SCALING DEXTEROUS MANIPULATION VIA ACTION REPRESENTATION PRETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large-scale vision, language, and video pretraining has substantially improved the perception and generalization capabilities of robotic manipulation models, but action generation for complex dexterous manipulation remains constrained by the scarcity of robot demonstrations. Limited robot demonstrations must be used to learn both action representations and task-conditioned action distributions, making it difficult to train action representations sufficiently and, in turn, limiting the ability of downstream generative models to generate complex dexterous actions. To address this limitation, we propose ARPG (Action Representation Pretraining and Generation), which explicitly separates action representation learning from action generation: we first pretrain an action representation encoder on large-scale human manipulation data. We then freeze the encoder and train a compact generative backbone (130M parameters) to learn task-conditioned action distributions in the encoder's representation space. Experimental results show that ARPG achieves state-of-the-art performance on the DexJoCo benchmark and improves the average success rate on bimanual dexterous manipulation from 10.1% for $\pi_{0.5}$ to 44.7%. In a nested data-scaling experiment, downstream performance improves as human-manipulation pretraining data increase from 10 to 500 hours. These results indicate that action representation pretraining provides an independent scaling dimension for improving complex dexterous manipulation.

1 INTRODUCTION

Building general-purpose robots capable of complex dexterous manipulation, particularly coordinated and precise manipulation with dexterous hands, is an important goal in robot learning. In recent years, large-scale vision, language, and video pretraining has endowed robot policies with strong visual representations, semantic understanding, and task generalization (Bi et al., 2026; Ye et al., 2026; Black et al., 2025a; Zheng et al., 2026b). However, robots must ultimately translate these capabilities into concrete manipulation actions, and reliable action generation remains a significant challenge. Mainstream policies learn action generation end to end from robot demonstrations, requiring the model to both learn representations suitable for action generation and model the action distribution under specific conditions from those demonstrations; thus, further improvements in complex action generation are constrained by both the scale of robot demonstrations and the capacity of the downstream model. This observation raises a question: Can scalable action representation pretraining continuously improve complex dexterous manipulation without simultaneously scaling robot demonstrations and the generative backbone?

Recent research in generative modeling suggests that generative models need not learn their modeling spaces entirely from scratch: pretrained representations can either guide internal representation learning in a generative model or directly define the generative space through a pretrained encoder, thereby decoupling representation learning from subsequent distribution modeling (Yu et al., 2025; Zheng et al., 2026a). However, extending this idea to robot actions is nontrivial: a representation space suitable for complex action generation must preserve sufficient action information for reliable action recovery while encoding motion and interaction semantics throughout manipulation. Pretrained visual encoders have learned rich motion and interaction semantics from large-scale video data (Assran et al., 2025; Mur-Labadia et al., 2026), but their representation spaces are not themselves designed for action representation and recovery; conversely, relying solely on action self-supervised learning requires the action encoder to learn representations from scratch, precludes

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

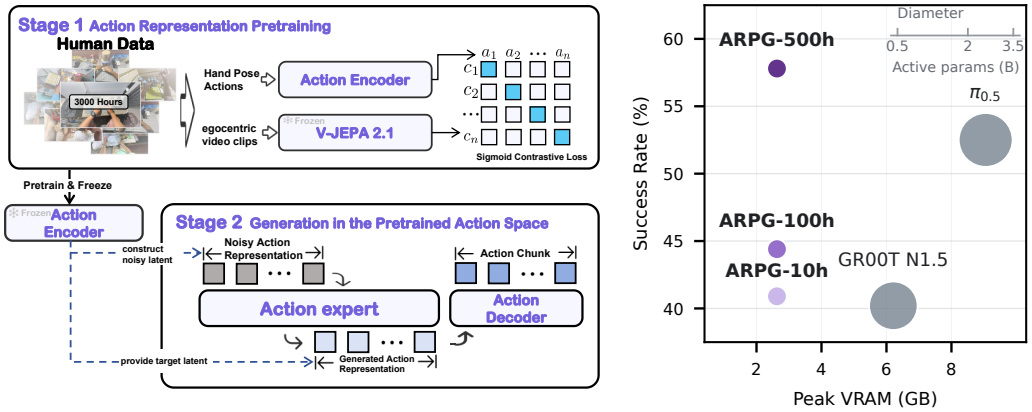


Figure 1: **Overview of ARPG. Left:** ARPG first pretrains action representations on large-scale human manipulation data, then freezes the action encoder and trains a robot policy to generate future action-chunk representations from noisy action representations. **Right:** Comparison of average success rate and peak inference VRAM on DexJoCo; marker area indicates the number of active parameters during inference. Exact values are reported in Table 1.

reusing the representation structure acquired through large-scale visual pretraining, and cannot leverage the scene context and interaction semantics contained in the visual modality.

Motivated by these observations, we propose ARPG (Action Representation Pretraining and Generation), which decouples action representation pretraining from downstream task-conditioned action generation and performs generation in the pretrained action representation space. Specifically, we use a vision–action dual-tower encoder: a frozen V-JEPA 2.1 (Mur-Labadia et al., 2026) extracts visual representations, an action Transformer encodes action trajectories, and a vision–action alignment objective based on sigmoid loss (Zhai et al., 2023) aligns action representations with visual representations, thereby leveraging representation priors acquired through large-scale video pretraining. Because action representation learning is decoupled from generative learning, the action representation need not adapt to the action details of a specific robot embodiment through the generative objective and can instead learn reusable motion representations across manipulation data; its pretraining can therefore be scaled to large-scale egocentric human manipulation data (Hoque et al., 2026; Punamiya et al., 2026). After pretraining, we freeze the action tower and use it as an action representation encoder, training an action decoder on robot demonstrations to recover executable continuous actions from the representations. On this basis, the downstream policy only needs to learn the task-conditioned action distribution in the fixed action representation space: ground-truth future action chunks (Zhao et al., 2023) are mapped by the frozen action encoder into generation targets; conditioned on the task, a DiT generative backbone (Peebles & Xie, 2023) generates future action representations through Flow Matching (Lipman et al., 2023), which the action decoder then recovers as executable actions.

We extensively evaluate ARPG on the DexJoCo simulation benchmark (Wang et al., 2026) and a real-robot platform. Despite using a compact generative backbone with only approximately 130M parameters, ARPG achieves better overall performance than $\pi_{0.5}$ (Black et al., 2025a) under joint multi-task training on DexJoCo, increasing the average success rate on the more challenging bimanual dexterous manipulation tasks from 10.1% to 44.7%. At the same time, ARPG maintains strong multi-task performance while substantially reducing deployment costs. Further scaling experiments show improved downstream performance as nested human-manipulation pretraining sets grow from 10 to 500 hours, increasing both data volume and behavioral coverage. We further analyze which action representation spaces are better suited to complex action generation and verify the downstream generative advantage of representations learned through vision–action alignment. Thus, rather than being merely an internal representation formed jointly during end-to-end policy learning, action representation can become an independently pretrained and scalable component, providing a new scaling path for complex dexterous manipulation.

2 Related Work

Generalist Robot Policies. In recent years, vision–language–action (VLA) models have combined large-scale pretrained vision–language models with robot data to transfer rich visual and semantic knowledge to robot control, demonstrating strong generalization across tasks and scenes (Black et al., 2025b;a; Zheng et al., 2026b). Recent world action models (WAMs) further leverage large-scale pretrained video models to introduce physical dynamics and interaction priors by jointly modeling future world states and actions (Bi et al., 2026; Ye et al., 2026). Unlike these generalist policies, which primarily target generalization across tasks and scenes, ARPG places greater emphasis on action generation for complex dexterous manipulation and explores an alternative path to improving robotic manipulation capabilities from the action side.

Learning from Human Manipulation Data. Because high-quality robot demonstrations are expensive and difficult to scale, a large body of recent work has explored using human manipulation data to scale up robot learning. One line of work recovers hand, object, or interaction trajectories from human videos and converts them into robot-executable training data through action retargeting (Yang et al., 2025; Zhang et al., 2026a;b). Another line of work requires no robot action annotations, instead learning latent actions or motion tokens from state changes in videos and using them as intermediate supervision connecting large-scale video pretraining and robot control (Ye et al., 2025; Chen et al., 2025; Luo et al., 2026b; Routray et al., 2026). Other work directly pretrains complete vision–language–action models (VLAs) or robot policies using large-scale human videos and their associated action information, followed by adaptation with robot data (Luo et al., 2026a; Li et al., 2026; Zheng et al., 2026c). Unlike these methods, ARPG uses large-scale human manipulation data for action representation pretraining, learning reusable action representations for bimanual dexterous manipulation from rich human manipulation experience.

Learning Action Spaces for Generation. Generative robot policies typically model complex multimodal action distributions directly in the raw continuous action space (Chi et al., 2023). To reduce the difficulty of modeling high-dimensional, high-frequency action sequences, subsequent work constructs more compact action parameterizations through frequency-domain compression, vector quantization, or action autoencoders, and performs prediction and generation using discrete action tokens or learned latent action spaces (Lee et al., 2024; Pertsch et al., 2025; Mete et al., 2024; Tan et al., 2025). The above work primarily reduces the difficulty of policy modeling by compressing or reparameterizing the action space, while paying less attention to the pretrained structure of the generative space itself. Related research in visual generation offers another perspective: classical latent-space generative methods typically learn latent spaces using reconstruction objectives (Rombach et al., 2022), whereas recent methods have begun to explicitly use pretrained representations to shape the generative space, either by aligning the internal features of generative models (Yu et al., 2025) or by directly adopting pretrained encoders to define the generative space (Zheng et al., 2026a; Singh et al., 2026). Inspired by this perspective, ARPG pretrains action representations independently and models task-conditioned action distributions for downstream tasks in a fixed pretrained representation space.

3 Action Representation Pretraining

ARPG explicitly decouples action representation learning from downstream conditional generation: it first independently learns a well-structured, decodable action representation space and then fixes this space for action generation in the subsequent stage. The overall pipeline is shown in Figure 2.

