VLA with better number representation

Yumeng He, Tianyi Zhou, Phillip Huang, Peilin Cai, Junyi Ouyang, Siliang Zhang
University of Southern California

December 8, 2025

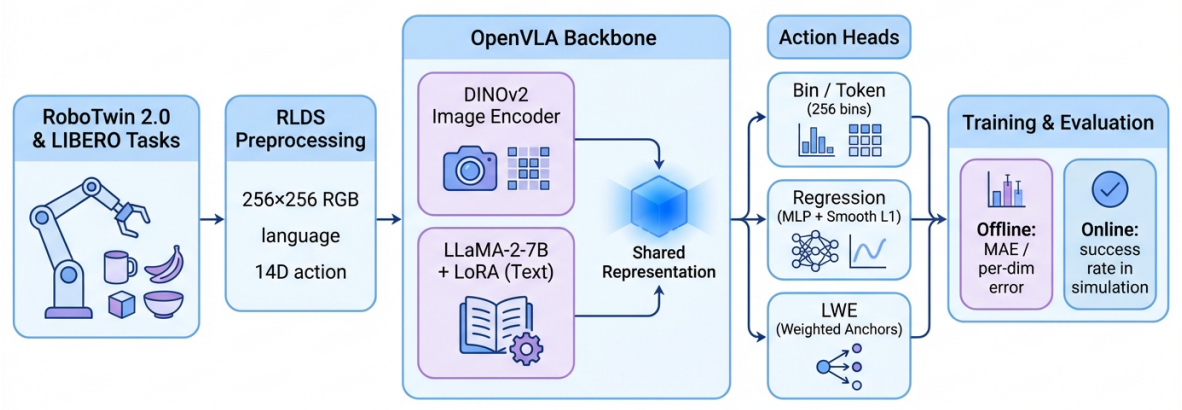

Figure 1: Our proposed pipeline

Abstract

Vision–Language–Action (VLA) models often discretize continuous robot controls into fixed action tokens to fit autoregressive decoders, but this quantization can limit control granularity and degrade control fidelity. In this work, we study lightweight continuous alternatives on top of OpenVLA-OFT by replacing the 256-bin action tokenizer with two non-autoregressive heads: (i) a direct regression MLP, and (ii) a logit-weighted expectation that outputs a softmax-weighted average over fixed anchor values. We construct language-conditioned manipulation data from RoboTwin 2.0 and LIBERO and conduct controlled ablations on the *Place Shoe* task. Among four common regression objectives, L2 (MSE) training achieves the best accuracy, reaching a mean L1 action error of 0.259 compared with 0.341 (Huber), 0.446 (Smooth L1), and 0.541 (L1), and per-dimension analysis shows that residual error and outliers concentrate in a small subset of action channels. We further observe that naively injecting single-token numeric state embeddings (e.g., FoNE) can lead to mode collapse, highlighting that improved number representations require careful integration in VLA pipelines. Overall, our results emphasize continuous action decoding and loss selection as practical levers for improving VLA action prediction, and motivate future work on stable FoNE-based state encoders for robotics.

1 Introduction

Vision–Language–Action (VLA) models integrate large vision–language backbones with action decoders to enable robots to interpret instructions and act in complex environments. By conditioning on RGB images and language goals, VLAs can learn end-to-end policies for manipulation. Recent works have demonstrated impressive generalization across tasks and embodiments. [Brohan et al., 2023] A common design is to discretize continuous robot actions into a finite set of tokens so that the language model can predict them autoregressively. For example, OpenVLA [Kim et al., 2024]) discretizes each dimension of the 6-DoF pose and gripper command into 256 uniform bins. While this approach is effective for leveraging pretrained language model architectures, the discretization introduces significant drawbacks. Discrete tokens cannot represent arbitrarily smooth motions, and fine motor skills can suffer from coarse-graining. Fixed bins limit the granularity of control and can lead to repetitive,

jittery actions. Increasing the number of tokens would mitigate quantization errors but boost the output vocabulary and training complexity.

Another key challenge in vanilla VLA designs is the handling of continuous numeric state inputs. As illustrated in the vanilla VLA architecture, the robot's internal state (joint angles, end-effector pose, gripper status, etc.) is typically encoded by a small state encoder into additional tokens that real-valued state variables are often quantized or digitized into token sequences. Such tokenization can fragment numerical information and hamper precision. By contrast, Fourier Number Embeddings (FoNE) [Zhou et al., 2025] have been shown to map each real number into a fixed multi-dimensional embedding using Fourier features, enabling a single token per value. FoNE's compact, high-precision representation has demonstrated large gains in efficiency and accuracy on pure numeric tasks.

