CIALLO: LEARNING SHARED INTENT REPRESENTATIONS FOR CROSS-EMBODIMENT MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Robot demonstrations contain reusable manipulation knowledge, but transferring this knowledge across embodiments requires representations that bridge differences in control interfaces. Action reconstruction alone does not ensure that similar behaviors across robots share a representation. To address this gap, we introduce CIALLO, an action codec that aligns actions, visual changes, and observation-grounded language in a shared space of short-horizon behavioral intent. An embodiment-conditioned decoder maps this intent back to robot-specific actions, allowing a shared representation to support different control interfaces. After pretraining on eight embodiments, CIALLO adapts to a new robot from one recording of at most 10 seconds by updating only its embodiment embeddings. The calibrated codec then remains frozen while a policy learns to predict intent from observations and instructions. We instantiate this policy with Qwen3.5-4B to form Qwen-CIALLO, whose backbone requires no large-scale robot-action pretraining before policy finetuning. With 20 demonstrations per task, Qwen-CIALLO achieves 75.0% average success across real SO-101 and single-arm Piper robots, compared with 68.3% for fully finetuned π_0. In simulation, we additionally finetune the action decoder, achieving 97.3% success on LIBERO and 52.7% on RoboTwin 2.0. The LIBERO policy also achieves 80.8% on LIBERO-Plus without further finetuning. Together, these results support a shared action codec as a transferable control interface that separates embodiment calibration from task learning.

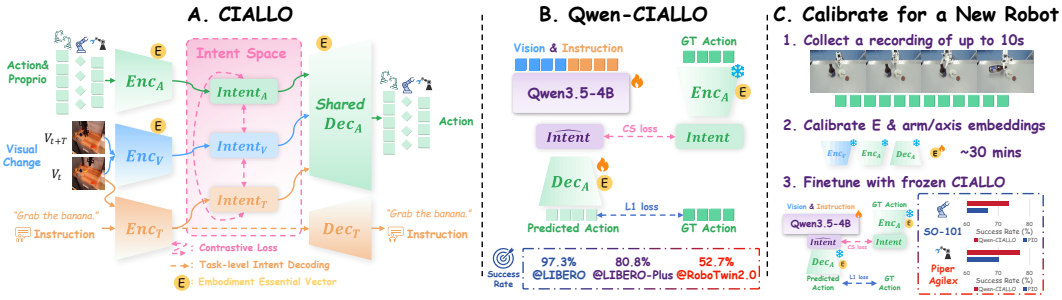

Figure 1: **Overview of CIALLO. (A)** During pretraining, three encoders map an action chunk with proprioception, a pair of current and future frames, and an instruction grounded in the current frame into a shared intent space. We align their outputs with contrastive losses and train a shared decoder to reconstruct the demonstrated actions from each intent, conditioned on the robot's embodiment parameters. **(B)** To learn a policy, Qwen-CIALLO predicts intent from observations and instructions, using the frozen action encoder as a teacher and the demonstrated actions to supervise the decoder output. CS denotes cosine similarity loss. **(C)** To transfer the codec to a new robot, we first calibrate its essential vector and arm and axis embeddings while keeping the shared weights fixed, then train the policy with the calibrated codec frozen.

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.

1 INTRODUCTION

A grasp demonstrated by one robot can teach another what to reach for and how an object should move, even when their joint coordinates and control rates differ. Cross-embodiment learning seeks to reuse experience from existing robots to provide control knowledge for a new body (Open X-Embodiment Collaboration et al., 2024; Ghosh et al., 2024; Liu et al., 2025). Pretrained vision-language models supply broad semantic knowledge (Kim et al., 2025b; Black et al., 2025), but grounding that knowledge in an unfamiliar action interface still requires substantial amounts of robot demonstration data. A reusable action codec could make these demonstrations more useful for task learning by reducing the need to adapt the underlying control representation.

Recorded actions entangle behavior with the coordinates and control conventions used to execute it. Similar motions can be expressed as end effector displacements or joint angles, with different dimensions, numbers of arms, and temporal resolutions. Shared action layouts and embodiment-specific modules allow policies to consume heterogeneous data (Ghosh et al., 2024; Liu et al., 2025; Wang et al., 2024). Yet action reconstruction alone, even with embodiment-conditioned encoding and decoding, does not require similar behaviors across robots to have similar representations. Visual changes and grounded language connect different control sequences that produce similar visual effects or serve compatible task goals. We therefore align action representations through these shared cues to learn a codec that transfers across embodiments with only brief calibration.

We introduce CIALLO (**C**ross-embodiment **I**ntent **All** In **O**ne Space), an action codec for transferring control knowledge across robots. CIALLO separates a shared representation of short-horizon behavior, *intent*, from the body and control schema used to realize it. Building on latent action learning and action–vision alignment (Ye et al., 2025; Bu et al., 2025; Zhang et al., 2026; Chen et al., 2026a), CIALLO aligns actions, visual changes, and observation-grounded language in one space. The visual and text branches anchor actions to their effects and task context, encouraging nearby representations for compatible behaviors across embodiments. A shared decoder reconstructs actions from all three representations, preserving the control information needed to execute them (Figure 1A). An essential vector E, arm and axis embeddings, and execution times supply the embodiment-specific context. The same codec weights therefore support different action dimensions, control rates, and single- or dual-arm configurations.

The aligned action codec then provides the interface for policy learning. From one recording of at most 10 seconds, we fit only the new robot's essential vector and arm and axis embeddings, keeping the shared network fixed. We freeze the calibrated codec and train a policy to predict intent from observations and instructions. The action encoder and decoder supervise this prediction (Figure 1B–C). Calibration establishes how the codec realizes behavior on the new body, while task demonstrations teach the policy which intent to produce. We pair the codec with Qwen3.5-4B (Qwen Team, 2026) through StarVLA (StarVLA Community, 2026), forming Qwen-CIALLO without large-scale robot-action pretraining of the Qwen backbone.

We pretrain CIALLO on eight embodiments and evaluate reuse on real SO-101 and single-arm Piper configurations absent from pretraining (Piper pretraining uses a simulated dual-arm configuration). Calibration takes approximately 30 minutes per robot, after which the codec remains frozen. With 20 demonstrations per task, Qwen-CIALLO achieves 75.0% average success, compared with 68.3% for fully finetuned π_0. In simulation, where the action decoder is also finetuned, Qwen-CIALLO reaches 97.3% on LIBERO after 10k policy steps and 80.8% on LIBERO-Plus without further training. One policy trained jointly on RoboTwin 2.0 achieves 52.7% after 40k steps. These evaluations test both adaptation to new robots and policy learning across tasks with the shared action codec.

2 RELATED WORK

2.1 VISION-LANGUAGE-ACTION MODELS AND WORLD MODELS

Language-conditioned robot policies map observations and instructions to actions (Brohan et al., 2023; Ghosh et al., 2024). Vision-language-action (VLA) models often build on pretrained vision-language backbones (Kim et al., 2025b; Black et al., 2025; Physical Intelligence et al., 2025; NVIDIA et al., 2025). World models learn dynamics for behavior learning (Hafner et al., 2020) or controllable video generation (Bruce et al., 2024; Liao et al., 2026; Guo et al., 2026). Video-action

models jointly learn prediction and control (Zhou et al., 2026), while JEPA-VLA transfers frozen video-predictive features into a policy (Miao et al., 2026). Fast-WAM studies the value of predictive representation learning without test-time future imagination (Yuan et al., 2026b). CIALLO focuses on a shared intent interface with explicit embodiment-conditioned encoding and decoding.

2.2 Cross-Embodiment Policy Learning

Pooling robot demonstrations (Open X-Embodiment Collaboration et al., 2024) requires handling different action dimensions, semantics, and control rates. Approaches include padded unified action spaces (Liu et al., 2025), embodiment- or action-space-specific modules (Ghosh et al., 2024; NVIDIA et al., 2025; Doshi et al., 2025), and embodiment-specific stems feeding a shared trunk (Wang et al., 2024). CIALLO conditions shared encoding and decoding networks on an essential vector and control schema. Adaptation fits the new robot's essential vector and arm and axis embeddings, then freezes the calibrated codec during policy learning.

2.3 Aligning Actions with Visual Change

Inverse dynamics models recover actions from observation pairs, enabling video labeling from limited action supervision (Baker et al., 2022). Latent-action methods learn a bottleneck between frames without action labels (Schmidt & Jiang, 2024; Bruce et al., 2024; Ye et al., 2025). UniVLA learns task-centered latent actions for transfer across embodiments (Bu et al., 2025). CLAP uses contrastive action–vision alignment (Zhang et al., 2026), while UniT uses cross-reconstruction to unify visual and kinematic representations (Chen et al., 2026a). CIALLO aligns actions, visual change, and observation-grounded language, with a shared action decoder supervising all three paths under the target embodiment context.

2.4 Action Representations

Diffusion and flow-matching heads model action distributions through iterative sampling (Chi et al., 2023; Liu et al., 2025; Black et al., 2025). Efficient alternatives include direct regression (Kim et al., 2025a). Compact policies such as TinyVLA reduce model cost while retaining diffusion decoding (Wen et al., 2025). Discrete representations include per-dimension bins (Brohan et al., 2023), frequency-domain action tokens (Pertsch et al., 2025), and hierarchical vector quantization (Lee et al., 2024). Other approaches use spatially structured grids or heatmaps (James et al., 2022; Shridhar et al., 2023; Goyal et al., 2023; Yang et al., 2026). CIALLO uses ordered numerical bins as a shared decoding basis, with transposed convolutions producing logits and a softmax expectation yielding continuous actions in one forward pass.

3 Method

3.1 Shared Intent and Embodiment Parameterization

A training example contains an instruction, RGB observations, proprioception, and an action chunk $a_{t:t+T-1} \in \mathbb{R}^{T \times A}$ together with its execution times. The chunk length, control rate, and semantics of the A action axes vary across datasets. CIALLO therefore represents a chunk as a masked set of timestamped control scalars rather than assigning semantic meaning to the position of a value in one global action vector. It compresses this set into an *intent* $z \in \mathbb{R}^{K \times D}$, with $K = 2$ tokens of width $D = 512$ in our experiments. The compact bottleneck provides the same interface for every embodiment, while the encoders and decoder receive the body and control context separately. The text branch also maintains a four-token *task-level intent* that summarizes the instruction. Combining it with the current observation produces a chunk-level intent.

