A. Training Details

Training hyper-parameters UniAct-0.5B utilizes the pre-trained parameters from LLava-One-Vision-0.5B [35] to initialize the VLM. For visual feature extraction in heterogeneous decoding heads, we deploy an ImageNet pretrained ResNet18, which is commonly employed in vision-based policy learning [64]. This model was jointly trained during the large-scale pre-training process to enhance perceptual capability in manipulation task scenarios. We adopt resolutions of 384×384 for the VLM and 224×224 for the ResNet18, consistent with their original configurations. Image augmentation settings and more hyper-parameters can be found in Table 3 and Table 4, respectively.

Augmentation	value
RandomResize	ratio=(0.75, 1.3333) scale=(0.5, 1.0) interpolation=BICUBIC
RandomHorizontalFlip	p=0.5
ColorJitter	contrast=(0.6, 1.4) brightness=(0.6, 1.4) saturation=(0.6, 1.4)

Table 3. Image augmentation settings during training

config	value
optimizer	AdamW
batch size	1024
learning rate	2×10^{-5}
weight decay	0.
optimizer momentum	β_1, β_2=0.9,0.95
iters	500K
model precision	BFloat16

Table 4. Training hyper-parameters

Data construction As illustrated in Table 5, we detail the composition of our training data, which includes the number of trajectories, samples, and the control interfaces for the 28 distinct embodiments. Following previous works [31, 61], we assign different sampling rates to each dataset during training to ensure a balanced mix of embodiments, tasks, and scenes. These sampling rates are specified in Table 5. It is important to note that many of the datasets may contain distinctly action spaces such as EEF position and Joint position. In addition, images from different datasets may contain multiple view points. By default, we use only the first third-person perspective following [31, 61] to maintain consistency.

Dataset	Trajectory	Samples	Sample rate(%)	Control Interface
Utaustin Mutex	1500	361871	1.0	EEF Position
Berkeley Cable Routing	1557	38789	0.2	EEF velocity
NYU Franka Play	454	44412	1.0	EEF velocity
Kuka	557893	7130157	12.7	EEF Position
Austin Sailor	240	352110	2.2	EEF velocity
Fmb	8611	1136907	1.0	EEF velocity
Berkeley Autolab	1000	95310	1.2	EEF Position
Viola	150	75614	0.9	EEF Position
Dobbe	5208	1139874	1.4	EEF Position
Iamlab CMU	631	146241	0.9	EEF Position
Austin Buds	50	34110	0.2	EEF Position
Language Table	441911	6602077	4.4	EEF Position
Stanford Hydra	569	357137	4.4	EEF Position
Robo Set	18246	1419838	5.0	Joint position
Austin Sirius	559	279724	1.7	EEF velocity
Dlr Edan	104	8824	0.1	EEF Position
Fractal	86599	3607028	12.7	EEF Position
TOTO	1003	324669	2.0	Joint position
Berkeley Fanuc	415	58660	0.7	EEF Position
CMU Stretch	135	25012	0.2	EEF Position
Roboturk	1934	186910	2.3	EEF Position
Jaco Play	976	66094	0.4	EEF Position
Taco Play	3603	230966	3.0	EEF Position
BC-Z	42811	5957097	7.5	EEF Position
Droid	92115	27043929	10.0	EEF Position
Furniture Bench	5100	3905717	2.4	EEF velocity
Bridge	28933	899685	13.3	EEF Position
Libero	6500	1007204	5.0	EEF Position

Table 5. Details of the training data composition, including the number of trajectories, number of samples, and control interfaces of the 28 different data sources. Following [31], we set different sampling probabilities for different data, which we also report in this table.

Categorical Reparameterization. Except for the Gumbel-Softmax utilized for training UniAct, we also explored another commonly used reparameterization technique: the Straight-Through Estimator (STE) [63]. However, empirical findings indicate that using STE can lead to severe collapses in the universal action codebook. An intuitive explanation for this is that each universal action requires numerous optimization steps to learn the highly abstracted behaviors. Since STE involves hard sampling, it results in only one universal action being selected and optimized for each training sample. This lack of gradient distribution among all potential actions can stifle the learning process and reduce the diversity of learned actions.

B. Evaluation Setups

In this section, we delve deeper into the evaluation experiments for the three robotic embodiments used in our study: WidowX Robot, Franka Robot, and AIRBOT. We provide detailed scores for each embodiment to offer a clearer, more intuitive comparison of their performance.

