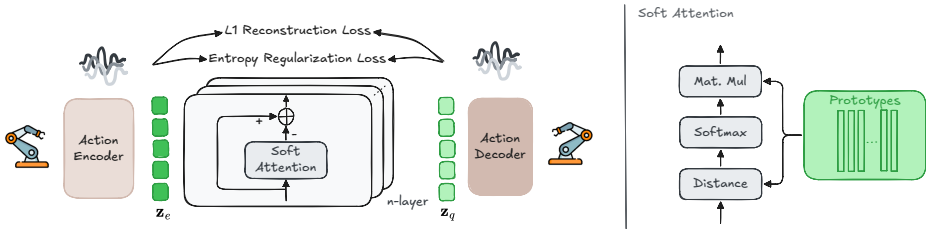

Fig. 2: Stage 1: Action Structure Learning via MSRQ.

Similarly to SoftVQ-VAE [8], the core idea is to allow each latent code to adaptively aggregate multiple prototypes from a learnable codebook via soft attention, rather than a hard one-hot selection. MSRQ uses L quantization levels. Each level $l \in \{1, \ldots, L\}$ maintains K learnable prototypes $\mathbf{P}^{(l)} \in \mathbb{R}^{K \times D_c}$ where D_c is the latent code dimension, a learnable temperature $\tau^{(l)} > 0$, and a learnable scale $s^{(l)} > 0$.

Given an encoder output $\mathbf{z}_e \in \mathbb{R}^{T \times D_c}$, MSRQ performs coarse-to-fine residual quantization across L levels. The process begins by initializing the residual $\mathbf{r}^{(0)}$ to the encoder output, which represents the full signal to be quantized, *i.e.*, $\mathbf{r}^{(0)} = \mathbf{z}_e$. At each level l, MSRQ computes a **soft assignment** $\boldsymbol{\alpha}^{(l)} \in \mathbb{R}^{T \times K}$ over the K prototypes using squared ℓ_2 distance, which preserves action magnitude and naturally generalizes VQ's nearest-neighbor assignment to the soft regime:

$$\boldsymbol{\alpha}^{(l)} = \mathrm{softmax}\left(-\frac{\|\mathbf{r}^{(l-1)} - \mathbf{P}^{(l)}\|_2^2}{\tau^{(l)}} \right), \tag{2}$$

where $\tau^{(l)}$ is a learnable temperature. The level's output $\Delta\mathbf{z}^{(l)}$ is the **weighted combination** of prototypes, scaled by a learnable factor $s^{(l)}$:

$$\Delta\mathbf{z}^{(l)} = s^{(l)} \cdot \left(\boldsymbol{\alpha}^{(l)} \mathbf{P}^{(l)} \right). \tag{3}$$

In practice, both $\tau^{(l)}$ and $s^{(l)}$ converge to monotonically decreasing patterns across levels, confirming that the multi-level structure captures a meaningful coarse-to-fine hierarchy. Finally, the **residual** $\mathbf{r}^{(l)}$ is updated by subtracting this level's contribution, so the next level operates on the remaining error:

$$\mathbf{r}^{(l)} = \mathbf{r}^{(l-1)} - \Delta\mathbf{z}^{(l)}. \tag{4}$$

The final quantized representation aggregates all levels:

$$\mathbf{z}_q = \sum_{l=1}^{L} \Delta\mathbf{z}^{(l)} \in \mathbb{R}^{T \times D_c}. \tag{5}$$

This formulation is **fully differentiable**: gradients flow through the softmax to both the encoder and prototypes without the straight-through estimator, and the resulting codes $\mathbf{z}_q$ are continuous vectors rather than discrete indices. Empirically, the latent space exhibits compact, zero-centered, unimodal marginal

distributions—arising naturally from the soft weighted average without explicit distributional constraints such as KL regularization [42], providing a well-conditioned regression target for Stage 2.

Action Encoder and Decoder. Action trajectories exhibit both local and global temporal structure: adjacent timesteps are strongly correlated due to smooth motion dynamics, while the overall trajectory must be globally coherent across multi-step sequences such as reach-then-grasp. To preserve both levels of temporal structure, we adopt the BiCT architecture for both the ActionEncoder and ActionDecoder: 1D convolutions capture local patterns, while bidirectional Transformers model global dependencies across the full action horizon.

The ActionEncoder maps $\mathbf{a} \in \mathbb{R}^{T \times D_a}$ to $\mathbf{z}_e \in \mathbb{R}^{T \times D_c}$:

$$\mathbf{z}_e = \gamma \cdot \mathrm{Linear}\big(\mathrm{BiTransformer}\big(\mathrm{CNN}_{1D}^{\times 2}(\mathbf{a}) + \mathbf{E}_{\mathrm{pos}}^{\mathrm{enc}}\big)\big), \tag{6}$$

where $\mathrm{CNN}_{1D}^{\times 2}$ stacks two 1D convolution blocks (kernel size 3, GroupNorm, GELU) that project $D_a \to D_h \to D_h$ with D_h being the hidden dimension; $\mathbf{E}_{\mathrm{pos}}^{\mathrm{enc}} \in \mathbb{R}^{T \times D_h}$ are learnable positional embeddings; BiTransformer is a 4-layer bidirectional Transformer with 8 attention heads; Linear projects $D_h \to D_c$; and γ is a learnable output scale.