CIALLO makes the complementary, body-specific part of the factorization explicit. Each robot is assigned an *essential vector* $E \in \mathbb{R}^n$, with $n = 128$ in practice. Axis embeddings identify whether a scalar controls an end effector component, a joint coordinate, or a gripper command. Arm embeddings further specify the manipulator to which the scalar belongs. Together, these codes provide the information needed to translate between a robot's native signals and the shared intent space. They also provide the parameters used for adaptation. For an unseen robot, we estimate a new

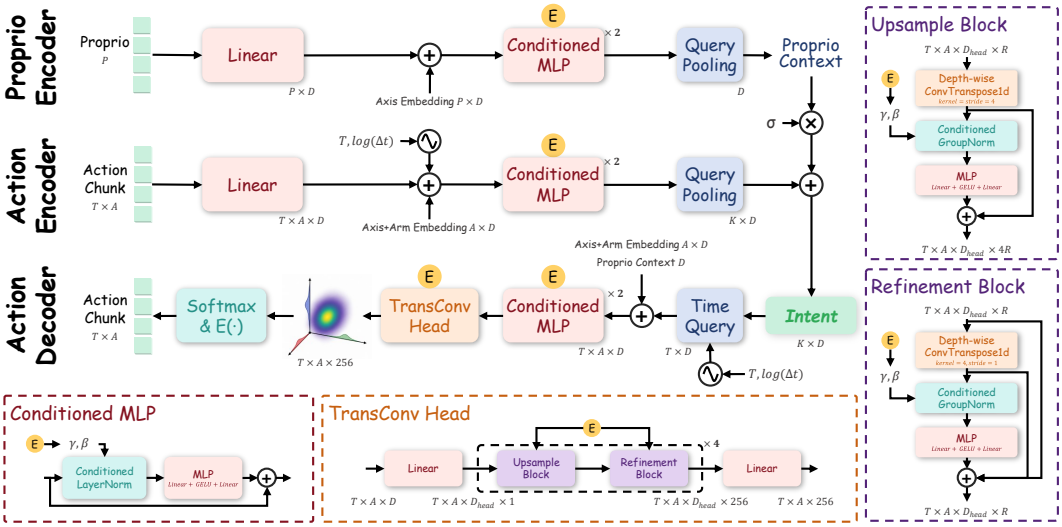

Figure 2: **The action codec.** The action encoder (middle) uses the essential vector E to map control signals into shared intent, incorporating context from the proprioceptive encoder (top). The decoder (bottom) combines this intent with the target action schema and current proprioception, remodulates the result for the target embodiment, and uses the TransConv Head to predict each control scalar on a shared basis of 256 ordered bins. The decoded action is the softmax expectation over the bin centers.

E together with its arm and axis embeddings while keeping the shared network fixed, as illustrated in Figure 1C.

The three encoders produce representations of the same demonstrated chunk,

$$z_a = q_a(a, p, E), \qquad z_v = q_v(o_t, o_{t+T-1}, E), \qquad z_t = q_t(l, o_t), \tag{1}$$

where p, o, and l denote proprioception, RGB observations, and the instruction. Execution times, axis and arm codes, and validity masks are omitted here for brevity. A shared action decoder reconstructs the chunk from each representation,

$$\hat{a}^m = p_a(z_m, p, E), \qquad m \in \{a, v, t\}. \tag{2}$$

Contrastive alignment encourages agreement across the three representations, while reconstruction requires each to retain control information. The text path uses the current observation to ground the instruction and receives neither E nor proprioception. The four-token task-level intent is internal to this path and is distinct from its grounded output z_t.

We inject E through conditional normalization (Huang & Belongie, 2017; Perez et al., 2018; Peebles & Xie, 2023). A learned projection produces feature-wise scales and shifts (γ, β), which transform a normalized activation as

$$\mathrm{CondNorm}(x; E) = \mathrm{Norm}(x) \odot (1 + \gamma(E)) + \beta(E). \tag{3}$$

The same operation appears on both sides of the intent bottleneck. In the action and visual encoders, it *demodulates* embodiment-specific signals into a shared representation. In the action decoder, it *remodulates* that representation into the coordinates of the target robot. This symmetry provides a common mechanism for the forward and inverse mappings in Eq. 1 and Eq. 2. Architectural and initialization details are given in Appendix A.1.

3.2 The Action Codec

The action codec implements both q_a and p_a with parameters shared across embodiments, as illustrated in Figure 2. The encoder represents each valid action scalar by its value, execution time, control axis type, and arm identity. Embodiment-conditioned blocks map these tokens into the common feature space, after which the Query Pooling module compresses the entire chunk into K intent

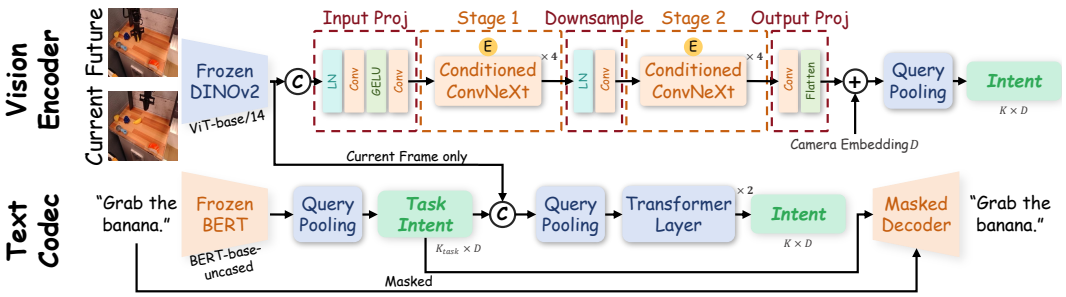

Figure 3: **Vision and text paths.** The vision encoder compares current and future observations with a shared frozen DINOv2 backbone and an E-conditioned local interaction encoder, then pools valid tokens from multiple cameras into chunk-level intent. The text codec compresses the instruction into a task-level intent, grounds it using features reused from the current frame to form the same $K \times D$ intent interface, and reconstructs masked instruction tokens from the task-level intent.

tokens. A separate proprioceptive encoder summarizes the current body state and injects it through a learned gate. The resulting intent is grounded primarily in the executed motion, while proprioception supplies the state and coordinate context needed to interpret that motion.

The decoder reverses this mapping for a requested target interface. The Time Query module reads the intent at the target execution times. It combines the resulting features with target axis and arm codes and current proprioceptive context, then remodulates them under the target essential vector. The temporal schedule and action schema are inputs rather than fixed network properties. One decoder therefore supports different horizons, control rates, action dimensions, and single- or dual-arm layouts. Padding is used only for batching and is removed by masks. It does not impose a shared semantic layout on otherwise unrelated control axes.

For each output scalar, the TransConv Head predicts logits over 256 uniformly spaced bins in $[-1, 1]$, sharing local convolutional kernels along the value axis. A softmax expectation converts these logits into a continuous action, following integral pose regression (Sun et al., 2018) and DSNT (Nibali et al., 2018). The head is trained with masked ℓ_1 regression and decodes in one forward pass. Appendix A.2 gives the architecture and expectation readout.

3.3 Vision Encoder

The vision path recovers intent over a short horizon from the change between the current and future observations, as illustrated in Figure 3. Both frames from every camera are processed by the same frozen DINOv2 backbone (Oquab et al., 2024). Freezing this backbone preserves its semantic representation while the interaction encoder learns from heterogeneous robot data. These backbone features remain fixed during embodiment calibration, while the downstream visual representation can change through its dependence on E. The paired patch features are concatenated along the channel dimension and processed by a two-stage ConvNeXt interaction encoder conditioned on the essential vector E (Liu et al., 2022). This local hierarchy is well suited to the small visual changes produced over a short action chunk. It avoids patch-level global attention and a full cost volume while retaining a sufficient receptive field to relate nearby changes.

The resulting tokens from each available view receive a camera embedding. A validity mask removes absent cameras and padded tokens, so the same encoder handles variable camera counts. Query Pooling summarizes all valid views into the $K \times D$ intent interface shared with the action path. Its learned queries select task-relevant changes across cameras without forming all patch pairs. The DINOv2 features for the current frame are also retained for the text path. Input resolution, stage dimensions, mask definitions, and complexity are detailed in Appendix A.3.

3.4 Text Codec

An instruction usually describes an entire task, whereas chunk-level intent specifies what should happen over the next short horizon. The text path separates task understanding from temporal grounding

to bridge these two scopes, as illustrated in Figure 3. A frozen BERT encoder (Devlin et al., 2019) represents the instruction, and Query Pooling compresses its valid token features into a task-level intent. These task-level tokens are fused with visual tokens extracted from the current frame by the frozen vision backbone. A second Query Pooling module extracts K grounded tokens, and two Transformer Layers refine them into a chunk-level intent in $\mathbb{R}^{K\times D}$. This design uses the current observation to identify instruction content relevant to the present chunk, with little additional visual computation.

The task-level intent bottleneck is trained through masked text reconstruction. The Masked Decoder receives an instruction with selected tokens hidden and attends to the task-level intent as its latent memory. The loss is evaluated only at the masked positions. Reconstructing the missing content encourages the task-level intent bottleneck to preserve distinctions expressed by the instruction. The text path intentionally differs from the action and vision paths in its conditioning inputs. It is neither conditioned on E nor given proprioception because language specifies the task independently of the body that performs it. The current observation supplies visual grounding, while alignment across modalities connects the resulting intent to an embodiment-relevant realization. Architectural details and the reconstruction objective are given in Appendix A.4.

3.5 Supervision

The supervision structure is summarized in Figure 1A. The action, vision, and text paths each pass their intent through the same action decoder and reconstruct the demonstrated action chunk with masked ℓ_1 loss. Sharing the decoder makes alignment functional rather than purely geometric. A visual or textual intent must expose the same information that the action codec needs to recover the embodiment-specific controls under the corresponding target context.

The task-level intent is additionally supervised by masked text reconstruction, as described in Section 3.4. Finally, symmetric InfoNCE losses (Radford et al., 2021) align the three pairs of representations using negatives from the global batch. The complete training objective is

$$\mathcal{L} = \sum_{m\in\{a,v,t\}} \mathcal{L}^m_{\text{act}} + \mathcal{L}_{\text{text}} + 0.1\mathcal{L}_{a\leftrightarrow v} + 0.05\mathcal{L}_{a\leftrightarrow t} + 0.05\mathcal{L}_{v\leftrightarrow t}. \tag{4}$$

Here, a, v, and t index the three paths, $\mathcal{L}^m_{\text{act}}$ and $\mathcal{L}_{\text{text}}$ are the masked action and text reconstruction losses, and $\mathcal{L}_{p\leftrightarrow q}$ is symmetric InfoNCE. Unit reconstruction weights keep reconstruction primary, while the smaller alignment weights encourage a shared intent space.