B.1. WidowX Robot in Real World

The experiments on WidowX aim to assess the models' generalization capabilities across five distinct dimensions, as illustrated in Figure 8. Instead of merely reporting task success rates, we calculate scores based on the progress made towards completing the task for some complex tasks. This scoring method can more intuitively reflect the model's understanding and generalization ability [31]. Detailed scores are available in Table 6. In the subsequent sections, we will provide a comprehensive description of the task settings.

Baseline models: Given that Octo, CrossFormer, and OpenVLA were trained using same data recipes (e.g., OXE) as UniAct, we directly evaluated their performance on the WidowX robot without further fine-tuning, consistent with their original implementations. For LAPA, we employed the OXE-pretrained model and adhered to the official pro-

tocol for fine-tuning with Bridge Data. This fine-tuning process involved 2,000 gradient steps with a batch size of 64. An intriguing observation is that, despite CrossFormer's utilization of more diverse pretraining data, its performance was inferior to that of its codebase model, Octo. We hypothesize that this discrepancy stems from a potentially suboptimal training pipeline, wherein all single-arm robots, regardless of their control interfaces, were treated as a homogeneous embodiment. This approach may have induced negative transfer, resulting in performance degradation. In contrast, while UniAct also leverages diverse and heterogeneous data, similar to CrossFormer, its well-defined universal action space effectively mitigates negative transfer.

Visual Generalization: This dimension evaluates the model's ability to adapt to different visual environments characterized by variations in lighting, background, and object textures. Following the setups outlined in [31], we design three specific tasks, detailed in Figure 8. Each task is set against a distinct background environment, features different lighting brightness levels, and involves target objects in two colors: red and green. Additionally, we introduce various visual distractors unrelated to the task to assess the model's ability to generalize visually and maintain robustness against such distractions. To ensure a fair comparison, all variables related to the model's testing conditions are kept consistent across all models. All tasks in this suite are assigned binary score: 0 for failure and 1 for success.

Motion Generalization: Tasks in this dimension aim to evaluate the robot's capability to perform appropriate motions while recognizing the target object's position and orientation, which may not have been encountered during training. Specifically, we have designed two tasks: *Lift eggplant* and *Put carrot on plate*, as detailed in Figure 8. The target objects are placed in several predetermined positions and oriented in pre-designed directions. This helps in assessing the robot's adaptability to changes in physical task parameters. All tasks in this suite are assigned binary score: 0 for failure and 1 for success.

Physical Generalization: In this dimension, we examine the model's capability to manage physical variations in objects, such as size, weight, and material properties. The tasks are crafted to evaluate the robot's adaptability to these fluctuations, which significantly influence manipulation strategies. Detailed task setups are listed in Figure 8. The objects involved in these tasks, such as carrots and AAA batteries, often have irregular shapes or are placed in unusual positions (e.g., an overturned pot). Successfully handling and completing tasks with these objects requires a generalized policy that can accurately recognize the physical attributes of the target objects and execute appropriate strategies. All tasks in this suite are assigned binary score: 0 for failure and 1 for success.

Semantic Generalization: In this dimension, we assess the model's ability to understand and generalize across various semantic contexts. Specifically, the tasks especially the target objects included in this dimension have never been encountered during the training procedures for both UniAct and the baseline models. This approach tests the model's capability to interpret and adapt to new instructions and environments, emphasizing its flexibility and learning efficiency. The purple grape is highly smooth, so we assign scores as follows: 0.25 for touching the grape, 0.5 for grasping it, 0.75 for moving it toward the pot, and 1 for successfully completing a pick-and-place. Also, for the stack green cup on red cup task, we assign scores as follows: 0.25 for touching the green cup, 0.5 for grasping it, 0.75 for moving it toward the red cup, and 1 for successfully stacking. Other tasks in this suite are assigned binary scores.

Language Grounding: This dimension evaluates the model's ability to comprehend and execute commands that are grounded in natural language. The focus is on assessing the model's proficiency in following novel instructions to manipulate specific objects as described verbally. While there are similarities with Semantic Generalization in terms of handling unseen scenarios, Language Grounding task distinctly tests the model's capacity to accurately understand and act on language-based directives within potentially misleading environments. An illustrative example is placing two cups of different colors on a table and instructing the model to manipulate one of them using language. The model must accurately ground the language instruction to the correct object in a complex real-world setting. This tests the model's ability to connect linguistic descriptions directly with physical actions in dynamic and visually diverse environments. For all tasks in this suite, we assign scores as follows: 0.5 for correct grounding, 1 for a successful task completion.