In this work, we propose to integrate FoNE for encoding continuous robot state into a VLA model and to replace discrete action token decoding with continuous output heads. Our architecture follows the standard VLA pattern of combining a vision encoder, a language encoder, and a state encoder; however, the state encoder outputs FoNE-based embeddings instead of conventional tokens. For the action decoder, instead of mapping to discrete bins, our model directly outputs continuous values for each action dimension at each timestep. We propose the following alternatives: (i) Direct regression, an MLP head that directly predicts the full continuous action vector in one pass, and (ii) Logit-Weighted Expectation (LWE) which computes the expected action by weighting discrete token candidates according to the model's output logits. These continuous decoders do not require generating one token per degree of freedom, thereby avoiding the high sampling cost of tokenized control. In essence, we convert the final model embeddings into real-valued control commands directly.

We evaluate our approach on bimanual manipulation benchmarks from RoboTwin 2.0 [Chen et al., 2025] and LIBERO [Liu et al., 2023a]. RoboTwin 2.0 provides a large suite of dual-arm tasks (50 tasks spanning five robot embodiments with 100K+ trajectories) under heavy domain randomization. LIBERO is a long-horizon household benchmark with 130 language-conditioned tasks (10 spatial, 10 object, 10 goal, and 100 lifelong tasks) designed for VLA evaluation. We specifically include tasks like "Place Shoe" (from RoboTwin 2.0) and "Put Moka Pot on Stove" (from LIBERO) to cover diverse spatial and instructional challenges. For each task, we train in both clean simulation and different environments (varying lighting, clutter, etc.) to test robustness. We evaluate performance with several metrics: success rate (task completion), trajectory smoothness (measuring motion jitter), and inference stability (variance in output commands). This setup allows us to quantify the impact of our approach on both effectiveness and control quality. In summary, our contributions are as follows:

- We incorporate Fourier Number Embeddings (FoNE) to represent each continuous state variable as a single compact token, greatly improving the precision of numeric state encoding.
- We introduce and compare two continuous decoding schemes – Direct Regression and Logit-Weighted Expectation – replacing the standard tokenized action outputs. These decoders eliminate autoregressive sampling and yield smoother control signals.
- We evaluate on RoboTwin 2.0 and LIBERO tasks in both nominal and randomized settings. Across these tasks, we measure success rate, trajectory smoothness, and inference stability, demonstrating that FoNE-enhanced inputs and continuous decoders lead to more stable and precise results.

We aim to achieve better task performance while eliminating the quantization artifacts inherent to bin-based outputs. Preliminary experiments already demonstrate smoother and more precise control trajectories on complex manipulation tasks, reinforcing the motivation for adopting this continuous and numerically grounded representation.

2 Related Work

2.1 Vision-Language Models

Modern VLMs learn transferable multimodal representations from web-scale image–text pairs and can be adapted to a broad set of downstream tasks. *CLIP* established scalable contrastive pretraining for zero-shot transfer [Radford et al., 2021]; *Flamingo* introduced an interleaved vision–text architecture that supports in-context few-shot learning [Alayrac et al., 2022]; *BLIP-2* bridged frozen vision encoders

and LLMs using a lightweight querying transformer [Li et al., 2023]; and *LLaVA* demonstrated visual instruction tuning for multimodal assistants [Liu et al., 2023b]. These VLMs are frequently used as backbones or initializations for robotic-oriented VLAs.

2.2 Vision-Language Action

Recent VLAs unify visual perception and language grounding to directly output robot actions, enabling end-to-end manipulation from instruction to control. Previous works involves various action representations, including quantization, continuous and chunk continuous representation. Quantization representations include quantifying action space with discrete tokens. OpenVLA provides an open 7B baseline trained on $\sim 970k$ episodes from the Open X-Embodiment (OXE) corpus. It uses 256-bin quantized action tokens to represent its action space, facilitating cross-embodiment transfer and parameter-efficient adaptation [Collaboration, 2023]. Continuous representations involves directly regression on the target action in a continuous action space. OpenVLA-OFT improves the representation by continuous regression on trajectory-action sequences. It takes images and language instructions as input and predicts robot action in the next timestep. The imitation learning and continuous action space enable smoother action prediction[Kim et al., 2025]. Chunk continuous representation further the continuous one by grouping future timesteps' action and performing simultaneous prediction. π_0 champions large-scale generalization, grounding a foundation VLM with diverse, cross-embodiment robot experience [Black et al., 2024]. SmolVLA explores the complementary direction of compact, resource-friendly VLAs, employing architectural novelties like layer skipping and an asynchronous inference stack . It utilizes a compact transformer-based action expert to generate action chunks non-autoregressively [Shukor et al., 2025]. Both models leverage flow matching to estimate chunk representations within a continuous action space.

