ActionUNet: Improving Robustness of VLA Models with Efficient Multi-scale Fine-tuning

Di Zhu* **Ziheng Yan*** **Fang Wan†**
University of Chinese Academy of Sciences
{zhudi25,yanziheng21}@mails.ucas.ac.cn
wanfang@ucas.ac.cn

Abstract

Vision-Language-Action (VLA) models have shown great promise for robotic manipulation by mapping multi-modal semantics to physical actions. However, this mapping inherently struggles to align these coarse-grained semantics with fine-grained temporal execution. It leaves VLA models with limited generalization and insufficient robustness in cluttered environments. To overcome this issue, we propose ActionUNet, an efficient multi-scale fine-tuning framework that enhances pre-trained VLA models with minimal computational cost. ActionUNet first constructs a lightweight temporal U-Net within the temporal-aligned action feature space to fuse hierarchical structural priors, effectively bridging the scale gap between semantics and temporal executions. Recognizing that multi-scale modeling can disrupt microscopic temporal continuity and cause mechanical oscillations, ActionUNet then employs a conditional SIREN as a continuous action decoder. Equipped with explicit second-order smoothness constraints, this decoder guarantees temporal continuity and reduces high-frequency motion jitter. By smoothing temporal discontinuities from multi-scale fusion, this continuous formulation reduces mechanical execution failures while preserving the base VLA model's generalization and manipulation robustness. Extensive experiments on RoboTwin 2.0 and LIBERO-Plus benchmarks, together with real-world hard evaluations, demonstrate that ActionUNet significantly improves $\pi_{0.5}$ success rates by absolute 9.8%, 6.1%, and 11.4%, respectively, while also generalizing to the regression-based OpenVLA-OFT backbone, highlighting its effectiveness and efficiency as a fine-tuning strategy. Code and implementation details are available at https://github.com/Di-Zhu123/ActionUNet.

1 Introduction

Vision-Language-Action (VLA) models, which combine language instructions with visual environmental perception, have emerged as a dominant paradigm for robotic manipulation tasks. With the rapid progress of vision-language models, a series of VLA approaches [1, 2, 3, 4] have demonstrated promising performance. However, existing VLA models still face critical challenges, including limited generalization ability [5] and insufficient robustness in cluttered or distracting environments [6].

Current VLA models [7, 8, 4] typically attach an action expert to the large vision-language model to predict future action sequences, prompted by a single language instruction. This action expert learns to translate the VLM's multi-modal representations directly into action trajectories, typically through a transformer decoder or a diffusion-based generative head. While VLMs excel at high-level task reasoning, they often struggle with precise, fine-grained action prediction [9, 10].

*Equal contribution.
†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

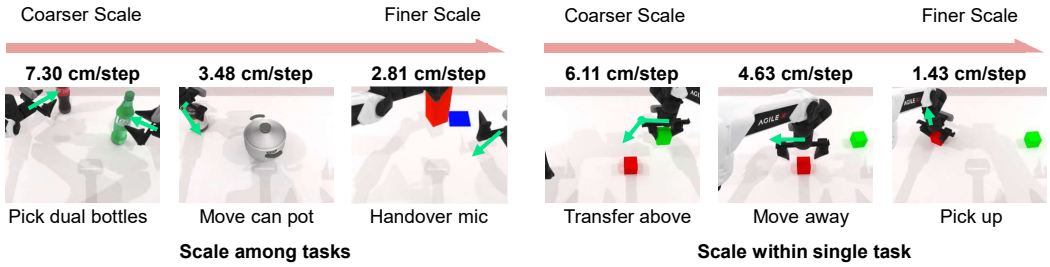

Figure 1: Illustration of multi-scale temporal dynamics. Per-step end-effector displacements vary significantly across different tasks (left) and over time within a single task "Stack blocks two" (right).

Central to robotic manipulation, action prediction must bridge high-level intentional inference for task planning with fine-grained, low-level control for execution. Furthermore, we observe that the per-step end-effector displacement varies significantly not only across different tasks (Fig. 1 left) but also across different stages within the same task (Fig. 1 right). We characterize execution scale using the per-step end-effector displacement, where larger displacements correspond to coarser-scale motions and smaller displacements correspond to finer-scale motions (see Appendix C). This variance in per-step displacement indicates the differences in the temporal scale of actions, implying that VLA models are inherently required to possess multi-scale temporal representation capabilities. Consequently, directly mapping the coarse-grained multi-modal semantics to fine-grained action sequences suffers from a severe *temporal scale mismatch* issue. This mismatch undermines the VLA models' robustness to visual observation noise, as the model over-relies on abstract semantics while failing to capture the multi-scale, fine-grained temporal cues necessary for resisting local perturbations.

To address this issue, we introduce multi-scale action modeling in the action feature space. However, implementing such multi-scale representations risks breaking the temporal continuity of actions across different scales. This occurs because aggregating actions from varying granularities often projects disjointed structural priors onto the continuous action space, causing interpolation errors at scale boundaries. Consequently, the generated trajectories often exhibit undesired non-smooth transitions [11], leading to mechanical oscillations and further reducing the execution robustness of VLA models. Therefore, a core challenge lies in equipping VLA models with multi-scale structural priors while simultaneously enforcing action smoothness constraints. Addressing this through efficient fine-tuning strategies has become an urgent necessity.

In this paper, we propose ActionUNet, an efficient multi-scale fine-tuning framework that significantly enhances the robustness of pre-trained VLA models with very low training and inference costs. Specifically, ActionUNet first constructs a lightweight temporal U-Net architecture within the temporal-aligned action feature space, injecting explicit multi-scale priors into the action generation process through hierarchical feature encoding and decoding. Then, ActionUNet introduces a conditional SIREN as a continuous action decoder to efficiently leverage the pre-trained VLA's action features. This architecture directly incorporates these action features as conditional inputs to preserve intrinsic temporal alignment, while simultaneously mitigating high-frequency motion jitter via explicit second-order smoothness constraints to guarantee temporal continuity.

By integrating the hierarchical structural priors from the temporal U-Net with the continuous temporal constraints of the conditional SIREN, ActionUNet effectively bridges the scale gap between coarse-grained multi-modal semantics and fine-grained physical execution. This ensures the generation of smooth, jitter-resistant action trajectories to enhance the model's resilience against environmental perturbations, while fully leveraging the pre-trained VLA to improve fine-tuning efficiency.

Our main contributions are summarized as follows:

- We propose ActionUNet, a simple-but-effective multi-scale fine-tuning framework that constructs a temporal U-Net to endow pre-trained VLA models with hierarchical structural priors, effectively mitigating the scale mismatch problem at a minimal computational cost.
- We design a conditional SIREN decoder for continuous action generation, which efficiently leverages the multi-scale fused action features as conditional inputs. This design preserves

intrinsic temporal alignment while imposing explicit second-order smoothness constraints to guarantee temporal continuity and mitigate high-frequency motion jitter.

- Extensive experiments on RoboTwin 2.0, LIBERO-Plus, and real-world dual-arm manipulation with hard setting demonstrate that ActionUNet significantly improves $\pi_{0.5}$ success rates by 9.8%, 6.1%, and 11.4% in hard/perturbed settings, respectively. Its consistent gains on OpenVLA-OFT further highlight its effectiveness and compatibility as an efficient fine-tuning strategy.

2 Related Work

Vision-Language-Action Model. Vision-Language-Action (VLA) models have emerged as a promising paradigm for robotic manipulation, typically building upon pre-trained Vision-Language Models (VLMs) [12, 13, 14] to map visual and linguistic inputs to actions. Existing VLA architectures can be broadly categorized into two families: autoregressive models that discretize continuous actions into tokens [15, 1, 2, 16], and diffusion-based or flow-matching-based models that generate continuous action chunks [17, 3, 18]. Despite these advances, robustness remains a key concern; prior work has shown that VLA models are vulnerable to visual corruptions [19], and existing robustification strategies often rely on external large models or focus solely on visual perturbations [20, 21]. Meanwhile, most VLA models adopt a single-scale action representation head and generate action sequences without explicit multi-scale or smoothness priors.

Multi-scale Visuomotor Policy Learning. Robotic manipulation requires modeling actions at multiple temporal scales, since a policy must simultaneously capture long-horizon task progression and fine-grained local control. Existing visuomotor approaches can be broadly grouped into two categories. The first category learns discrete multi-scale representations of actions, often through vector-quantized tokenization, and performs hierarchical or scale-wise prediction [22, 23]. While such discretization can improve long-horizon reasoning and autoregressive efficiency, it introduces an information bottleneck that may hinder the transfer of rich visual features to downstream action prediction [24]. The second category operates in continuous action space and introduces multi-scale structure either within the action generation process, by predicting action chunks at multiple resolutions [25] or decomposing actions into frequency components [26], or through multi-scale perception and hierarchical policy architectures that fuse features across sensor modalities [27, 28, 29]. These methods, together with VLA-based multi-scale prediction that requires full-model retraining [30], demonstrate that coordinating global planning with local refinement is crucial, but their specially designed policy heads or training paradigms limit direct reuse of strong pre-trained VLA priors. In contrast, we inject multi-scale structure directly into the action feature space, thereby bypassing the discretization bottleneck and the need for architecturally specialized heads, while retaining the base model's generalization with low fine-tuning overhead.

3 Method

3.1 Preliminary

In standard VLA fine-tuning, the action head inherits its weights from pre-training and decodes each timestep independently from high-level multi-modal features into a continuous action. However, this single-scale mapping often fails to bridge the gap between coarse-grained multi-modal semantics and the multi-scale, fine-grained requirements of physical execution, leading to the scale mismatch issue.

Our method, referred to as **ActionUNet**, addresses this issue by introducing two key modules: a temporal U-Net in the temporal-aligned action feature space that restores multi-scale structure, and a continuous action decoder that imposes explicit smoothness constraints. The temporal U-Net enriches the backbone action features with hierarchical priors, jointly capturing coarse task semantics and fine local control cues. These refined features are then decoded into a continuous-time trajectory via local SIREN fields and Matérn-weighted aggregation, producing actions that are both spatially precise and temporally smooth. By constructing the multi-scale prior directly in the continuous feature space, ActionUNet bridges the scale gap between coarse-grained multi-modal semantics and fine-grained actions, helps recover the multi-scale temporal cues needed for robust manipulation, and guarantees temporal continuity.