B.2. Franka Robot in Simulation

Baseline models. We include 6500 expert demonstrations of 130 different tasks collected with Franka Robot in LIBERO [38] simulation to train our UniAct-0.5B and then follow the LIBERO Benchmark [38] to evaluate our models. The input images are rendered by the emulator and we use the default 128×128 resolution. As all the open-source baseline models were not initially trained with the simulation data, substantial effort was put into fine-tuning them to facilitate fair comparisons with our UniAct framework. The fine-tuning process for OpenVLA, Octo and CrossFormer was conducted on 8 A6000 GPUs and 2 4090 GPUs, lasting 7 hours and 4 hours, respectively. Details of the training hyperparameters are provided in Table 7. Notably, we manually cleaned the "no-op'" data before fine-tuning OpenVLA following its official guidance, a step that proved crucial for achieving convergence. However, as shown in Figure 9, even with an increased number of training steps, the

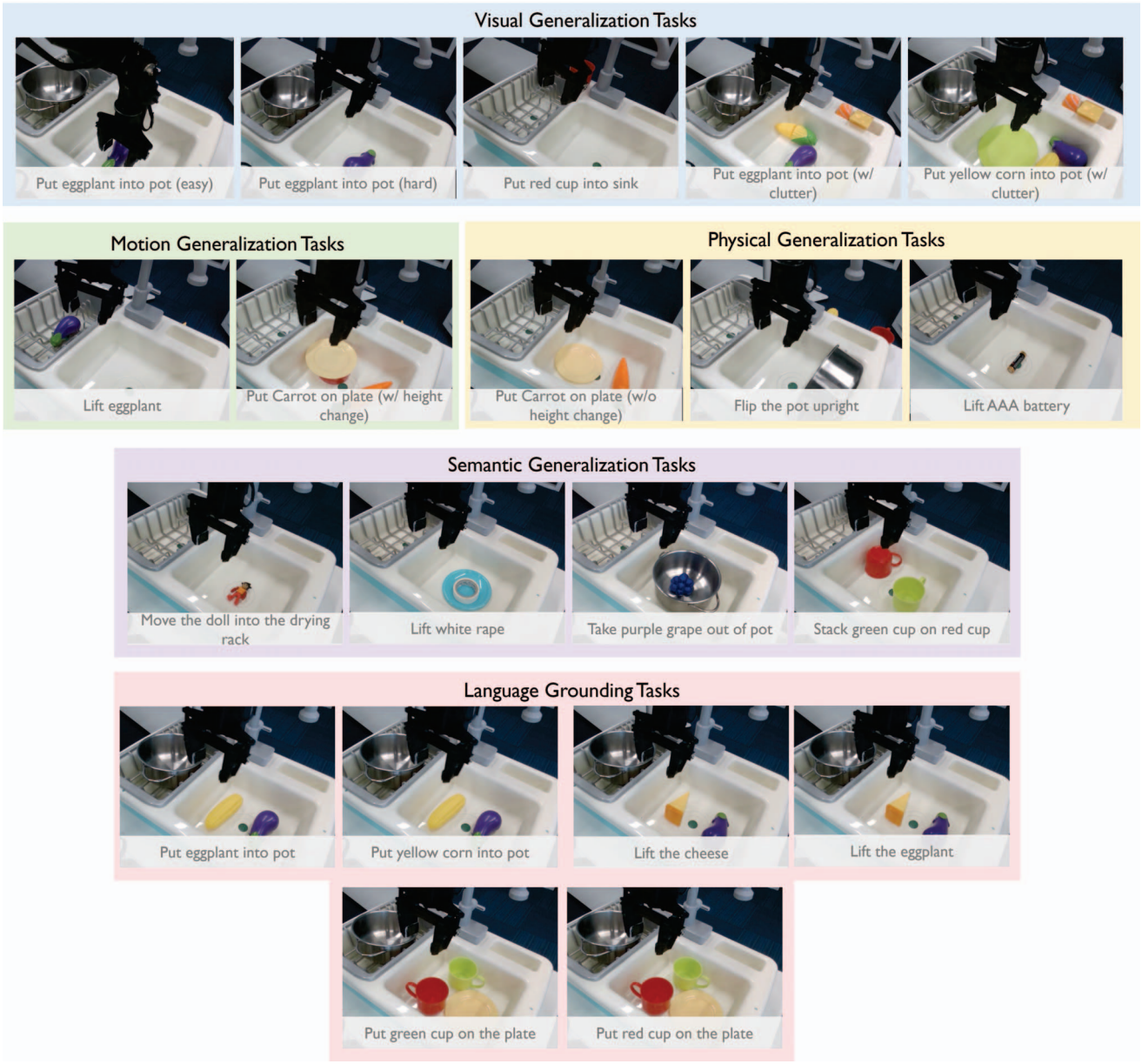

Figure 8. Illustration for WidowX evaluation tasks.