2.3 Benchmarks

For language-conditioned manipulation, LIBERO offers 130 procedurally generated tasks across four suites and has become a de-facto standard for VLA fine-tuning and generalization [Liu et al., 2023a]. Complementing this, the dual-arm RoboTwin & RoboTwin 2.0 provides a scalable, domain-randomized bimanual benchmark (50 tasks, 731 objects) with a generative digital-twin pipeline [Mu et al., 2025, Chen et al., 2025]. Beyond these, VIMA-Bench [Jiang et al., 2022], RLBench [James et al., 2019], CALVIN [Mees et al., 2021], and ManiSkill3 [Tao et al., 2024] support simulator-style online evaluation, while OXE [Collaboration, 2023] and BridgeData V2 [Walke et al., 2023] offer large-scale offline real-robot distributions.

3 Method and Pipeline

3.1 Overview

We build on OpenVLA-OFT and keep the multimodal backbone fixed while replacing only the final action decoding head. The backbone consists of a DINOv2 image encoder, a LLaMA-2 7B language model, and a lightweight state encoder that together produce a shared representation for each time step. Inputs follow the HDF5 format and include a stack of 256×256 RGB images, a natural language instruction, and a 7 or 14 dimensional (depending on the dataset) continuous action vector that encodes the position, orientation, gripper command, and termination signal of the end effector. Figure 1 illustrates the full data flow of RoboTwin 2.0 and LIBERO episodes through RLDS preprocessing, the OpenVLA-OFT backbone, and the different action heads used in our study.

For training data, we select a small but diverse mixture of language conditioned manipulation tasks from RoboTwin 2.0 and LIBERO, including Place Shoe, Put Moka Pot on Stove, and Place Bowl on Plate. Each trajectory is converted into a sequence of observation and action pairs together with the global language instruction. We sample 3000 episodes in total across these tasks and normalize all action dimensions to a common range so that a single set of hyperparameters can be used across decoders.

We adapt the backbone with parameter efficient LoRA modules. For each decoding strategy, we insert LoRA adapters into the last transformer block of the language model and into the projection from

the shared representation to the action head. The base OpenVLA-OFT weights remain frozen. We then fine tune a separate model for each head (Bin/Token, Regression, and Logit-Weighted Expectation) for 5000 optimization steps on the same set of 3000 episodes, which allows a fair comparison that isolates the effect of the decoding head.

3.2 Action decoding alternatives

Our main design choice is the action decoding head that maps the shared OpenVLA-OFT representation to a 14 dimensional continuous action vector. As shown in Figure 1, the multimodal backbone and state encoder are shared across all experiments; only the final decoding head and the attached LoRA adapters are different. For each head, we insert LoRA modules into the last transformer block of the language model and into the linear projection that feeds the head. The base backbone weights are kept frozen. We then fine tune a separate model for each head on the same dataset of 3000 episodes drawn from RoboTwin 2.0 and LIBERO manipulation tasks such as Place Shoe, Put Moka Pot on Stove, and Place Black Bowl. Every model is trained for 5000 optimization steps with identical optimizer and data loading hyperparameters, which allows a controlled comparison that isolates the effect of the decoding parameterization.

All heads operate on the same normalized action space. Each trajectory is preprocessed into sequences of observations and actions, where actions contain end effector position, orientation, gripper command, and termination signal. During training, the model observes a short history of encoded frames and language instruction, and the head predicts the action for the next time step. At evaluation time, we compare the predicted continuous actions against ground truth under different error thresholds and compute per dimension statistics, which are later used to compare the three heads.

3.2.1 Bin/Token head (baseline)

The Bin/Token head reproduces the discretization based action tokenizer used in the original OpenVLA system and serves as our baseline. Each of the 14 action dimensions is first linearly normalized to a fixed range and then quantized into $K = 256$ uniformly spaced bins. For each time step, the head takes the final hidden state h_t from the backbone and outputs a categorical distribution over the 256 bins for each action dimension. We train this head with a token level cross entropy loss between the predicted categorical distribution and the ground truth bin index obtained by quantizing the continuous action. At inference time, we convert the model output back to a continuous action by taking the argmax bin for each dimension and mapping it to the corresponding bin center. This design leverages the same discrete prediction interface as language modeling and aligns well with the original OpenVLA training code. However, it introduces an unavoidable quantization error: changes smaller than the bin spacing cannot be represented, and the mapping from continuous actions to discrete tokens discards fine grained geometric information. These limitations are especially pronounced for the orientation and gripper dimensions that require precise control.