3.1 Action Representation Learning

Action representation space. Given a continuous action sequence $A = (a_1, \ldots, a_T)$ of length T, where a_t denotes the action at time step t, we learn an action encoder E_a that maps it to a temporal action representation $Z_a = E_a(A) \in \mathbb{R}^{T \times d}$, where d is the representation dimension. The representation Z_a defines the action space directly modeled by the downstream generative model. We want this space to meet two requirements: it should have a well-structured representation that makes complex action distributions easier to model, and it should retain sufficient motion information to recover executable continuous actions.

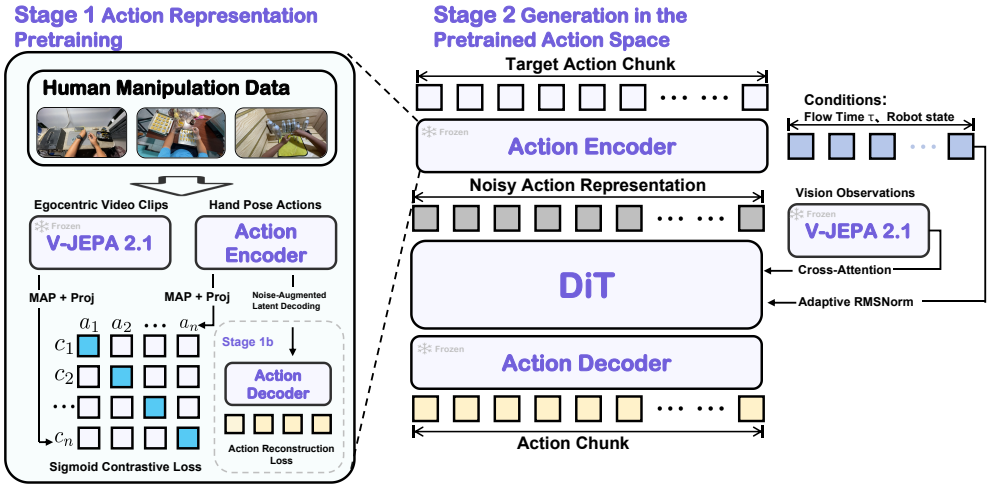


Figure 2: **ARPG framework. Stage 1a** pretrains the action encoder through vision–action alignment on human manipulation data; **Stage 1b** trains the action decoder using demonstrations from the target robot; **Stage 2** learns conditional action generation in the frozen action representation space. The action encoder is used only to construct training targets and does not participate in inference.

Self-supervised learning from action trajectories alone requires the representation structure to be learned from scratch, whereas pretrained video models have already acquired rich motion and interaction structure from large-scale videos; however, their visual representations cannot directly serve as a decodable action space. Therefore, rather than directly reusing visual representations, we use them as a fixed representation reference and train an independent action encoder to align with them, thereby leveraging the structural prior provided by large-scale video pretraining while preserving action information.

Vision–action dual-tower model. Given a manipulation video V temporally synchronized with the action sequence A, the visual branch uses a frozen video encoder E_v to extract spatiotemporal visual features $H_v = E_v(V)$. We instantiate E_v as V-JEPA 2.1 ViT-L/16 (Mur-Labadia et al., 2026); it serves only as a fixed visual reference during representation pretraining and does not participate in subsequent policy inference. The action branch uses an L-layer Transformer to encode continuous action sequences; let $H_a^{(\ell)}$ denote the output of layer ℓ, where $\ell \in \{1, \dots, L\}$.

Inspired by the multi-layer representation aggregation design of RAEv2 (Singh et al., 2026), we exploit complementary information at different depths by normalizing the features from the last K action layers separately and aggregating them with equal weights as $Z_a = \frac{1}{K}\sum_{\ell=L-K+1}^{L} \mathrm{Norm}(H_a^{(\ell)})$, where K is the number of layers included in the aggregation and $\mathrm{Norm}(\cdot)$ denotes a shared output LayerNorm. The resulting Z_a serves both as the input to the action decoder and as the action space for the subsequent generative model. The encoder architecture and network configuration are detailed in Appendix A.

Vision–action representation alignment. Following the cross-modal representation alignment design of SigLIP (Zhai et al., 2023), we apply it to synchronized video–action data. For sample i, the visual features H_v^i and action representation Z_a^i each pass through Multi-head Attention Pooling (MAP) and an independent projection head, and are then normalized to obtain the global visual representation v_i and action representation u_i. For a mini-batch of B video–action samples, (v_i, u_i) from the same clip forms a positive pair, while all other cross-sample combinations form negative pairs. We define the matching score as $s_{ij} = \tau v_i^\top u_j + b$, where τ and b are learnable scale and bias parameters, respectively; let $y_{ij} = 1$ when $i = j$ and $y_{ij} = -1$ otherwise. We optimize the pairwise sigmoid alignment loss

$$\mathcal{L}_{\text{align}} = -\frac{1}{B}\sum_{i=1}^{B}\sum_{j=1}^{B} \log \sigma(y_{ij} s_{ij}) \,, \tag{1}$$

where $\sigma(\cdot)$ is the sigmoid function. During training, only the action encoder E_a, the MAP modules on both branches, and the projection heads are updated, while the visual encoder E_v remains frozen; after pretraining, only E_a is retained for subsequent action modeling. The complete training configuration is provided in Appendix E.

3.2 Pretraining on Human Manipulation Data

Cross-embodiment action representation. In ARPG, action representation learning is decoupled from downstream conditional generation, so the action encoder E_a need not fit the control details of a specific robot embodiment through the generative objective of a robot policy and can instead focus on reusable motion and interaction structure across manipulation data. Accordingly, action representation pretraining is no longer limited to robot demonstrations: we can use larger-scale egocentric human manipulation data with broader behavioral coverage to learn the action space, and subsequently use demonstrations from the target robot for action decoding and conditional generation.

Unified action representation. To allow human manipulation data from different sources and subsequent robot data to be processed by the same action encoder, we convert action trajectories into a fixed-dimensional bimanual action representation. Specifically, the action representation at time step t is $a_t = [a_t^L; a_t^R] \in \mathbb{R}^{324}$, where a_t^L and a_t^R denote the 162-dimensional action slots for the left and right hands, respectively. Each action slot contains joint rotations, palm rotation, absolute wrist position, and wrist-relative keypoints; missing or unavailable hand dimensions are handled with a validity mask. This unified representation maps human hand trajectories and actions from different robot embodiments to the same data interface without changing the input format of the action encoder. The complete dimension definitions, action conversion, and adaptations for different dexterous hands are given in Appendix B.

Human-data pretraining. We construct temporally synchronized one-second vision–action clips from human manipulation data: the visual stream is sampled at $4\,\mathrm{Hz}$ to obtain four egocentric frames, and the action stream is sampled at $30\,\mathrm{Hz}$ to obtain an action sequence of length $T = 30$. The synchronized video V and action sequence A are fed into the frozen visual encoder E_v and the trainable action encoder E_a, respectively, for pretraining with the vision–action alignment objective defined in Section 3.1. The pretraining data are drawn primarily from EgoDex (Hoque et al., 2026) and EgoVerse (Punamiya et al., 2026); data processing, the construction of different pretraining scales, and optimization configurations are detailed in Appendices C and E, respectively.

3.3 Action Decoder

Robot action decoding. After action representation pretraining, we freeze the encoder E_a and train an action decoder D_a using demonstrations from the target robot. For a robot action sequence A, the encoder yields $Z_a = E_a(A)$, and the decoder then predicts executable actions $\hat{A}_{\text{exec}} = D_a(Z_a)$. The decoder is optimized with a Smooth-L_1 reconstruction loss using a validity mask. The unified action format and executable action definition are given in Appendix B, and the decoder architecture and training configuration are given in Appendices A and E.

Noise-augmented decoding. Following the noise-augmented decoding design introduced in RAE (Zheng et al., 2026a), we apply local perturbations to Z_a during training and use the same executable actions as supervision to mitigate the distribution shift between generated action representations and real encoded representations, thereby improving the decoder's robustness to generation errors. The specific perturbation scheme and training configuration are given in Appendix E.

4 Generation in the Pretrained Action Space

After obtaining the pretrained action representation space in Section 3, ARPG learns a conditional action distribution in this fixed space instead of directly generating raw continuous actions. During training, both the action encoder E_a and the action decoder D_a remain frozen; they are used to construct generation targets and recover executable actions, respectively.

Action Representation Targets. At time t, let $A_t = (a_t, \ldots, a_{t+H-1})$ denote a future action chunk, O_t the visual observation, and q_t the robot state. The frozen action encoder first maps

the action chunk into the pretrained action space, and normalization yields the generation target $\bar{Z}_t = N_z(E_a(A_t))$. Thus, ARPG learns the conditional distribution $p_\theta(\bar{Z}_t \mid O_t, q_t)$ rather than directly modeling $p(A_t \mid O_t, q_t)$ in the raw control space. After denormalization, the frozen action decoder D_a recovers continuous actions from the generated result.

Conditional Representation Generation. We use DiT (Peebles & Xie, 2023) as the generative backbone and train it with Flow Matching (Lipman et al., 2023). The frozen V-JEPA 2.1 (Mur-Labadia et al., 2026) extracts the visual condition C_t^v, and the robot state forms the state condition C_t^s. Given Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$ and flow time $\tau \in [0, 1]$, we construct the interpolated representation $Z_t^\tau = (1-\tau)\bar{Z}_t + \tau\epsilon$ and optimize

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}\left[\left\| G_\theta(Z_t^\tau, \tau; C_t^v, C_t^s) - (\epsilon - \bar{Z}_t) \right\|_2^2\right]. \tag{2}$$

Training Objective. In addition to $\mathcal{L}_{\mathrm{FM}}$, we introduce representation-level and action-level auxiliary supervision. Following REPA (Yu et al., 2025), we align the intermediate DiT feature $H_t^{(m)}$ with the target representation produced by the frozen action encoder and define $\mathcal{L}_{\mathrm{REPA}} = \mathbb{E}[\|P_\phi(H_t^{(m)}) - \mathrm{sg}(\bar{Z}_t)\|_2^2]$, where P_ϕ is a projection head and $\mathrm{sg}(\cdot)$ denotes stop-gradient. In addition, $\mathcal{L}_{\mathrm{dec}}$ constrains the decoded action values, while $\mathcal{L}_{\mathrm{smooth}}$ constrains changes between adjacent actions. The final training objective is

$$\mathcal{L} = \lambda_{\mathrm{FM}}\mathcal{L}_{\mathrm{FM}} + \lambda_{\mathrm{REPA}}\mathcal{L}_{\mathrm{REPA}} + \lambda_{\mathrm{dec}}\mathcal{L}_{\mathrm{dec}} + \lambda_{\mathrm{smooth}}\mathcal{L}_{\mathrm{smooth}}. \tag{3}$$

Inference. Starting from Gaussian noise, we generate an action representation through the conditional velocity field and obtain a future action chunk through denormalization and the frozen action decoder D_a. The action encoder E_a is used only to construct supervision targets during training and does not participate in inference. The solver and inference settings are detailed in Appendix E.