Inspired by FlashAttention and Inf-CL (Dao et al., 2022; Cheng et al., 2025), we evaluate exact InfoNCE over the global batch by streaming log-sum-exp blocks and recomputing them during the backward pass. This avoids the full similarity matrix, as detailed in Appendix A.5.

3.6 Policy Learning and Embodiment Calibration

With the action codec pretrained, we can train a policy to predict intent from observations and instructions, as illustrated in Figure 1B. Qwen-CIALLO uses a Qwen3.5-4B backbone followed by an intent projector to produce this $K \times D$ representation. The frozen action encoder provides teacher tokens from the demonstrated actions, which supervise the predicted intent through a cosine alignment loss. We also decode the predicted tokens and apply an action regression loss against the demonstrations, training the policy to produce intent that supports accurate control. For the simulation benchmarks, we jointly finetune the backbone, projector, and action decoder. At inference, the backbone predicts intent directly, so the teacher encoder is no longer needed.

When transferring to a new robot, we first fit the embodiment parameters that allow it to use the pretrained codec. As shown in Figure 1C, we calibrate its essential vector and arm and axis embeddings from a recording of at most 10 seconds while keeping the shared network weights fixed. This optimization takes approximately 30 minutes on one GPU. We then freeze the calibrated codec and train the backbone and projector on demonstrations of the new robot's tasks. Keeping the codec fixed during this second stage lets us evaluate whether it can support policy learning after calibration alone. The real-robot evaluation reports the calibration recording and the policy demonstrations separately to distinguish the data used for these two stages.

4 EXPERIMENTS

We first evaluate whether the three intent paths align and retain information needed by the shared action decoder. We then evaluate policies trained through this interface on LIBERO and RoboTwin 2.0, with LIBERO-Plus testing the LIBERO policy under distribution shifts. Finally, we assess adaptation to real robots by calibrating the codec and keeping it frozen during policy training.

4.1 CIALLO PRETRAINING

We pretrain CIALLO on 125,434 episodes comprising 1,354 Panda episodes from the four LIBERO suites (Liu et al., 2023), 45,541 WidowX episodes from Bridge and 69,339 Google Robot episodes from Fractal as used by SimplerEnv (Li et al., 2025), and 9,200 episodes from five RoboTwin 2.0 embodiments (Chen et al., 2026b). The RoboTwin portion covers Aloha AgileX, Franka Dual, ARX X5, Piper, and UR5. Every source uses only its main camera. We balance sampling across the eight embodiments. The maximum and minimum sample counts differ by 0.30% of the mean count.

Training runs for 16,000 optimizer steps with a global batch size of 1,024, which processes 16.38 million examples. We use AdamW with weight decay 0.01, a peak learning rate of 3×10^{-4}, 400 warmup steps, cosine decay, and BF16 precision. The model has 262.05M parameters, of which 65.99M are trainable because DINOv2 and BERT remain frozen. Training takes 15.7 hours on 8 NVIDIA H20 GPUs, including validation and checkpoint writes.

On 2,048 held-out samples, bidirectional top-1 retrieval reaches 96.68% for A↔V, 91.94% for A↔T, and 96.88% for V↔T. All three representations support the shared decoder, with normalized masked action mean absolute errors (MAEs) of 0.00761, 0.01853, and 0.02429 for the action, visual, and grounded language paths, respectively. Replacing the decoder's essential vector raises these errors to 0.19–0.21, confirming its use of embodiment conditioning. Appendix B details the evaluation protocol, per-embodiment results, ablations on embodiment count, intent capacity, and contrastive alignment, and direct closed-loop evaluation without policy finetuning.

4.2 QWEN-CIALLO

4.2.1 IMPLEMENTATION DETAILS

We use Qwen3.5-4B with the intent prediction and action supervision described in Section 3.6. For LIBERO and RoboTwin 2.0, the action encoder remains frozen while the backbone, intent projector, and action decoder are finetuned. Appendix C.1 gives the training hyperparameters.

4.2.2 LIBERO

Evaluation protocol. We evaluate Qwen-CIALLO on the four standard LIBERO task suites, namely LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long. Together, these suites assess spatial reasoning, object-centric generalization, goal understanding, and long-horizon manipulation (Liu et al., 2023). Baseline results in Table 1 come from sources that use different statistical protocols. Results marked with † are averaged over three random seeds, with 500 evaluation trials per suite and 50 trials for each of the 10 tasks. Diffusion Policy, Octo, OpenVLA, and OpenVLA-OFT results are taken from the OpenVLA-OFT evaluation (Kim et al., 2025a). The GR00T-N1 and π_0 results are reported by Hume (Song et al., 2025), and the StarVLA results come from its official evaluations (StarVLA Community, 2026). CLAP-RF results are taken from its reported LIBERO evaluation (Zhang et al., 2026), using joint training across all four suites for 30k steps. Both the StarVLA variants and Qwen-CIALLO use single-seed evaluations.

Results. Qwen-CIALLO attains 97.3% average success across the four LIBERO suites, the highest average in Table 1. It exceeds CLAP-RF by 0.1%, OpenVLA-OFT by 0.2%, and all reported StarVLA variants. Both Qwen-CIALLO and CLAP-RF use joint training across all four suites, with 10k and 30k policy finetuning steps, respectively. StarVLA also uses 30k steps. This comparison concerns downstream optimization steps and excludes CIALLO pretraining. It is consistent with the motivation that a pretrained action codec can reduce the burden of learning robot control during policy finetuning, although the comparison does not isolate that mechanism.

Table 1: LIBERO success rates in %. Results marked with † are averages over three seeds. CLAP-RF and Qwen-CIALLO are trained jointly on all four task suites for 30k and 10k optimization steps, respectively.

Method	Steps	Spatial	Object	Goal	Long	Avg.
Diffusion Policy† (Chi et al., 2023)	100–250k	78.3	92.5	68.3	50.5	72.4
Octo† (Ghosh et al., 2024)	100–250k	78.9	85.7	84.6	51.1	75.1
OpenVLA† (Kim et al., 2025b)	50–150k	84.7	88.4	79.2	53.7	76.5
OpenVLA-OFT† (Kim et al., 2025a)	50–150k	97.6	98.4	97.9	**94.5**	97.1
GR00T-N1† (NVIDIA et al., 2025)	–	94.4	97.6	93.0	90.6	93.9
π_0† (Black et al., 2025)	–	96.8	98.8	95.8	85.2	94.2
StarVLA-PI (StarVLA Community, 2026)	30k	**98.8**	**99.6**	95.8	88.4	95.7
StarVLA-OFT	30k	97.8	98.6	96.2	93.8	96.6
StarVLA-GR00T	30k	97.8	98.8	97.4	92.0	96.5
CLAP-RF (Zhang et al., 2026)	30k	<u>98.6</u>	<u>99.2</u>	<u>98.0</u>	93.0	<u>97.2</u>
Qwen-CIALLO	**10k**	98.4	98.4	**98.2**	<u>94.0</u>	**97.3**

Table 2: LIBERO-Plus success rates in %. Total reports overall success across all evaluation tasks for every method. Qwen-CIALLO uses its LIBERO checkpoint without further finetuning.

Method	Camera	Robot	Language	Light	Background	Noise	Layout	Total
π_0	15.8	6.6	61.0	79.6	78.5	<u>79.4</u>	70.4	53.6
UniVLA (Bu et al., 2025)	1.8	46.2	69.6	69.0	81.0	21.2	31.9	42.9
OpenVLA	0.8	3.5	23.0	8.1	34.8	15.2	28.5	15.6
OpenVLA-OFT	<u>56.4</u>	31.9	79.5	88.7	<u>93.3</u>	75.8	<u>74.2</u>	69.6
Fast-WAM (Yuan et al., 2026b)	16.3	44.1	66.8	79.8	52.6	38.5	61.4	49.9
DreamWAM (Yuan et al., 2026a)	53.8	<u>63.6</u>	**94.8**	**96.7**	71.6	67.2	**80.7**	<u>74.6</u>
Qwen-CIALLO	**57.5**	**66.7**	<u>90.5</u>	<u>95.1</u>	**94.6**	**88.9**	**80.7**	**80.8**

4.2.3 LIBERO-PLUS

Evaluation protocol. LIBERO-Plus extends LIBERO with controlled perturbations to camera viewpoints, robot initial states, language instructions, light conditions, background textures, sensor noise, and object layouts (Fei et al., 2026). We evaluate the LIBERO checkpoint without further finetuning. Following the standard protocol, we run one trial per task across the four task suites for each perturbation category. Baseline results in Table 2 are taken from LIBERO-Plus, except the Fast-WAM and DreamWAM results, which are taken from DreamWAM (Yuan et al., 2026a).

Results. The same LIBERO checkpoint achieves 80.8% overall success, exceeding DreamWAM by 6.2% (Table 2). Qwen-CIALLO has the highest camera, robot, background, and noise scores among the listed methods and ties on layout. This performance is obtained without training specifically for these perturbations. We hypothesize that the short policy finetuning schedule helps preserve the backbone's pretrained capabilities and that intent pretraining across embodiments and environments contributes to scene generalization. The current evaluation measures overall robustness and does not separate these two possible contributions.

4.2.4 ROBOTWIN 2.0

Evaluation protocol. We evaluate Qwen-CIALLO on RoboTwin 2.0 using the official clean-to-clean protocol (Chen et al., 2026b). The benchmark contains 50 tasks, each with 50 demonstrations for training. We train a single policy jointly across all tasks. In Table 3, Single Task methods use one checkpoint per task, while Co-Train methods use a single checkpoint jointly trained across all 50 tasks. All baseline results are taken from the official RoboTwin 2.0 leaderboard.

Results. After 40k policy finetuning steps, one Qwen-CIALLO checkpoint achieves 52.7% overall success across all 50 tasks, the highest score in Table 3. It exceeds the jointly trained StarVLA baseline by 6.2% and the strongest listed task-specific baseline, π_0, by 6.3%. The gains over RDT, ACT, and Diffusion Policy are 18.2%, 23.0%, and 24.7%, respectively. These results show that

Table 3: RoboTwin 2.0 success rates (%) under the clean-to-clean protocol. Single Task uses one checkpoint per task, and Co-Train uses one checkpoint jointly trained on all 50 tasks.