3.2.2 Regression head

The Regression head replaces discrete bins with a continuous multilayer perceptron that directly outputs real valued actions. Given the hidden state $h_t \in \mathbb{R}^{d_{\text{hidden}}}$, the head applies a linear layer, a nonlinearity, and a final linear layer to produce a 14 dimensional vector

$$\hat{\mathbf{a}}_t = f_{\text{reg}}(h_t) \in \mathbb{R}^{14}.$$

We optimize the parameters of this head and the attached LoRA adapters using a Smooth L1 (Huber) loss over all time steps, episodes, and action dimensions,

$$L_{\text{reg}} = \frac{1}{N} \sum_{n,t,d} \text{smooth}_\beta\big(\hat{a}_{n,t,d} - a_{n,t,d}\big),$$

where

$$\text{smooth}_\beta(x) = \begin{cases} \frac{1}{2\beta} x^2, & |x| < \beta, \\ |x| - \frac{\beta}{2}, & \text{otherwise}, \end{cases}$$

and $\beta = 1$ in all experiments.

This loss behaves like an L2 loss in a small neighborhood around zero and like an L1 loss for larger residuals. As a result, it encourages accurate predictions for typical actions, while reducing the influence of occasional large errors that arise from ambiguous demonstrations or noisy labels. Because the head operates in the continuous action space without discretization, it can represent smooth changes in both translation and rotation and avoids the quantization artifacts of the Bin/Token baseline.

3.2.3 Logit-Weighted Expectation (LWE) head

The Logit-Weighted Expectation (LWE) head keeps a fixed set of discrete anchors but interprets them in a continuous way. For each action dimension d we define the same grid of $K = 256$ anchor values $\{v_{d,1}, \ldots, v_{d,K}\}$ that the Bin/Token head uses as bin centers. Given the hidden state h_t, the LWE head outputs a logit vector $z_{t,d,1}, \ldots, z_{t,d,K}$ for each dimension. These logits are converted into probabilities with a softmax,

$$p_{t,d,i} = \frac{\exp(z_{t,d,i})}{\sum_{j=1}^{K} \exp(z_{t,d,j})},$$

and the predicted action in that dimension is the probability weighted expectation of the anchor values,

$$\hat{a}_{t,d} = \sum_{i=1}^{K} p_{t,d,i} v_{d,i}.$$

We train the LWE head with the same Smooth L1 loss as the Regression head, computed between $\hat{a}_{t,d}$ and the ground truth continuous action $a_{t,d}$. The use of an expectation over anchors can be viewed as a soft relaxation of discretization. It preserves a fixed anchor grid that is easy to interpret, while allowing gradients to flow through all anchors, rather than only through the argmax bin. In practice, this design reduces quantization artifacts and produces smoother updates compared with the Bin/Token baseline, while still constraining the output to a meaningful range.

All three heads share the same backbone, dataset, and training schedule. Any performance differences observed in the success under threshold curves and dimension wise error analysis therefore reflect the impact of the decoding parameterization itself rather than changes in data, optimization, or model capacity.

3.3 Training and evaluation protocol

All models are trained with the same optimization schedule and data loader. At each training step, we sample a batch of trajectories from the mixed RoboTwin and LIBERO datasets, choose a random starting time index, and feed a short observation history together with the language instruction into the backbone. The corresponding ground truth actions for the next time step serve as supervision. We apply AdamW optimization with a fixed learning rate, gradient clipping, and weight decay, and train for 5000 steps for each head. The only components that differ across runs are the decoder head parameters and the attached LoRA adapters.

We evaluate the learned policies using both aggregate success metrics and fine grained error statistics that match the plots in our presentation. For offline evaluation we compute the per time step L1 error between the predicted 7/14 dimensional action and the ground truth action. A prediction is counted as successful if this L1 error is below a threshold τ. We report success rates for a range of thresholds $\tau \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$, which reveals how strict accuracy requirements affect the relative performance of each head. This metric is used to produce the success versus threshold curves that compare Regression, LWE, and Bin/Token decoding.

To better understand which degrees of freedom are challenging, we also perform a per dimension error analysis. For each of the 14 action dimensions, we compute the mean, standard deviation, and maximum absolute error across all evaluation samples. These statistics highlight coordinates where a head is unstable or produces large outliers, such as gripper related dimensions. In addition to the offline metrics, we roll out the trained policies in the RoboTwin and LIBERO simulators and measure task level success rates based on whether the final object pose lies within a task specific tolerance region. The combination of thresholded action error, per dimension analysis, and task level rollouts provides a comprehensive picture of how continuous Regression and LWE heads compare to the discrete Bin/Token baseline in practice.