5 Experiments

In this section, we conduct extensive experiments focusing on the following questions: **Q1.** Does ARPG improve complex dexterous manipulation, particularly bimanual coordination, while retaining a compact and efficient downstream policy? (Table 1, Figure 3) **Q2.** How should action representation pretraining be scaled to improve downstream action generation? (Figure 4, Table 1) **Q3.** What representation and generative modeling choices are critical for effective conditional action generation in the pretrained action space? (Table 2)

5.1 Experimental Setups

We evaluate ARPG on a simulation benchmark and a real-world robot platform. Complete training and evaluation details are provided in Appendices E, F, and G.

Simulation benchmark. We adopt DexJoCo (Wang et al., 2026) as our primary simulation benchmark. DexJoCo contains 11 task-level dexterous manipulation tasks, including six single-arm tasks and five bimanual coordination tasks. Following its original task definitions and success criteria, we report per-task success rates and compute macro-average success rates across all tasks, the single-arm tasks, and the bimanual tasks.

Real-world robot evaluation. We further evaluate ARPG on three long-horizon dexterous manipulation tasks on a real-world robot platform: hammer handover, pouring coffee beans, and cabinet retrieval. Each task is divided into three ordered stages to characterize stage-wise policy performance over long-horizon execution. ARPG and $\pi_{0.5}$ are evaluated using the same robot demonstrations, initial-state distributions, and success criteria.

Baselines. We compare against two representative large-scale generalist robot policies: $\pi_{0.5}$ (Black et al., 2025a) and GR00T N1.5 (Bjorck et al., 2025). $\pi_{0.5}$ represents the current large-scale VLA paradigm and has shown strong performance in open-world generalization and long-horizon dexterous manipulation; GR00T N1.5 is a large-scale foundation model for general-purpose humanoid robots that supports bimanual and dexterous-hand manipulation and exhibits strong generalization across robot embodiments.

Table 1: **Dexterous manipulation performance and deployment efficiency on the DexJoCo benchmark.** We report macro-average success rates (%) over single-arm, bimanual, and all tasks under single-task and multi-task training, together with inference costs. Complete per-task results are in Appendix H. Bold indicates the best result within each setting.

Method	Inference cost			Success rate (%)		
	Params (B) ↓	VRAM (GB) ↓	Latency (ms) ↓	Single arm ↑	Bimanual ↑	Overall ↑
Single-task training						
$\pi_{0.5}$	3.35	9.05	84.9	69.7	35.3	54.1
GR00T N1.5	2.72	6.21	93.6	56.2	21.1	40.2
ARPG	**0.46**	**2.63**	**50.8**	**75.2**	**44.5**	**61.3**
Multi-task training						
$\pi_{0.5}$	3.35	9.05	84.9	55.8	10.1	35.0
ARPG	**0.46**	**2.63**	**50.8**	**68.8**	**44.7**	**57.8**

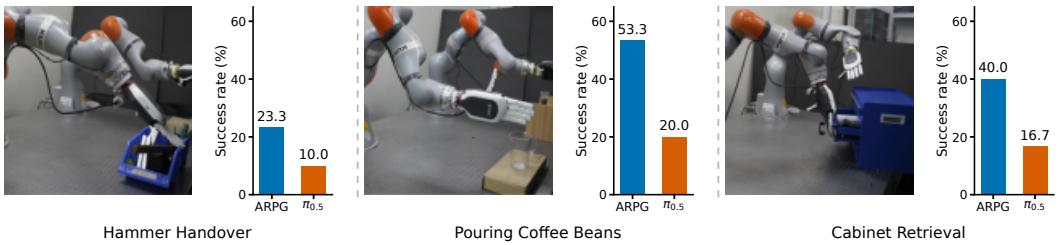


Figure 3: **Real-world task success rates.** We compare ARPG and $\pi_{0.5}$ on Hammer Handover, Pouring Coffee Beans, and Cabinet Retrieval. Each result is the proportion of successful rollouts over 30 trials; photographs show the task setups.

5.2 Effective and Efficient Dexterous Manipulation

We first investigate **Q1**, evaluating ARPG's overall performance and deployment efficiency on complex dexterous manipulation tasks, with results shown in Table 1 and Figure 3.

Dexterous manipulation performance. We first evaluate ARPG's overall dexterous manipulation performance on DexJoCo, using the baseline results reported by DexJoCo (Wang et al., 2026). As shown in Table 1, under single-task training, ARPG achieves an overall average success rate of 61.3%, exceeding the 54.1% of $\pi_{0.5}$ and the 40.2% of GR00T N1.5. Under the more challenging multi-task joint-training setting, ARPG still achieves an overall success rate of 57.8%, while $\pi_{0.5}$ drops to 35.0%. This advantage is particularly pronounced on bimanual dexterous manipulation tasks: ARPG reaches an average success rate of 44.7%, substantially above the 10.1% of $\pi_{0.5}$. These results show that ARPG effectively improves complex bimanual dexterous manipulation.

Deployment efficiency and multi-task scalability. At the same time, ARPG achieves this performance with substantially fewer active parameters, lower peak VRAM, and lower inference latency, demonstrating greater deployment efficiency. More notably, despite its smaller deployment footprint, ARPG's overall success rate changes only from 61.3% under single-task training to 57.8% under multi-task joint training, whereas $\pi_{0.5}$ drops from 54.1% to 35.0%. We believe this robustness is at least partly due to the separation of action representation learning from downstream generation, which allows the compact generative backbone to devote more capacity to modeling task-conditioned action distributions.

Real-world evaluation. We further evaluate ARPG on a real robot and compare it with $\pi_{0.5}$ on three long-horizon dexterous manipulation tasks: hammer handover, coffee-bean pouring, and cabinet retrieval. As shown in Figure 3, ARPG achieves success rates of 23.3%, 53.3%, and 40.0% on the three tasks, respectively, exceeding the 10.0%, 20.0%, and 16.7% achieved by $\pi_{0.5}$. The largest gap is on coffee-bean pouring, at 33.3 percentage points. These results show that ARPG's dexterous manipulation advantage in simulation is also reflected in long-horizon manipulation on a real robot.

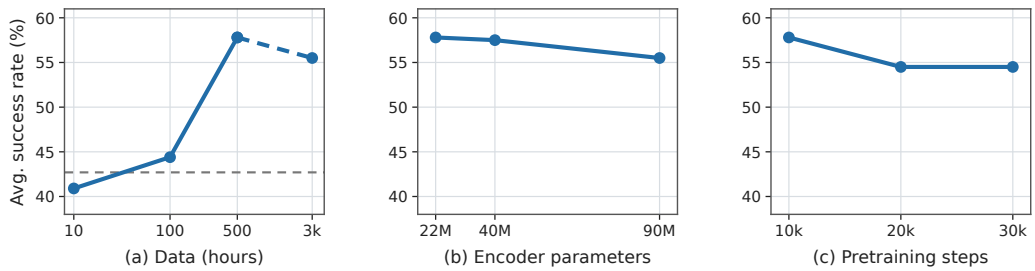


Figure 4: **Scaling action representation pretraining.** DexJoCo average success rate as a function of (a) human pretraining data, (b) action encoder parameter count, and (c) pretraining steps. In (a), the solid line connects the 10–500-hour results, the dashed blue segment connects 500 and 3,000 hours, and the dashed gray line marks the Robot-only reference.

5.3 Scaling Action Representation Pretraining for Downstream Generation

We next investigate **Q2**. To this end, we scale pretraining data, action encoder size, and pretraining compute separately, always holding other configurations fixed and changing only action representation pretraining itself.

We first investigate how to scale the pretraining data. Specifically, we construct nested human-manipulation datasets of 10, 100, 500, and 3,000 hours: each larger set retains every clip in the preceding set and adds new data. We additionally pretrain a Robot-only encoder on 8.8 hours of robot demonstrations from DexJoCo as an in-domain reference. From 10 to 500 hours, the cumulative expansion increases both data volume and the range of manipulation behaviors. From 500 to 3,000 hours, the added clips are concentrated in task categories already represented in the smaller set, although the number of distinct tasks also increases; the data composition is detailed in Appendix C. As shown in Figure 4, downstream success rate rises from 40.9% to 57.8% as the nested human-data sets grow from 10 to 500 hours. Pretraining on 8.8 hours of in-domain robot data yields a 42.7% success rate, below that of the 100-hour human-data setting. Expanding from 500 to 3,000 hours does not improve downstream success rate in this construction.

These observations show that adding clips from this data pool does not always yield further downstream gains. We hypothesize that the limited benefit of the 500-to-3,000-hour expansion may have two causes. First, our contrastive learning objective treats all unpaired video–action sequences as negatives; similar manipulation sequences may therefore become false negatives. Second, the added clips are concentrated in already represented task categories and may contribute less new motion and interaction structure. Because data volume and behavioral coverage change together in the nested 10-to-500-hour expansion, these experiments do not isolate the causal effect of either factor. A fixed-volume comparison with different behavioral coverage is needed to test that hypothesis.