model's action accuracy remained low. In contrast, our UniAct framework demonstrated robustness to noisy data. For LAPA, we employed an identical training recipe to that used for the WidowX robots. The OXE-pretrained LAPA model was fine-tuned with 2,000 gradient steps on Libero, utilizing a batch size of 64.

Adapt to ACT head. To evaluate UniAct's adaptability to heterogeneous decoder heads with varying model architectures, we augmented UniAct-0.5B with an additional ACT head and fine-tuned it on the Libero dataset. Leveraging ACT's sophisticated architectural design, we incorporated image observations from a wrist-mounted perspective and proprioceptive data as supplementary inputs to the decoder head. We adhered to the official implementation settings for the ACT model. Fine-tuning was conducted for 200K gradient steps, utilizing a batch size of 256 across 8 A100 GPUs.

B.3. Fast adaptation to AIRBOT

To assess the fast adaptation capability of UniAct-0.5B and baselines, we fine-tune them using newly collected demonstrations on an unseen embodiment during the pretraining phase, AIRBOT. In this section, we provide detailed information about the fine-tuning processes.

Task Category	Task Name	Octo	OpenVLA	UniAct-0.5B(ours)
Visual Generalization	*put eggplant into pot(easy)*	3.0	**10.0**	**10.0**
	put eggplant into pot	6.0	6.0	**10.0**
	put cup from counter into sink	0.0	**2.0**	1.0
	put eggplant into pot	3.0	**8.0**	5.0
	put corn on plate	0.0	7.0	**8.0**
	Average Score	2.4	6.6	**6.8**
Motion Generalization	*lift eggplant*	4.0	3.0	**6.0**
	put carrot on plate	2.0	4.0	**6.0**
	Average Score	3.0	3.5	**6.0**
Physical Generalization	*put carrot on plate*	2.0	**9.0**	**9.0**
	flip pot upright	1.0	2.0	**7.0**
	lift AAA battery	0.0	**7.0**	5,0
	Average Score	1.0	6.0	**7.0**
Sematic Generalization	*move doll into drying rack*	0.0	5.0	**6.0**
	lift white rape	0.0	1.0	**2.0**
	take purple grapes out of pot	4.5	**5.0**	**5.0**
	stack green cup on red cup	3.25	**8.25**	2.5
	Average Score	1.9	**4.8**	3.9
Language Grounding	*put eggplant into pot*	6.0	7.0	5.0
	put yellow corn into pot	7.0	10.0	10.0
	lift cheese	1.0	**8.0**	6.0
	lift eggplant	6.0	8.0	**9.0**
	put green cup on plate	3.0	**10.0**	**6.0**
	put red cup on plate	6.0	**10.0**	**8.0**
	Average Score	4.8	**8.8**	7.3

Table 6. Comparison between UniAct-0.5B and baseline models on detailed scores in WidowX evaluations.

Fine-tuning Settings For UniAct: We utilized 4 A100 GPUs to fine-tune UniAct-0.5B with DeepSpeed. Notably, we train a new MLP network from scratch as the heterogeneous head for AIRBOT while keeping the other modules frozen. The fine-tuning was conducted over a span of 1 hours. The training hyper-parameters employed during this process are detailed in Table 8.

Fine-tuning Settings For Baseline Models. The fine-tuning for the baseline models, OpenVLA and Octo, was executed on 8 A6000 GPUs and 2 4090 GPUs, lasting 1.5 hours and 0.5 hours, respectively. To ensure robust performance, the training of OpenVLA continued until it reached an accuracy rate of 95% on the training dataset, aligning with the official requirements in [31]. Training hyperparameters for this process are illustrated in Tab 10.