The ActionDecoder mirrors the encoder in reverse order and reconstructs actions from $\mathbf{z}_q$:

$$\hat{\mathbf{a}} = \mathrm{Conv}_{\mathrm{out}}\big(\mathrm{CNN}_{1\mathrm{D}}\big(\mathrm{BiTransformer}\big(\mathrm{Linear}(\mathbf{z}_q) + \mathbf{E}_{\mathrm{pos}}^{\mathrm{dec}}\big)\big)\big), \tag{7}$$

where Linear projects from D_c back to D_h, the BiTransformer and $\mathrm{CNN}_{1\mathrm{D}}$ share the same configuration as the encoder, and $\mathrm{Conv}_{\mathrm{out}}$ is a final 1D convolution that maps from D_h to D_a without normalization or activation to produce unconstrained action predictions.

Training Objective. The encoder $\mathcal{E}$, MSRQ quantizer $\mathcal{Q}$, and decoder $\mathcal{D}$ are jointly trained with two losses that ensure faithful action reconstruction and maximum prototype utilization.

Reconstruction Loss. We adopt L1 loss for reconstruction, as L2 disproportionately penalizes large-magnitude commands (*e.g.*, gripper open/close) and biases toward over-smoothed trajectories:

$$\mathcal{L}_{\mathrm{recon}} = \|\hat{\mathbf{a}} - \mathbf{a}\|_1 . \tag{8}$$

Entropy Regularization Loss. To prevent codebook collapse, we regularize the batch-averaged prototype usage toward uniformity:

$$\mathcal{L}_{\mathrm{entropy}} = \frac{1}{L} \sum_{l=1}^{L} \left(1 - \frac{H(\bar{\boldsymbol{\alpha}}^{(l)})}{\log K}\right), \tag{9}$$

where $\bar{\boldsymbol{\alpha}}^{(l)} = \frac{1}{BT} \sum_{b,t} \boldsymbol{\alpha}_{b,t}^{(l)}$ is the attention distribution averaged over batch size B and time steps T, $H(\bar{\boldsymbol{\alpha}}^{(l)}) = -\sum_{k=1}^{K} \bar{\alpha}_k^{(l)} \log \bar{\alpha}_k^{(l)}$ is the Shannon entropy, and $\frac{H(\cdot)}{\log K} \in [0, 1]$ uses $\log K$ as the normalization term for $H(\cdot)$, equaling 1 when all prototypes are equally used and 0 when usage collapses to a single prototype.

The total Stage 1 objective is:

$$\mathcal{L}_{\text{Stage1}} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{entropy}} \,. \tag{10}$$

3.4 Stage 2: Latent Action Aligner

With the action latent space frozen, Stage 2 trains a **Latent Action Aligner** that maps VLM representations onto the learned latent space, conditioned on vision, language, and proprioception (Fig. 1). The Aligner operates **externally** to the VLM—we freeze the VLM backbone and apply only lightweight LoRA [14] adapters, preserving the pretrained representations without structural modifications. The Aligner consists of four components: (1) a Text Compressor that extracts compact textual features, (2) an Action Initializer that constructs seed action representations from visual features, (3) a stack of Aligner Attention blocks that fuse all modalities into refined action representations, and (4) a Prototype-Aware Output Head that maps the refined features onto the action latent space to produce the predicted latent code $\hat{\mathbf{z}}_q$.

Multi-Layer Feature Extraction. The VLM produces hidden states of dimension d from all N_L transformer layers. Following prior work [34] showing that multi-layer features significantly outperform last-layer-only extraction, we decompose each layer's hidden states by token position into per-layer **visual representations** $C_R \in \mathbb{R}^{B\times N_L\times N_v\times d}$ and **textual representations** $C_T \in \mathbb{R}^{B\times N_L\times N_t\times d}$, where N_v and N_t denote the number of visual patches and text tokens. Unlike methods that inject action tokens into the VLM, we leave the input sequence unmodified and extract features externally through a Text Compressor and an Action Initializer.

Text Compressor. Language instructions vary in length across tasks, yet the downstream Aligner requires fixed-size conditioning signals. We compress the variable-length textual representations C_T into a fixed set of k query vectors per layer via a **Text Compressor**—a learnable cross-attention module:

$$\hat{C}_T^{(l)} = \text{LN}\Big(\text{CrossAttn}\Big(Q_{\text{text}},\, C_T^{(l)}\Big)\Big) \in \mathbb{R}^{B\times k\times d}, \tag{11}$$

where $Q_{\text{text}} \in \mathbb{R}^{k\times d}$ is a set of learnable query parameters shared across all layers, CrossAttn denotes multi-head cross-attention with Q_{text} as queries and $C_T^{(l)}$ as keys/values, and LN is layer normalization. This explicit text-processing pathway preserves linguistic intent in a compact, fixed-size representation before fusion, rather than being diluted among the far more numerous visual tokens.

Action Initializer. The Aligner Attention blocks iteratively refine an action representation $\mathbf{x} \in \mathbb{R}^{B\times T\times d}$. Instead of learnable queries [7] or noise, $\mathbf{x}_0$ is initialized from visual features for a scene-grounded starting point. We first fuse multi-layer visual representations with learnable layer weights, and reshape/mean-pool across patches to produce T tokens matching the action horizon:

$$\mathbf{x}_0 = \text{MeanPool}\left(\text{Reshape}\left(\sum_{l=1}^{L}\lambda_l C_R^{(l)},\; T\times\frac{N_v}{T}\times d\right)\right) \in \mathbb{R}^{B\times T\times d}\,, \tag{12}$$

where λ_l are learnable layer weights, and $C_R^{(l)}$ denotes the visual representation from VLM layer l. This initialization bridges visual patches and action timesteps while preserving spatial structure.