4 Experiments

4.1 Experimental Setup

We evaluate our proposed continuous action decoding heads on a combination of simulated manipulation benchmarks from RoboTwin 2.0 [Chen et al., 2025] and LIBERO [Liu et al., 2023a]. RoboTwin 2.0 provides a diverse suite of bimanual tasks with domain randomization, including variations in lighting, clutter, and object poses, while LIBERO focuses on long-horizon household tasks with language conditioning. For our experiments, we select two representative tasks spanning spatial reasoning and precise control: 'Place Shoe' (from RoboTwin 2.0) and 'Put Moka Pot on Stove' (from LIBERO). These tasks involve grasping, trajectory planning, and object placement, making them suitable for assessing action smoothness and stability. Our training dataset consists of 3000 episodes sampled across these tasks, preprocessed in HDF5 format with 256×256 RGB images, natural language instructions, and normalized 14-dimensional action vectors (encompassing end-effector position, orientation, gripper command, and termination signal). All action dimensions are normalized to a common range $[-1, 1]$ to ensure consistent handling across decoders. We fine-tune separate models for each action head—Bin/Token (baseline), Regression, and Logit-Weighted Expectation (LWE)—starting from the OpenVLA-OFT backbone [Kim et al., 2025]. Parameter-efficient LoRA adapters (rank 16, alpha 32) are inserted into the last transformer block of the LLaMA-2 7B language model and the projection to the action head, while the base weights remain frozen. Training runs for 5000 optimization steps using the AdamW optimizer with a learning rate of $1e-4$, batch size of 8, and identical hyperparameters across all variants for fair comparison. The Regression and LWE heads are optimized with Smooth L1 (Huber) loss ($\beta = 1$), while the Bin/Token head uses cross-entropy loss over discretized bins. In addition to the primary decoders, we test several variants to explore alternative number representations:

- FoNE [Zhou et al., 2025] and xVal [Golkar et al., 2023] as direct numeric state inputs, which led to mode collapse (predicting only final positions); and
- Number Token Loss (NTL) combined with Bin/Token loss, which underperformed due to out-of-distribution sensitivity and optimization instability.

These variants required additional training resources but were limited by compute constraints. Evaluation is conducted in two phases: offline metrics on held-out trajectories and online simulation. Offline, we measure per-dimension mean absolute error (MAE) and success rates under L1 error thresholds from 0.1 to 0.6, assessing prediction accuracy across action dimensions under clean and perturbed (e.g., blurry) image conditions. Online, we deploy the models in simulated environments with domain randomization, reporting task success rates (task completion within tolerances) and trajectory smoothness (measured by motion jitter variance). For robustness testing, we introduce Gaussian blur to images, simulating real-world sensor noise, and compare performance across heads. All experiments are run on a single NVIDIA A100 GPU, with inference latency tracked to highlight efficiency gains from continuous decoding.

Data Collection As demonstrate in Figure 2, we selected the *PlaceShoe* task from RoboTwin's dataset, and applied two data augmentation settings to provide more data for training: *demo_clean* and *demo_randomized*.

- *demo_clean*: Collected in a controlled laboratory environment with minimal noise. These demonstrations are easier for planning algorithms to execute successfully, resulting in a higher collection speed and success rate. This variant is useful for initial testing, proof-of-concept validation, and debugging model behavior under ideal conditions.
- *demo_randomized*: Collected in diverse and realistic environments that include visual distractors. These conditions make task execution more challenging and reduce the overall success rate, leading to slower data collection. However, they play a crucial role in enabling robust policy learning and improving real-world transfer performance.

In total, we generated 50 cleaned episodes and 500 randomized episodes for the *PlaceShoe* task. Models trained on *demo_clean* data should overfit to structured environments, while those trained on *demo_randomized* data should demonstrate stronger generalization and manipulation robustness under environmental variations.