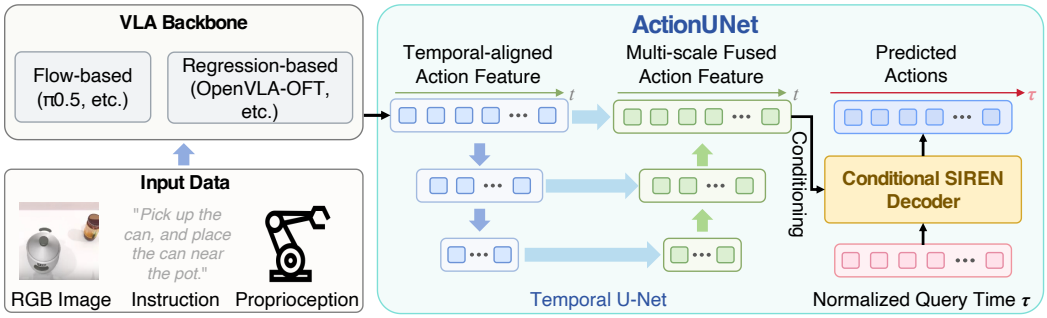

Figure 2: Overview of ActionUNet. Given input observations and instructions, the VLA backbone first produces temporal-aligned action features. ActionUNet refines these features in the action feature space through multi-scale temporal mapping, and then decodes the fused features with a conditional SIREN into smooth continuous action trajectories.

3.2 Temporal U-Net in Action Feature Space

To address the scale mismatch and recover the fine-grained dynamics lacking in standard fine-tuning approaches, ActionUNet constructs a lightweight temporal U-Net that explicitly encodes multi-scale priors in the temporal-aligned action feature space. Let the action feature be denoted as $\mathbf{h}^{(0)} \in \mathbb{R}^{H\times d}$, where H is the action horizon and d is the feature dimension. The temporal U-Net progressively encodes the action feature into lower-resolution temporal representations and subsequently decodes it back to the original resolution. Skip connections recover fine-grained local details during reconstruction. The whole process is represented as:

$$\mathbf{h}^{(\ell+1)} = \mathrm{LN}\Big(\mathrm{GeLU}\big(\mathrm{Conv1D}^{(\ell)}(\mathbf{h}^{(\ell)})\big)\Big), \quad \ell = 0, \dots, L-1, \tag{1}$$

$$\tilde{\mathbf{h}}^{(\ell)} = \mathrm{Fuse}^{(\ell)}\Big(\mathrm{Up}\big(\tilde{\mathbf{h}}^{(\ell+1)}\big),\ \mathbf{h}^{(\ell)}\Big), \quad \ell = L-1, \dots, 0. \tag{2}$$

In Eq. (1), $\mathrm{Conv1D}^{(\ell)}$ denotes a one-dimensional convolution with a downsampling rate r that halves the temporal resolution at each stage. The output is passed through a GeLU activation followed by Layer Normalization. Starting from $\mathbf{h}^{(0)} \in \mathbb{R}^{H\times d}$, after L stages the hidden state $\mathbf{h}^{(L)}$ resides in $\mathbb{R}^{\lfloor H/r^L \rfloor \times d}$. A fully connected layer transforms $\mathbf{h}^{(L)}$ into $\tilde{\mathbf{h}}^{(L)}$. Eq. (2) restores temporal resolution in a coarse-to-fine manner. At each stage ℓ, the current hidden state is first upsampled by a factor of two using nearest-neighbor interpolation ($\mathrm{Up}(\cdot)$). The upsampled feature is then concatenated with the corresponding encoder skip feature $\mathbf{h}^{(\ell)}$ and processed by a fusion block $\mathrm{Fuse}^{(\ell)}$, which consists of a single linear layer. The final output $\tilde{\mathbf{h}}^{(0)} \in \mathbb{R}^{H\times d}$ serves as the refined action feature.

3.3 Conditional SIREN Decoder for Continuous Action Generation

Rather than directly mapping each refined feature vector $\tilde{\mathbf{h}}^{(0)}$ to an isolated action step, we interpret the action chunk as samples from a temporal-continuous trajectory over a normalized time $\tau \in [-1, 1]$. To achieve this goal, we propose the conditional SIREN decoder, where the decoder produces a smooth function that can represent either the action trajectory itself (regression mode) or the flow-matching vector field (flow mode), depending on the training objective. This continuous formulation is motivated by standard robot trajectory-generation practice, where the robot's actions are often represented as smooth twice-differentiable or piecewise C^2 continuous curves to ensure continuity [31].

Local SIREN Fields. Let $\tilde{\mathbf{h}}_1^{(0)}, \dots, \tilde{\mathbf{h}}_H^{(0)} \in \mathbb{R}^d$ be the refined action features from the temporal U-Net, each associated with a uniformly spaced normalized time $\tau_1, \dots, \tau_H \in [-1, 1]$. For i-th feature $\tilde{\mathbf{h}}_i^{(0)}$, we construct a local implicit field $\Phi_i(\tau)$ that continuously maps the query time τ to the action space, conditioned on $\tilde{\mathbf{h}}_i^{(0)}$. To endow every layer with explicit temporal awareness, we replace the standard static biases of SIREN [32] with time-dependent biases defined as simple affine functions

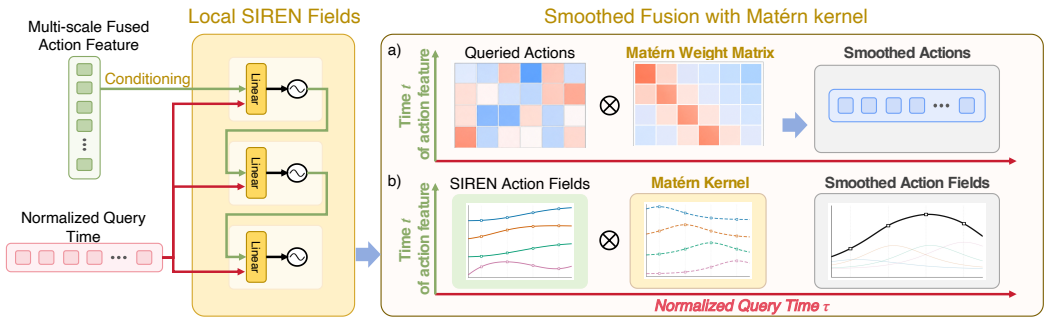

Figure 3: Overview of the conditional SIREN decoder. Local SIREN fields conditioned on multi-scale action features are queried over continuous time and fused with Matérn-kernel weights to generate temporally smooth actions.

of τ. Formally, each hidden layer is a mapping

$$\phi_m^\tau : \mathbb{R}^d \to \mathbb{R}^d, \quad \phi_m^\tau(\mathbf{z}) = \sin\big(\mathbf{W}_m \mathbf{z} + \mathbf{U}_m \tau + \mathbf{b}_m\big), \quad m = 0, \dots, M-1, \tag{3}$$

where $\mathbf{W}_m \in \mathbb{R}^{d\times d}$, $\mathbf{U}_m \in \mathbb{R}^{d\times 1}$, and $\mathbf{b}_m \in \mathbb{R}^d$ are learnable parameters. Starting from the refined feature $\tilde{\mathbf{h}}_i^{(0)}$, the local SIREN field is the composition of these layers followed by a linear readout:

$$\Phi_i(\tau) \;=\; \mathbf{W}_M \, (\phi_{M-1}^\tau \circ \cdots \circ \phi_0^\tau)(\tilde{\mathbf{h}}_i^{(0)}) \;+\; \mathbf{b}_M, \tag{4}$$

with $\mathbf{W}_M \in \mathbb{R}^D \times d$, $\mathbf{b}_M \in \mathbb{R}^D$ where D is the action dimension. Because every ϕ_m^τ composes an affine transformation with the analytic sine, and the time dependence enters solely through the parameter $\mathbf{b}_m(\tau)$, the overall function $\Phi_i(\tau)$ remains infinitely differentiable with respect to τ, inheriting the smoothness guarantees of SIREN. This formulation preserves the architectural simplicity of the original SIREN while seamlessly embedding temporal τ information into every hidden layer.

Smooth Fusion with Matérn Kernel. The global continuous function $\psi(\tau)$ is obtained by aggregating the local SIREN fields with a normalized Matérn-$5/2$ kernel [33] of length-scale parameter ρ as

$$\psi(\tau) = \sum_{i=1}^{H} \alpha_i(\tau)\, \Phi_i(\tau), \quad \text{s.t.} \quad \alpha_i(\tau) \propto \kappa_{5/2}(|\tau - \tau_i|/\rho). \tag{5}$$

This soft assignment encourages second-order temporal smoothness across neighboring feature positions, reducing high-frequency inconsistencies while preserving local expressiveness. The aggregated output $\{\psi(\tau_j)\}_{j=1}^H$ is interpreted as the predicted action chunk (regression) or as the estimated velocity field (flow matching), depending on the training objective of the base policy.

3.4 Training Objective

ActionUNet is compatible with both direct regression and flow-matching policies. We keep the pre-trained backbone frozen and only fine-tune the action expert with ActionUNet, so the method can be trained under the original objective of the base policy.

Regression Objective. For direct regression, we supervise the output sequence $\{\psi(\tau_j)\}_{j=1}^H$ with the ground-truth action chunk $\mathbf{a}_{1:H}$ using an ℓ_1 loss:

$$\mathcal{L}_{\text{reg}} = \frac{1}{H} \sum_{j=1}^{H} \big\| \psi(\tau_j) - \mathbf{a}_j \big\|_1. \tag{6}$$

Flow-matching Objective. For flow-based training, the decoder $\psi(\tau)$ is trained to predict the conditional vector field that transports noise to data. Following the standard conditional flow-matching formulation [34], we construct a linear interpolation path between a noise sequence $\epsilon_{1:H}$ and the clean action chunk $\mathbf{a}_{1:H}$. Given an observation and the noisy trajectory $\epsilon_{1:H}$, the model predicts $\psi(\tau_j)$, and we minimize the mean squared error against this velocity:

$$\mathcal{L}_{\text{fm}} = \frac{1}{H} \sum_{j=1}^{H} \big\| \psi(\tau_j) - (\mathbf{a}_j - \epsilon_j) \big\|_2^2. \tag{7}$$

Temporally Correlated Flow Noise. To align the flow target with the smoothness prior of our continuous decoder, we replace the standard white Gaussian noise with Matérn-$5/2$ process noise. Concretely, we sample noise vectors with a temporal covariance $K_{ij} = \kappa_{5/2}(|i-j|/\sigma)+\varepsilon\delta_{ij}$, where σ controls the correlation length and ε ensures numerical stability. This correlated perturbation reduces high-frequency jitter in the transport target and better matches the C^2 continuity induced by the SIREN decoder and Matérn aggregation.