Aligner Attention. The core of the Aligner is a stack of L attention blocks, one per VLM layer. Each block refines the action representation $\mathbf{x}$ through **3-way joint attention**—a single softmax over three heterogeneous key-value sources that we term **Aligner Attention**:

1. **Self channel** (K_s, V_s): key-values derived from $\mathbf{x}$ itself, enabling inter-timestep reasoning so that each action step can coordinate with others across the horizon.
2. **Language-proprioception channel** (K_a, V_a): key-values from the concatenation of compressed text $\hat{C}_T^{(l)}$ and projected proprioceptive state $\mathbf{p}$, providing task-goal conditioning and embodiment awareness.
3. **Visual channel** (K_v, V_v): key-values from $C_R^{(l)}$, grounding the action representation in the current scene observation.

Each source has independent key-value projections (no cross-channel sharing), and rotary position embeddings (RoPE) [32] is applied independently per source to encode intra-source positional structure without cross-source interference. At block l, per-channel attention logits are computed first:

$$L_s = QK_s^\top / \sqrt{d_h}, \tag{13}$$

$$L_a = QK_a^\top / \sqrt{d_h}, \tag{14}$$

$$L_v = \tanh(g) \cdot QK_v^\top / \sqrt{d_h}, \tag{15}$$

where W_q, W_{k_*}, W_{v_*}, and W_p are learnable projection matrices, $Q = W_q\mathbf{x}$ is the query projection from the action representation, $(K_s, V_s) = (W_{k_s}\mathbf{x}, W_{v_s}\mathbf{x})$ are the self key-values, $(K_a, V_a) = (W_{k_a}[\hat{C}_T^{(l)} \| \mathbf{p}], W_{v_a}[\hat{C}_T^{(l)} \| \mathbf{p}])$ are the language-proprioception key-values with $\mathbf{p} = W_p\mathbf{s}$ being the projected proprioceptive state, $(K_v, V_v) = (W_{k_v}C_R^{(l)}, W_{v_v}C_R^{(l)})$ are the visual key-values, $d_h = d/n_{\text{heads}}$ is the per-head dimension, and g is a learnable gating scalar initialized to zero. The three logit matrices are then concatenated along the token dimension and passed through a single softmax to produce the fused output:

$$\text{Attn} = \text{softmax}([\, L_s \parallel L_a \parallel L_v \,]) \cdot [\, V_s \parallel V_a \parallel V_v \,], \tag{16}$$

where $\|$ denotes concatenation.

Because visual tokens substantially outnumber the other sources, we modulate the visual logits with a learnable gate g initialized to zero ($\tanh(0)=0$), preventing early visual dominance and allowing the model to first establish task conditioning from language and proprioception; as training progresses, g is learned and $\tanh(g) \to 1$, progressively opening the visual channel. Each block ends with a residual connection followed by a feedforward network (LayerNorm $\to$ Linear $\to$ ReLU):

$$\mathbf{x} \leftarrow \text{FFN}(\text{Attn}(\mathbf{x}) + \mathbf{x}) \,. \tag{17}$$

Prototype-Aware Output Head. After the final Aligner block, we apply layer normalization to obtain the refined features $\mathbf{h} \in \mathbb{R}^{B\times T\times d}$. Rather than directly regressing $\hat{\mathbf{z}}_q$ via a linear projection, which risks producing codes outside the learned latent space, we design a **Prototype-Aware Output Head** that constrains predictions to the latent space by construction.

For each residual level l, a learned projection maps $\mathbf{h}$ to a distribution over the K frozen prototypes, and the level's output is the weighted combination:

$$\Delta\hat{\mathbf{z}}_q^{(l)} = s^{(l)} \cdot \mathrm{softmax}\left(\frac{\mathrm{Linear}_l(\mathbf{h})}{\tau^{(l)}}\right)\mathbf{P}^{(l)}, \tag{18}$$

where $\mathbf{P}^{(l)}$, $\tau^{(l)}$, and $s^{(l)}$ are the frozen prototypes, temperature, and scale from Stage 1. Freezing these parameters preserves the latent manifold geometry learned in Stage 1; the Aligner learns to navigate this fixed space rather than deforming it. The final prediction accumulates residuals across all levels:

$$\hat{\mathbf{z}}_q = \sum_{l=1}^{R} \Delta\hat{\mathbf{z}}_q^{(l)} \in \mathbb{R}^{B\times T\times D_c}\,. \tag{19}$$

This output head performs soft selection over prototypes using the same residual structure as MSRQ. Since $\hat{\mathbf{z}}_q$ is always a weighted combination of Stage 1 prototypes, it is guaranteed to lie within the learned action latent space. The predicted latent is then decoded into executable actions via the frozen Stage 1 decoder: $\hat{\mathbf{a}} = \mathcal{D}(\hat{\mathbf{z}}_q)$.

Training Objectives. Stage 2 trains the Aligner with three complementary losses, supervised in both the latent space and the action space. Let $\mathbf{z}_q = \mathcal{Q}(\mathcal{E}(\mathbf{a}))$ denote the ground-truth latent code obtained by encoding expert actions through the frozen Stage 1 encoder and quantizer.

