

Dynin-Robotics: Omnimodal Unified Diffusion Vision-Language-Action Model

Hoeun Lee^{§¶}, Jaeik Kim[¶], Jusang Oh[¶], Jinhyeok Kim, Geon Choi, Hyeonggeun Kim, Jaeyoung Do[†]

AIDAS Lab
Seoul National University

[🏠 Project Page](#) [🔗 Code](#) [📄 Model](#)

Abstract

Visual goal and dynamics prediction can provide language-conditioned robot policies with both a target outcome and a representation of action-dependent scene changes. We bring these predictions into action generation and selection through a shared trajectory model. Dynin-Robotics implements this formulation on Dynin-Omni, an omnimodal masked-diffusion backbone, representing language, visual observations, goals, and actions as discrete tokens. By varying the conditioning and target spans, the same model learns action prediction, action-conditioned next-observation prediction, terminal goal-state prediction, and trajectory-to-instruction reconstruction. These conditional interfaces support test-time scaling by allocating additional computation to goal prediction and action-candidate evaluation, as well as joint refinement of action and future-state predictions. We continually pretrain the model on approximately 1.33 million trajectories from 48 Open X-Embodiment datasets and adapt it separately to downstream domains. On two VLABench tasks, robot pretraining improves adaptation within a fixed Stage-2 step budget, and the full objective mixture improves shifted-instruction success over Policy-only post-training under the same coupled decoder. Combining goal guidance with joint action-next-state denoising further improves shifted-instruction success over action-only decoding; the benefit depends on how the predictions are composed. Dynin-Robotics achieves competitive performance on LIBERO and zero-shot LIBERO-Plus, together with a 78.4% average success rate across four manipulation conditions on a Franka Research 3 robot. An optimized block-parallel implementation accelerates model-side action decoding by up to 29.2 $\times$ relative to the base implementation under the reported profiling setup. These results support shared trajectory modeling as a common interface for learning complementary robot objectives and composing their predictions during control.

1 Introduction

Vision-language-action (VLA) models [1–6] connect natural-language instructions and visual observations to robot actions. Language-conditioned manipulation spans target identification, object interaction, and continuous control under contact, clutter, and occlusion [7–12]. For example, following the instruction “hand me something to cut the package” involves selecting an object by its function and grasping it in the current scene. Visual goal and dynamics prediction offer representations of the intended outcome and the scene changes associated with an action. Bringing these predictions into action inference provides a way to connect task semantics with physical execution.

Recent work approaches this connection from different pretraining and modeling choices. VLM-based policies build on large vision-language backbones [1–6, 13–19], adapting language-vision representations for instruction following and action prediction. Video-based policies and related predictive approaches [20–27] incorporate

^{§¶} Project lead. [¶] Core contributors. [†] Corresponding author.

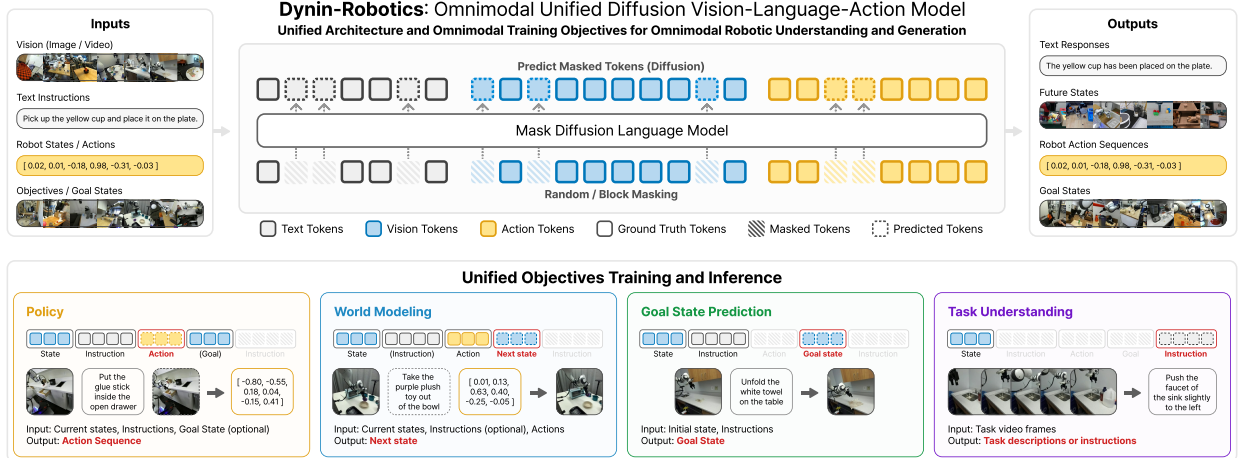


Figure 1 Overview of Dynin-Robotics. A single omnimodal masked-diffusion backbone represents language, image/video, and robot-action tokens in a shared sequence and reconstructs masked target spans. By changing the visible context, objective token, and masked target, the same model supports Policy, World Modeling, Goal-State Prediction, and Task Understanding, generating action chunks, next visual states, goal states, or task instructions.

future visual observations or latent scene representations into policy learning. These approaches emphasize different sources of supervision for linking language, visual changes, and control.