4 Experiments

4.1 Experimental Setup

Simulation Benchmarks. We evaluate ActionUNet on three simulation benchmarks:

1) **RoboTwin 2.0** [35] contains 50 bimanual tasks, each with easy and hard settings (varying distractors, backgrounds, lighting, table heights, and language). We select 12 tasks covering diverse temporal scale compositions. Each single-task policy is trained on 50 clean demonstrations and evaluated over 100 rollouts in both settings. 2) **LIBERO** [36] comprises four suites (Spatial, Object, Goal, Long), each with 10 tasks and 50 demonstrations. We train on a mixture from all suites. 3) **LIBERO-Plus** [37] extends LIBERO with seven perturbation types (camera, robot state, language, lighting, background, sensor noise, layout). Models trained on LIBERO are directly evaluated on LIBERO-Plus to measure robustness. All tasks adopt the average success rate as the evaluation metric.

Real-World Robot Experiments. We conduct real-world experiments on the Realman Gen72 dual-arm platform. Three Intel RealSense D435 cameras are used: two wrist-mounted cameras placed on the end-effectors and one head camera positioned between the two arms, providing multi-view observations of the workspace. We train and evaluate models on three dual-arm manipulation tasks: Place Object Basket, Stack Blocks Two, and Stack Bowls Two. To test real-world robustness, we consider two evaluation settings: an easy setting matching the training environment and a hard setting with additional distractor objects.

Implementation Details. We apply ActionUNet to both $\pi_{0.5}$, a flow-based policy, and OpenVLA-OFT, a regression-based policy, to evaluate its compatibility with different VLA frameworks. For ActionUNet, we set the temporal pyramid downsampling rate $r = 2$, the Matérn kernel length-scale $\rho = 0.2$, and use standard SIREN initialization [32]. We reproduce $\pi_{0.5}$ following the official PyTorch version and OpenVLA-OFT baselines. All other baseline results are cited from [35, 38, 37]. All methods adopt the same dataset-specific training data, inference setting, and evaluation protocol. Unless otherwise specified, fine-tuning hyperparameters follow the settings reported in the original papers or official benchmark reports.

For RoboTwin 2.0, $\pi_{0.5}$ and $\pi_{0.5}$+ActionUNet are trained on 2×RTX 4090 GPUs for 8K steps with a batch size of 64 and a 5×10^{-5} learning rate (cosine decay). For LIBERO and LIBERO-Plus, they are trained on 8×A800 GPUs for 40K steps with a batch size of 256. OpenVLA-OFT and OpenVLA-OFT+ActionUNet follow official settings, except OpenVLA-OFT+ActionUNet on LIBERO uses a reduced learning rate of 5×10^{-5} for stable fine-tuning.

4.2 Results on RoboTwin 2.0

Table 1 reports the results on RoboTwin 2.0, where tasks are grouped according to their scale composition. Based on $\pi_{0.5}$, ActionUNet consistently improves performance across both easy and hard settings. Averaged over all 12 tasks, ActionUNet significantly improves the success rate by 8.8% in the easy setting, and 9.8% in the hard setting. The improvement under the hard setting is especially important, since this setting introduces stronger environmental perturbations and requires the policy to maintain reliable action execution under distractors.

The gains are consistent across scale compositions. On single-scale tasks, ActionUNet improves the average success rate from 66.0% to 72.7% and from 34.0% to 44.0% in the easy and hard settings, respectively. On dual-scale tasks, ActionUNet improves the average success rate from 60.2% to 69.5% and from 29.7% to 40.0% under easy and hard settings, respectively. On three-scale tasks, where successful execution requires coordinating more diverse temporal action patterns, ActionUNet improves the average success rate from 52.0% to 62.0% and from 25.7% to 34.3% in the

Table 1: Task-wise success rates on RoboTwin 2.0 grouped by scale composition.

Task	RDT [7]		π_0 [17]		ACT [39]		DP3 [40]		$\pi_{0.5}$ [4]		$\pi_{0.5}$+ActionUNet	
	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard	Easy	Hard
Single-scale Tasks												
Pick Dual Bottles	42	13	57	12	31	0	60	1	65	34	**71**	**39**
Handover Mic	90	31	98	13	85	0	**100**	3	**100**	55	**100**	**76**
Handover Block	45	14	45	8	42	0	**70**	0	33	13	47	**17**
Dual-scale Tasks												
Beat Block Hammer	77	37	43	21	56	3	72	8	77	28	**89**	**43**
Move Can Pot	25	12	58	21	22	4	70	6	66	52	**77**	**69**
Place A2B Left	3	1	31	1	1	0	46	2	46	7	**55**	**10**
Place Object Stand	15	5	36	11	1	0	60	0	51	29	**62**	**38**
Place Phone Stand	15	6	35	7	2	0	44	2	47	21	**61**	**27**
Press Stapler	41	24	62	29	31	6	69	3	**74**	41	73	**53**
Three-scale Tasks												
Stack Blocks Two	21	2	42	1	25	0	24	0	68	24	**74**	**35**
Blocks Ranking RGB	3	0	19	5	1	0	3	0	36	16	**50**	**24**
Put Bottles Dustbin	21	4	54	13	27	1	60	21	52	37	**62**	**44**
Average	33.2	12.4	48.3	11.8	27.0	1.2	56.5	3.8	59.6	29.8	**68.4**	**39.6**

Table 2: LIBERO benchmark results. We report the average success rate (%) across four LIBERO task suites.

Method	SPATIAL	OBJECT	GOAL	LONG	Avg.
Diffusion Policy [41]	78.3	92.5	68.3	50.5	72.4
WorldVLA [42]	87.6	96.2	83.4	60.0	81.8
SmolVLA [43]	93.0	94.0	91.0	77.0	88.8
π_0 [17]	96.8	98.8	95.8	85.2	94.2
π_0-FAST [16]	96.4	96.8	88.6	60.2	85.5
UniVLA [44]	96.5	96.8	95.6	92.0	95.2
VLA-Adapter [38]	97.8	99.2	97.2	95.0	97.3
OpenVLA-OFT [8]	98.4	99.0	98.4	95.8	97.9
OpenVLA-OFT+**ActionUNet**	**99.2**	99.0	**99.6**	**96.4**	**98.6**
$\pi_{0.5}$ [4]	95.4	98.4	97.0	91.6	95.6
$\pi_{0.5}$**+ActionUNet**	98.6	**99.4**	98.8	93.8	97.7

Table 3: Performance under environmental perturbations on LIBERO-Plus. All models are trained on LIBERO and evaluated on LIBERO-Plus. We report success rates (%) under 7 perturbation types.

Method	Camera	Robot	Lang.	Light	Back.	Noise	Layout	Avg.
OpenVLA [2]	0.8	3.5	23.0	8.1	34.8	15.2	28.5	15.6
WorldVLA [42]	0.1	27.9	41.6	43.7	17.1	10.9	38.0	25.0
UniVLA [44]	1.8	46.2	69.6	69.0	81.0	21.2	31.9	43.9
π_0 [17]	13.8	6.0	58.8	85.0	81.4	**79.0**	68.9	53.6
π_0-FAST [16]	**65.1**	21.6	61.0	73.2	73.2	74.4	68.8	61.6
OpenVLA-OFT [8]	36.0	35.9	67.0	82.1	93.1	47.1	82.5	60.9
OpenVLA-OFT+**ActionUNet**	41.2	40.8	68.3	83.4	**93.7**	54.5	85.3	64.6
$\pi_{0.5}$ [4]	48.2	45.9	68.8	93.0	87.4	52.0	81.7	65.8
$\pi_{0.5}$**+ActionUNet**	56.9	**54.4**	**73.3**	**94.6**	89.8	61.1	**85.7**	**71.9**

easy and hard settings, respectively. These results show that multi-scale temporal action modeling is beneficial not only for simple action patterns but also for tasks involving more complex scale composition.

4.3 Results on LIBERO and LIBERO-Plus

Table 2 reports results on the standard LIBERO benchmark. With $\pi_{0.5}$, ActionUNet improves the performance of all four suites, achieving an average success rate of 97.7% against 95.6%. This improvement extends to the regression-based OpenVLA-OFT backbone (98.6% vs 97.9%), con-

Table 4: Comparison of our method against the baseline under easy and hard conditions in real-world experiments.

Task	$\pi_{0.5}$		$\pi_{0.5}$+ActionUNet	
	Easy	**Hard**	**Easy**	**Hard**
Place Object Basket	60	42	78	52
Stack Bowls Two	88	62	96	74
Stack Blocks Two	80	68	92	80
Average	76.0	57.3	**88.7**	**68.7**

Table 5: Ablation results on four RoboTwin 2.0 tasks. Each task reports success rates under easy and hard settings. All non-baseline variants are built on $\pi_{0.5}$.

Task	$\pi_{0.5}$		+ U-Net		+ SIREN		+ ActionUNet w/o Matérn		+ ActionUNet	
	Easy	**Hard**	**Easy**	**Hard**	**Easy**	**Hard**	**Easy**	**Hard**	**Easy**	**Hard**
Move Can Pot	66	52	64	56	73	53	73	68	**77**	**69**
Stack Blocks Two	68	24	**75**	27	74	24	71	34	74	**35**
Beat Block Hammer	77	28	65	33	87	30	**89**	39	**89**	**43**
Pick Dual Bottles	65	34	66	33	68	26	67	**40**	**71**	39
Average	69.0	34.5	67.5	37.3	75.5	33.3	75.0	45.3	**77.8**	**46.5**
Gain over $\pi_{0.5}$	–	–	-1.5	+2.8	+6.5	-1.2	+6.0	+10.8	**+8.8**	**+12.0**

firming that ActionUNet generalizes across VLA models while preserving strong in-distribution performance.

Table 3 evaluates the robustness of ActionUNet and comparison methods under controlled perturbations on LIBERO-Plus. $\pi_{0.5}$+ActionUNet improves the average success rate from 65.8% to 71.9%, with consistent gains across all seven perturbation types, particularly for camera, robot-state, and sensor-noise perturbations. This result demonstrates that the temporal U-Net refines action features for multi-scale representations, and the continuous decoder reduces high-frequency inconsistency to benefit the robotic execution. With the more powerful regression-based OpenVLA-OFT, ActionUNet still improves the average success rate from 60.9% to 64.6%, indicating its robustness across VLA frameworks.