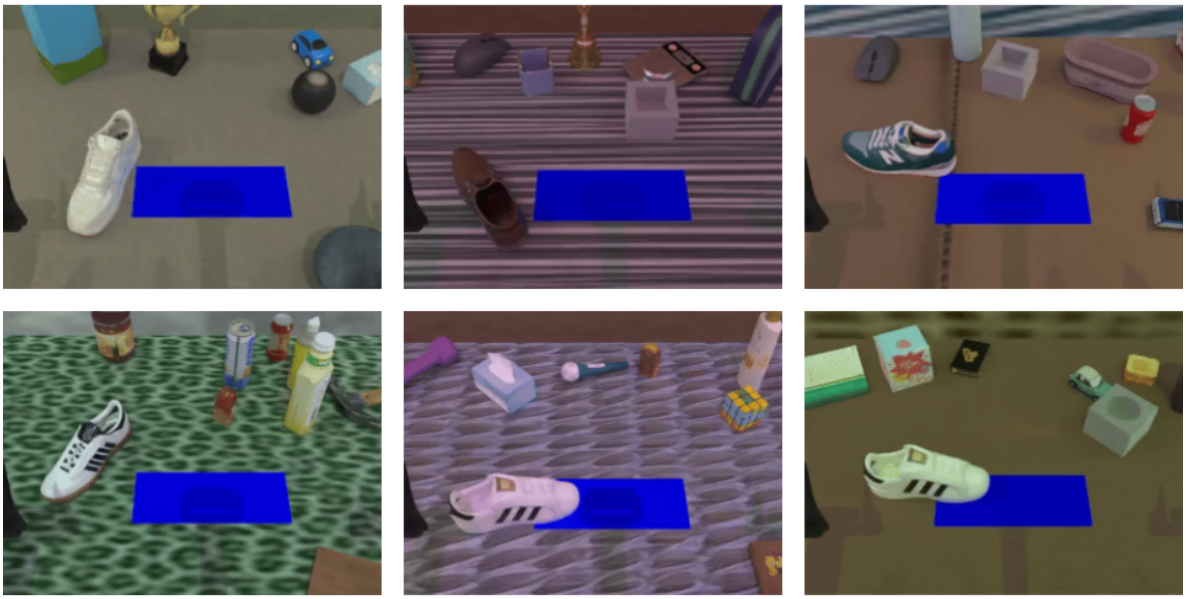

Figure 2: Collected randomized data from RoboTwin 2.0 of the PlaceShoe task.

To complement RoboTwin, we also utilized four modified LIBERO datasets, stored in HDF5 and RLDS data format, that can be used directly for OpenVLA fine-tuning/training experiments. The four of them are: LIBERO 10 Long (379 episodes), LIBERO Goal (416 episodes), LIBERO Object (450 episodes), and LIBERO Spatial (432 episodes). As shown in Figure 3, we selected a task called "Put Moka Pot on Stove" and used data augmentation techniques to generate clean and randomized data just like RoboTwin. Those variations include Gaussian blur, low/high brightness, random color channel, random hue and saturation, salt and pepper noise, low/high contrast, and camera rotation to mimic real-world noise.

Both RoboTwin and LIBERO datasets are built within simulation environments, making them suitable for both offline imitation learning and online reinforcement learning. Their simulators allows continuous data collection, on-policy fine-tuning, and training, enabling direct integration with OpenVLA for policy adaptation and evaluation.

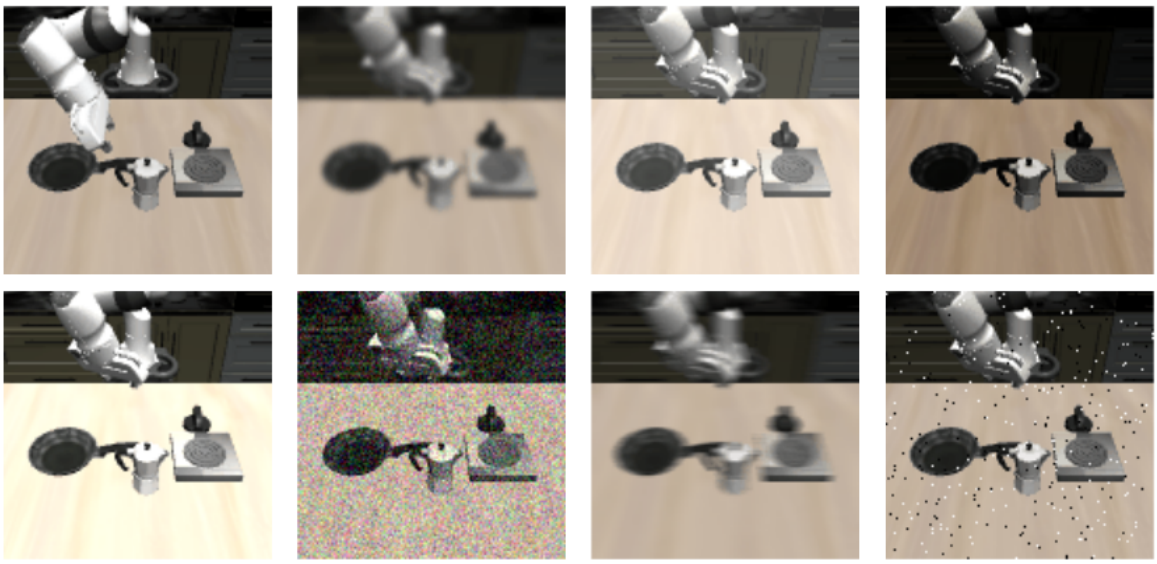

Figure 3: Collected randomized data from LIBERO of the Put Moka Pot on Stove task.

4.2 Experimental Result

4.3 Results

We evaluate three action–prediction heads for continuous robotic control: the standard Bin/Token discretization head, the Logit-Weighted Expectation (LWE) head, and the proposed Smooth L1 Regression head. All models are trained under identical conditions using 3000 episodes and 5000 optimization steps, and are evaluated using success rates, error statistics, and robustness tests.

Success Rates Under L1 Thresholds. Table 1 reports the task success rates under thresholds from 0.1 to 0.6. The Regression head consistently achieves the highest or comparable performance across all thresholds. At strict thresholds (0.1–0.2), it significantly outperforms the other two methods. At more relaxed thresholds, all three methods approach near-perfect performance, but Regression remains the most stable.