Further increasing the action encoder size or pretraining compute likewise does not consistently improve downstream success rate. This does not imply that action representation quality has not improved; rather, any such improvement has not translated into better downstream action generation. This indicates that optimizing the representation pretraining objective more thoroughly is not equivalent to obtaining action representations better suited to generation. One possible reason is that our pretraining objective constrains sequence-level vision–action alignment after pooling, whereas the downstream DiT actually models token-level temporal action representations. Therefore, increasing model capacity or training compute may further improve global alignment without necessarily improving the local information on which downstream generation depends. Within the scope of our experiments, the nested 10-to-500-hour expansion improves downstream performance, while increasing data volume beyond 500 hours, encoder capacity, or pretraining compute yields no consistent additional benefit.

Table 2: Ablations of action representation and generative design. We report the overall success rate (%) on DexJoCo.

(a) Action representation		(b) Aggregation depth		(c) Auxiliary objectives	
Representation	Overall ↑	Last-K	Overall ↑	Objective	Overall ↑
Raw Action	13.3	1	52.5	w/o $\mathcal{L}_{\text{REPA}}$	54.85
Action-MAE	36.4	2	**57.8**	w/o $\mathcal{L}_{\text{dec}}$	56.0
Action-iBOT	30.5	3	51.6	Full	**57.8**
Visual Alignment + iBOT	45.6				
Visual Alignment	**57.8**				

5.4 Ablations and Analysis

Finally, we investigate **Q3**. To this end, we hold the robot demonstrations, observation conditions, and downstream generative model fixed, vary only how the action representation is learned, and further ablate multilayer feature aggregation and the DiT training objective, with results shown in Table 2(a)–(c).

Action representation learning. As shown in Table 2(a), both Action-MAE and Action-iBOT improve over Raw Action, supporting the effectiveness of generation in a pretrained action representation space. However, the gains from learning representations solely from action trajectories remain limited. Introducing Visual Alignment further improves downstream performance and achieves the best result. We believe this improvement arises primarily because visual representations provide the action space with a structural reference shaped by large-scale pretraining: compared with action-only self-supervision, which must learn representation structure from scratch, Visual Alignment transfers the visual model's existing representation structure and interaction semantics into the action representation while retaining the information needed for action decoding. The resulting action space is therefore easier for the downstream generative model to learn.

Additional ablations. We further ablate key design choices in representation construction and downstream generative training. As shown in Table 2(b), when the last $K = 1, 2, 3$ layers are used to construct the action representation, $K = 2$ achieves the best result, supporting the use of multilayer deep-feature aggregation in the main model. Meanwhile, the ablations of $\mathcal{L}_{\text{REPA}}$ and $\mathcal{L}_{\text{dec}}$ in Table 2(c) show that the full DiT training objective achieves the best downstream performance, validating the effectiveness of these auxiliary constraints for generating action representations.

6 Limitations and Conclusion

We present ARPG, which decouples action representation pretraining from downstream conditional action generation, enabling action representations to be pretrained on scalable human manipulation data and providing downstream policies with a representation space better suited to generation. Experiments show that ARPG achieves stronger dexterous manipulation performance with a compact downstream policy and demonstrates substantial advantages in multi-task bimanual manipulation. Scaling experiments further show improved downstream performance as nested human-manipulation pretraining sets grow from 10 to 500 hours. Overall, ARPG shows that scalable action representation pretraining offers a path to scaling complex dexterous manipulation independently of the scale of robot demonstrations and the capacity of the downstream policy.

The current work has several limitations. First, ARPG has so far been validated only on dexterous-hand manipulation; its ability to transfer to other robot embodiments, such as grippers, remains to be studied. Second, this work focuses primarily on complex action generation and has not systematically evaluated semantic understanding or task generalization in open environments. Furthermore, the current pretraining data and compute scales remain limited, and scaling behavior at larger scales requires further validation. We leave these directions for future work.

AI USE STATEMENT

In this work, we used generative AI tools to assist with the preparation of the manuscript and research code. Specifically, generative AI tools were used for Chinese–English translation and language editing of technical text, assistance with manuscript wording and LaTeX editing, and software-engineering support for the research code, including code reading, refactoring, debugging, and documentation. All AI-assisted text, code, and documentation were reviewed by the authors, and all content concerning methodological descriptions, mathematical expressions, citations, and experimental results was manually verified by the authors. Generative AI tools were not used as a source of experimental data or as a substitute for simulation or real-robot experiments; all quantitative results reported in this paper were obtained from experiments executed and verified by the authors. The authors take full responsibility for the final content of this work, including any text, code, or other artifacts produced or modified with the assistance of generative AI.

ETHICS STATEMENT

This work uses existing large-scale egocentric human manipulation data for action representation pretraining, with the current training corpus primarily consisting of EgoDex and EgoVerse. ARPG uses manipulation videos and information related to hand motion and actions solely to learn representations for robotic manipulation; the work does not aim to perform identity recognition, biometric identification, or inference of sensitive personal attributes. For third-party human data, we follow the access, usage, and redistribution conditions specified by the respective data sources, and we do not redistribute raw human videos whose redistribution is restricted by the original licenses or access terms. Information regarding participant consent, privacy protection, and ethical approval is governed by the official documentation provided by the original dataset creators. Because egocentric videos may contain privacy-sensitive visual information, we restrict their use to manipulation representation learning within the scope permitted by the corresponding data sources.

This work also includes dexterous manipulation experiments on physical robot platforms. These experiments are conducted in controlled environments with operator supervision, emergency-stop mechanisms, robot joint and workspace constraints, motion-speed limits, and the native safety mechanisms of the robotic platforms to reduce risks associated with unintended motion, collisions, and object damage. Outputs of the learned policy are not used to bypass the existing safety layers of the robotic system. We further acknowledge that learned policies may exhibit unexpected behavior when deployed outside the robot embodiments, tasks, and environmental distributions evaluated in this work; deployment on new platforms therefore requires independent calibration, safety validation, and human supervision. The intended use of this work is research on robotic manipulation, rather than identity surveillance or applications targeting sensitive personal information.

REPRODUCIBILITY STATEMENT

We support the reproducibility of ARPG through the main paper, the appendix, and an anonymized code repository. The code is available at `https://anonymous.4open.science/r/review-code-7f2a/`. Section 3 describes the action representation encoder, the vision–action representation alignment objective, and the action decoder; Section 4 describes conditional generation in the pretrained action space, the Flow Matching training objective, and the inference procedure; and Section 5 specifies the primary evaluation settings for DexJoCo and the real-robot experiments. The appendix further provides the complete network architectures and training hyperparameters, the preprocessing pipeline that converts human manipulation data into the unified D324 action representation, data normalization procedures, action-decoder training and latent-perturbation settings, downstream policy training and inference configurations, and the construction of human-pretraining subsets at different data scales. For both simulation and real-robot experiments, the appendix additionally documents the task settings, success criteria, training and evaluation protocols, and other implementation details. The anonymized code repository contains the implementations, configuration files, and software-environment specifications required for data conversion, model training, inference, and evaluation. For third-party datasets subject to their original licenses or access restrictions, we do not redistribute restricted raw data; instead, we provide the data sources and deterministic preprocessing procedures so that researchers with appropriate access can reconstruct the training inputs used in this work.

REFERENCES

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025. URL https://arxiv.org/abs/2506.09985.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

Johan Bjorck, Valts Blukis, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Xiaowei Jiang, Kaushil Kundalia, Jan Kautz, Zhiqi Li, Kevin Lin, Zongyu Lin, Loic Magne, Yunze Man, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Jing Wang, Qi Wang, Shihao Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Seonghyeon Ye, Zhiding Yu, Yizhou Zhao, Zhe Zhang, Ruijie Zheng, and Yuke Zhu. GR00T N1.5: An improved open foundation model for generalist humanoid robots. NVIDIA Research, 2025. URL https://research.nvidia.com/labs/gear/gr00t-n1_5/.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 2025a. URL https://proceedings.mlr.press/v305/black25a.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems*, 2025b.

Yi Chen, Yuying Ge, Weiliang Tang, Yizhuo Li, Yixiao Ge, Mingyu Ding, Ying Shan, and Xihui Liu. Moto: Latent motion token as the bridging language for learning robot manipulation from videos. In *IEEE/CVF International Conference on Computer Vision*, pp. 19752–19763, 2025.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Ryan Hoque, Peide Huang, David J. Yoon, Mouli Sivapurapu, and Jian Zhang. EgoDex: Learning dexterous manipulation from large-scale egocentric video. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=FFxkFMU89E.

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. In *International Conference on Machine Learning*, pp. 26991–27008, 2024.

Qixiu Li, Yu Deng, Yaobo Liang, Lin Luo, Lei Zhou, Chengtang Yao, Lingqi Zeng, Zhiyuan Feng, Huizhi Liang, Sicheng Xu, Yizhong Zhang, Xi Chen, Hao Chen, Lily Sun, Dong Chen, Jiaolong Yang, and Baining Guo. Scalable vision-language-action model pretraining for robotic manipulation with real-life human activity videos. In *IEEE International Conference on Robotics and Automation*, 2026. URL https://arxiv.org/abs/2510.21571.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Hao Luo, Yicheng Feng, Wanpeng Zhang, Sipeng Zheng, Ye Wang, Haoqi Yuan, Jiazheng Liu, Chaoyi Xu, Qin Jin, and Zongqing Lu. Being-H0: Vision-language-action pretraining from large-scale human videos. In *International Conference on Machine Learning*, 2026a.

Hao Luo, Ye Wang, Wanpeng Zhang, Haoqi Yuan, Yicheng Feng, Haiweng Xu, Sipeng Zheng, and Zongqing Lu. Joint-aligned latent action: Towards scalable VLA pretraining in the wild. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026b.

Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. QueST: Self-supervised skill abstractions for learning continuous control. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-JEPA 2.1: Unlocking dense features in video self-supervised learning. In *Computer Vision – ECCV 2026*, volume 17005 of *Lecture Notes in Computer Science*, pp. 671–689. Springer, 2026. doi: 10.1007/978-3-032-37369-4_36.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Peebles_Scalable_Diffusion_Models_with_Transformers_ICCV_2023_paper.html.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. In *Robotics: Science and Systems*, 2025.