4.4 Real-World Experiments

Evaluation Protocol. We select three representative tasks from the RoboTwin 2.0 benchmark: place Object Basket, Stack Blocks Two, and Stack Bowls Two. Each task contains easy and hard evaluation settings. The easy setting is the same as the training settings. For the hard setting, we randomly add some distractor objects to the environment to introduce visual and physical interference. We report the success rate as the percentage of successful trials out of the 50 rollouts.

Results. Table 4 reports the real-world evaluation results on the Realman Robot. ActionUNet consistently improves the baseline across all three tasks under both easy and hard evaluation settings. The average success rate is significantly improved from 76.0% to 88.7% in the easy setting and from 57.3% to 68.7% in the hard setting. These results demonstrate its robustness in complex real-world environments without compromising performance in standard scenes.

4.5 Ablation Study

Module Ablation. We ablate ActionUNet with respect to the temporal U-Net, the conditional SIREN decoder, and the Matérn-weighted aggregation in Table 5. Using only the temporal U-Net on $\pi_{0.5}$ improves hard-setting performance from 34.5% to 37.3% while slightly dropping in the easy setting. This suggests that while the temporal U-Net enhances robustness against distractors, it compromises the spatial continuity of actions. In contrast, by adding only the SIREN decoder, the success rate in the easy setting is improved from 69.0% to 75.5% while dropping in the hard setting. It indicates the SIREN decoder is capable of maintaining the spatial continuity of actions

to suppress trajectory jitter, but fails to capture multi-scale action representations, leading to a lack of robustness against distractors. The combination of U-Net and SIREN without the Matérn kernel (ActionUNet w/o Matérn) achieves 75.0% and 45.3% success rates in easy and hard settings, significantly outperforming the baseline by 6.0% and 10.8%, demonstrating that multi-scale modeling, aided by SIREN to mitigate action discontinuity, effectively provides crucial robustness against distractors. Incorporating the Matérn kernel further increases performance to 77.8% and 46.5% in easy and hard settings, suggesting that the Matérn kernel supplies a lightweight smoothness prior over neighboring local fields, complementing the learned components.

Effect of Multi-scale Fine-tuning. We compute the response of the 1st, 4th, and 6th layers of temporal U-Net to reflect fine-, mid-, and coarse-scale action feature activations using Grad-CAM on RoboTwin 2.0. The responses are grouped into five bins by ground-truth action scales, and Grad-CAM responses are min-max normalized and averaged per bin. As shown in Fig. 4, action features at each scale respond most strongly to the corresponding ground-truth action scales, suggesting that ActionUNet learns scale-aware temporal representations.

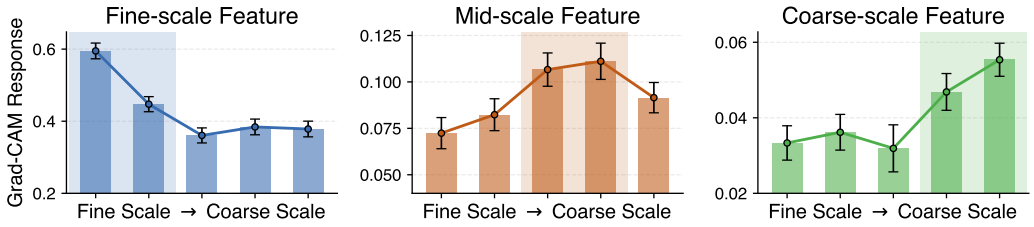

Figure 4: Scale-conditioned temporal Grad-CAM responses on RoboTwin 2.0. Fine-, mid-, and coarse-resolution temporal features show stronger responses to the corresponding action scales.

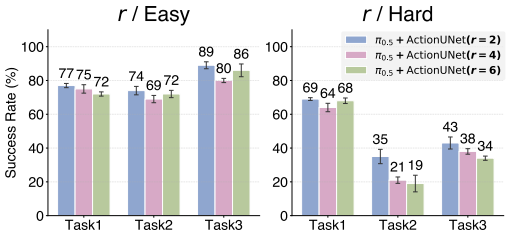

Figure 5: Ablation study on the downsampling rate r. Task1, Task2, and Task3 correspond to Move Can Pot, Stack Blocks Two, and Beat Block Hammer, respectively.

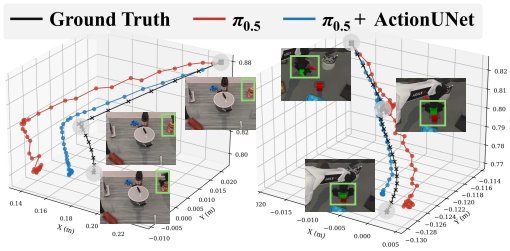

Figure 6: Predicted end-effector action chunks under the same initial settings. ActionUNet produces actions closer to the ground-truth with smoother local action than $\pi_{0.5}$.

Effect of Hyper-parameters r. Fig. 5 shows that the downsampling rate $r = 2$ in the temporal U-Net performs best under both easy and hard settings. Smaller r provides denser intermediate scales for gradual coarse-to-fine refinement, which is especially beneficial in hard settings where contact-sensitive actions are required.

Visualization Analysis. As shown in Fig. 6, we further visualize predicted end-effector action chunks under the same initial setting. Compared with $\pi_{0.5}$, ActionUNet produces smaller spatial deviation and smoother trajectories in these examples. This demonstrates ActionUNet's effectiveness in generating precise and stable continuous actions.

5 Conclusion

We propose ActionUNet, a lightweight multi-scale fine-tuning module for temporal action mapping in pre-trained VLA policies. ActionUNet injects hierarchical temporal priors into the action feature space through a temporal U-Net, and decodes the refined features with a conditional SIREN decoder to preserve smooth action trajectories. This design provides an architecture-compatible interface for both flow-based and regression-based policies, while introducing negligible computational overhead. On RoboTwin 2.0, LIBERO, LIBERO-Plus, and real-world dual-arm experiments,

ActionUNet consistently improves task success and robustness under environmental distractions, showing the effectiveness of multi-scale temporal mapping for efficient VLA adaptation.

Limitations and Future Work.

ActionUNet currently uses fixed temporal scales, models action chunks independently, and introduces scale-aware modeling only at downstream action stage. Future work will explore adaptive scales, cross-chunk temporal modeling, and scale-aware VLA pre-training.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant 62472402 and 62176260, and the Fundamental Research Funds for the Central Universities (E2ET1104).

The authors declare no competing interests.

References

[1] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Conf. Robot Learn.*, 2023, pp. 2165–2183.

[2] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong *et al.*, "Openvla: An open-source vision-language-action model," in *Conf. Robot Learn.*, 2025, pp. 2679–2713.

[3] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang *et al.*, "Gr00t n1: An open foundation model for generalist humanoid robots," *arXiv preprint arXiv:2503.14734*, 2025.

[4] K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. R. Equi, C. Finn, N. Fusai, M. Y. Galliker *et al.*, "$\pi_{0.5}$: a vision-language-action model with open-world generalization," in *Conf. Robot Learn.*, 2025.

[5] G. Ma, S. Wang, Z. Zhang, S. Yu, and H. Tang, "Generalvla: Generalizable vision-language-action models with knowledge-guided trajectory planning," *arXiv preprint arXiv:2602.04315*, 2026.

[6] A. Rasouli, M. Alban, S. Pakdamansavoji, Z. Li, Z. Zhang, A. Wu, and X. Zhao, "Distracted robot: How visual clutter undermine robotic manipulation," *arXiv preprint arXiv:2511.22780*, 2025.

[7] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu, "Rdt-1b: a diffusion foundation model for bimanual manipulation," in *Int. Conf. Learn. Represent.*, 2025.

[8] M. J. Kim, C. Finn, and P. Liang, "Fine-tuning vision-language-action models: Optimizing speed and success," in *Robot.: Sci. Syst.*, 2025.

[9] W. Yuan, J. Duan, V. Blukis, W. Pumacay, R. Krishna, A. Murali, A. Mousavian, and D. Fox, "Robopoint: A vision-language model for spatial affordance prediction in robotics," in *Conf. Robot Learn.*, 2024.

[10] E. Zhao, V. Raval, H. Zhang, J. Mao, Z. Shangguan, S. Nikolaidis, Y. Wang, and D. Seita, "Manipbench: Benchmarking vision-language models for low-level robot manipulation," in *Conf. Robot Learn.*, 2025.

[11] Y. Liu, H. Yu, J. Zhao, B. Li, D. Zhang, M. Li, W. Wu, Y. Hu, J. Xie, J. Guo *et al.*, "Learning native continuation for action chunking flow policies," *arXiv preprint arXiv:2602.12978*, 2026.

[12] L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, D. Salz, M. Neumann, I. Alabdulmohsin, M. Tschannen, E. Bugliarello *et al.*, "Paligemma: A versatile 3b vlm for transfer," *arXiv preprint arXiv:2407.07726*, 2024.

[13] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," *Adv. Neural Inform. Process. Syst.*, vol. 36, pp. 34 892–34 916, 2023.

[14] P. Wang, S. Bai, S. Tan, S. Wang, Z. Fan, J. Bai, K. Chen, X. Liu, J. Wang, W. Ge *et al.*, "Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution," *arXiv preprint arXiv:2409.12191*, 2024.

[15] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," in *Robot.: Sci. Syst.*, 2023.

[16] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine, "Fast: Efficient action tokenization for vision-language-action models," *arXiv preprint arXiv:2501.09747*, 2025.

[17] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, "π_0: A vision-language-action flow model for general robot control," in *Robot.: Sci. Syst.*, 2025.

[18] Q. Li, Y. Liang, Z. Wang, L. Luo, X. Chen, M. Liao, F. Wei, Y. Deng, S. Xu, Y. Zhang *et al.*, "Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation," *arXiv preprint arXiv:2411.19650*, 2024.

[19] Z. Wang, Z. Zhou, J. Song, Y. Huang, Z. Shu, and L. Ma, "Vlatest: Testing and evaluating vision-language-action models for robotic manipulation," *Proceedings of the ACM on Software Engineering*, vol. 2, no. FSE, pp. 1615–1638, 2025.

[20] A. J. Hancock, A. Z. Ren, and A. Majumdar, "Run-time observation interventions make vision-language-action models more visually robust," in *Int. Conf. Robot. Autom.*, 2025, pp. 9499–9506.