Method	Protocol	Avg.	Method	Protocol	Avg.
Diffusion Policy	Single Task	28.0	π_0	Single Task	46.4
ACT (Zhao et al., 2023)	Single Task	29.7	StarVLA	Co-Train	<u>46.5</u>
RDT (Liu et al., 2025)	Single Task	34.5	**Qwen-CIALLO**	Co-Train	**52.7**

Table 4: Real robot task success rates (%) on SO-101 and single-arm Piper, with 20 training demonstrations and 10 evaluation trials per task on each robot. Qwen-CIALLO uses a frozen codec after calibration from one recording per robot, while π_0 is fully finetuned.

Method	SO-101			Piper			Avg.
	Water bottle	Pour beer	Wooden fish	Water bottle	Pour beer	Wooden fish	
Qwen-CIALLO	80	70	70	90	100	40	75.0
π_0	90	80	30	100	100	10	68.3

a single policy trained through the shared intent interface can handle diverse manipulation tasks, outperforming both the jointly trained StarVLA baseline and the listed task-specific policies.

4.3 Adaptation to New Embodiments

Evaluation protocol. We evaluate adaptation on two real robot platforms, SO-101 and a single-arm Piper, using three manipulation tasks with 20 demonstrations per task on each robot. Appendix E provides the task instructions and example observations. Following Figure 1C, we calibrate the essential vector and axis and arm embeddings using one recording of at most 10 seconds from each robot. Calibration takes approximately 30 minutes per robot. We then pair the calibrated codec with Qwen3.5-4B and freeze the entire codec during policy training, updating only the backbone and intent projector. This setting tests whether the calibrated action interface can support downstream control without further changes to the codec.

We compare against full finetuning of the pretrained π_0 generalist (Black et al., 2025) on the same demonstrations. Our Qwen3.5-4B backbone starts from a pretrained vision-language model without large-scale robot-action pretraining. Reusable control knowledge is supplied by the separately pretrained CIALLO codec. Appendix C.2 gives the calibration and policy training settings. Each method is evaluated over 10 trials per task on each robot, with success rates reported in Table 4.

Results. With 20 demonstrations per task, Qwen-CIALLO achieves 75.0% average success, compared with 68.3% for π_0 (Table 4). It performs comparably on water-bottle placement and beer pouring, with gaps of at most 10%. On the wooden-fish task, it achieves higher success, reaching 70% versus 30% on SO-101 and 40% versus 10% on Piper. This task requires grasping a thin screwdriver before striking the wooden fish, demanding greater grasp precision. These results support rapid embodiment adaptation without large-scale robot-action pretraining of the policy backbone.

5 Conclusion

We presented CIALLO, an action codec that separates shared behavioral intent from embodiment-specific control. During pretraining, visual changes and grounded language connect actions through shared effects and task context. The aligned action codec then supports policy learning across embodiments. On each of two real robots, one short recording calibrates the codec, which remains frozen during policy learning from 20 demonstrations per task. Together with the simulation results, these findings support separating embodiment calibration from task learning. Future work will align human egocentric videos with humanoid actions in the shared intent space and scale CIALLO pretraining to larger, more diverse datasets. We also aim to use the codec to pretrain generalist vision-language-action policies from vision-language models at scale, extending transfer across embodiments.

AI USE STATEMENT

In this work, we used generative AI tools for language editing, manuscript organization, LaTeX formatting, and summarizing experiment records provided by the authors for presentation in the manuscript. We did not use generative AI tools to design the methodology, write or modify experimental code, run experiments, or generate experimental data or results. Other disclosure tasks are not applicable to this work. We reviewed all work produced with AI assistance. The authors checked text produced with AI assistance against the manuscript plan and original experiment records, verified citation metadata against primary sources, and compiled and visually inspected the affected pages. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies transferable robot control using existing datasets, simulation benchmarks, and real robot manipulation tasks. Transferring policies across embodiments can introduce physical risks when the target robot or environment differs from the evaluation setting. The reported success rates do not establish safety for unsupervised deployment. Practical use requires application-specific safety validation and appropriate human oversight. Future extensions to human egocentric video should also address consent, privacy, and responsible data use.

REPRODUCIBILITY STATEMENT

Section 4.1 describes the pretraining data composition and training configuration. Appendix A specifies the model architecture and objectives, and Appendix C documents policy training and embodiment calibration. Section 4 reports the evaluation protocols, while Appendix E provides real robot task instructions and example observations. Appendix D specifies the hardware, numerical precision, input dimensions, and timing protocol used for benchmarking.

REFERENCES

Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video PreTraining (VPT): Learning to act by watching unlabeled online videos. In *Advances in Neural Information Processing Systems*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9c7008aff45b5d8f0973b23e1a22ada0-Abstract-Conference.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 *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.010. URL https://www.roboticsproceedings.org/rss21/p010.html.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan H Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics transformer for real-world control at scale. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.025. URL https://www.roboticsproceedings.org/rss19/p025.html.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Maria Elisabeth Bechtle, Feryal Behbahani, Stephanie C.Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon

Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando De Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *Proceedings of International Conference on Machine Learning*, 2024. URL https://proceedings.mlr.press/v235/bruce24a.html.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Learning to act anywhere with task-centric latent actions. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.014. URL https://www.roboticsproceedings.org/rss21/p014.html.

Boyu Chen, Yi Chen, Lu Qiu, Jerry Bai, Yuying Ge, and Yixiao Ge. UniT: Toward a unified physical language for human-to-humanoid policy learning and world modeling. *CoRR*, 2026a. URL https://arxiv.org/abs/2604.19734.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. In *Proceedings of International Conference on Machine Learning*, 2026b. URL https://icml.cc/virtual/2026/poster/62192.

Zesen Cheng, Hang Zhang, Kehan Li, Sicong Leng, Zhiqiang Hu, Fei Wu, Deli Zhao, Xin Li, and Lidong Bing. Breaking the memory barrier of contrastive loss via tile-based strategy. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Cheng_Breaking_the_Memory_Barrier_of_Contrastive_Loss_via_Tile-Based_Strategy_CVPR_2025_paper.html.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://www.roboticsproceedings.org/rss19/p026.html.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract.html.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Ria Doshi, Homer Walke, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. In *Proceedings of Conference on Robot Learning*, 2025. URL https://proceedings.mlr.press/v270/doshi25a.html.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 38574–38583, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Fei_LIBERO-Plus_A_Progressive_Robustness_Benchmark_for_Visual-Language-Action_Models_CVPR_2026_paper.html.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, 2024. doi:

10.15607/RSS.2024.XX.090. URL https://www.roboticsproceedings.org/rss20/p090.html.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. RVT: Robotic view transformer for 3D object manipulation. In *Proceedings of Conference on Robot Learning*, pp. 694–710, 2023. URL https://proceedings.mlr.press/v229/goyal23a.html.

Kerui Gu, Linlin Yang, Michael Bi Mi, and Angela Yao. Bias-compensated integral regression for human pose estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9): 10687–10702, 2023. doi: 10.1109/TPAMI.2023.3264742. URL https://ieeexplore.ieee.org/document/10093110.

Yanjiang Guo, Lucy Shi, Jianyu Chen, and Chelsea Finn. Ctrl-World: A controllable generative world model for robot manipulation. In *Proceedings of International Conference on Learning Representations*, pp. 6121–6138, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/0ae94013da7cd459402fd77874e09ee3-Abstract-Conference.html.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *Proceedings of International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1912.01603.

Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of IEEE/CVF International Conference on Computer Vision*, 2017. URL https://openaccess.thecvf.com/content_iccv_2017/html/Huang_Arbitrary_Style_Transfer_ICCV_2017_paper.html.

Stephen James, Kentaro Wada, Tristan Laidlow, and Andrew J. Davison. Coarse-to-Fine Q-attention: Efficient learning for visual robotic manipulation via discretisation. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13739–13748, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/James_Coarse-To-Fine_Q-Attention_Efficient_Learning_for_Visual_Robotic_Manipulation_via_Discretisation_CVPR_2022_paper.html.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems*, 2025a. doi: 10.15607/RSS.2025.XXI.017. URL https://www.roboticsproceedings.org/rss21/p017.html.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of Conference on Robot Learning*, pp. 2679–2713, 2025b. URL https://proceedings.mlr.press/v270/kim25c.html.

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. In *Proceedings of International Conference on Machine Learning*, 2024. URL https://proceedings.mlr.press/v235/lee24y.html.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. In *Conference on Robot Learning*, 2025. URL https://proceedings.mlr.press/v270/li25c.html.

Yue Liao, Pengfei Zhou, Siyuan Huang, Donglin Yang, Shengcong Chen, Yuxin Jiang, Yue Hu, Si Liu, Jianlan Luo, Liliang Chen, Shuicheng YAN, Maoqing Yao, and Guanghui Ren. Genie Envisioner: A unified world foundation platform for robotic manipulation. In *Proceedings of International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/8e773e9e463e97b930c47621b6777159-Abstract-Conference.html.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1B: a diffusion foundation model for bimanual manipulation. In *Proceedings of International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/49f80e4d2471ad4f2edf4f5f1ab62339-Abstract-Conference.html.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Liu_A_ConvNet_for_the_2020s_CVPR_2022_paper.html.

Shangchen Miao, Ningya Feng, Jialong Wu, Ye Lin, Xu He, Dong Li, and Mingsheng Long. JEPA-VLA: Video predictive embedding is needed for VLA models. *CoRR*, 2026. URL https://arxiv.org/abs/2602.11832.

Aiden Nibali, Zhen He, Stuart Morgan, and Luke Prendergast. Numerical coordinate regression with convolutional neural networks. *CoRR*, 2018. URL https://arxiv.org/abs/1801.07372.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Jim Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. *CoRR*, 2025. URL https://arxiv.org/abs/2503.14734.

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi Jim Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit

Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick Tree Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, and Zipeng Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. In *Proceedings of International Conference on Robotics & Automation*, pp. 6892–6903, 2024. doi: 10.1109/ICRA57147.2024.10611477. URL https://ieeexplore.ieee.org/document/10611477.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2304.07193.

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

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2018. doi: 10.1609/aaai.v32i1.11671. URL https://aaai.org/papers/11671-film-visual-reasoning-with-a-general-conditioning-layer/.