Ryan Punamiya, Simar Kareer, Zeyi Liu, Josh Citron, Ri-Zhao Qiu, Xiongyi Cai, Alexey Gavryushin, Jiaqi Chen, Davide Liconti, Lawrence Y. Zhu, Patcharapong Aphiwetsa, Baoyu Li, Aniketh Cheluva, Pranav Kuppili, Yangcen Liu, Dhruv Patel, Aidan Gao, Hye-Young Chung, Ryan Co, Renee Zbizika, Jeff Liu, Xiaomeng Xu, Haoyu Xiong, Geng Chen, Sebastiano Oliani, Chenyu Yang, Xi Wang, James Fort, Richard Newcombe, Josh Gao, Jason Chong, Garrett Matsuda, Aseem Doriwala, Marc Pollefeys, Robert Katzschmann, Xiaolong Wang, Shuran Song, Judy Hoffman, and Danfei Xu. EgoVerse: An egocentric human dataset for robot learning from around the world. In *Robotics: Science and Systems*, 2026. URL https://arxiv.org/abs/2604.07607.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Sandeep Kumar Routray, Hengkai Pan, Unnat Jain, Shikhar Bahl, and Deepak Pathak. ViPRA: Video prediction for robot actions. In *International Conference on Learning Representations*, 2026.

Jaskirat Singh, Boyang Zheng, Zongze Wu, Richard Zhang, Eli Shechtman, and Saining Xie. Improved baselines with representation autoencoders. *arXiv preprint arXiv:2605.18324*, 2026.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Wenhui Tan, Bei Liu, Junbo Zhang, Ruihua Song, and Jianlong Fu. RoLD: Robot latent diffusion for multi-task policy modeling. In *MultiMedia Modeling*, pp. 340–353. Springer, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Hanwen Wang, Weizhi Zhao, Xiangyu Wang, Siyuan Huang, He Lin, Boyuan Zheng, Rongtao Xu, Gang Wang, Yao Mu, He Wang, Lue Fan, Hongsheng Li, Zhaoxiang Zhang, and Tieniu Tan. DexJoCo: A benchmark and toolkit for task-oriented dexterous manipulation on MuJoCo. *arXiv preprint arXiv:2605.16257*, 2026.

Ruihan Yang, Qinxi Yu, Yecheng Wu, Rui Yan, Borui Li, An-Chieh Cheng, Xueyan Zou, Yunhao Fang, Xuxin Cheng, Ri-Zhao Qiu, Hongxu Yin, Sifei Liu, Song Han, Yao Lu, and Xiaolong Wang. EgoVLA: Learning vision-language-action models from egocentric human videos. *arXiv preprint arXiv:2507.12440*, 2025.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *International Conference on Learning Representations*, 2025.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi Fan, and Joel Jang. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026. URL https://arxiv.org/abs/2602.15922.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/d9e42b4d7163931f3689d6d6fbaa11d0-Abstract-Conference.html.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *International Conference on Computer Vision*, 2023.

Gu Zhang, Qicheng Xu, Haozhe Zhang, Jianhan Ma, Long He, Yiming Bao, Zeyu Ping, Zhecheng Yuan, Chenhao Lu, Chengbo Yuan, Tianhai Liang, Xiaoyu Tian, Maanping Shao, Feihong Zhang, Mingyu Ding, Yang Gao, Hao Zhao, Hang Zhao, and Huazhe Xu. UniDex: A robot foundation suite for universal dexterous hand control from egocentric human videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026a.

Yuhong Zhang, Zihan Gao, Shengpeng Li, Ling-Hao Chen, Kaisheng Liu, Runqing Cheng, Xiao Lin, Junjia Liu, Zhuoheng Li, Jingyi Feng, Ziyan He, Jintian Lin, Zheyan Huang, Zhifang Liu, and Haoqian Wang. RoboWheel: A data engine from real-world human demonstrations for cross-embodiment robotic learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026b.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016. URL https://roboticsproceedings.org/rss19/p016.html.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *International Conference on Learning Representations*, 2026a.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Ya-Qin Zhang, Jiangmiao Pang, Jingjing Liu, Tai Wang, and Xianyu Zhan. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, 2026b.

Ruijie Zheng, Dantong Niu, Yuqi Xie, Jing Wang, Mengda Xu, Yunfan Jiang, Fernando Castañeda, Fengyuan Hu, You Liang Tan, Letian Fu, Trevor Darrell, Furong Huang, Yuke Zhu, Danfei Xu, and Linxi Fan. EgoScale: Scaling dexterous manipulation with diverse egocentric human data. *arXiv preprint arXiv:2602.16710*, 2026c.

Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. iBOT: Image BERT pre-training with online tokenizer. In *International Conference on Learning Representations*, 2022.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5745–5753, 2019.

A Architecture Details

A.1 Action Encoder

The action encoder E_a uses a Transformer (Vaswani et al., 2017) to model the temporal structure of continuous bimanual action sequences. For an action clip $A \in \mathbb{R}^{30\times 324}$ of length $T = 30$, the action at each time step is first encoded in groups according to the anatomical structure of the two hands and then mapped to a frame-level action token. After multi-layer Transformer encoding, features from multiple terminal layers are aggregated into the final action representation $Z_a \in \mathbb{R}^{30\times 512}$.

Action tokenization. The 324-dimensional action in each frame is divided into the wrist and five fingers for the left and right hands, yielding 12 action groups in total. Each action group is first mapped to a 32-dimensional local feature by an independent linear layer with GELU activation; the 12 group features are then concatenated and further projected into a 512-dimensional action token. Invalid action dimensions are zeroed before tokenization; when an entire time step contains no valid action, the corresponding token is masked in temporal attention.

Transformer encoder. The action encoder contains eight Transformer blocks with a hidden dimension of 512, eight attention heads, an MLP ratio of 4, dropout of 0.1, and rotary positional embeddings (Su et al., 2024) for temporal position encoding.

Multi-layer action representation. To exploit complementary action information in deep Transformer layers, we apply a shared LayerNorm to the outputs of the last K layers and aggregate them uniformly along network depth:

$$E_a(A) = \frac{1}{K} \sum_{\ell=L-K+1}^{L} \mathrm{Norm}\left(H_a^{(\ell)}\right). \tag{4}$$

The main model uses $K = 2$, namely, it aggregates the outputs of the last two Transformer blocks, ultimately producing $Z_a \in \mathbb{R}^{30\times 512}$.

Sequence pooling and projection. During action-representation pretraining, the visual and action branches use separate Multi-head Attention Pooling (MAP) modules to aggregate token sequences into sequence-level representations. Each MAP module uses a learnable query, applies LayerNorm separately to the query and input tokens, and performs sequence aggregation through eight-head cross-attention. The 1024-dimensional pooled feature on the visual side and the 512-dimensional pooled feature on the action side are mapped into a shared 512-dimensional space by separate linear projection heads and then ℓ_2-normalized.

A.2 Action Encoder Training Objectives

Given a time-synchronized first-person video clip V_i and action sequence A_i, the frozen V-JEPA 2.1 encoder (Mur-Labadia et al., 2026) produces visual tokens, while the action encoder produces the corresponding action tokens. The two branches are passed through their respective MAP modules and projection heads to obtain normalized sequence-level representations v_i and a_i. The main model in this paper uses Visual Alignment as the action-representation pretraining objective.

Visual–action alignment. For a mini-batch containing B synchronized video–action sequence pairs, the visual and action representations from the same clip form a positive pair, while all other cross-sequence combinations form negative pairs. The matching score between visual representation v_i and action representation a_j is

$$s_{ij} = \tau v_i^\top a_j + b, \tag{5}$$

where τ and b are learnable parameters. The action encoder learns an action structure consistent with the visual interaction process through a pairwise sigmoid objective (Zhai et al., 2023). During training, V-JEPA 2.1 remains frozen, and only the action encoder, MAP modules, and projection heads are updated.

Action-MAE. Action-MAE (He et al., 2022) uses the same action tokenization and Transformer encoder as the main method but does not use paired video supervision. During training, valid action groups are randomly masked, and a training-only reconstruction decoder predicts the masked raw actions. This decoder concatenates the 30 encoded latent tokens with 30 learnable query tokens

carrying temporal embeddings, processes them with four pre-norm Transformer layers of hidden dimension 512 and eight attention heads, and applies LayerNorm and a linear layer to output a 30×324 action sequence; reconstruction loss is applied only to valid masked action dimensions. The reconstruction objective consists of a Smooth-L_1 reconstruction loss and a temporal velocity loss to learn local structure and temporal variation within action sequences.

Action-iBOT. Action-iBOT likewise keeps the action tokenizer and Transformer encoder unchanged and uses only action-side masked self-distillation (Zhou et al., 2022). During training, 50% of the valid action content is masked, and teacher–student prediction is used to learn representations of masked action tokens. The projection head uses a 2,048-dimensional hidden layer and a 256-dimensional bottleneck and predicts 8,192 prototypes; its teacher update, temperature schedule, and other training configurations are provided in Appendix E.

Visual Alignment. Visual Alignment uses only visual–action sigmoid alignment and does not introduce action-side masked self-distillation. This setting uses the motion and interaction semantic constraints provided by synchronized human-manipulation videos to structure the action-representation space.

Visual Alignment + iBOT. Visual Alignment + iBOT jointly uses visual–action alignment and action-side masked self-distillation, retaining the self-supervised learning signal within action sequences while exploiting cross-modal interaction information.

Raw Action. Raw Action does not use a pretrained action encoder or action decoder; instead, the generative model directly predicts continuous actions in the robot-executable action space, enabling comparison between conditional action generation in the raw action space and in the pretrained representation space.

A.3 Action Decoder

The action decoder D_a maps the latent sequence produced by the action encoder back to the continuous actions required by the robot-control interface. Its input is $Z_a \in \mathbb{R}^{30\times512}$, and its output is a bimanual executable action sequence of the same length, $\hat{A}_{\text{exec}} \in \mathbb{R}^{30\times198}$.