[21] H. Zhang, P. Ding, S. Lyu, Y. Peng, and D. Wang, "Gevrm: Goal-expressive video generation model for robust visual manipulation," *arXiv preprint arXiv:2502.09268*, 2025.

[22] Z. Gong, P. Ding, S. Lyu, S. Huang, M. Sun, W. Zhao, Z. Fan, and D. Wang, "Carp: Visuomotor policy learning via coarse-to-fine autoregressive prediction," in *Int. Conf. Comput. Vis.*, 2025, pp. 13 460–13 470.

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

[24] T. Shiba, "The compression gap: Why discrete tokenization limits vision-language-action model scaling," *arXiv preprint arXiv:2604.03191*, 2026.

[25] D. Yashima, K. Seno, S. Kurita, Y. Oda, and K. Sugiura, "Hiflow: Tokenization-free scale-wise autoregressive policy learning via flow matching," *arXiv preprint arXiv:2603.27281*, 2026.

[26] J. Zhang, Z. Han, J. Wang, X. Wu, S. Lin, J. Li, H. Fan, R. Wu, D. Li, and H. Dong, "Hipolicy: Hierarchical multi-frequency action chunking for policy learning," *arXiv preprint arXiv:2604.06067*, 2026.

[27] Y. Su, X. Zhan, H. Fang, H. Xue, H.-S. Fang, Y.-L. Li, C. Lu, and L. Yang, "Dense policy: Bidirectional autoregressive learning of actions," *arXiv preprint arXiv:2503.13217*, 2025.

[28] Y. Lu, Y. Tian, Z. Yuan, X. Wang, P. Hua, Z. Xue, and H. Xu, "H^3dp: Triply-hierarchical diffusion policy for visuomotor learning," in *Int. Conf. Learn. Represent.*, 2026.

[29] H. Xue, J. Ren, W. Chen, G. Zhang, Y. Fang, G. Gu, H. Xu, and C. Lu, "Reactive diffusion policy: Slow-fast visual-tactile policy learning for contact-rich manipulation," *arXiv preprint arXiv:2503.02881*, 2025.

[30] R. Huang, C. Zeng, W. Tang, J. Cai, C. Lu, and P. Cai, "Mimic intent, not just trajectories," *arXiv preprint arXiv:2602.08602*, 2026.

[31] S. Kim, J. Kim, and J. J. Lim, "Time optimal execution of action chunk policies beyond demonstration speed," in *Int. Conf. Learn. Represent.*, 2026.

[32] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *Adv. Neural Inform. Process. Syst.*, vol. 33, pp. 7462–7473, 2020.

[33] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[34] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *Int. Conf. Learn. Represent.*, 2023.

[35] T. Chen, Z. Chen, B. Chen, Z. Cai, Y. Liu, Z. Li, Q. Liang, X. Lin, Y. Ge, Z. Gu *et al.*, "Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation," *arXiv preprint arXiv:2506.18088*, 2025.

[36] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, "Libero: Benchmarking knowledge transfer for lifelong robot learning," *Adv. Neural Inform. Process. Syst.*, vol. 36, pp. 44 776–44 791, 2023.

[37] S. Fei, S. Wang, J. Shi, Z. Dai, J. Cai, P. Qian, L. Ji, X. He, S. Zhang, Z. Fei *et al.*, "Libero-plus: In-depth robustness analysis of vision-language-action models," *arXiv preprint arXiv:2510.13626*, 2025.

[38] Y. Wang, P. Ding, L. Li, C. Cui, Z. Ge, X. Tong, W. Song, H. Zhao, W. Zhao, P. Hou *et al.*, "Vla-adapter: An effective paradigm for tiny-scale vision-language-action model," in *AAAI Conf. Artif. Intell.*, vol. 40, no. 22, 2026, pp. 18 638–18 646.

[39] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," in *Robot.: Sci. Syst.*, 2023.

[40] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations," in *Robot.: Sci. Syst.*, 2024.

[41] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *Int. J. Robot. Res.*, vol. 44, no. 10-11, pp. 1684–1704, 2025.

[42] J. Cen, C. Yu, H. Yuan, Y. Jiang, S. Huang, J. Guo, X. Li, Y. Song, H. Luo, F. Wang *et al.*, "Worldvla: Towards autoregressive action world model," *arXiv preprint arXiv:2506.21539*, 2025.

[43] M. Shukor, D. Aubakirova, F. Capuano, P. Kooijmans, S. Palma, A. Zouitine, M. Aractingi, C. Pascal, M. Russi, A. Marafioti *et al.*, "Smolvla: A vision-language-action model for affordable and efficient robotics," *arXiv preprint arXiv:2506.01844*, 2025.

[44] Q. Bu, Y. Yang, J. Cai, S. Gao, G. Ren, M. Yao, P. Luo, and H. Li, "Univla: Learning to act anywhere with task-centric latent actions," in *Robot.: Sci. Syst.*, 2025.

[45] T. Chen, Y. Chen, Z. Li, J. Tang, K. Su, H. Lu, W. Wan, B. Chen, S. Liu, H. Yan, H. Su, Z. Dou, K. Wang, D. Zhang, Y. Liu, Y. Qin, Q. Liang, Q. Wu, Z. Lin, W. Lin, Y. Wang, M. He, T. Wu, R. Wu, J. Zhou, K.-C. Lei, H. Yu, Y. Ji, W. Jin, G. Lin, X. Li, Q. Xiong, R. Xu, Z. Li, W. Chai, E. Xie, Z. Wang, Y. Mu, H. Dong, W. Matusik, M. Ding, W. Ding, P. Luo, and M. Tomizuka, "Robodojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies," *arXiv preprint arXiv:2607.04434*, 2026.

A Additional Method Details

Conditional SIREN Decoder. Given the refined action features $\tilde{\mathbf{h}}_1^{(0)}, \ldots, \tilde{\mathbf{h}}_H^{(0)}$, ActionUNet represents an action chunk as samples from a continuous function over the normalized time axis $\tau \in [-1, 1]$. The temporal anchors are uniformly placed as

$$\tau_i = -1 + \frac{2(i-1)}{H-1}, \qquad i = 1, \ldots, H. \tag{8}$$

For each anchor i, the local SIREN field $\Phi_i(\tau)$ maps a query time τ to the action space while conditioning on the refined action feature $\tilde{\mathbf{h}}_i^{(0)}$. The final continuous function is obtained by Matérn-weighted aggregation:

$$\psi(\tau) = \sum_{i=1}^{H} \alpha_i(\tau)\Phi_i(\tau), \qquad \alpha_i(\tau) = \frac{\kappa_{5/2}(|\tau - \tau_i|/\rho)}{\sum_{j=1}^{H} \kappa_{5/2}(|\tau - \tau_j|/\rho)}. \tag{9}$$

The output sequence $\{\psi(\tau_j)\}_{j=1}^{H}$ is interpreted as the predicted action chunk in regression mode, and as the estimated flow-matching velocity field in flow mode.

Smoothness of Matérn-weighted Decoding. We verify that Matérn-$5/2$ weighting preserves second-order smoothness of the decoded function. Let $c = \sqrt{5}/\rho$. The kernel and its derivative are

$$\begin{aligned} g_\rho(t) &= \left(1 + ct + \frac{c^2 t^2}{3}\right) e^{-ct}, \\ g'_\rho(t) &= -\frac{c^2}{3} t(1 + ct) e^{-ct}. \end{aligned} \tag{10}$$

For each anchor, define $q_i(\tau) = g_\rho(|\tau - \tau_i|)$. Since $g'_\rho(0) = 0$ and $g''_\rho(0) = -c^2/3$ is finite, the left and right first and second derivatives of q_i agree at τ_i. Hence $q_i(\tau)$ is C^2 on $[-1, 1]$.

Since $q_j(\tau) > 0$ for all j and τ, the denominator $Z(\tau) = \sum_{j=1}^{H} q_j(\tau)$ is strictly positive. Therefore, each normalized weight $\alpha_i(\tau) = q_i(\tau)/Z(\tau)$ is also C^2. Each local SIREN field $\Phi_i(\tau)$ is smooth in τ, because it is composed of affine maps and sinusoidal nonlinearities. Consequently,

$$\psi(\tau) \in C^2([-1, 1]; \mathbb{R}^D). \tag{11}$$

For implementation on the uniform temporal grid, the normalized-time length-scale ρ is equivalent to an index-space length-scale $\rho_{\text{idx}} = \rho/\Delta_\tau$, where $\Delta_\tau = 2/(H-1)$.

Temporally Correlated Noise for Flow Matching. For flow-based training, we use temporally correlated Gaussian noise on the action grid. The temporal covariance matrix is defined as

$$K_{ij} = \kappa_{5/2}\left(\frac{|i-j|}{\sigma}\right) + \varepsilon\delta_{ij}, \tag{12}$$

where σ controls the correlation length and ε ensures numerical stability. Given the Cholesky decomposition $K = \mathbf{R}\mathbf{R}^\top$, we sample

$$\boldsymbol{\epsilon}_{1:H} = \mathbf{R}\boldsymbol{\eta}_{1:H}, \qquad \boldsymbol{\eta}_{1:H} \sim \mathcal{N}(0, I). \tag{13}$$

For multi-dimensional actions, this temporal covariance is applied independently to each action dimension, equivalently sampling $\boldsymbol{\epsilon}_{1:H} \sim \mathcal{N}(0, K \otimes I_D)$.

We denote the flow interpolation time by $s \in [0, 1]$ to avoid overloading the normalized action time τ. The linear interpolation path and target velocity are

$$\begin{aligned} \mathbf{x}_s &= (1-s)\boldsymbol{\epsilon}_{1:H} + s\mathbf{a}_{1:H}, \\ \mathbf{u}_s &= \frac{d\mathbf{x}_s}{ds} = \mathbf{a}_{1:H} - \boldsymbol{\epsilon}_{1:H}. \end{aligned} \tag{14}$$

The flow-mode decoder predicts this target velocity at each temporal query τ_j. This construction aligns the flow target with the second-order temporal prior used by the continuous decoder, instead of injecting temporally white perturbations into the action sequence.

Table 6: Compiled policy-call latency on a single RTX 4090 GPU. We report mean latency with 95% confidence intervals computed over repeated policy-call measurements.