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 *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.012. URL https://www.roboticsproceedings.org/rss21/p012.html.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael 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. *CoRR*, 2025. URL https://arxiv.org/abs/2504.16054.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya

Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of International Conference on Machine Learning*, 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Dominik Schmidt and Minqi Jiang. Learning to act without actions. In *Proceedings of International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/27985d21f0b751b933d675930aa25022-Abstract-Conference.html.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-Actor: A multi-task transformer for robotic manipulation. In *Proceedings of Conference on Robot Learning*, pp. 785–799, 2023. URL https://proceedings.mlr.press/v205/shridhar23a.html.

Haoming Song, Delin Qu, Yuanqi Yao, Qizhi Chen, Qi Lv, Yiwen Tang, Modi Shi, Guanghui Ren, Maoqing Yao, Bin Zhao, Dong Wang, and Xuelong Li. Hume: Introducing system-2 thinking in visual-language-action model. *CoRR*, 2025. URL https://arxiv.org/abs/2505.21432.

StarVLA Community. StarVLA: A Lego-like codebase for vision-language-action model developing. *CoRR*, 2026. URL https://arxiv.org/abs/2604.05014.

Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8922–8931, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Sun_LoFTR_Detector-Free_Local_Feature_Matching_With_Transformers_CVPR_2021_paper.html.

Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. Integral human pose regression. In *Proceedings of European Conference on Computer Vision*, 2018. URL https://openaccess.thecvf.com/content_ECCV_2018/html/Xiao_Sun_Integral_Human_Pose_ECCV_2018_paper.html.

Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. In *Advances in Neural Information Processing Systems*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/e0f393e7980a24fd12fa6f15adfa25fb-Abstract-Conference.html.

Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, and Jian Tang. TinyVLA: Toward fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025. doi: 10.1109/LRA.2025.3544909. URL https://ieeexplore.ieee.org/document/10900471/.

Pei Yang, Hai Ci, Yanzhe Chen, Qi Lv, Han Cai, and Mike Zheng Shou. ActionMap: Robot policy learning via voxel action heatmap. *CoRR*, 2026. URL https://arxiv.org/abs/2606.06904.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *Proceedings of International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/45d74e190008c7bff2845ffc8e3facd3-Abstract-Conference.html.

Shanglin Yuan, Weiheng Zhao, Xin Shi, Haoyi Jiang, Xianda Guo, Liu Liu, Wenyu Liu, Wei Sui, and Xinggang Wang. DreamWAM: Beyond RGB future prediction for world action models. *CoRR*, 2026a. URL https://arxiv.org/abs/2608.04996.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *CoRR*, 2026b. URL https://arxiv.org/abs/2603.16666.

Chubin Zhang, Jianan Wang, Zifeng Gao, Yue Su, Tianru Dai, Cai Zhou, Jiwen Lu, and Yansong Tang. CLAP: Contrastive latent action pretraining for learning vision-language-action models from human videos. *CoRR*, 2026. URL https://arxiv.org/abs/2601.04061.

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. URL https://www.roboticsproceedings.org/rss19/p016.html.

Pengfei Zhou, Shengcong Chen, Di Chen, Jiaxu Wang, Rongjun Jin, Bingwen Zhu, Yike Pan, Songen Gu, Kuanning Wang, Shufeng Nan, Xingyu Qiu, Chenhao Qiu, Pu Yang, Yunuo Cai, Jianxiong Gao, Yifan Li, Yanwei Fu, Xiangyu Yue, Zhi Chen, and Jianlan Luo. τ_0-WM: A unified video-action world model for robotic manipulation. *CoRR*, 2026. URL https://arxiv.org/abs/2606.01027.

A Model Details

A.1 Embodiment Conditioning

Each essential vector is a row of a learned embedding table initialized from $\mathcal{N}(0, n^{-1/2})$. We set $n = 128$ in practice. A conditioned residual block takes the form

$$F_E(x) = x + \mathrm{MLP}(\mathrm{LN}(x) \odot (1 + \gamma(E)) + \beta(E)) . \tag{5}$$

LayerNorm has no affine parameters. The projection that produces (γ, β) and the final linear layer of the MLP are zero-initialized, so F_E begins as an identity map. The TransConv Head uses GroupNorm in place of LayerNorm and applies the same E-conditioned scale and shift.

A.2 Action Codec

A.2.1 Action Normalization and Chunk Construction

For action axis j, the raw command $a_{i,j}$ is normalized with a positive scale s_j defined for each source according to

$$\tilde{a}_{i,j} = \mathrm{clip}\left(\frac{a_{i,j}}{s_j}, -1, 1\right), \qquad \hat{a}_{i,j} = s_j \hat{\tilde{a}}_{i,j}. \tag{6}$$

This zero-preserving transform keeps every valid axis on the shared interval $[-1, 1]$. Let f be the native control frequency and H the selected physical horizon. We choose the number of steps as $T = \max(1, \lfloor Hf + 0.5 \rfloor)$. Table 5 lists the values used by the final eight-embodiment mixture. Most sources use a horizon of 0.5 seconds. Bridge and Fractal operate at only 5 Hz and 3 Hz, respectively, so we manually extend their horizons to 1 and 2 seconds to retain 5 and 6 action steps.

Table 5: Action normalization and chunk construction. Scale vectors follow the native axis order, and $\times 2$ denotes a vector shared by both arms. RT2 denotes RoboTwin 2.0. The 4 LIBERO suites share one entry.

Source / embodiment	f (Hz)	T	H (s)	Scale $\mathbf{s}$
LIBERO / Panda	20	10	0.5	$(1, 1, 1, 0.375, 0.375, 0.375, 1)$
Bridge / WidowX	5	5	1.0	$(0.05, 0.075, 0.075, 0.15, 0.15, 0.3, 1)$
Fractal / Google	3	6	2.0	$(0.3, 0.25, 0.375, 0.75, 0.6, 0.75, 1)$
RT2 / Aloha AgileX	30	15	0.5	$(1.25, 3, 3, 2, 1.5, 3, 1) \times 2$
RT2 / Franka	30	15	0.5	$(2, 2, 2, 3, 3, 3.75, 3, 1) \times 2$
RT2 / ARX X5	30	15	0.5	$(1.5, 3, 2.5, 2, 1.5, 3, 1) \times 2$
RT2 / Piper	30	15	0.5	$(1, 3, 2.5, 2, 1.25, 3, 1) \times 2$
RT2 / UR5	30	15	0.5	$(2.5, 3, 2.5, 2, 3, 3, 1) \times 2$

A.2.2 Action and Proprioceptive Encoders

The codec uses token width $D = 512$ and eight attention heads. For normalized action $\tilde{a}_{i,j}$ at time step i and axis j, the input token is

$$x_{i,j} = W_a \tilde{a}_{i,j} + \psi(\tau_i) + e_j^{\mathrm{axis}} + e_j^{\mathrm{arm}}. \tag{7}$$

Here, W_a is a learned scalar projection, τ_i denotes the temporal metadata of step i, and e_j^{axis} and e_j^{arm} are the axis and arm embeddings. The embedding ψ parameterizes continuous time through the normalized phase $u_i \in [0, 1]$ and control period Δt_i,

$$\psi(\tau_i) = \mathrm{MLP}\Big(\Big[u_i, \log \Delta t_i, \big\{\sin(2\pi 2^k u_i), \cos(2\pi 2^k u_i)\big\}_{k=0}^{7}\Big]\Big) . \tag{8}$$

Let $X = \{x_{i,j}\}$ collect all action tokens and let $M_a \in \{0, 1\}^{T \times A}$ denote their validity mask. An entry of M_a is one only when the corresponding time step and action axis are both valid. Two Conditioned MLP blocks, each with expansion ratio four, demodulate X. The proprioceptive path adds an axis embedding to each projected scalar and applies two Conditioned MLP blocks with the

same architecture. Its Query Pooling module uses a single learned query to produce the context vector c_p. With $Q = \{q_k\}_{k=1}^{K}$ denoting the K learned action queries, the action path is

$$\bar{z}_a = \mathrm{QueryPooling}_Q\left(F_E^{(2)}(X), M_a\right), \qquad z_a = \mathrm{LN}(\bar{z}_a + \sigma(g)c_p)\,. \tag{9}$$

The operator $\mathrm{QueryPooling}_Q(\cdot, M_a)$ performs attention from Q to the valid tokens selected by M_a, while $\sigma(g)$ is the learned proprioceptive gate shown in Figure 2. The pooling queries are initialized from $\mathcal{N}(0, D^{-1/2})$. We initialize $g = -2$ and drop the complete proprioceptive context with probability 0.1 during training. The embedding tables support 128 action axis types, eight arms, and 64 proprioceptive coordinates.

A.2.3 Action Decoder and TransConv Head

During decoding, the Time Query module combines the requested timestamps, encoded by Eq. 8, with a chunk-level intent $z \in \mathbb{R}^{K\times D}$. It uses two attention blocks to produce one temporal feature per requested step. The resulting features are specialized for each target action axis and remodulated by two Conditioned MLP blocks,

$$r_{1:T} = \mathrm{TimeQuery}\big(z, \{\psi(\tau_i)\}_{i=1}^{T}\big)\,, \qquad y_{i,j} = G_E^{(2)}\big(r_i + e_j^{\mathrm{axis}} + e_j^{\mathrm{arm}} + c_p\big)\,. \tag{10}$$

Here, r_i is the temporal feature conditioned on intent and $G_E^{(2)}$ denotes the two Conditioned MLP blocks. Both the temporal and action axis masks are retained throughout this computation.

The TransConv Head first uses a linear layer to lift $y_{i,j}$ to $D_{\mathrm{head}} = 128$ channels. It then repeats an Upsample Block followed by a Refinement Block 4 times, expanding the value axis as $1 \to 4 \to 16 \to 64 \to 256$. Each Upsample Block applies a depthwise transposed convolution with kernel size and stride 4, followed by E-conditioned GroupNorm, GELU, and a residual MLP. Each Refinement Block follows the same structure with a depthwise transposed convolution of kernel size 4 and stride 1. A final linear layer produces the bin logits. Writing H_E for the complete TransConv Head, the projection and soft expectation are

$$\ell_{i,j} = H_E(y_{i,j}) \in \mathbb{R}^{256}, \qquad p_{i,j,b} = \mathrm{softmax}_b(\ell_{i,j}), \tag{11}$$

$$\hat{\tilde{a}}_{i,j} = \mathbb{E}_{b\sim p_{i,j}}[c_b] = \sum_{b=1}^{256} p_{i,j,b}c_b, \qquad c_b \in [-1, 1]. \tag{12}$$

The symbols $\ell_{i,j}$ and $p_{i,j,b}$ denote the bin logits and the corresponding softmax probabilities, and c_b is the center of bin b. With output validity mask $M \in \{0,1\}^{T\times A}$, the action reconstruction objective is

$$\mathcal{L}_{\mathrm{act}} = \frac{\sum_{i,j} M_{i,j}\left|\hat{\tilde{a}}_{i,j} - \tilde{a}_{i,j}\right|}{\sum_{i,j} M_{i,j}}. \tag{13}$$

The bin logits are supervised only through this expected action. They form a structured decoding basis rather than a calibrated estimate of action uncertainty. Dropout is set to 0.1 in the attention and residual MLP blocks.