Latent memory and query tokens. For the 30 latent tokens, the decoder introduces 30 learnable query tokens with temporal indices and concatenates the latent memory with the query tokens. Token-type embeddings and temporal embeddings are added separately to the two token types.

Transformer decoder. The concatenated sequence is processed by a five-layer pre-norm Transformer. The decoder has a hidden dimension of 512, eight attention heads, an MLP ratio of 4, and dropout of 0.1.

Action output. Only the query-token features are retained at the output and passed through LayerNorm and a linear prediction head to produce a $[30, 198]$ continuous action sequence. The 198-dimensional output concatenates the 99-dimensional executable action of each hand:

$$\hat{A}_{\text{exec}} = [\hat{A}^L_{0:99}; \hat{A}^R_{0:99}] \in \mathbb{R}^{30\times198}. \tag{6}$$

The additional 126 dimensions of wrist-relative keypoint positions in the unified action representation are not output by the action decoder; their precise definition is provided in Appendix B.

A.4 Generative Backbone

The downstream policy uses a conditional DiT (Peebles & Xie, 2023) to generate future action latents in the representation space defined by the frozen action encoder. The action latent sequence serves as the main sequence of the generative model, multi-view visual tokens are injected into the generative backbone through cross-attention, and the robot proprioceptive state and flow time modulate the action features as global conditions.

DiT backbone. The generative backbone contains ten Transformer blocks with a hidden dimension of 768, 12 attention heads, an MLP ratio of 4, bidirectional self-attention, temporal rotary positional embeddings, and zero dropout. The input and output projections transform between 512-dimensional action latents and 768-dimensional DiT hidden features. The main sequence contains 30 latent tokens, corresponding to a one-second, 30-step future action chunk.

Visual conditioning. ARPG uses three views: a first-person camera and left and right wrist cameras. Each view provides two consecutive 384×384 RGB frames, which are encoded by a frozen V-JEPA 2.1 ViT-L/16. With a tubelet size of 2 and a spatial token grid of 24×24, each view yields 576 visual tokens of dimension 1024. Separate learnable camera embeddings are added to the three fixed views; the visual tokens are then concatenated and injected into the action latent sequence through cross-attention in every DiT block.

Robot-state conditioning. The 324-dimensional robot proprioceptive state is first mapped to a 768-dimensional conditioning feature by a three-layer GELU MLP.

Flow-time conditioning. The Flow Matching time τ is encoded with a sinusoidal embedding and jointly fed with the robot-state feature into a two-layer SiLU modulation network. For every DiT block, the modulation network produces the shift, scale, and residual gate required by adaptive RMSNorm, thereby injecting the current robot state and flow time into the self-attention and feed-forward sublayers.

Intermediate representation alignment. REPA supervision (Yu et al., 2025) is applied to the intermediate feature from the fourth DiT block. After RMSNorm, this feature is mapped to 512 dimensions through a linear projection head and aligned with the clean action representation produced by the frozen action encoder. Its loss weight and other auxiliary-supervision configurations are provided in Appendix E.

B Unified Action Format

To unify the action forms of human-manipulation data and different robot platforms, we convert all action trajectories into a fixed-dimensional bimanual action format $A \in \mathbb{R}^{T \times 324}$. Each hand corresponds to a 162-dimensional action slot consisting of local joint rotations, palm rotation, absolute wrist position, and wrist-relative keypoints. All rotations use a continuous 6D representation (Zhou et al., 2019), and positions are measured in meters; hand keypoints are represented relative to the wrist so that global translation information is concentrated in the absolute wrist position. The left and right hands use the same action layout, and a validity mask handles missing or unavailable action dimensions.

Action format. The 162-dimensional action for one hand consists of 15 local joint rotations, palm rotation, absolute wrist position, and 21 wrist-relative keypoints. The two 162-dimensional slots for the left and right hands are concatenated to form the complete 324-dimensional action input. The specific layout of each component is shown below, where indices use zero-based, half-open intervals.

Executable actions. The complete 324-dimensional action is input to the action encoder to learn an action representation containing both kinematic state and local hand geometry. The final 63 wrist-relative keypoint dimensions of each hand serve only as auxiliary geometric information for action-representation learning and are not direct control quantities required by the robot-control interface. Therefore, the action decoder predicts only the first 99 dimensions of each hand and concatenates the left- and right-hand outputs as

$$A_{\text{exec}} = [A^L_{0:99}; A^R_{0:99}] \in \mathbb{R}^{T \times 198}. \tag{7}$$

Thus, the 324-dimensional unified action format is used for action encoding and representation learning, whereas the policy ultimately outputs and sends a 198-dimensional executable action to the robot controller; the additional 126 wrist-relative keypoint dimensions are not output as control commands.

Dexterous-hand adaptation. To accommodate dexterous hands with different structures, we use fixed bimanual action slots and handle missing or unavailable action dimensions through a validity mask. For the four-finger Allegro hand used in DexJoCo, we use MuJoCo forward kinematics to construct the wrist-relative keypoints required by the unified action format, while dimensions corresponding to the nonexistent fifth finger are set to zero and marked invalid; these invalid dimensions are ignored when computing the relevant training losses.

Table 3: Detailed layout of the unified action format. Indices are zero-based and half-open.

Hand	Action component	Index range	Dim.	Decoder output
Left	15 local joint rotations (6D)	$[0, 90)$	90	Yes
Left	Palm rotation (6D)	$[90, 96)$	6	Yes
Left	Absolute wrist position	$[96, 99)$	3	Yes
Left	21 wrist-relative keypoints	$[99, 162)$	63	No
Right	15 local joint rotations (6D)	$[162, 252)$	90	Yes
Right	Palm rotation (6D)	$[252, 258)$	6	Yes
Right	Absolute wrist position	$[258, 261)$	3	Yes
Right	21 wrist-relative keypoints	$[261, 324)$	63	No

C Human Manipulation Pretraining Data

The pretraining corpus for the action encoder consists of two large-scale first-person human-manipulation datasets, EgoDex and EgoVerse, which contain time-synchronized first-person videos and 3D hand-motion trajectories. We uniformly convert the different data sources into one-second visual–action clips: the visual stream is sampled at 4 Hz to obtain four first-person images, while the action stream is sampled at 30 Hz to obtain 30 consecutive actions and converted into the 324-dimensional unified action format defined in Appendix B. Human-manipulation data are used only for representation pretraining of the action encoder; the action decoder and downstream generative policy are both trained exclusively on target-robot demonstrations.

C.1 Main Datasets

EgoDex. EgoDex (Hoque et al., 2026) provides first-person human-manipulation videos and 3D hand and finger motion trajectories captured with Apple Vision Pro. We use the first-person RGB videos and their corresponding hand trajectories to construct synchronized visual–action training samples, and the task labels in the dataset are not used as inputs to the action encoder.

EgoVerse. EgoVerse (Punamiya et al., 2026) provides geographically distributed first-person manipulation videos with hand-motion trajectories for robot learning. For clips in which only one hand is fully tracked, we retain the action trajectory of the valid hand and handle the missing second hand through the validity mask in Appendix B rather than discarding the entire clip.

C.2 Data Filtering

The two data sources use the same action-conversion and quality-control procedure. We use a hand-tracking confidence threshold of 0.05 and require each frame to contain at least 12 valid joints. For samples whose forward-kinematics reconstruction error exceeds 0.02 m or whose inter-frame rotation change exceeds 90°, the corresponding frame is marked invalid. To reduce high-frequency jitter from hand tracking, we apply five-frame temporal smoothing within contiguous valid action intervals and avoid interpolation or smoothing across temporal intervals with missing hands. After conversion and quality control, all valid trajectories are further divided into uniform one-second training clips.

C.3 Training-Data Scale

We construct the 10-, 100-, 500-, and 3,000-hour pretraining sets from the same converted data pool while keeping the action format, data-preprocessing procedure, and model architecture unchanged. We order complete data shards deterministically and expand the sets cumulatively: every clip in the 10-hour set is retained in the 100-hour set, every clip in the 100-hour set is retained in the 500-hour set, and the 500-hour set is contained in the full 3,000-hour set. Each expansion adds shards without removing earlier clips; the 500-hour set approximately preserves the proportions of the two data sources in the full pool. The full set contains 11,117,819 one-second training clips. Its approximately 3,088 hours, referred to as 3,000 hours in the main paper, are the sum of clip durations rather than the duration of unique source recordings; overlapping clips, if present, are counted separately.

Table 4: Scale and task distribution of human-manipulation pretraining data.

Metric	500 h	3,000 h
Summed clip duration	500 h 20 m 16 s	3,088 h 16 m 59 s
Total samples	1,801,216	11,117,819
Number of tasks	17,382	23,316
Largest single-task share	0.09%	9.85%
Top-10 task share	0.77%	23.87%
Mean samples per task	103.63	476.83
Median samples per task	39	54
P90 samples per task	213	453
EgoDex / EgoVerse share	22.00% / 78.00%	22.01% / 77.99%

D Robot Demonstrations and Downstream Data

Both the action decoder and the downstream generative policy are trained exclusively on target-robot demonstrations. For simulation experiments, we use the robot demonstrations provided by DexJoCo; for real-robot experiments, we collect data for three long-horizon manipulation tasks on a dual-arm dexterous-hand platform. Robot actions from different platforms are converted into the unified action format defined in Appendix B, and the same data interface provides visual observations, robot proprioceptive states, and future action targets.

D.1 DexJoCo Data

DexJoCo (Wang et al., 2026) contains approximately 1.1K robot demonstrations covering 11 dexterous-manipulation tasks. We use the official `rand-obj` setting, which contains six single-arm tasks and five bimanual tasks. The simulation platform consists of Franka Panda robot arms and Wonik Allegro dexterous hands.

- **Single-arm tasks:** Hammer Nail, Click Mouse, Pick Bucket, Pinch Tongs, Fold Glasses, and Water Plant.
- **Bimanual tasks:** Unlock iPad, Hanoi, Assembly, Microwave, and Photograph.