Proximity Loss. The MSE loss minimizes the Euclidean distance between the predicted and target latent codes in the action latent space:

$$\mathcal{L}_{\text{prox}} = \|\hat{\mathbf{z}}_q - \mathbf{z}_q\|_2^2\,. \tag{20}$$

Directional Consistency Loss. Beyond distance, we enforce that the predicted code points in the same direction as the target at each timestep via cosine similarity:

$$\mathcal{L}_{\text{dir}} = 1 - \frac{1}{T}\sum_{t=1}^{T}\cos(\hat{\mathbf{z}}_{q,t},\, \mathbf{z}_{q,t})\,. \tag{21}$$

Together, these two terms form the **Latent Alignment Loss** $\mathcal{L}_{\text{align}} = \mathcal{L}_{\text{prox}} + \lambda_{\text{dir}}\,\mathcal{L}_{\text{dir}}$ with $\lambda_{\text{dir}} = 0.5$ to balance the scale difference between MSE and cosine terms, which jointly ensures proximity and directional consistency in the action latent space.

Action Reconstruction Loss. The predicted latent is decoded through the frozen Stage 1 decoder, and the L1 loss provides end-to-end supervision in the action space:

$$\mathcal{L}_{\text{action}} = \|\hat{\mathbf{a}} - \mathbf{a}\|_1\,. \tag{22}$$

This complements the latent-space losses by ensuring that the final decoded actions are accurate, even when multiple latent codes could produce similar latent-space distances. The total Stage 2 objective is:

$$\mathcal{L}_{\text{Stage2}} = \mathcal{L}_{\text{align}} + \mathcal{L}_{\text{action}}. \tag{23}$$

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate on two benchmarks: **LIBERO** [22] and **CALVIN** [24]. **LIBERO** [22] contains four task suites of increasing difficulty (Spatial, Object, Goal, Long), each with 10 tasks and 50 demonstrations per task; we run 50 rollouts per task. **CALVIN** [24] is a long-horizon language-conditioned benchmark; we use the ABC→D protocol (train on A/B/C, evaluate on held-out D) with 1000 evaluation sequences, each comprising 5 chained subtasks. We report success rate (%) on both benchmarks; for CALVIN we additionally report average completed sequence length.

Baselines. We compare against representative paradigms, including autoregressive token prediction, flow matching, chain-of-thought reasoning, and large-scale multi-task pretraining, with all method-level details reported in Tabs. 4 and 5. All baselines are peer-reviewed publications.

Implementation details. We use InternVL3-1B [45] as the VLM backbone with LoRA adapters of rank 32. The observation input consists of two camera views (third-person and wrist-mounted), and the action horizon is T=8. Stage 1 trains the MSRQ encoder-decoder with K=256 prototypes, R=4 residual levels, and code dimension D_c=32 for 10K steps on action-only data. The choice of K=256 follows the codebook size widely adopted in prior action tokenization methods [6, 19, 20, 33]; ablations on codebook size K and residual levels L are provided in the supplementary material. Stage 2 trains the Aligner for 100K steps with batch size 64, learning rate 1×10^{-4}, and cosine schedule. The total trainable parameter count is ∼100M across LoRA adapters and the Aligner, with a frozen VLM backbone of ∼0.9B. Stage 1 is trained on 4× NVIDIA RTX 3090 Ti GPUs; Stage 2 is trained on 4× NVIDIA H100 GPUs.

4.2 Stage 1: Action Representation Quality

We first validate that MSRQ learns a high-quality action latent space before connecting it to the VLM. All Stage 1 experiments in the main paper use LIBERO-Spatial as the primary testbed and CALVIN ABC→D for cross-environment verification; additional Stage 1 results on the remaining LIBERO suites are provided in the supplementary material.

Quantization Method. We report codebook utilization via the entropy ratio $H(\cdot)/\log K\times100\%$ and the effective count $\exp(H(\cdot))$, *i.e.*, the equivalent number of uniformly used prototypes. Tab. 1 compares MSRQ against 11 baselines, all using our BiCT encoder/decoder with $K = 256$ prototypes. Multi-level RVQ

Table 1: Quantization method comparison on LIBERO-Spatial and CALVIN ABC→D with fixed BiCT encoder/decoder. All methods use $K = 256$, 10K steps. †Entropy ratio $H(\cdot)/\log K \times 100\%$: entropy-based utilization. ‡Effective count $\exp(H(\cdot))$: equivalent uniform codebook size ($\max = 256$). The relative ranking is consistent across both benchmarks. Best results are denoted in **bold**, and second best are underlined.