Metric	$\pi_{0.5}$	$\pi_{0.5}$+**ActionUNet**	Difference
Mean latency / policy call (ms)	60.947 ± 0.251	64.886 ± 0.376	$+3.939 \quad (+6.46\%)$

Table 7: Rollout-level policy computation on the 12 RoboTwin2.0 tasks. Average taken steps are task-level unweighted means over successful rollouts. Latency values are estimated by multiplying the average taken steps by the compiled policy-call latency in Table 6; their 95% confidence intervals are propagated from the task-independent policy-call latency measurements. The paired difference is computed as $\pi_{0.5}$+**ActionUNet** minus $\pi_{0.5}$ latency, and its 95% confidence interval is propagated from the two latency confidence intervals.

Setting	Average Taken Steps		Latency (s)		
	$\pi_{0.5}$	$\pi_{0.5}$+**ActionUNet**	$\pi_{0.5}$	$\pi_{0.5}$+**ActionUNet**	Difference
Easy	238.80	238.69	14.554 ± 0.060	15.488 ± 0.090	$+0.934 \pm 0.108$
Hard	276.43	264.29	16.848 ± 0.069	17.149 ± 0.099	$+0.301 \pm 0.121$

Action Representation and Normalization. ActionUNet inherits the action representation of the backbone VLA model without modification. For $\pi_{0.5}$, actions are represented as continuous end-effector poses in the flow-matching framework. For OpenVLA-OFT, actions are represented as continuous joint/task-space commands in a regression framework.

During preprocessing, actions are normalized to zero mean and unit variance. The discrete action horizon H is mapped to the continuous normalized time axis $\tau \in [-1, 1]$ for SIREN decoding, improving training stability and gradient flow through the sinusoidal layers.

B Inference Efficiency

ActionUNet is designed as a lightweight module for pre-trained VLA policies. We report policy-level inference latency to quantify the computational overhead introduced by the proposed module.

Measurement Protocol. We measure the latency of sampling one action chunk on a single NVIDIA RTX 4090 GPU. Both $\pi_{0.5}$ and $\pi_{0.5}$+**ActionUNet** use the same input observation, action horizon of 50, and 10 flow-matching steps. All measurements are conducted under the same PyTorch inference setting with `torch.compile` enabled. We report the mean policy inference latency with 95% confidence intervals computed over repeated policy-call measurements.

These measurements correspond to policy computation only. They should not be interpreted as end-to-end robot execution latency, which is also affected by mechanical motion, low-level controller constraints, safety-limited velocity, gripper actuation, and contact-rich interactions.

Rollout-level Policy Computation. We compute the average number of taken action steps over successful rollouts on the same 12 RoboTwin2.0 main experimental tasks used in the task-wise success-rate evaluation in Table 1. We then estimate cumulative policy inference latency using the compiled policy-call latency in Table 6. This metric measures policy computation only and excludes physical robot execution time.

As shown in Table 7, $\pi_{0.5}$+**ActionUNet** introduces only a small rollout-level increase in policy computation. In the Easy setting, the cumulative policy-computation latency increases by 0.934 seconds. In the Hard setting, $\pi_{0.5}$+**ActionUNet** uses fewer average taken steps than the baseline, which partially offsets its higher per-call latency; the cumulative policy-computation latency increases by only 0.301 seconds. These results indicate that the additional policy-computation overhead remains small relative to the full successful rollout.

C Action Scale Computation

We quantify task-level and stage-level action scales using per-step end-effector displacement to characterize temporal dynamics and group RoboTwin tasks.

C.1 End-effector Displacement

Let $\mathbf{p}_t^L, \mathbf{p}_t^R \in \mathbb{R}^3$ denote the Cartesian positions of the left and right end-effectors at recorded step t. The per-step displacement is defined as

$$d_t = \left\|\mathbf{p}_{t+1}^L - \mathbf{p}_t^L\right\|_2 + \left\|\mathbf{p}_{t+1}^R - \mathbf{p}_t^R\right\|_2 .$$

Displacement is measured in m/step rather than m/s, since demonstrations are generated from planned waypoints under a fixed recording protocol. It serves as a dataset-level action-scale proxy rather than physical execution velocity.

The overall task-level scale is averaged over valid consecutive steps from successful episodes:

$$D_{\text{task}}(\mathcal{T}) = \frac{1}{|\mathcal{I}_{\mathcal{T}}|} \sum_{t \in \mathcal{I}_{\mathcal{T}}} d_t,$$

where $\mathcal{I}_{\mathcal{T}}$ denotes the set of valid step transitions from task $\mathcal{T}$.

C.2 Stage-level Scale Regimes

We segment trajectories into stages according to stable binary gripper-state transitions (0 and 1), ignoring intermediate transition states. The stage-level scale is

$$D_{\text{stage}}(s) = \frac{1}{|\mathcal{I}_s|} \sum_{t \in \mathcal{I}_s} d_t,$$

where $\mathcal{I}_s$ contains valid step transitions within stage s.

C.3 Scale Thresholds

We use empirical tertile thresholds to classify task-level and non-idle stage-level displacements into low-, medium-, and high-scale regimes.

The task-level thresholds, computed over all 50 tasks, are

$$\gamma_1^{\text{task}} = 0.03130, \qquad \gamma_2^{\text{task}} = 0.03714.$$

The task-level scale label is

$$c_{\text{task}}(\mathcal{T}) = \begin{cases} \text{Low}, & D_{\text{task}}(\mathcal{T}) < \gamma_1^{\text{task}}, \\ \text{Medium}, & \gamma_1^{\text{task}} \le D_{\text{task}}(\mathcal{T}) < \gamma_2^{\text{task}}, \\ \text{High}, & D_{\text{task}}(\mathcal{T}) \ge \gamma_2^{\text{task}}. \end{cases}$$

The stage-level thresholds, computed over all non-idle stages, are

$$\gamma_1^{\text{stage}} = 0.01998, \qquad \gamma_2^{\text{stage}} = 0.04693.$$

The stage-level scale label is

$$c_{\text{stage}}(s) = \begin{cases} \text{Low}, & D_{\text{stage}}(s) < \gamma_1^{\text{stage}}, \\ \text{Medium}, & \gamma_1^{\text{stage}} \le D_{\text{stage}}(s) < \gamma_2^{\text{stage}}, \\ \text{High}, & D_{\text{stage}}(s) \ge \gamma_2^{\text{stage}}. \end{cases}$$

The number of distinct active scale regimes in task $\mathcal{T}$ is

$$K(\mathcal{T}) = \left|\{c_{\text{stage}}(s) : s \in \mathcal{T},\ D_{\text{stage}}(s) > 0\}\right| .$$

Tasks with $K(\mathcal{T}) = 1$, 2, or 3 are categorized as single-scale, dual-scale, or three-scale, respectively.

C.4 Scale Analysis on 50 Tasks

Table 8 reports the scale analysis of all 50 RoboTwin tasks.

Table 8: Action scale analysis on 50 RoboTwin tasks. Overall displacement is measured in m/step. L, M, and H denote low-, medium-, and high-scale stages. Stage sequences exclude idle stages; composition lists distinct active regimes.

Task	Overall	Overall Scale	Stage Sequence	Composition	#Regimes
Adjust Bottle	0.0442	High	H–M	M+H	2
Beat Block Hammer	0.0372	High	H–L	L+H	2
Blocks Ranking RGB	0.0363	Medium	M–L–H–L–H–L–L	L+M+H	3
Blocks Ranking Size	0.0359	Medium	H–L–H–L–H–L–L	L+H	2
Click Alarmclock	0.0412	High	H–M	M+H	2
Click Bell	0.0373	High	H–L	L+H	2
Dump Bin Bigbin	0.0335	Medium	M–L–H–L	L+M+H	3
Grab Roller	0.0675	High	H–M	M+H	2
Handover Block	0.0348	Medium	M–M–M	M	1
Handover Mic	0.0281	Low	M–M–M	M	1
Hanging Mug	0.0382	High	H–L–H–L–L	L+H	2
Lift Pot	0.0570	High	H–M	M+H	2
Move Can Pot	0.0348	Medium	H–L	L+H	2
Move Pillbottle Pad	0.0346	Medium	H–M	M+H	2
Move Playingcard Away	0.0361	Medium	H–M	M+H	2
Move Stapler Pad	0.0276	Low	M–L	L+M	2
Open Laptop	0.0179	Low	M–L	L+M	2
Open Microwave	0.0116	Low	H–L–M–L	L+M+H	3
Pick Diverse Bottles	0.0736	High	H–H	H	1
Pick Dual Bottles	0.0730	High	H–H	H	1
Place A2B Left	0.0307	Low	M–L	L+M	2
Place A2B Right	0.0299	Low	M–M	M	1
Place Bread Basket	0.0370	Medium	H–M–M–L	L+M+H	3
Place Bread Skillet	0.0438	High	H–M	M+H	2
Place Burger Fries	0.0440	High	H–M–M–L	L+M+H	3
Place Can Basket	0.0460	High	H–M–H–L	L+M+H	3
Place Cans Plasticbox	0.0554	High	H–M–H–H	M+H	2
Place Container Plate	0.0301	Low	H–L–L	L+H	2
Place Dual Shoes	0.0556	High	H–M–H	M+H	2
Place Empty Cup	0.0249	Low	M–L–L	L+M	2
Place Fan	0.0312	Low	H–L	L+H	2
Place Mouse Pad	0.0269	Low	M–L	L+M	2
Place Object Basket	0.0460	High	H–L–H–L	L+H	2
Place Object Scale	0.0316	Medium	M–M	M	1
Place Object Stand	0.0306	Low	H–L	L+H	2
Place Phone Stand	0.0279	Low	M–L	L+M	2
Place Shoe	0.0267	Low	M–L	L+M	2
Press Stapler	0.0296	Low	H–L	L+H	2
Put Bottles Dustbin	0.0384	High	M–L–H–M–M–M–H–M–L–H–H	L+M+H	3
Put Object Cabinet	0.0315	Medium	M–M–M	M	1
Rotate QRcode	0.0237	Low	M–L	L+M	2
Scan Object	0.0448	High	H–M	M+H	2
Shake Bottle Horizontally	0.0188	Low	H–L	L+H	2
Shake Bottle	0.0213	Low	H–L	L+H	2
Stack Blocks Three	0.0345	Medium	M–L–H–L–H–L–L	L+M+H	3
Stack Blocks Two	0.0333	Medium	M–L–H–L–L	L+M+H	3
Stack Bowls Three	0.0356	Medium	M–L–H–L–H–M–L	L+M+H	3
Stack Bowls Two	0.0328	Medium	M–L–H–M–L	L+M+H	3
Stamp Seal	0.0343	Medium	H–L	L+H	2
Turn Switch	0.0320	Medium	M	M	1

D Additional Experimental Results in Real World

Training and Evaluation Details. We evaluate ActionUNet on three representative tasks from the RoboTwin 2.0 benchmark:

(1) Place Object Basket: the robot must grasp one of five kinds of objects and place it in the basket; success is recorded when the object is put in the basket.