A.2.4 Connection to Integral Coordinate Regression

Our readout follows an established approach to coordinate regression. Integral pose regression (Sun et al., 2018) shows that a softmax expectation over a heatmap can be trained with coordinate ℓ_1 loss alone, without direct heatmap supervision. Expectation-based coordinate readout also appears in LoFTR's fine matching stage (Sun et al., 2021). DSNT (Nibali et al., 2018) similarly provides a parameter-free expectation operator that connects convolutional heatmaps to continuous coordinates. These results motivate learning the bin representation indirectly through the decoded action.

The convolutional parameterization supplies a separate design motivation. Nibali et al. (2018) argue that spatial parameter sharing helps coordinate models generalize across image locations and that a learned fully connected readout can compromise this property. In CIALLO, the corresponding axis indexes numerical action values, and the TransConv Head reuses local kernels along that axis as it expands the bin resolution. This introduces locality and weight sharing into the output representation. The analogy motivates our architecture, but does not establish the image-translation generalization studied by DSNT for robot actions.

Coordinate supervision does not uniquely determine the intermediate representation. DSNT discusses this ambiguity and explores additional regularization to control heatmap shape (Nibali et al., 2018). Bias-Compensated Integral Regression further identifies decoding bias from the softmax–expectation combination and slower convergence under implicit heatmap supervision in pose estimation (Gu et al., 2023). These findings caution against interpreting a differentiable readout as a guarantee of faithful distribution learning or efficient optimization. CIALLO uses neither heatmap regularization nor bias compensation. We treat its bin weights as a learned decoding basis and evaluate the resulting actions, without assuming a particular distribution shape, number of modes, or calibrated uncertainty.

Figure 4 illustrates how the bin weights evolve under scalar supervision in a simplified setting. We pass a fixed input through two transposed convolution layers and a shared linear projection to produce logits over 16 ordered bins. We train the network by applying an ℓ_1 loss to the softmax expectation, using noisy labels around a fixed nominal target of 0.75.

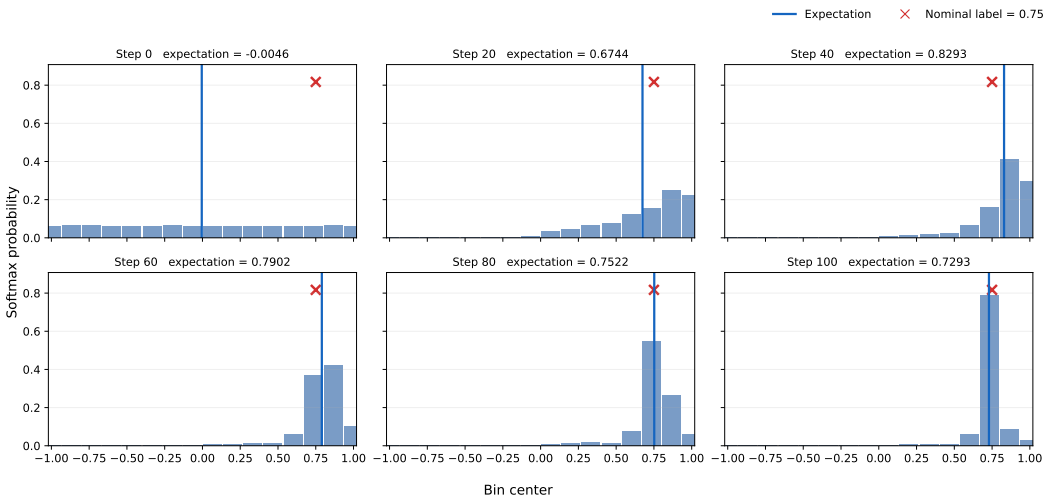

Figure 4: **Evolution of bin weights under scalar supervision.** Bars show the softmax weights at six training steps. The blue line indicates the predicted scalar, computed as their expectation, and the red cross marks the nominal target of 0.75. The prediction approaches the target as the weights change, although no loss is applied directly to the bin weights.

A.3 Vision Encoder

The vision encoder compares the observations at the two endpoints of the same action chunk. For camera c, we denote the current frame by o_c^t and the future frame by o_c^{t+T-1}, reusing the time index and chunk length T from the action path. Both frames are resized to 252×252 and processed by the same frozen DINOv2-base backbone Φ. With a patch size of 14 and the class token removed, the resulting features are

$$F_c^t = \Phi(o_c^t), \qquad F_c^{t+T-1} = \Phi(o_c^{t+T-1}), \qquad F_c^t, F_c^{t+T-1} \in \mathbb{R}^{18\times18\times768}. \tag{14}$$

The same Φ is shared across times and cameras, and its parameters receive no gradient updates.

For each camera, we concatenate the two 18×18 feature grids along the channel dimension and process them through the modules shown in Figure 3. Input Proj applies LayerNorm, convolution, GELU, and convolution to produce features with 256 channels. Stage 1 contains four E-conditioned ConvNeXt blocks at this resolution. Downsample uses LayerNorm and a convolution with stride 2 to obtain a 9×9 grid with 384 channels, which is processed by the E-conditioned blocks in Stage 2. Output Proj maps the result to width $D = 512$ and flattens it into 81 tokens per camera.

We support up to 4 camera slots. The 81 tokens from each slot receive its learned camera embedding, and a camera validity mask excludes every token from an absent view. Query Pooling attends from $K = 2$ learned queries to the remaining tokens and returns the visual intent in $\mathbb{R}^{K\times D}$. For fixed

K, this pooling cost grows linearly with the number of visual tokens. The encoder therefore avoids both global self-attention across patches and a pairwise cost volume between the two frames.

A.4 TEXT CODEC

The frozen BERT encoder produces one feature per instruction token. A learned linear projection maps these features to width $D = 512$, while the instruction attention mask removes padding. Query Pooling uses four learned queries to compress the valid features into a four-token task-level intent. The projection and pooling modules are trained while BERT remains frozen.

To ground each chunk, the text path reuses the DINOv2 features F_c^t from the current frame in Eq. 14, before the interaction encoder for the frame pair. Each camera contributes 324 patch tokens from the current frame. These tokens are projected to width D, augmented with camera and visual modality embeddings, and concatenated with the task-level intent after adding a task modality embedding. The combined validity mask keeps all four task-level intent tokens and only the patches from available cameras. A second Query Pooling module reduces this sequence to $K = 2$ tokens, and two Transformer Layers produce the chunk-level intent in $\mathbb{R}^{K\times D}$. This path receives neither the essential vector E nor proprioception.

The two-layer Masked Decoder receives the corrupted instruction with learned position embeddings and attends to the task-level intent as memory. Its vocabulary projection is tied to the frozen BERT input embedding, and cross-entropy is averaged only over the selected masked positions. The chunk-level intent is not used as decoder memory. This direct reconstruction path forces the task-level intent to preserve instruction content rather than only the identity of a task class.

A.5 MEMORY-EFFICIENT CONTRASTIVE ALIGNMENT

For modalities p and q, we mean-pool each valid chunk-level intent and apply ℓ_2 normalization to obtain $h_i^p, h_i^q \in \mathbb{R}^D$. A differentiable gather across all devices provides B global pairs. With a learnable inverse temperature α initialized to $1/0.07$, the directional loss is

$$\mathcal{L}_{p\to q} = \frac{1}{B}\sum_{i=1}^{B}\left[\log\sum_{j=1}^{B}\exp(s_{ij}) - s_{ii}\right], \qquad s_{ij} = \alpha\,(h_i^p)^\top h_j^q. \tag{15}$$

Diagonal pairs are positive, and all other samples are negatives. We symmetrize the objective as $\mathcal{L}_{p\leftrightarrow q} = \frac{1}{2}(\mathcal{L}_{p\to q} + \mathcal{L}_{q\to p})$.

The dense evaluation of Eq. 15 stores B^2 similarities in order to compute $\log\sum_{j=1}^{B}\exp(s_{ij})$ for each row. Following FlashAttention and Inf-CL (Dao et al., 2022; Cheng et al., 2025), we implement this log-sum-exp as an exact blockwise streaming computation. We split both axes into blocks of at most C samples and never materialize the full similarity matrix. Let $\mathcal{J}_c$ denote the indices in column block c. For an active row i, we first compute the largest similarity $\mu_i^{(c)}$ in this block and use it to evaluate the block log-sum-exp without exponentiating unshifted logits:

$$\mu_i^{(c)} = \max_{j\in\mathcal{J}_c} s_{ij}, \qquad \lambda_i^{(c)} = \mu_i^{(c)} + \log\sum_{j\in\mathcal{J}_c}\exp\Big(s_{ij} - \mu_i^{(c)}\Big). \tag{16}$$

Here $\lambda_i^{(c)}$ is the exact log-sum-exp over column block c. We merge it with the running normalization using a second max shift. Starting from $L_i^{(0)} = -\infty$, define

$$\rho_i^{(c)} = \max\Big(L_i^{(c-1)}, \lambda_i^{(c)}\Big), \tag{17}$$

$$L_i^{(c)} = \rho_i^{(c)} + \log\Big[\exp\Big(L_i^{(c-1)} - \rho_i^{(c)}\Big) + \exp\Big(\lambda_i^{(c)} - \rho_i^{(c)}\Big)\Big]. \tag{18}$$

Here $\rho_i^{(c)}$ is the larger of the previous running normalization and the current block contribution. After processing blocks 1 through c, the recurrence satisfies $L_i^{(c)} = \log\sum_{d=1}^{c}\sum_{j\in\mathcal{J}_d}\exp(s_{ij})$. The final L_i is therefore exactly the dense log-sum-exp for row i, giving the loss $B^{-1}\sum_i(L_i - s_{ii})$. It is accumulated in FP32 while retaining only the current similarity block and one normalization per row.