Single-Level					Multi-Level (4L)				
Method	**Venue**	**Ent.**†	**Eff.**‡	**MAE↓** ($\times 10^{-3}$)	**Method**	**Venue**	**Ent.**†	**Eff.**‡	**MAE↓** ($\times 10^{-3}$)
LIBERO-Spatial									
VQ-VAE (grad.) [26]	NeurIPS'17	58.3%	25.3	86.6	SimRVQ	–	90.0%	164.4	7.7
VQ-VAE (EMA) [26]	NeurIPS'17	54.6%	20.6	77.8	OptRVQ	–	97.1%	217.8	37.7
SimVQ [46]	ICCV'25	69.9%	48.2	55.8	RVQ-EMA [37]	TASLP'22	81.7%	122.6	9.1
OptVQ [38]	arXiv'24	98.6%	237.3	79.4	RVQ-Commit	–	81.5%	129.6	9.0
SoftVQ [8]	CVPR'25	90.6%	151.6	4.9	SoftRVQ	–	90.4%	151.5	6.0
MSRQ-single (ours)	–	**99.8%**	**253.5**	2.5	**MSRQ (ours)**	–	99.6%	250.8	**1.5**
CALVIN ABC→D									
VQ-VAE (grad.) [26]	NeurIPS'17	59.3%	26.8	62.3	SimRVQ	–	91.9%	175.2	7.6
VQ-VAE (EMA) [26]	NeurIPS'17	58.5%	25.6	48.0	OptRVQ	–	97.6%	224.8	43.3
SimVQ [46]	ICCV'25	74.8%	63.4	41.4	RVQ-EMA [37]	TASLP'22	84.5%	150.5	8.5
OptVQ [38]	arXiv'24	98.4%	234.5	77.7	RVQ-Commit	–	86.3%	147.3	9.0
SoftVQ [8]	CVPR'25	93.0%	173.8	2.7	SoftRVQ	–	92.9%	173.6	3.6
MSRQ-single (ours)	–	**99.6%**	**250.8**	1.2	**MSRQ (ours)**	–	99.4%	247.9	**1.1**

Table 2: Per-level utilization comparison of multi-level methods on LIBERO-Spatial. All methods use $K = 256$, 4 levels, same encoder/decoder, 10K steps. Ent.: entropy ratio. Eff.: effective count ($\max = 256$). Best results are denoted in **bold**.

	RVQ-Commit		SimRVQ		SoftRVQ		**MSRQ (ours)**	
Level	Ent.	Eff.	Ent.	Eff.	Ent.	Eff.	Ent.	Eff.
0 (coarse)	51.9%	17.8	74.1%	60.8	87.7%	129.1	**99.5%**	**248.6**
1	82.8%	98.4	91.7%	161.9	88.4%	134.6	**99.6%**	**250.6**
2	96.2%	207.0	97.0%	216.8	92.2%	165.8	**99.7%**	**251.6**
3 (fine)	95.1%	195.2	97.1%	218.3	93.3%	176.5	**99.7%**	**252.2**
Average	81.5%	129.6	90.0%	164.4	90.4%	151.5	**99.6%**	**250.8**
MAE↓ ($\times 10^{-3}$)	9.0		7.7		6.0		**1.5**	

variants are our extensions of their single-level counterparts to a 4-level residual hierarchy. MSRQ achieves the lowest MAE with near-perfect utilization (99.6%); even MSRQ-single outperforms all multi-level discrete methods, confirming that L2-based soft assignment is the dominant factor.

Per-Level Utilization. Tab. 2 shows utilization by residual level. Discrete RVQ methods exhibit severe imbalance—RVQ-Commit collapses to 51.9% at the coarse level 0—while SoftRVQ plateaus around 90%. MSRQ maintains ≥99.5% at every level, confirming uniform collapse prevention across the hierarchy.

Encoder/Decoder Architecture. Tab. 3 compares four encoder/decoder architectures with identical MSRQ settings. A causal Transformer degrades MAE to 71.9×10^{-3} and utilization to 76%, since unidirectional attention cannot leverage future context. BiCT achieves the lowest MAE of 1.5×10^{-3} with near-perfect utilization, confirming that combining local convolutions with bidirectional attention is optimal.

Table 3: Encoder/Decoder architecture comparison on LIBERO-Spatial with fixed MSRQ, $K = 256$, $R = 4$, 10K steps. Ent.: entropy ratio. Eff.: effective count (max = 256). Best results are denoted in **bold**.

Encoder/Decoder	Ent.	Eff.	MAE↓ ($\times 10^{-3}$)
MLP only	**100.0%**	**255.3**	15.4
CNN only	99.7%	251.8	8.5
CNN + Causal Transformer	76.0%	68.1	71.9
BiCT	99.6%	250.8	**1.5**

Table 4: LIBERO success rate (%) comparison. All baseline results are reported from their original publications. Methods are sorted by backbone parameter count (descending). Robot PT: pretrained on ✓ = large-scale robot data (OXE, DROID, *etc.*); △ = additional non-robot data (video, *etc.*); ✗ = VLM backbone only.

Model	Venue	Params (B)	Robot PT	Spatial	Object	Goal	Long	Avg
UD-VLA [9]	ICLR'26	8.5	△	94.1	95.7	91.2	89.6	92.7
OpenVLA [19]	CoRL'24	7	✓	84.7	88.4	79.2	53.7	76.5
UniVLA [6]	RSS'25	7	✓	96.5	96.8	95.6	92.0	95.2
CoT-VLA [41]	CVPR'25	7	✓	87.5	91.6	87.6	69.0	81.1
ThinkAct [16]	NeurIPS'25	7	✓	88.3	91.4	87.1	70.9	84.4
SpatialVLA [28]	RSS'25	4	✓	88.2	89.9	78.6	55.5	78.1
π_0 [3]	RSS'25	3	✓	96.8	98.8	95.8	85.2	94.2
π_0-FAST [27]	RSS'25	3	✓	96.4	96.8	88.6	60.2	85.5
Fast-ThinkAct [15]	CVPR'26	3	✓	92.0	97.2	90.2	79.4	89.7
LEAP-VLA (Ours)	–	1.0	✗	**98.4**	**99.4**	**97.6**	**93.0**	**97.1**

4.3 Stage 2: End-to-End Evaluation

Having established that MSRQ produces a high-fidelity, fully-utilized action latent space, we now evaluate the complete LEAP-VLA pipeline on closed-loop robotic manipulation.