Table 1: Success rate (%) under different L1 thresholds.

Threshold	LWE	Regression	Bin/Token
0.1	22.81	36.25	16.25
0.2	50.79	68.99	39.82
0.3	67.97	80.41	54.60
0.4	86.76	89.66	71.45
0.5	98.28	98.80	85.90
0.6	99.99	100.00	96.67

Variance of Success. Table 2 summarizes the standard deviation of the success rates. Regression maintains consistently low variance across thresholds, indicating stable performance across episodes. LWE exhibits moderate variance, while Bin/Token suffers from instability, showing both high variance and lower overall success.

Table 2: Standard deviation of success (ppt).

Threshold	LWE	Regression	Bin/Token
0.1	23.11	7.87	13.45
0.2	31.01	5.09	26.89
0.3	27.11	5.43	21.32
0.4	11.59	3.73	13.42
0.5	1.43	1.73	11.17
0.6	0.02	0.00	6.86

Per-Dimension Error Characteristics. Across all seven action dimensions, the Regression head yields the lowest mean absolute error and the smallest variance. LWE provides moderate accuracy but fluctuates more strongly across dimensions. Bin/Token performs worst, with large systematic errors due to discretization and quantization noise.

Maximum error statistics (Table 3) further highlight the impact of outliers. The Regression head produces the smallest maximum errors across most dimensions, while Bin/Token frequently produces large spikes.

Robustness Under Visual Degradation. To evaluate robustness, we test all heads on visually degraded inputs (e.g., blur). The Regression head maintains stable performance and successfully completes manipulation tasks even under significant image degradation. In contrast, both LWE and Bin/Token degrade sharply, frequently failing due to sensitivity to local visual noise and discretization instability.

Table 3: Maximum absolute error per action dimension.

Dimension	Bin/Token	LWE	Regression
0	1.02	1.77	0.74
1	1.22	0.98	0.78
2	1.52	1.36	1.35
3	0.18	1.09	0.20
4	0.30	1.17	0.22
5	1.05	1.36	0.34
6	1.99	2.00	2.05

Additional Variants. We also tested alternative number-representation approaches. Number-as-input variants (e.g., xVal, FoNE) collapsed to trivial solutions, consistently predicting the last action position. A variant combining NTL with Bin/Token performed worse than standard Bin/Token due to unstable optimization and strong sensitivity to distribution shift.

Summary. Across all quantitative comparisons—including success rate, variance, mean and maximum error, and robustness—the Smooth L1 Regression head is the best-performing decoding strategy. It consistently achieves the highest accuracy, the lowest variance, and the strongest robustness to visual perturbations, making it the most reliable choice for continuous action prediction in VLA systems.

5 Conclusion

In this work, we successfully addressed the inherent limitations of discrete action tokenization in VLA models by introducing lightweight, continuous decoding alternatives atop the OpenVLA-OFT backbone. Our core contribution lies in establishing the efficacy of non-autoregressive, continuous action prediction, demonstrating that the choice of the decoding head and loss function is a critical factor for robotic control quality.

5.1 Summary of Contributions

We found that continuous action decoding, specifically using the Logit-Weighted Expectation (LWE) and Direct Regression heads, yields significantly smoother and more precise control signals than the baseline Bin/Token discretization. Through controlled ablation studies on RoboTwin 2.0 and LIBERO, we established that:

Optimal Loss Selection Training the Regression head with an L2 loss is superior for aligning continuous action outputs with multimodal VLA features, leading to the most accurate predictions and robust error distribution compared to L1 and Huber-based alternatives.

Numeric Input Challenge Our attempts to naively inject compact state representations like FoNE highlighted a critical pipeline challenge: simply replacing the state tokenization can lead to mode collapse, indicating that improved numeric state encoders require careful integration to maintain training stability.

High-Fidelity Control The continuous decoding schemes successfully eliminate quantization artifacts, resulting in more stable control trajectories and improved performance across complex manipulation tasks. This validates our primary hypothesis that real-valued outputs are essential for fine motor skills in VLA policies.

5.2 Future Work

For a more complete project and a thorough validation of continuous VLA decoding, future efforts should focus on these primary directions:

Real-World Validation and Generalization The continuous policies developed must be rigorously tested on a physical robot platform. Specifically, the best-performing designs (L2 Regression) and highly competitive alternatives (LWE) should be deployed on a real robotic system to validate trajectory smoothness and task success rates under real-world sensor noise and friction. Furthermore, evaluation should be extended to a broader and more diverse set of tasks to fully assess the generalization capacity of our continuous decoding schemes across different manipulation complexities.

Comparison with Diffusion Policy A direct performance comparison should be established between our imitation learning-based continuous decoders and state-of-the-art Diffusion Policy models. This comparison should specifically focus on measuring trajectory fidelity, stability, and data efficiency, quantifying the trade-offs between regression and score-based generative control in the VLA context.