We follow the official DexJoCo demonstration data and environment configuration and convert the original robot trajectories into the unified training format. Each time step contains visual observations from the first-person camera and the left and right wrist cameras, the robot proprioceptive state, the action target, and the corresponding validity mask. The original TCP targets and Allegro hand-control quantities are further converted into the 324-dimensional unified action format for training the action encoder, action decoder, and downstream generative policy.

D.2 Real-Robot Data-Collection Hardware

The real-robot platform consists of two 7-DoF KUKA iiwa robot arms and two 12-DoF DexHand dexterous hands. Visual observations are provided by three camera views, including one Intel RealSense L515 scene camera and two Intel RealSense D405 wrist cameras mounted at the ends of the left and right robot arms. The L515 records the scene view at a resolution of 960×540, and the two D405 cameras record the wrist views at a resolution of 640×480. Robot states and multi-camera images are temporally synchronized during data recording, with a maximum permitted temporal offset of 50 ms between the robot state and visual frames.

D.3 Real-Robot Data Content

We collect robot demonstrations for three long-horizon bimanual dexterous-manipulation tasks: Hammer Handover, Pouring Coffee Beans, and Cabinet Retrieval. Each demonstration records synchronized three-view RGB images, dual-arm joint states, bimanual hand states, and the corresponding action targets, which are uniformly converted into the 324-dimensional action format in Appendix B. The data scale for each task is as follows:

Table 5: Data scale for the three real-robot tasks. Durations are estimated original recording times (h:mm:ss).

Task	Task description	Demonstrations	Duration
Hammer Handover	Pick up, hand over, and place a hammer	40	3:19:31
Pouring Coffee Beans	Pick up a container and pour coffee beans	40	2:15:03
Cabinet Retrieval	Open a cabinet, retrieve the target, and close it	40	1:47:31
Total	—	**120**	**7:22:05**

Table 6: Hyperparameters for action representation pretraining.

Pretraining objective	Visual Alignment
Optimizer	AdamW
Batch size	1,536
Learning rate	2×10^{-4}
Minimum learning rate	2×10^{-6}
Weight decay	0.05
Warmup steps	1,000
LR schedule	Cosine
Training steps	21,714
Training epochs	3
Precision	bfloat16
Gradient clipping	1.0
MAP attention heads	8
Projection dimension	512
Initial logit scale	10
Initial bias	-10

Table 7: Hyperparameters for downstream policy training.

Optimizer	AdamW
Batch size	128
Learning rate	1×10^{-4}
Minimum learning rate	1×10^{-6}
Weight decay	0.05
Warmup steps	500
LR schedule	Cosine
Training steps	120,000
Precision	bfloat16
Gradient clipping	1.0
EMA decay	0.9999
Flow-time sampling	$0.999\,\mathrm{Beta}(1.5, 1) + 0.001$
λ_{FM}	1.0
λ_{REPA}	0.5
λ_{dec}	1.0
$\lambda_{\mathrm{smooth}}$	0.01
ODE solver	Euler
Inference steps	10

All real-robot demonstrations are collected using hand motion-capture equipment. The action decoder and downstream policy use the same target-robot demonstrations to learn the latent-to-action mapping and the task-conditioned action-generation distribution, respectively. The durations in the table are computed from the processed training data at 30 Hz and multiplied by three to estimate the original recording times before clipping and acceleration.

E Training Details

Action-encoder pretraining. The action encoder is pretrained using Visual Alignment. The frozen V-JEPA 2.1 model provides visual representations, while only the action encoder, the visual-side and action-side MAP modules, and the projection heads are optimized during training. For a mini-batch containing B synchronized video–action clips, the visual–action pair from the same clip is treated as a positive, whereas all cross-clip combinations are treated as negatives and optimized with a pairwise sigmoid objective. The main training hyperparameters are summarized in Table 6.

Downstream generative-policy training. Both the action encoder and action decoder remain frozen throughout downstream policy training. The Conditional DiT uses the action representations produced by the frozen action encoder as the generation targets for Flow Matching and learns the conditional distribution of future action latents given visual observations and robot proprioceptive states. In addition to the main Flow Matching objective, we employ REPA representation alignment, decoded-action supervision, and trajectory-smoothness supervision. The main training and inference configurations are summarized in Table 7.

Action-decoder training. After action-encoder pretraining, the action encoder remains frozen, and the action decoder is trained using only demonstrations from the target robot. Given an action sequence A, the frozen encoder first produces an action latent $Z_a = E_a(A)$, after which the decoder predicts the 198-dimensional executable action defined in Appendix B. The decoder uses a Smooth-L_1 reconstruction loss with a validity mask, and invalid action dimensions do not contribute to the loss.

To improve the decoder's robustness to deviations in generated latents, we employ noise-augmented decoding in the standardized latent space. For each sample, we first draw

$$\sigma \sim \mathcal{U}(0, \delta), \qquad \epsilon \sim \mathcal{N}(0, \sigma^2 I), \tag{8}$$

and feed the perturbed latent

$$\widetilde{Z}_a = Z_a + \epsilon \tag{9}$$

to the decoder. Twenty-five percent of training samples retain clean latents, and the maximum noise scale δ increases linearly from 0 to 0.05 over the first 5,000 training steps. The decoder is trained with AdamW for 100,000 steps using a batch size of 512, a learning rate of 2×10^{-4}, and a weight decay of 0.05; the first 1,000 steps use linear warmup, followed by a cosine schedule that decays to 2×10^{-6}. Training uses bfloat16 mixed precision and clips the gradient norm to 1.0.

Auxiliary training objectives. The complete downstream-policy training objective is

$$\mathcal{L} = \lambda_{\text{FM}}\mathcal{L}_{\text{FM}} + \lambda_{\text{REPA}}\mathcal{L}_{\text{REPA}} + \lambda_{\text{dec}}\mathcal{L}_{\text{dec}} + \lambda_{\text{smooth}}\mathcal{L}_{\text{smooth}}. \tag{10}$$

Here, $\mathcal{L}_{\text{REPA}}$ aligns intermediate DiT features with the clean action representations produced by the frozen action encoder; $\mathcal{L}_{\text{dec}}$ directly supervises the executable actions decoded from generated latents; and $\mathcal{L}_{\text{smooth}}$ constrains action changes between adjacent time steps. All action-space losses use the corresponding validity masks; for the smoothness loss, a pair is excluded whenever an action dimension is invalid at either of the two adjacent time steps.

F Simulation Experiment Details

Training setup. We conduct both single-task training and multi-task joint training under the `rand-obj` setting of DexJoCo. In the single-task setting, we train a separate policy on the corresponding robot demonstrations for each task for 60,000 steps; in the multi-task setting, we jointly use data from all 11 tasks and train for 120,000 steps. The two settings use the same model architecture and optimization configuration; the detailed training hyperparameters are provided in Appendix E.

No additional validation set is constructed during multi-task training, and the checkpoint at 120,000 steps is used for final evaluation. Multi-task training uses a task-aware shard-local batch sampler that draws samples without replacement within each shard group and favors coverage of distinct tasks in each mini-batch; it does not enforce equal global sampling probabilities across tasks.

Baseline implementations. We compare ARPG with $\pi_{0.5}$ and GR00T N1.5 as reported by DexJoCo. To avoid additional implementation differences introduced by retraining large generalist policies, all baseline results use the values officially reported by DexJoCo under the same `rand-obj` setting (Wang et al., 2026).

- $\pi_{0.5}$**.** Both the single-task and multi-task results are taken from the official DexJoCo report.
- **GR00T N1.5.** The single-task results are taken from the official DexJoCo report.
- **ARPG.** All ARPG models are retrained using our action-format conversion and training pipeline and are evaluated separately in the single-task and multi-task settings.

Evaluation protocol. All methods follow the official DexJoCo environment settings and task-success criteria. For each task, we evaluate with random seeds 0, 1, and 2, running 50 episodes per seed for a total of 150 rollouts. Per-task results report the mean and standard deviation of the success rates across the three random seeds.

For the single-arm tasks, bimanual tasks, and all tasks, we compute unweighted macro-averages as

$$S_{\text{single}} = \frac{1}{6}\sum_{i=1}^{6} S_i, \qquad S_{\text{bimanual}} = \frac{1}{5}\sum_{i=1}^{5} S_i, \qquad S_{\text{all}} = \frac{1}{11}\sum_{i=1}^{11} S_i. \tag{11}$$

Both the policy and control frequencies are 30 Hz. Each policy query generates a 30-step action chunk, and the inference scheduler uses a nominal eight-step execution window for rolling replanning. The evaluation configuration uses strict RTC (PI-VJP) execution with inference-time guidance from the previous action chunk; the prefix-attention schedule is exponential, the maximum guidance weight is 5.0, and the inference-latency safety margin is 40 ms.

Table 8: Stage definitions for the real-world evaluation tasks.

Task	Stage 1	Stage 2	Stage 3
Hammer Handover	Pick up the hammer	Complete the bimanual handover	Place the hammer at the target
Pouring Coffee Beans	Pick up the bean container	Successfully pour out the coffee beans	Completely pour out the coffee beans
Cabinet Retrieval	Open the cabinet door	Retrieve the target object	Close the cabinet door

Note: The three stages of each task must be completed in order.