LIBERO. Tab. 4 presents the main comparison. LEAP-VLA achieves **97.1%** average success rate—the highest among all peer-reviewed methods—with the best score on all four suites, while using the smallest backbone (1B) and fewest trainable parameters (∼100M).

Architecturally, LEAP-VLA predicts the full action chunk in a single forward pass without iterative denoising, resulting in a simpler inference pipeline compared to flow-matching or diffusion policies.

CALVIN ABC→D. We further evaluate cross-environment generalization on CALVIN ABC→D (Tab. 5), where the model must execute long-horizon task chains in a held-out environment with unseen scene layouts and lighting.

The CALVIN results provide strong support for our claim. Under the same VLM-only setting (✗), LEAP-VLA achieves the best score (4.28) with the smallest model (1B), surpassing VLM4VLA [39] (4.06) with 8.3× fewer parameters. LEAP-VLA also outperforms robot-pretrained methods, including OpenVLA-

Table 5: CALVIN ABC→D benchmark [24] comparison. Methods are sorted by backbone parameter count (descending). Best results in **bold**. Robot PT: pretrained on ✓ = large-scale robot data (OXE, DROID, *etc.*); △ = additional non-robot data (video, *etc.*); ✗ = VLM backbone only.

Model	Venue	Params (B)	Robot PT	Tasks completed in a row ↑					Avg. len ↑
				1	2	3	4	5	
VLM4VLA [39]	ICLR'26	8.3	✗	93.5	86.4	80.7	75.8	69.3	4.06
UniVLA [6]	RSS'25	7	✓	95.5	85.8	75.4	66.9	56.5	3.80
OpenVLA [19]	CoRL'24	7	✓	91.3	77.8	62.0	52.1	43.5	3.27
OpenVLA-OFT [18]	RSS'25	7	✓	96.3	89.1	82.4	75.8	66.5	4.10
VLAS [43]	ICLR'25	7	✗	87.2	64.2	40.9	28.1	19.6	2.40
LCB [30]	IROS'24	7	✗	73.6	50.2	28.5	16.0	9.9	1.78
DeeR [36]	NeurIPS'24	3	✗	86.2	70.1	51.8	41.5	30.4	2.82
RoboFlamingo [21]	ICLR'24	3	✗	82.4	61.9	46.6	33.1	23.5	2.48
UP-VLA [40]	ICML'25	1.3	△	92.8	86.5	81.5	76.9	69.9	4.08
SuSIE [4]	ICLR'24	1.3	✗	87.0	69.0	49.0	38.0	26.0	2.69
LEAP-VLA (Ours)	–	1.0	✗	**96.5**	**92.2**	**86.1**	**79.8**	**73.0**	**4.28**

Table 6: Action representation ablation on LIBERO-Spatial. All variants use single-pass direct regression with the same VLM backbone and Aligner; only the prediction target and output head differ.

Prediction Target	Output Head	Spatial (%)
Raw action	Linear	92.4
Latent $\mathbf{z}_q$	Linear	94.6
Latent $\mathbf{z}_q$	**Prototype-Aware**	**98.4**

OFT [18] (4.10), despite using no external robot data. This highlights action-space structure as a key driver of robustness and data efficiency.

Action representation ablation. Tab. 6 compares three variants sharing the same VLM backbone, Aligner, and training recipe, differing only in prediction target and output head. Moving from raw actions to the latent target yields +2.2% (92.4→94.6%), and replacing the linear head with the Prototype-Aware Output Head adds a further +3.8% (94.6→98.4%). These gains show that structured latent-space prediction is the key driver of performance.

5 Conclusion

We presented LEAP-VLA, a two-stage VLA framework that turns action prediction into structured latent-space alignment: Stage 1 learns a continuous coarse-to-fine latent action space with MSRQ, and Stage 2 aligns multimodal features to this frozen space with a lightweight external aligner for single-pass inference.

Experiments validate this design. On LIBERO-Spatial, moving from raw actions to latent targets improves success by +2.2% (92.4→94.6%), and the Prototype-Aware Output Head adds +3.8% (94.6→98.4%). LEAP-VLA achieves strong performance on both LIBERO and CALVIN with a 1B backbone and no

embodied pretraining, outperforming methods using 3–8.5× larger models; on CALVIN, it also surpasses several OXE-pretrained baselines.

Limitations and future work. Our evaluation is currently limited to simulation, and performance remains bounded by VLM capacity. Future work will validate LEAP-VLA on physical robots and study cross-embodiment transfer via a shared MSRQ latent space.