(2) Stack Blocks Two: the robot must grasp two blocks and stack one on the other one; success is recorded when the block is centrally stacked on the other one.

(3) Stack Bowls Two: the robot must grasp two bowls and stack one on the other one; success is recorded when the bowl is centrally stacked on the other one.

Each task includes 50 demonstrations used for training. Demonstrations are collected through keyboard-based teleoperation. In the hard setting, distractor layouts are randomly sampled across episodes but kept identical across compared models. Success is judged by a blinded human evaluator according to the task-specific criteria above.

Visualizations. We further provide visualizations of real-world task executions in Fig. 7, Fig. 8, and Fig. 9. Across Place Object Basket, Stack Bowls Two, and Stack Blocks Two, $\pi_{0.5}$+ActionUNet successfully completes the manipulation process in both easy and hard settings, demonstrating stable execution under additional distractor objects.

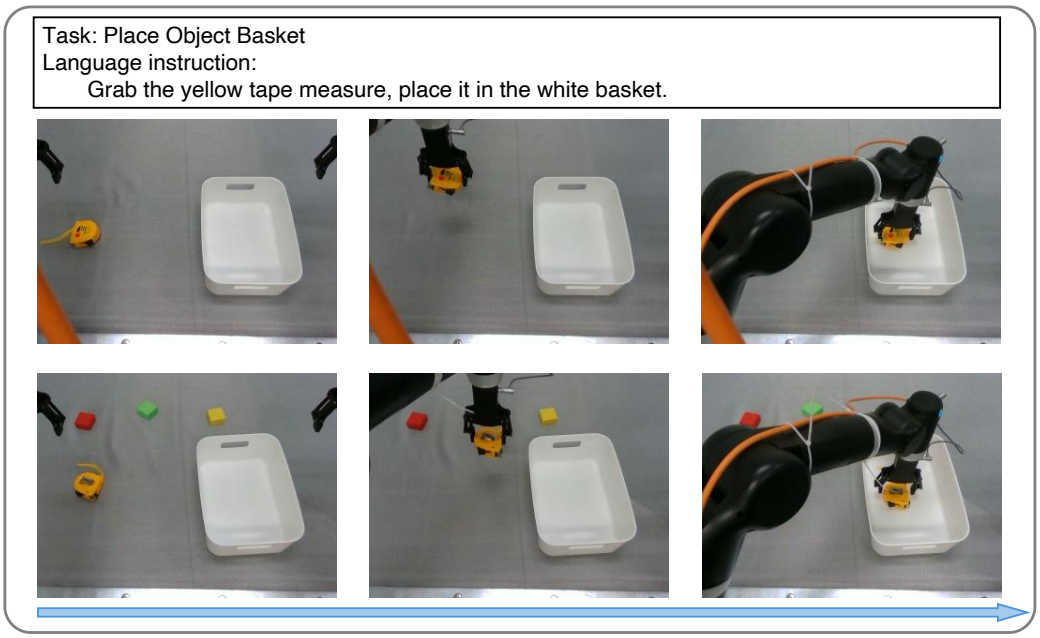

Figure 7: Visualization of Place Object Basket task execution with $\pi_{0.5}$+ActionUNet. The top row shows the keyframes of the task execution in the easy setting, while the bottom row presents the keyframes in the hard setting.

E Architecture Exploration: Where Should Multi-scale Features Enter the Continuous Decoder?

We further study an alternative design that directly injects multiple U-Net decoder features into different layers of the SIREN decoder. Specifically, given decoder features from the finest, middle, and coarsest resolutions, this variant conditions the first, second, and third SIREN layers on these features, respectively. This design appears natural because the temporal U-Net produces a feature pyramid whose levels correspond to different temporal resolutions. However, as shown in Table 9, this layer-wise pyramid conditioning does not improve performance and is consistently worse than our default ActionUNet design.

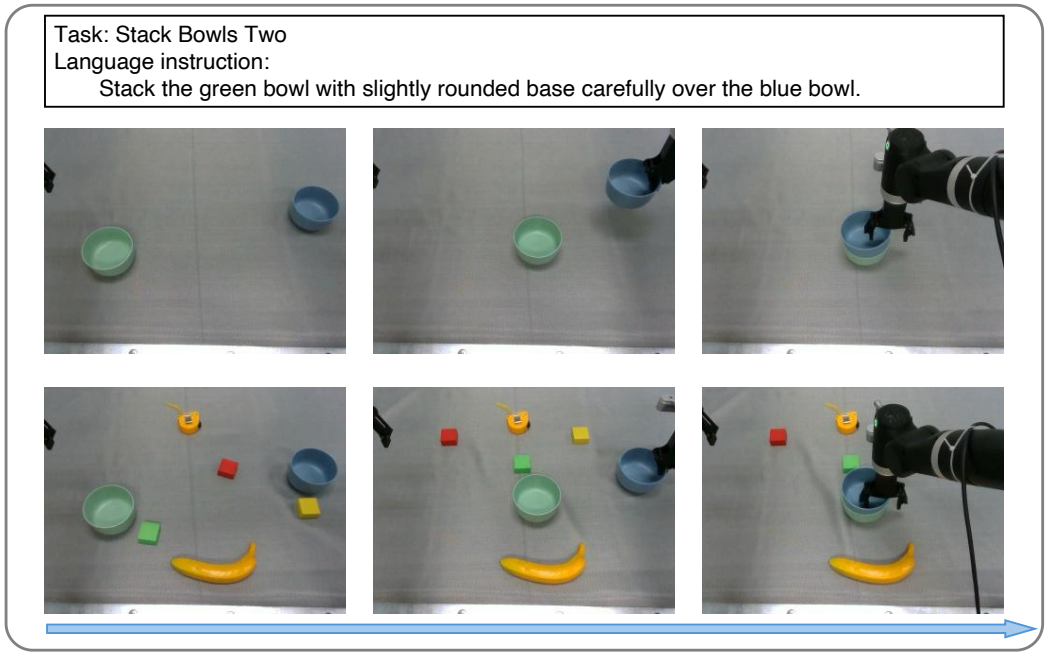

Figure 8: Visualization of Stack Bowls Two task execution with $\pi_{0.5}$+ActionUNet. The top row shows the keyframes of the task execution in the easy setting, while the bottom row presents the keyframes in the hard setting.

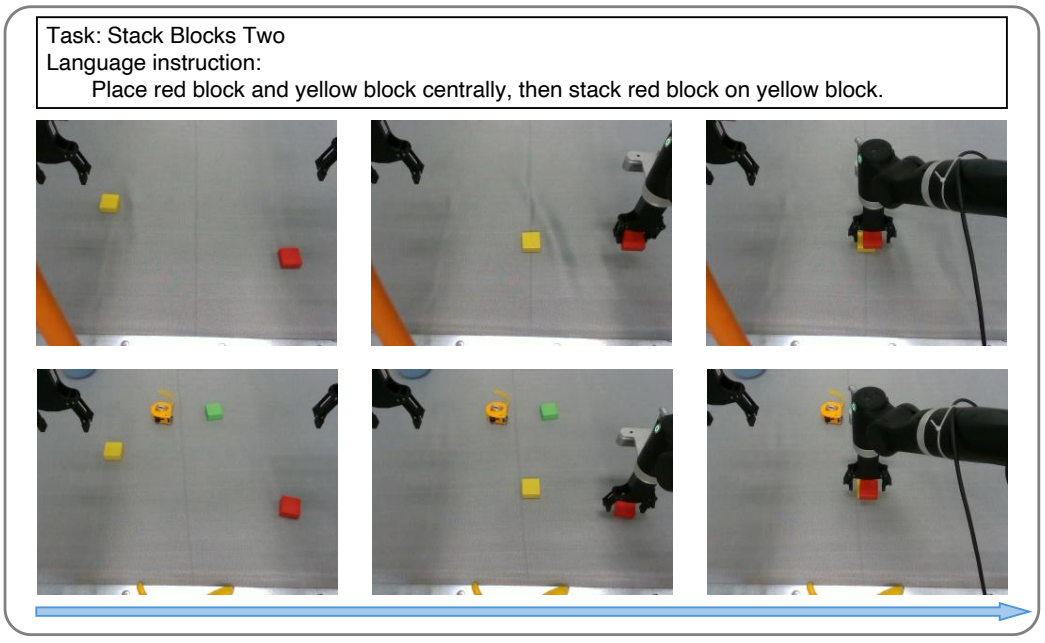

Figure 9: Visualization of Stack Blocks Two task execution with $\pi_{0.5}$+ActionUNet. The top row shows the keyframes of the task execution in the easy setting, while the bottom row presents the keyframes in the hard setting.

We attribute this result to a mismatch between the temporal scale hierarchy and the SIREN layer hierarchy. The depth of a SIREN represents nonlinear composition in an implicit continuous field, but it does not provide an ordered coarse-to-fine temporal scale axis. Injecting U-Net features with different temporal resolutions into different sinusoidal layers, therefore forces the continuous de-

Table 9: Architecture ablation on where multi-scale features enter the continuous decoder. Layer-wise Pyramid Conditioning directly injects U-Net decoder features at different temporal resolutions into different SIREN layers. All non-baseline variants are built on $\pi_{0.5}$.