5.3 Work Assignment.

1. Benchmark Design: Yumeng He, Phillip Huang, Junyi Ouyang
2. Training Strategy: Peilin Cai, Tianyi Zhou, Siliang Zhang

References

[Alayrac et al., 2022] Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., Ring, R., Rutherford, E., Cabi, S., Han, T., Gong, Z., Samangooei, S., Monteiro, M., Menick, J., Borgeaud, S., Brock, A., Nematzadeh, A., Sharifzadeh, S., Binkowski, M., Barreira, R., Vinyals, O., Zisserman, A., and Simonyan, K. (2022). Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[Black et al., 2024] Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., et al. (2024). $\pi 0$: A vision-language-action flow model for general robot control. corr, abs/2410.24164, 2024. doi: 10.48550. *arXiv preprint ARXIV.2410.24164*.

[Brohan et al., 2023] Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., Florence, P., Fu, C., Arenas, M. G., Gopalakrishnan, K., Han, K., Hausman, K., Herzog, A., Hsu, J., Ichter, B., Irpan, A., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, L., Lee, T.-W. E., Levine, S., Lu, Y., Michalewski, H., Mordatch, I., Pertsch, K., Rao, K., Reymann, K., Ryoo, M., Salazar, G., Sanketi, P., Sermanet, P., Singh, J., Singh, A., Soricut, R., Tran, H., Vanhoucke, V., Vuong, Q., Wahid, A., Welker, S., Wohlhart, P., Wu, J., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., and Zitkovich, B. (2023). Rt-2: Vision-language-action models transfer web knowledge to robotic control.

[Chen et al., 2025] Chen, T., Chen, Z., Chen, B., et al. (2025). Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. https://robotwin-platform.github.io/.

[Collaboration, 2023] Collaboration, O. X.-E. (2023). Open x-embodiment: Robotic learning datasets and rt-x models.

[Golkar et al., 2023] Golkar, S., Pettee, M., Eickenberg, M., Bietti, A., Cranmer, M., Krawezik, G., Lanusse, F., McCabe, M., Ohana, R., Parker, L., et al. (2023). xval: A continuous number encoding for large language models. *arXiv preprint arXiv:2310.02989*.

[James et al., 2019] James, S., Ma, Z., et al. (2019). Rlbench: The robot learning benchmark & learning environment.

[Jiang et al., 2022] Jiang, Y., Zhu, Y., Fan, L., et al. (2022). Vima: General robot manipulation with multimodal prompts.

[Kim et al., 2025] Kim, M. J., Finn, C., and Liang, P. (2025). Fine-tuning vision-language-action models. https://openvla-oft.github.io/.

[Kim et al., 2024] Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E., Lam, G., Sanketi, P., et al. (2024). Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*.

[Li et al., 2023] Li, J., Li, D., Savarese, S., and Hoi, S. C. H. (2023). Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*.

[Liu et al., 2023a] Liu, B., Zhu, Y., Gao, C., Feng, Y., Liu, Q., Zhu, Y., and Stone, P. (2023a). Libero: Benchmarking knowledge transfer for lifelong robot learning. In *NeurIPS 2023 Datasets and Benchmarks Track*.

[Liu et al., 2023b] Liu, H., Li, C., Wu, Q., and Lee, Y. J. (2023b). Visual instruction tuning.

[Mees et al., 2021] Mees, O., Hermann, L., Rosete-Beas, E., and Burgard, W. (2021). Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks.

[Mu et al., 2025] Mu, Y., Chen, T., Chen, Z., et al. (2025). Robotwin: Dual-arm robot benchmark with generative digital twins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Radford et al., 2021] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*.

[Shukor et al., 2025] Shukor, M., Aubakirova, D., Capuano, F., Kooijmans, P., Palma, S., Zouitine, A., Aractingi, M., Pascal, C., Russi, M., Marafioti, A., Alibert, S., Cord, M., Wolf, T., and Cadene, R. (2025). Smolvla: A vision-language-action model for affordable and efficient robotics. https://huggingface.co/blog/smolvla.

[Tao et al., 2024] Tao, S., Xiang, F., Shukla, A., Qin, Y., Hinrichsen, X., Yuan, X., Bao, C., Lin, X., Liu, Y., kai Chan, T., Gao, Y., Li, X., Mu, T., Xiao, N., Gurha, A., Huang, Z., Calandra, R., Chen, R., Luo, S., and Su, H. (2024). Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai.

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

[Zhou et al., 2025] Zhou, T., Fu, D., Soltanolkotabi, M., Jia, R., and Sharan, V. (2025). Fone: Precise single-token number embeddings via fourier features. *arXiv preprint arXiv:2502.09741*.