References

1. Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013)
2. Bjorck, J., Castañeda, F., Cherniadev, N., Da, X., Ding, R., Fan, L., Fang, Y., Fox, D., Hu, F., Huang, S., et al.: GR00T N1: An open foundation model for generalist humanoid robots. arXiv preprint arXiv:2503.14734 (2025)
3. Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., et al.: π_0: A vision-language-action flow model for general robot control. In: Robotics: Science and Systems (RSS) (2025). https://doi.org/10.15607/RSS.2025.XXI.010
4. Black, K., Nakamoto, M., Atreya, P., Walke, H., Finn, C., Kumar, A., Levine, S.: Zero-shot robotic manipulation with pre-trained image-editing diffusion models. In: International Conference on Learning Representations (ICLR) (2024)
5. Bode, J., Memmesheimer, R., Behnke, S.: El3dd: Extended latent 3d diffusion for language conditioned multitask manipulation. arXiv preprint arXiv:2511.13312 (2025)
6. 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. In: Robotics: Science and Systems (RSS) (2025). https://doi.org/10.15607/RSS.2025.XXI.014
7. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European Conference on Computer Vision (ECCV). pp. 213–229 (2020). https://doi.org/10.1007/978-3-030-58452-8_13
8. Chen, H., Wang, Z., Li, X., Sun, X., Chen, F., Liu, J., Wang, J., Raj, B., Liu, Z., Barsoum, E.: SoftVQ-VAE: Efficient 1-dimensional continuous tokenizer. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 28358–28370 (2025). https://doi.org/10.1109/CVPR52734.2025.02641
9. Chen, J., Song, W., Ding, P., Zhou, Z., Zhao, H., Tang, F., Wang, D., Li, H.: Unified diffusion VLA: Vision-language-action model via joint discrete denoising diffusion process. In: International Conference on Learning Representations (ICLR) (2026)
10. Chen, Y., Ge, Y., Tang, W., Li, Y., Ge, Y., Ding, M., Shan, Y., Liu, X.: Moto: Latent motion token as the bridging language for learning robot manipulation from videos. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 19752–19763 (2025). https://doi.org/10.1109/ICCV51701.2025.01837
11. 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 (RSS) (2023). https://doi.org/10.15607/RSS.2023.XIX.026
12. Cui, C., Ding, P., Song, W., Bai, S., Tong, X., Ge, Z., Suo, R., Zhou, W., Liu, Y., Jia, B., et al.: OpenHelix: A short survey, empirical analysis, and open-source

dual-system VLA model for robotic manipulation. arXiv preprint arXiv:2505.03912 (2025)
13. Hou, Z., Zhang, T., Xiong, Y., Duan, H., Pu, H., Tong, R., Zhao, C., Zhu, X., Qiao, Y., Dai, J., Chen, Y.: Dita: Scaling diffusion transformer for generalist vision-language-action policy. In: IEEE/CVF International Conference on Computer Vision (ICCV). pp. 7686–7697 (2025). https://doi.org/10.1109/ICCV51701.2025.00721
14. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: LoRA: Low-rank adaptation of large language models. In: International Conference on Learning Representations (ICLR) (2022)
15. Huang, C.P., Man, Y., Yu, Z., Chen, M.H., Kautz, J., Wang, Y.C.F., Yang, F.E.: Fast-thinkact: Efficient vision-language-action reasoning via verbalizable latent planning. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5070–5081 (2026)
16. Huang, C.P., Wu, Y.H., Chen, M.H., Wang, Y.C.F., Yang, F.E.: Thinkact: Vision-language-action reasoning via reinforced visual latent planning. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 38 (2025)
17. Khazatsky, A., Pertsch, K., Nair, S., Balakrishna, A., Dasari, S., Karamcheti, S., Nasiriany, S., Srirama, M.K., Chen, L.Y., Ellis, K., et al.: DROID: A large-scale in-the-wild robot manipulation dataset. In: Robotics: Science and Systems (RSS) (2024). https://doi.org/10.15607/RSS.2024.XX.120
18. Kim, M.J., Finn, C., Liang, P.: Fine-tuning vision-language-action models: Optimizing speed and success. In: Robotics: Science and Systems (RSS) (2025). https://doi.org/10.15607/RSS.2025.XXI.017
19. Kim, M.J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E.P., 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. In: Conference on Robot Learning (CoRL) (2024)
20. Lee, S., Wang, Y., Etukuru, H., Kim, H.J., Shafiullah, N.M.M., Pinto, L.: Behavior generation with latent actions. In: International Conference on Machine Learning (ICML) (2024)
21. Li, X., Liu, M., Zhang, H., Yu, C., Xu, J., Wu, H., Cheang, C., Jing, Y., Zhang, W., Liu, H., et al.: Vision-language foundation models as effective robot imitators. In: International Conference on Learning Representations (ICLR) (2024)
22. Liu, B., Zhu, Y., Gao, C., Feng, Y., Liu, Q., Zhu, Y., Stone, P.: LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In: Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track (2023)
23. Liu, S., Li, B., Ma, K., Wu, L., Tan, H., Ouyang, X., Su, H., Zhu, J.: Rdt2: Exploring the scaling limit of umi data towards zero-shot cross-embodiment generalization. arXiv preprint arXiv:2602.03310 (2026)
24. Mees, O., Hermann, L., Rosete-Beas, E., Burgard, W.: CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. IEEE Robotics and Automation Letters **7**(3), 7327–7334 (2022). https://doi.org/10.1109/LRA.2022.3180108
25. O'Neill, A., Rehman, A., Maddukuri, A., Gupta, A., Padalkar, A., Lee, A., Pooley, A., Gupta, A., Mandlekar, A., Jain, A., et al.: Open X-Embodiment: Robotic learning datasets and RT-X models. In: IEEE International Conference on Robotics and Automation (ICRA). pp. 6892–6903 (2024). https://doi.org/10.1109/ICRA57147.2024.10611477