Task	$\pi_{0.5}$		+ Layer-wise Pyramid Condition		+ ActionUNet	
	Easy	Hard	Easy	Hard	Easy	Hard
Move Can Pot	66	52	71	64	**77**	**69**
Stack Blocks Two	68	24	**74**	25	**74**	**35**
Beat Block Hammer	77	28	86	34	**89**	**43**
Average	70.3	34.7	77.0	41.0	**80.0**	**49.0**
Gain over $\pi_{0.5}$	–	–	+6.7	+6.3	**+9.7**	**+14.3**

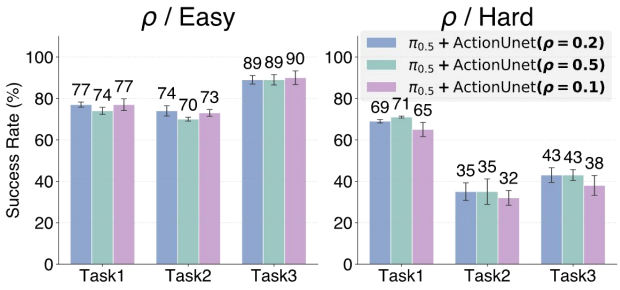

Figure 10: Ablation study on the Matérn length-scale parameter ρ. Changing ρ within the tested range only introduces minor performance variations, showing that ActionUNet is robust to the choice of the Matérn aggregation bandwidth.

coder to simultaneously align multi-scale features and decode smooth trajectories. Because sine activations are sensitive to feature-induced phase shifts, unaligned coarse and fine conditions can interfere inside the hidden field, producing less stable action predictions. In contrast, ActionUNet first reconciles multi-scale information in the temporal action feature space through the U-Net decoder and then passes a single temporal-aligned refined feature to the conditional SIREN. This separation lets the U-Net model have a hierarchical temporal structure, while the SIREN focuses on continuous and smooth action decoding, which explains the stronger performance of our default design.

F Ablation on the Matérn Length-Scale Parameter ρ

We evaluate the length-scale parameter ρ in the Matérn aggregation kernel. The parameter ρ controls temporal bandwidth of the fixed Matérn weights which aggregate neighboring local SIREN fields.

As shown in Fig. 10, the Matérn length-scale parameter ρ has only a minor effect: the frozen Matérn kernel mainly provides lightweight aggregation, while the conditional SIREN does the heavy lifting in representing the continuous action field. We thus fix $\rho = 0.2$ in all experiments.

G Quantitative Analysis of Trajectory Accuracy and Smoothness

To quantitatively support the trajectory analysis in Fig. 6, we compute two metrics on four RoboTwin 2.0 tasks: Move Can Pot, Stack Blocks Two, Beat Block Hammer, and Pick Dual Bottles.

Jerk RMS is the root mean square of the third temporal derivative of the end-effector position averaged over the action horizon. Lower values indicate smoother trajectories with less high-frequency jitter. **Manipulation Drift** is the positional offset of the end-effector at the first grasp attempt relative to the target object. It measures spatial accuracy in the critical grasping phase, with lower values indicating more precise localization.

The results confirm that the SIREN-only variant achieves the lowest Jerk RMS and the best smoothness, but its Hard-setting success rate decreases from 34.5% to 33.3%, while Manipulation Drift

Table 10: Quantitative trajectory analysis on four RoboTwin 2.0 tasks. Lower Manipulation Drift and Jerk RMS indicate better spatial accuracy and smoother trajectories, respectively.

Setting	Method	Success Rate (%) ↑	Drift (mm) ↓	Jerk RMS (m/s^3) ↓
Easy	$\pi_{0.5}$	69.0	32.53	32.90
	+ U-Net	67.5	29.07	34.49
	+ SIREN	75.5	28.85	**29.43**
	+ ActionUNet	**77.8**	**27.60**	32.70
Hard	$\pi_{0.5}$	34.5	81.64	48.31
	+ U-Net	37.3	76.05	49.34
	+ SIREN	33.3	83.03	**42.01**
	+ ActionUNet	**46.5**	**62.28**	45.88

Table 11: Module-wise ablation on LIBERO-Plus. All variants are trained on LIBERO and directly evaluated under seven controlled perturbations. We report mean success rate (%) ± standard error over three repeated evaluations.

Method	Camera	Robot	Lang.	Light	Back.	Noise	Layout	Avg.
$\pi_{0.5}$	48.4±0.65	48.0±1.75	67.4±0.65	93.0±0.20	87.1±0.30	51.1±1.20	81.8±0.50	66.0±0.40
+ U-Net	50.8±0.18	52.4±0.51	71.3±0.34	93.8±0.86	88.7±0.86	52.9±0.17	83.1±0.31	68.4±0.18
+ SIREN	47.4±0.23	53.1±0.30	67.4±0.25	92.7±0.55	86.8±0.42	48.7±0.63	81.4±0.53	66.1±0.17
+ ActionUNet	**56.4**±0.49	**55.2**±1.55	**74.6**±0.51	**94.8**±0.06	**89.7**±0.20	**60.2**±0.97	**85.5**±0.56	**72.0**±0.32

increases from 81.64 mm to 83.03 mm. This shows that temporal smoothing alone is insufficient to explain the robustness gains.

In contrast, the U-Net-only variant improves Hard-setting success from 34.5% to 37.3% and reduces Manipulation Drift from 81.64 mm to 76.05 mm, while increasing Jerk RMS. The full ActionUNet achieves the highest Hard-setting success rate of 46.5% and the lowest drift of 62.28 mm while maintaining relatively low jerk. These results indicate that the temporal U-Net provides the main robustness improvement through multi-scale temporal feature aggregation, while the SIREN decoder complements it by resolving fusion-induced temporal discontinuities.

H Module-Wise Robustness Ablation on LIBERO-Plus

To directly separate the contribution of multi-scale temporal refinement from that of the smoothing decoder, we conduct the module-wise ablation on LIBERO-Plus. All variants are trained under the same LIBERO protocol and directly evaluated on the seven controlled LIBERO-Plus perturbations. We repeat the evaluation three times and report the mean values and standard errors.

Performance on LIBERO-Plus provides a direct measure of robustness, as the models are trained on clean data but tested under diverse observation perturbations.

Under LIBERO-Plus perturbations, U-Net-only improves the average success rate by 2.4 percentage points and improves all seven perturbation categories. In contrast, SIREN-only improves the average by only 0.1 points: it improves the Robot perturbation, matches the baseline under Language, and decreases performance under the other perturbations.

The U-Net therefore exhibits the broad standalone robustness effect under observation corruptions. The full ActionUNet achieves the best performance under every perturbation and improves the average by 6.0 percentage points, substantially exceeding either individual module.

These results show that the U-Net alone provides more consistent performance gains across observation perturbations. Together with the RoboTwin 2.0 module-wise analysis, these results support that the multi-scale scheme improves the robustness of downstream action-feature prediction, rather than making the VLM encoder itself more robust.

Table 12: Evaluation on precision-demanding RoboDojo tasks. We report mean success rate and Progress Score $\pm$ standard error over three repeated evaluations. Lower Jerk RMS indicates smoother trajectories.

Task	Method	Success (%) ↑	Progress ↑	Jerk RMS ↓	GT Jerk RMS
Insert Tubes	$\pi_{0.5}$	$4.7_{\pm 3.1}$	$22.0_{\pm 3.2}$	127.81	50.62
	+ ActionUNet	$\mathbf{14.0}_{\pm 2.0}$	$\mathbf{30.8}_{\pm 0.4}$	**122.99**	50.62
Build Tower	$\pi_{0.5}$	$26.0_{\pm 2.0}$	$37.3_{\pm 0.9}$	119.75	49.54
	+ ActionUNet	$\mathbf{34.7}_{\pm 2.3}$	$\mathbf{42.5}_{\pm 1.5}$	**109.53**	49.54

I Precision-Demanding Manipulation on RoboDojo

We evaluate ActionUNet on high-precision manipulation tasks from RoboDojo [45], specifically on the precision dimension, which contains tasks requiring strict spatial accuracy and contact-rich control.

We evaluate two representative precision tasks: *Insert Tubes*, where the robot must insert tubes into narrow holes, and *Build Tower*, where the robot must precisely arrange wooden blocks and boards into a stable tower. We compare $\pi_{0.5}$ and $\pi_{0.5}$+ActionUNet under the same training protocol with 100 demonstrations and single-task fine-tuning. Each evaluation includes 50 episodes, and we repeat it three times to calculate the mean $\pm$ standard error.

In addition to success rate, we report the RoboDojo Progress Score and Jerk RMS. Progress Score measures partial completion. Insert Tubes assigns 20, 40, or 100 points when one, two, or three tubes satisfy the position, insertion-depth, and orientation conditions; Build Tower assigns 10, 30, or 100 points for completing the bottom structure, middle structure, or full tower. Jerk RMS is the root mean square of the third temporal derivative of end-effector position over the full episode. GT Jerk RMS is computed from the demonstration trajectories.

On Insert Tubes, ActionUNet improves the success rate by 9.3 percentage points and the Progress Score by 8.8 points. On Build Tower, it improves the success rate by 8.7 percentage points and the Progress Score by 5.2 points. These consistent gains in both complete success and partial task progress demonstrate that ActionUNet remains effective on precision-demanding manipulation tasks.

The quantitative results further show that ActionUNet is only slightly smoother than the $\pi_{0.5}$ baseline and remains substantially less smooth than the ground-truth trajectory. On Insert Tubes, Jerk RMS decreases from 127.81 to 122.99 $\mathrm{m/s^3}$, compared with a GT Jerk RMS of 50.62 $\mathrm{m/s^3}$. On Build Tower, it decreases from 119.75 to 109.53 $\mathrm{m/s^3}$, compared with a GT Jerk RMS of 49.54 $\mathrm{m/s^3}$. The simultaneous improvements in complete success, partial task progress, and trajectory smoothness indicate that the lower Jerk RMS does not come from suppressing the fine-grained motions required for precise manipulation. These results support that ActionUNet remains effective on precision-demanding tasks without over-smoothing the trajectory.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We state the main claims and contributions in the Abstract and Section 1.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitations of ActionUNet in Section 5.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: We provide the method formulation in Section 3 and additional derivations in Appendix A.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide the experimental setup, benchmarks, protocols, and implementation details in Section 4.1.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We provide public access to the implementation of ActionUNet, with instructions for reproducing the main experiments. All datasets used in this work are publicly available, and their original sources are cited in the paper.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: We specify the datasets, evaluation protocols, training settings, and hyperparameters in Section 4.1.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: We report statistical experiments on the baseline and our method.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We report training compute resources in Section 4.1 and inference latency measurements in Appendix B.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We conduct the study using public benchmarks and controlled robot experiments, as described in Section 4.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [No]

    Justification: We currently do not include a dedicated broader-impact discussion.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: We do not release high-risk models, scraped datasets, or assets requiring special safeguards.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All datasets used are from publicly available sources. They are released under open-access or research licenses, which are respected and cited in our paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: We release the implementation of ActionUNet as a new code asset. The repository provides documentation for the software environment, dependencies, configuration, training, and evaluation. We do not introduce a new dataset or benchmark.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: We do not conduct crowdsourcing or human-subject experiments.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: We do not involve human-subject research, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.