The backward pass recomputes L_i and revisits the same tiles. Let $H_p^{(r)}$ and $H_q^{(c)}$ be the normalized feature blocks, and define $G_{ij} = [\exp(s_{ij} - L_i) - \mathbb{I}(i = j)]/B$. The feature gradients accumulate as

$$\nabla H_p^{(r)} \mathrel{+}= \alpha G^{(r,c)} H_q^{(c)}, \qquad \nabla H_q^{(c)} \mathrel{+}= \alpha \big(G^{(r,c)}\big)^\top H_p^{(r)}. \tag{19}$$

The reverse direction exchanges p and q, and the same tiles yield $\nabla\alpha$. This exact reordering reduces memory for the similarity matrix from $B \times B$ to $C \times C$ while retaining quadratic computational complexity.

B Pretraining Diagnostics

B.1 Pretraining Results

We evaluate the final checkpoint from one pretraining seed using an episode-level validation split. The evaluation includes 2,048 held-out samples from 1,575 episodes and 512 tasks. Each 2×512 intent is flattened in slot order, and exact nearest neighbors are found with raw ℓ_2 distance. Cross-modal retrieval measures how accurately each representation retrieves its paired demonstration sample. We denote reconstruction from action, visual, and grounded language intent by AA, VA, and TA, respectively. The language representation is the chunk-level intent conditioned on the current observation, distinct from the four-token task-level intent used for instruction reconstruction.

Table 6 reports retrieval within each embodiment and reconstruction through the shared decoder. The higher VA and TA errors for Panda, WidowX, and Google Robot may reflect differences in task coverage, camera setup, control rate, action normalization, and environment. Per-embodiment MAE should therefore not be interpreted as a ranking of representation quality. In the All row, MAE is aggregated over valid action elements, while retrieval searches the complete sample pool.

Table 6: Intent retrieval and action reconstruction at the final pretraining checkpoint. Retrieval entries report mean top-1 accuracy across both directions in %. Per-embodiment retrieval uses candidates from the same embodiment, while the final row uses the complete pool. Reconstruction entries are normalized masked MAE.

		Top-1 retrieval			Action MAE		
Embodiment	Samples	A↔V	A↔T	V↔T	AA	VA	TA
Panda	245	94.90	89.18	95.31	0.01363	0.06924	0.08772
WidowX	263	98.86	94.87	98.86	0.02245	0.08469	0.11743
Google Robot	269	98.88	93.49	99.07	0.01851	0.09376	0.11423
Aloha AgileX	259	97.68	94.98	97.88	0.00572	0.00840	0.01098
Franka Dual	267	95.88	89.51	95.51	0.00691	0.01198	0.01510
ARX X5	257	95.14	90.27	95.33	0.00580	0.00919	0.01346
Piper	245	95.71	91.02	95.31	0.00670	0.01093	0.01580
UR5	243	96.30	92.18	97.53	0.00620	0.00943	0.01201
All	**2,048**	**96.68**	**91.94**	**96.88**	**0.00761**	**0.01853**	**0.02429**

B.2 Number of Pretraining Embodiments

We compare pretraining on three, five, and eight embodiments using the same training objective and number of optimization steps. The three-embodiment setting uses Panda, WidowX, and Google Robot. The five-embodiment setting adds Aloha AgileX and Franka Dual from RoboTwin 2.0, and the eight-embodiment setting further adds ARX X5, Piper, and UR5. Table 7 reports AA and VA reconstruction errors for each setting, with eight-embodiment results taken from Table 6.

AA and VA MAE decrease for each of the three original embodiments as the number of training embodiments increases from three to five to eight. Both errors also decrease for Aloha AgileX and Franka Dual when training expands from five to eight embodiments. This trend suggests that greater embodiment diversity, together with the addition of RoboTwin data, helps CIALLO learn shared cross-embodiment intent representations.

Table 7: AA and VA MAE with increasing numbers of pretraining embodiments. Dashes indicate embodiments absent from the corresponding training setting. Lower values are better. The five-embodiment evaluation uses 819, 857, 799, 814, and 807 validation samples for the first five rows, respectively.

	3 embodiments		5 embodiments		8 embodiments	
Embodiment	AA	VA	AA	VA	AA	VA
Panda	0.0807	0.1002	0.0339	0.0766	0.0136	0.0692
WidowX	0.0920	0.1107	0.0331	0.0986	0.0225	0.0847
Google Robot	0.0899	0.1129	0.0336	0.1074	0.0185	0.0938
Aloha AgileX	–	–	0.0467	0.0493	0.0057	0.0084
Franka Dual	–	–	0.0506	0.0553	0.0069	0.0120
ARX X5	–	–	–	–	0.0058	0.0092
Piper	–	–	–	–	0.0067	0.0109
UR5	–	–	–	–	0.0062	0.0094

B.3 Intent Capacity Ablation

We compare intent capacities of $K = 1$, 2, and 4 tokens, keeping the token width fixed at $D = 512$. All other training settings remain fixed, including the full three-modal objective in Eq. 4. We evaluate each run after 2,000 optimization steps. Table 8 reports validation losses, macro AA and VA MAE, and peak GPU memory. Macro MAE averages the per-embodiment MAEs equally across all eight embodiments, regardless of action dimensionality or horizon.

Table 8: Intent capacity comparison after 2,000 training steps. Lower values are better for all metrics. Losses follow the aggregation used by the validation objective, whereas macro MAE weights embodiments equally. Memory is reported in GiB. Bold values indicate the lowest loss or MAE in each column.

	Validation loss			Macro MAE		Peak memory (GiB)	
K	Total	AA	VA	AA	VA	Allocated	Reserved
1	0.264283	0.024807	0.036571	**0.035501**	0.056521	37.412	38.494
2	**0.261221**	**0.023484**	**0.034908**	0.036392	**0.056142**	37.414	38.500
4	0.304822	0.033762	0.045132	0.044961	0.065114	37.418	38.508

Two tokens yield the lowest total, AA, and VA validation losses and the lowest macro VA MAE. One token performs similarly and yields a slightly lower macro AA MAE. Four tokens increase total validation loss by 16.7% relative to two tokens and increase both macro MAEs. Peak memory is nearly unchanged across capacities, with ranges of 0.006 GiB for allocated memory and 0.014 GiB for reserved memory. These results support our choice of $K = 2$.

B.4 Contrastive Alignment Ablation

We compare four combinations of contrastive losses while retaining AA, VA, TA, and masked text reconstruction in every setting. All models are trained on eight embodiments with $K = 2$ for 2,000 optimization steps, using the same validation set and otherwise identical training settings. The configurations comprise reconstruction alone, reconstruction with action–vision alignment, additional action–text alignment, and the full objective with vision–text alignment. Table 9 reports macro AA and VA MAE. The full-model values are taken from the $K = 2$ setting in Table 8.

Action–vision alignment alone yields the lowest macro VA MAE, whereas the full objective yields the lowest macro AA MAE. We prioritize AA reconstruction accuracy because downstream policy learning uses the action codec, and therefore adopt all three contrastive terms for CIALLO. This objective also reduces both AA and VA MAE relative to reconstruction alone.

Table 9: Ablation of contrastive alignment terms with all reconstruction losses retained. A, V, and T denote action, vision, and grounded text representations. ✓ and ✗ indicate enabled and disabled terms, respectively. Macro MAE is the unweighted mean across the eight embodiments. Lower values are better, and bold denotes the minimum in each MAE column.

A↔V	A↔T	V↔T	Macro AA MAE	Macro VA MAE
✗	✗	✗	0.039171	0.057500
✓	✗	✗	0.040415	**0.054645**
✓	✓	✗	0.038087	0.054947
✓	✓	✓	**0.036392**	0.056142

B.5 Direct Closed-Loop Control

We test the pretrained TA path directly in closed loop without policy finetuning. This CIALLO-Direct setting reaches 73.4% success on LIBERO Spatial, 52.2% on Object, 65.6% on Goal, and 17.8% on Long. It establishes a direct control baseline for the pretrained representation before coupling the codec to Qwen3.5.

B.6 Intent Geometry

Figure 5 provides a complementary view of the learned geometry. The five RoboTwin embodiments occupy a broad shared region, consistent with their common task set and similar simulated environments. Panda, WidowX, and Google Robot occupy more distinct, partially overlapping regions, which may reflect differences in their tasks and environments. Within each embodiment, the three modalities exhibit similar local structure. These observations agree with the retrieval results in Table 6, but the t-SNE axes and global cluster distances have no direct meaning and do not establish strict embodiment invariance.

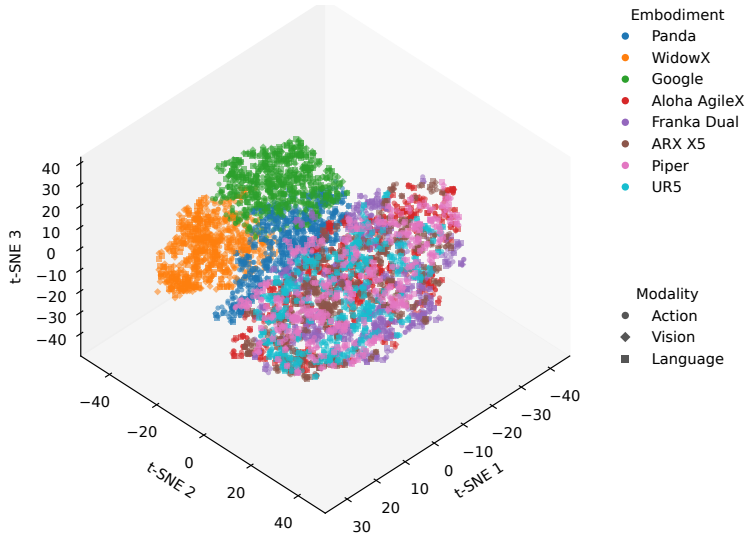

Figure 5: Intent geometry at the final pretraining checkpoint. Color identifies the embodiment and marker shape identifies the input modality. A shared three-dimensional t-SNE embedding is fitted after reducing the flattened intents to 50 dimensions with PCA. The projection emphasizes local neighborhood structure, while global distances should not be interpreted quantitatively.

B.7 Embodiment Conditioning

We test the functional role of the essential vector on the same 2,048 validation samples used in Section 4.1. Each sample is assigned the next embodiment in a fixed cycle from Panda through

WidowX, Google Robot, Aloha AgileX, Franka Dual, ARX X5, Piper, and UR5 before returning to Panda. We keep the action, observations, instruction, proprioception, timestamps, masks, and target action schema unchanged. The wrong encoder condition replaces E only in the action or visual encoder and its proprioceptive context. The wrong decoder condition replaces E only in the shared action decoder. The all wrong condition applies both replacements. This protocol tests conditioning under a controlled mismatch. It does not retarget an action to another robot.