26. van den Oord, A., Vinyals, O., Kavukcuoglu, K.: Neural discrete representation learning. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 6306–6315 (2017)
27. 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. In: Robotics: Science and Systems (RSS) (2025). https://doi.org/10.15607/RSS.2025.XXI.012
28. Qu, D., Song, H., Chen, Q., Yao, Y., Ye, X., Gu, J., Wang, Z., Ding, Y., Zhao, B., Wang, D., Li, X.: SpatialVLA: Exploring spatial representations for visual-language-action models. In: Robotics: Science and Systems (RSS) (2025). https://doi.org/10.15607/RSS.2025.XXI.011
29. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2022). https://doi.org/10.1109/CVPR52688.2022.01042
30. Shentu, Y., Wu, P., Rajeswaran, A., Abbeel, P.: From LLMs to actions: Latent codes as bridges in hierarchical robot control. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 8539–8546 (2024). https://doi.org/10.1109/IROS58592.2024.10801683
31. Shi, H., Xie, B., Liu, Y., Sun, L., Liu, F., Wang, T., Zhou, E., Fan, H., Zhang, X., Huang, G.: MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In: International Conference on Learning Representations (ICLR) (2026)
32. Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., Liu, Y.: RoFormer: Enhanced transformer with rotary position embedding. Neurocomputing **568**, 127063 (2024). https://doi.org/10.1016/j.neucom.2023.127063
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. In: IEEE/CVF International Conference on Computer Vision (ICCV). pp. 11089–11099 (2025). https://doi.org/10.1109/ICCV51701.2025.01032
34. Wang, Y., Ding, P., Li, L., Cui, C., Ge, Z., Tong, X., Song, W., Zhao, H., Zhao, W., Hou, P., et al.: VLA-adapter: An effective paradigm for tiny-scale vision-language-action model. In: Proceedings of the AAAI conference on artificial intelligence. vol. 40, pp. 18638–18646 (2026). https://doi.org/10.1609/aaai.v40i22.38931
35. Ye, S., Jang, J., Jeon, B., Joo, S.J., Yang, J., Peng, B., Mandlekar, A., Tan, R., Chao, Y.W., Lin, B.Y., et al.: Latent action pretraining from videos. In: International Conference on Learning Representations (ICLR) (2025)
36. Yue, Y., Wang, Y., Kang, B., Han, Y., Wang, S., Song, S., Feng, J., Huang, G.: DeeR-VLA: Dynamic inference of multimodal large language models for efficient robot execution. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 37, pp. 56619–56643 (2024)
37. Zeghidour, N., Luebs, A., Omran, A., Skoglund, J., Tagliasacchi, M.: SoundStream: An end-to-end neural audio codec. IEEE/ACM Transactions on Audio, Speech, and Language Processing **30**, 495–507 (2022). https://doi.org/10.1109/TASLP.2021.3129994
38. Zhang, B., Zheng, W., Zhou, J., Lu, J.: Preventing local pitfalls in vector quantization via optimal transport. arXiv preprint arXiv:2412.15195 (2024)
39. Zhang, J., Chen, X., Wang, Q., Li, M., Guo, Y., Hu, Y., Zhang, J., Bai, S., Lin, J., Chen, J.: VLM4VLA: Revisiting vision-language-models in vision-language-action models. In: International Conference on Learning Representations (ICLR) (2026)

40. Zhang, J., Guo, Y., Hu, Y., Chen, X., Zhu, X., Chen, J.: UP-VLA: A unified understanding and prediction model for embodied agent. In: International Conference on Machine Learning (ICML) (2025)
41. Zhao, Q., Lu, Y., Kim, M.J., Fu, Z., Zhang, Z., Wu, Y., Li, Z., Ma, Q., Han, S., Finn, C., et al.: CoT-VLA: Visual chain-of-thought reasoning for vision-language-action models. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1702–1713 (2025). https://doi.org/10.1109/CVPR52734.2025.00166
42. Zhao, T., Kumar, V., Levine, S., Finn, C.: Learning fine-grained bimanual manipulation with low-cost hardware. In: Robotics: Science and Systems (RSS) (2023). https://doi.org/10.15607/RSS.2023.XIX.016
43. Zhao, W., Ding, P., Zhang, M., Gong, Z., Bai, S., Zhao, H., Wang, D.: VLAS: Vision-language-action model with speech instructions for customized robot manipulation. In: International Conference on Learning Representations (ICLR) (2025)
44. Zheng, J., Li, J., Liu, D., Zheng, Y., Wang, Z., Ou, Z., Liu, Y., Liu, J., Zhang, Y.Q., Zhan, X.: Universal actions for enhanced embodied foundation models. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 22508–22519 (2025). https://doi.org/10.1109/CVPR52734.2025.02096
45. Zhu, J., Wang, W., Chen, Z., Liu, Z., Ye, S., Gu, L., Tian, H., Duan, Y., Su, W., Shao, J., et al.: InternVL3: Exploring advanced training and test-time recipes for open-source multimodal models. arXiv preprint arXiv:2504.10479 (2025)
46. Zhu, Y., Li, B., Xin, Y., Xia, Z., Xu, L.: Addressing representation collapse in vector quantized models with one linear layer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 22968–22977 (2025). https://doi.org/10.1109/ICCV51701.2025.02132
47. 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 (CoRL). pp. 2165–2183. PMLR (2023)