B.4. Fast adaptation to Bi-manual AIRBOT

To assess UniAct's versatility across diverse embodiments, we performed fast adaptation experiments using the bi-manual AIRBOT. Given the requirement for precise perception of complex environments inherent in bi-manual robotic manipulation, which a simple MLP decoder head cannot fulfill, we employed an ACT decoder head for the bi-manual AIRBOT. We designed four challenging manipulation tasks for evaluation: 'sweep plate', 'fold towel', 'put cup on plate', and 'transport pen'. For each task, we collected 250 demonstration trajectories and subsequently fine-tuned UniAct-0.5B. Training hyperparameters are detailed in Tab 9

C. More Related Works

Embodied Models with Hierarchical Structures. UniAct resembles a hierarchical-style structure, which firstly infers an universal action and then translate it into actionable actions. Some works also explore hierarchical structures, utilizing the planning capabilities of LLMs or VLMs to decompose original instructions into linguistic [2, 46, 56, 57, 71] or visual [7, 19, 66] sequences of subgoals. However, transferring these language or visual subgoals into precise actions still requires extensive action labels and struggles to leverage the shared structures across diverse action spaces. The universal action space in UniAct, however,

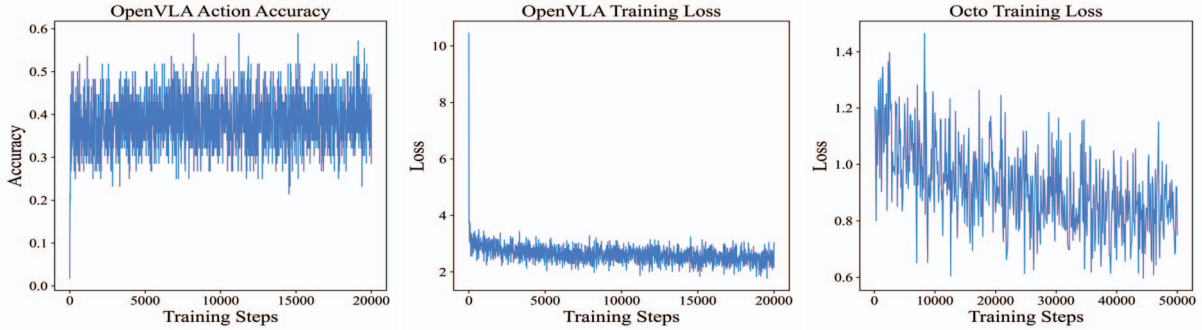

Figure 9. The learning curves for OpenVLA and Octo in LIBERO fintuning.

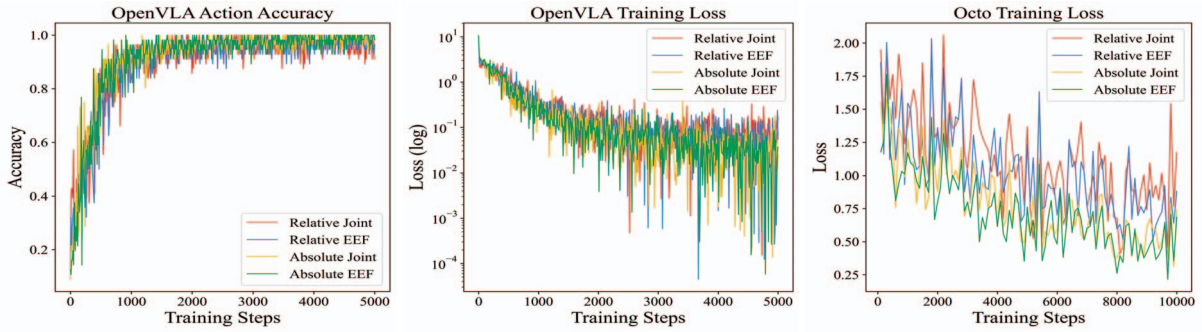

Figure 10. The learning curves for OpenVLA and Octo in AIRBOT finetuning.