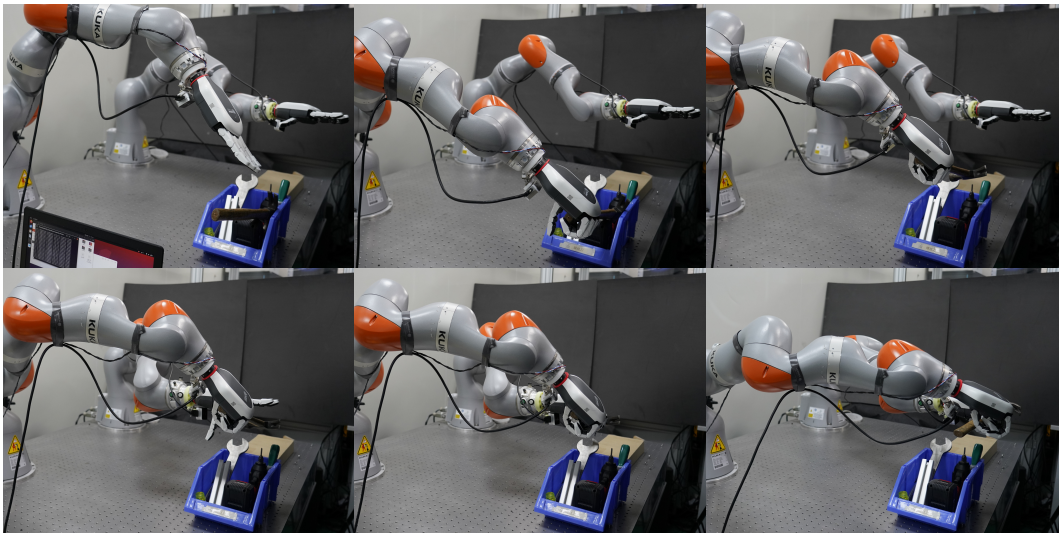


Figure 5: **Hammer Handover: Detailed Procedure.** The right hand picks up the hammer, hands it to the left hand, and the left hand puts it down.

G Real-World Experiment Details

Task setup. We evaluate the policy on three real-world robot tasks that require multiple manipulation stages to be executed sequentially: Hammer Handover, Pouring Coffee Beans, and Cabinet Retrieval. Each task is divided into three ordered stages to evaluate both local manipulation ability and long-horizon task completion.

The execution sequences of the three real-world tasks are shown in Figures 5, 6, and 7, respectively. The six frames in each figure are ordered from left to right and then from top to bottom.

For each rollout, completion of a later stage is counted only if the preceding stage has been completed successfully; final task success requires all three stages to be completed in order. **Evaluation protocol.** ARPG and $\pi_{0.5}$ are trained and evaluated using the same robot demonstrations, initial-state distribution, and task-success criteria.

The success rate of each stage is computed as the proportion of rollouts that satisfy the success criterion for that stage, while the final success rate is the proportion of rollouts that complete all three stages. For method m and stage k, the stage success rate is defined as

$$S_{m,k} = \frac{N_{m,k}^{\text{success}}}{N_m^{\text{rollout}}}. \tag{12}$$

Baseline setup. The real-world experiments primarily compare ARPG with $\pi_{0.5}$. To ensure a consistent comparison, $\pi_{0.5}$ and ARPG use exactly the same robot demonstrations and are evaluated with the same three-view visual inputs, initial-state distribution, and success criteria. For $\pi_{0.5}$, we initialize from the $\pi_{0.5}$ base weights and use mixed LoRA fine-tuning; the training configuration specifies 25,000 steps, a batch size of 32, a 1,000-step warmup, and checkpoint saving every 1,000 steps. The inputs comprise three-view images, task-language instructions, and a normalized 38-

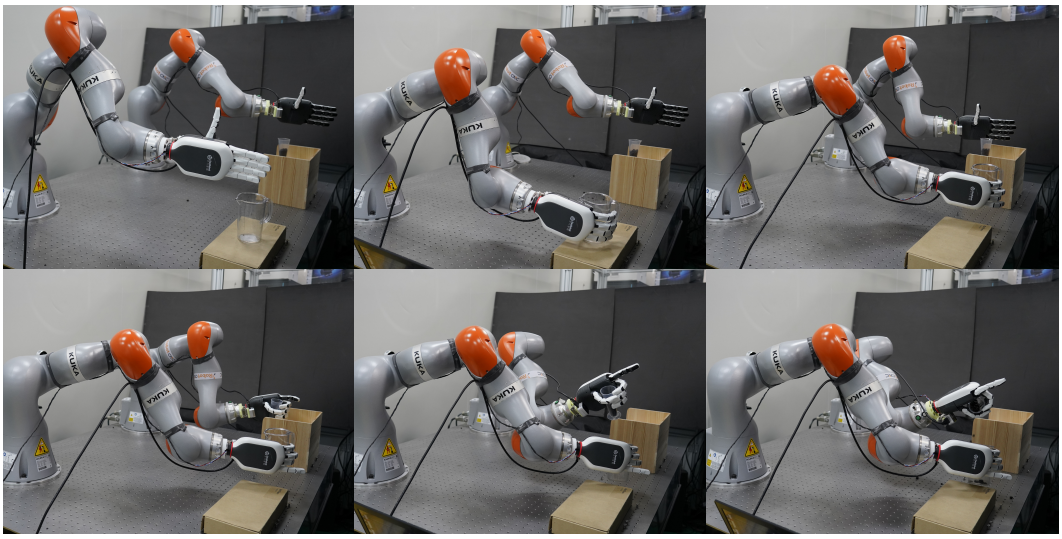


Figure 6: **Pouring Coffee Beans: Detailed Procedure.** Both hands pick up measuring cups, and the coffee beans are poured from the left-hand cup into the right-hand cup.

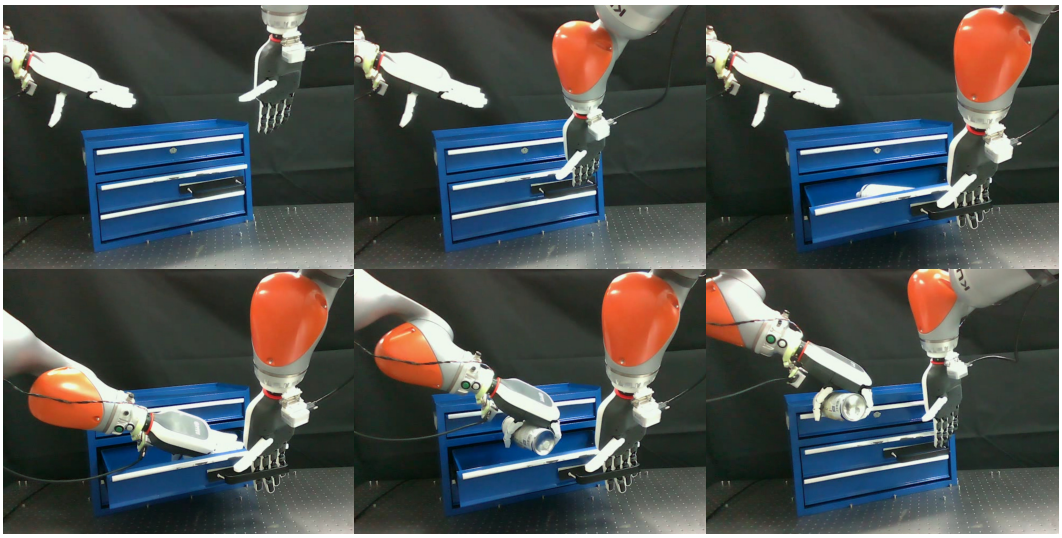


Figure 7: **Cabinet Retrieval: Detailed Procedure.** The right hand opens the cabinet, the left hand retrieves the object, and the right hand closes the cabinet.

dimensional robot state; `discrete_state_input=True` discretizes the state and includes it in the language prompt, with `max_token_len=250`. The three task instructions describe picking up the hammer with the right hand, handing it to the left hand, and putting it down; picking up the measuring cups with both hands and pouring coffee beans from the left cup into the right cup; and opening the cabinet with the right hand, retrieving the object with the left hand, and closing the cabinet with the right hand.

H Evaluation Details on Robotics Simulation

We report complete task-level success rates for all 11 DexJoCo tasks, comprising six single-arm tasks and five bimanual-coordination tasks. All results follow the evaluation protocol described in Appendix F. For experiments with multiple random seeds, we report the mean success rate and standard deviation across three seeds; we additionally report the macro-average success rates over the single-arm tasks, bimanual tasks, and all tasks.

Table 9: Full task-level results on the DexJoCo benchmark.

Task	Single-task training			Multi-task training	
	$\pi_{0.5}$	GR00T N1.5	ARPG	$\pi_{0.5}$	ARPG
Single-arm					
Hammer Nail	84.7 ± 5.0	67.3 ± 4.2	72.0 ± 9.2	74.7 ± 2.3	52.7 ± 5.0
Click Mouse	64.7 ± 8.1	85.3 ± 3.1	73.3 ± 8.1	48.0 ± 6.9	96.0 ± 3.3
Pick Bucket	84.0 ± 7.2	72.0 ± 6.0	80.0 ± 2.0	60.7 ± 2.3	78.0 ± 1.6
Pinch Tongs	24.0 ± 6.9	12.7 ± 2.3	72.7 ± 13.0	44.0 ± 15.9	50.0 ± 1.6
Fold Glasses	72.0 ± 3.5	27.3 ± 2.3	58.0 ± 2.0	28.0 ± 4.0	41.3 ± 4.1
Water Plant	88.7 ± 3.1	72.7 ± 1.2	95.3 ± 2.3	79.3 ± 9.5	94.7 ± 3.4
Bimanual					
Unlock iPad	36.0 ± 14.4	12.7 ± 11.0	30.0 ± 3.5	0.0 ± 0.0	25.3 ± 6.6
Hanoi	23.3 ± 7.0	0.7 ± 1.2	47.3 ± 11.5	4.7 ± 3.1	30.7 ± 3.4
Assembly	2.7 ± 2.3	0.7 ± 1.2	8.7 ± 2.3	4.0 ± 5.3	7.3 ± 0.9
Microwave	60.0 ± 5.3	50.7 ± 4.6	70.7 ± 6.1	34.7 ± 13.6	84.7 ± 5.3
Photograph	54.7 ± 8.1	40.7 ± 7.0	66.0 ± 7.2	7.3 ± 5.0	75.3 ± 6.2
Single-arm Avg.	**69.7**	**56.2**	**75.2 ± 2.5**	**55.8**	**68.8**
Bimanual Avg.	**35.3**	**21.1**	**44.5 ± 4.1**	**10.1**	**44.7**
All Avg.	**54.1 ± 1.8**	**40.2 ± 0.3**	**61.3 ± 2.8**	**35.0 ± 2.8**	**57.8 ± 0.4**

Note: Values are success rates (%); ± denotes the standard deviation across three seeds when available.