Table 10 shows that the wrong decoder condition raises MAE for all three paths to approximately 0.19–0.21. The wrong action encoder also raises AA MAE from 0.00761 to 0.04506. Replacing the visual encoder condition has a smaller effect on VA, whose MAE rises from 0.01853 to 0.02189. The text path does not consume E or proprioception, so its wrong encoder value is unchanged by construction. These results show that the action encoder and decoder use the essential vector, while the visual path remains sensitive to it to a lesser degree. The errors need not increase monotonically across conditions because simultaneous encoder and decoder mismatches can partially compensate for one another.

Table 10: Normalized masked action MAE under essential vector mismatch. Parentheses report the relative increase from the correct condition. Only embodiment conditioning is changed, while every sample retains its original target action schema.

Path	Correct	Wrong encoder	Wrong decoder	All wrong
AA	0.00761	0.04506 (+492.1%)	0.19124 (+2413.0%)	0.16470 (+2064.3%)
VA	0.01853	0.02189 (+18.1%)	0.20082 (+983.8%)	0.21534 (+1062.1%)
TA	0.02429	0.02429 (+0.0%)	0.20684 (+751.5%)	0.20684 (+751.5%)

C Training Details

C.1 Qwen-CIALLO

For LIBERO, we train a single Qwen-CIALLO policy jointly on all four task suites for 10,000 optimization steps on 8 NVIDIA H20 GPUs. The per-device batch size is 16. We use AdamW with weight decay 10^{-8}, 5,000 warmup steps, and cosine annealing. The Qwen3.5-4B backbone, CIALLO action decoder, and intent projector use learning rates of 2×10^{-5}, 2×10^{-6}, and 1×10^{-4}, respectively.

For RoboTwin 2.0, we train a single Qwen-CIALLO policy jointly across all 50 tasks for 40,000 optimization steps on 8 NVIDIA H20 GPUs. The per-device batch size is 4. We otherwise use the LIBERO training configuration, including AdamW with weight decay 10^{-8}, 5,000 warmup steps, cosine annealing, and learning rates of 2×10^{-5} for the Qwen3.5-4B backbone, 2×10^{-6} for the CIALLO action decoder, and 1×10^{-4} for the intent projector.

C.2 Adaptation to New Embodiments

Codec calibration. We use the pretrained CIALLO vision encoder and action codec to calibrate each robot from a recording of at most 10 seconds. The action encoder maps demonstrated action chunks and proprioception to action intent, while the vision encoder maps the corresponding current and future frames to visual intent. Both representations pass through the shared action decoder to reconstruct the demonstrated actions. Calibration combines masked ℓ_1 reconstruction along the action-to-action (AA) and vision-to-action (VA) paths with a contrastive loss between the action and visual intents, using the objectives described in Section 3.5. The text branch is not used. Shared network weights remain frozen, and gradients update only the target robot's 128-dimensional essential vector and its axis and arm embeddings. The embeddings contain 174.08k parameters.

For both SO-101 and single-arm Piper, calibration runs for 5,000 steps on one NVIDIA H20 GPU with a batch size of 64, AdamW, and a learning rate of 10^{-2}. The action chunk size and execution horizon are both 15 steps.

Qwen-CIALLO. After calibration, we freeze the entire action codec and train the policy for 30,000 steps on 8 NVIDIA H20 GPUs with a per-device batch size of 4. We use AdamW with 2,500

warmup steps and learning rates of 2×10^{-5} for the Qwen3.5-4B backbone and 10^{-4} for the intent projector. The action chunk size and execution horizon both remain 15 steps.

π_0. We initialize from the pretrained π_0-base checkpoint and finetune the full model for 30,000 steps on 8 NVIDIA H20 GPUs with a per-device batch size of 4. We use AdamW with a learning rate of 2×10^{-5} and 2,500 warmup steps. The action chunk size and execution horizon are both 50 steps.

D Action Head Efficiency

We compare three action decoding approaches: the direct point-regression head used by OpenVLA-OFT (Kim et al., 2025a), the flow-matching head of $\pi_{0.5}$ (Physical Intelligence et al., 2025), and the CIALLO action decoder. CIALLO uses a distribution-based regression readout that predicts softmax weights over 256 ordered bins and outputs their expectation in a single forward pass (Eq. 12). The bin weights are learned through action regression and serve as a structured decoding basis, rather than a calibrated action distribution.

Benchmark protocol. All heads are benchmarked on a single NVIDIA H20 GPU in BF16 precision, with batch size 1, an action horizon of 8 steps, and action dimension 7. We report mean latency over 100 inference runs after 20 warmup runs. Table 11 reports head parameter counts, FLOPs, and inference latency, excluding the policy backbone. For comparison within the framework used by $\pi_{0.5}$, we also implement the CIALLO decoder in JAX with the same architecture and action readout. This version is labeled CIALLO (JAX).

Table 11: Action head efficiency on a single NVIDIA H20 with BF16, batch size 1, an action horizon of 8 steps, and action dimension 7. Measurements exclude the policy backbone. Latency is averaged over 100 inference runs after 20 warmup runs.

Method	Params	FLOPs	Latency
OFT (Kim et al., 2025a)	151.12M	2.42G	0.37ms
$\pi_{0.5}$ (JAX) (Physical Intelligence et al., 2025)	430.10M	79.13G	19.88ms
CIALLO	19.89M	46.49G	4.89ms
CIALLO (JAX)	19.89M	46.49G	1.46ms

Results. CIALLO uses 19.89M parameters, approximately $7.6\times$ fewer than OFT and $21.6\times$ fewer than the $\pi_{0.5}$ action head. Under the same JAX framework, CIALLO decodes an action chunk in 1.46 ms, compared with 19.88 ms for $\pi_{0.5}$, a $13.6\times$ speedup. Its 46.49G FLOPs are also below the 79.13G FLOPs of the flow-matching head. The original CIALLO implementation takes 4.89 ms, showing that the implementation framework materially affects latency even with an unchanged architecture. The single-pass expectation readout avoids the iterative action refinement used by flow matching.

OFT remains the fastest head at 0.37 ms and uses the fewest FLOPs (2.42G). CIALLO instead combines a smaller parameter count with a structured action-value representation and substantially lower latency than the flow-matching baseline. At the policy level, Qwen-CIALLO achieves higher success than OpenVLA-OFT on LIBERO (97.3% versus 97.1%, Table 1) and LIBERO-Plus (80.8% versus 69.6%, Table 2).

E Real Robot Tasks

We use SO-101 and single-arm Piper, each equipped with a gripper, to evaluate manipulation in a tabletop workspace. Demonstrations are recorded at 30 Hz with a wrist camera and a fixed front camera, and actions are represented as joint angles. Each robot contributes 20 demonstrations for each of the three tasks in Table 12. Figure 6 shows front camera observations sampled from these demonstrations.

Table 12: Task instructions used for data collection and evaluation on both robots. The wording is reproduced from the dataset.

Task	Instruction
Water bottle	Grab the water bottle and place on the black plate.
Pour beer	Pour beer from bottle into the cup.
Wooden fish	Pick up the screwdriver and hit the wooden fish.

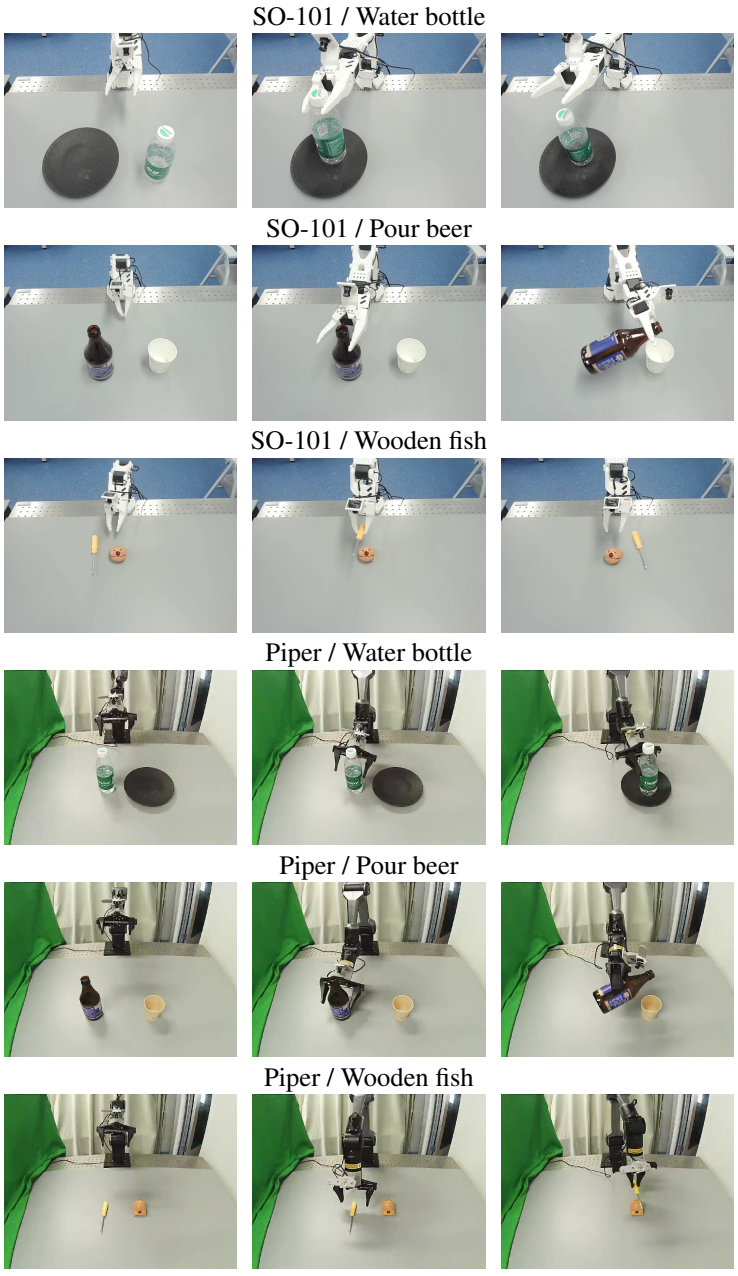

Figure 6: Demonstration sequences for the three real robot tasks on SO-101 and single-arm Piper. Each row shows front camera observations at 15%, 50%, and 85% of one demonstration, ordered from left to right. These examples illustrate the recorded trajectories and do not indicate evaluation success.