Model-Task	Config	Value
OpenVLA-LIBERO	Optimizer	AdamW
	Batch size	64
	Learning rate	3×10^{-4}
	Weight decay	1×10^{-2}
	Optimizer momentum	$\beta_1 = 0.9, \beta_2 = 0.999$
Octo-LIBERO	Optimizer	AdamW
	Batch size	64
	Learning rate	3×10^{-4}
	Weight decay	1×10^{-2}
	Optimizer momentum	$\beta_1 = 0.9, \beta_2 = 0.999$
	Warmup iters	2000
OpenVLA-AIRBOT	Optimizer	AdamW
	Batch size	64
	Learning rate	3×10^{-4}
	Weight decay	1×10^{-2}
	Optimizer momentum	$\beta_1 = 0.9, \beta_2 = 0.999$
Octo-AIRBOT	Optimizer	AdamW
	Batch size	64
	Learning rate	3×10^{-4}
	Weight decay	1×10^{-2}
	Optimizer momentum	$\beta_1 = 0.9, \beta_2 = 0.999$
	Warmup iters	2000

Table 7. Fine-tuning hyper-parameters for baseline models in simulation.

serves as a more fine-grained skill library that can be efficiently adapted to physically grounded actions. This space is also end-to-end trained, maximizing the VLMs' capabil-

config	value
optimizer	AdamW
batch size	128
learning rate	3×10^{-4}
weight decay	0.
optimizer momentum	β_1, β_2=0.9,0.95
iters	10K
model precision	BFloat16

Table 8. Hyper-parameters for fine-tuning UniAct-0.5B on AIRBOT

config	value
optimizer	AdamW
batch size	256
learning rate	5×10^{-4}
weight decay	0.
optimizer momentum	β_1, β_2=0.9,0.95
iters	100K
model precision	BFloat16

Table 9. Hyper-parameters for fine-tuning UniAct-0.5B on Bimanual AIRBOT

ities to develop a flexible and comprehensive structure.

D. Limitations and Future Works

In this section, we discuss limitations in this work and the corresponding solutions. We hope this will inspire more interesting works.

More Embodiments: In this study, we explore the concept of a universal action space that can be shared across different action spaces. In this current version, UniAct is mostly evaluated with varied control interfaces on single robotic arms. The underlying motivation stems from an intuitive observation: despite differences in control interfaces, these robotic arms inherently exhibit common physical movements. However, this raises a crucial question: Is the commonality of physical movement exclusive to robotic arms or similar embodiments?

We argue that the answer is no. Future work will explore the universal actions for more complex embodiments, such as dual robotic arms, dexterous robotic hands, quadrupeds and even autonomous driving cars, which differ in degrees of freedom and mechanical structures. Despite their strong heterogeneity, these systems may also share some fundamental movements with simpler robotic arms, holding the potential to be incorporated into the same universal action space. Therefore, we hope to develop a "truly" universal action space that is capable of encoding movements across ANY physical embodiment while recognizing unique characteristics and prioritizing shared commonalities. This offers a promising direction for future research, with strong potential for enabling cross-embodiment control across diverse systems.

More Flexible Network Design: In our current implementation, UniAct utilizes identical decoding heads—a simple MLP network—for each embodiment to minimize the risk of over-fitting and facilitating training for the universal action extractor. However, it seems more reasonable that the complexity and number of parameters in the decoding heads should be tailored according to the control complexity of each embodiment and the diversity of its training data. For example, a bi-manual robot, with its increased degrees of freedom, would logically require a more complex decoding head than a single-arm robot to effectively learn and decode universal actions.

Looking forward, as the dataset expands to include more demonstrations from more diverse embodiments, it will become crucial to develop specialized decoding heads that consider the specific control complexities of each embodiment. Additionally, incorporating embodiment-specific information, such as proprioceptive data or different views, into these decoding heads could significantly enhance their performance.

Scaling Law For Universal Action Training: While our efforts to train UniAct with a vast array of open-source data have yielded commendable results, an intriguing question has arisen: Does more data unequivocally improve the universal action space? This question calls for a thorough examination from multiple perspectives. Firstly, does the incorporation of more embodiments inherently enhance the action space, or is there greater value in accumulating diverse task demonstrations for the same embodiment? Moreover, it's critical to consider whether the more data always bring better result.

Studying the relationship between the amount of data and how well our universal action space performs could be very useful. By understanding these relationships, we can better plan our data collection to improve model training effectively and efficiently.

More Utilization of Universal Action. Our work has highlighted the robust capabilities of universal actions in deploying cross-embodiment robot policies. However, we believe that the potential applications of universal actions extend far beyond what has been explored so far.

One promising direction is in the development of world models, where universal actions serve as a form of 'tokenizer'. This role involves breaking down complex actions into standardized, understandable components. By employing a universal action space for planning, these world models can more accurately predict and simulate outcomes across various scenarios and environments. This uniform approach to understanding and interacting with the world is invaluable for advanced planning and decision-making in robotics.