We examine how these modeling choices relate to policy behavior through a two-task diagnostic on VLABench [28], comparing $\pi_{0.5}$ [4] and Mimic-Video [24]. On SelectFruit, which involves target selection across object and layout variation, $\pi_{0.5}$ performs better across the evaluated instruction tracks. On InsertFlower, which involves grasping, alignment, and insertion, Mimic-Video is competitive particularly under the indirect Track 4 instructions. Replacing the instruction with a length-matched random string also produces a larger success-rate decrease for $\pi_{0.5}$ than for Mimic-Video. The two policies thus exhibit different task-performance profiles and different sensitivity to linguistic input. These observations motivate studying language conditioning and visual prediction together within a shared policy model.

Recent unified action models [29–34] combine action prediction with visual generation or trajectory understanding. Existing approaches offer several ways to connect predictions to control: VLM features can condition a dedicated action expert [3, 17], while unified prediction-and-understanding models combine visual information with action learning [35]. We study a trajectory formulation in which instructions, observations, goals, and actions are represented as variables that can serve as either context or prediction targets. This formulation allows the same model to support several conditional learning tasks and to reuse their predictions through multiple inference compositions.

We introduce Dynin-Robotics, a unified multimodal VLA foundation model built on Dynin-Omni [36], an omnimodal masked-diffusion model. Dynin-Omni performs understanding and generation through iterative token prediction with a shared discrete interface and a single bidirectional backbone. Dynin-Robotics extends this interface with quantized robot-action tokens and a reserved typed interface for aligned continuous sensor streams. As illustrated in Figure 1, language instructions, visual observations, goal states, and robot actions occupy typed spans within a single trajectory sequence, with optional spans for sensor context.

Objective-conditioned masking specifies which trajectory spans remain visible and which the shared Transformer and prediction head reconstruct. POLICY predicts action chunks from observations and instructions; WORLD MODELING predicts the next visual observation conditioned on the current context and action; TASK UNDERSTANDING reconstructs an instruction from trajectory frames; and GOAL-STATE PREDICTION predicts a terminal visual state corresponding to the instructed goal. Training these conditional queries within one representation connects actions with task language, local visual transitions, and terminal outcomes.

The learned interfaces also support inference composition. As summarized in Figure 5, Dynin-Robotics

performs action-only decoding, predicts a goal before policy decoding, jointly denoises action and future-state spans, or uses WORLD MODELING to score and rerank action candidates. Goal guidance can be combined with joint denoising or candidate filtering. These modes reuse the same model for policy, goal, and world-model queries. They support test-time scaling by allocating additional inference computation to visual prediction and candidate evaluation. Our experiments examine which compositions improve action-only performance and the throughput costs of those choices.

Block-parallel masked decoding reduces the sequential work required to generate an action chunk. At each denoising step, Dynin-Robotics predicts masked action positions in parallel and progressively commits tokens. Combined with dInfer [37], an optimized inference stack for parallel decoding and context reuse, this improves model-side action-decoding throughput by up to $29.2\times$ under the reported profiling setup.

Our evaluation connects the training and inference design to downstream performance. On two VLABench tasks, robot pretraining improves adaptation within the same Stage-2 step budget, and the full objective mixture improves shifted-instruction success over Policy-only post-training under a fixed coupled decoder. With the checkpoint held fixed, selected inference compositions improve success over action-only decoding. We also evaluate the Stage-1 checkpoint on DROID [38] for action-conditioned visual prediction, goal-state generation, action prediction, and qualitative trajectory-to-instruction generation, without additional DROID-specific post-training. On policy benchmarks, Dynin-Robotics achieves a 98.1% average success rate on LIBERO [39], 73.0% on zero-shot LIBERO-Plus [40], and a 78.4% average across four manipulation conditions on a Franka Research 3 platform.

Our main contributions are:

- **Unified robot trajectory modeling with masked diffusion.** We extend an omnimodal masked-diffusion backbone with robot-action tokens and formulate POLICY, WORLD MODELING, TASK UNDERSTANDING, and GOAL-STATE PREDICTION as conditional denoising queries over a shared trajectory representation.
- **Compositional and accelerated inference from a single model.** The shared interface supports test-time scaling through goal prediction and action-candidate evaluation, with predicted visual states participating in action generation and selection. Block-parallel decoding and context reuse reduce sequential action-generation cost.
- **Diagnostics and controlled evaluation of unified VLA modeling.** A two-task VLABench diagnostic identifies differences in task performance and instruction sensitivity between $\pi_{0.5}$ and Mimic-Video. Within-model ablations evaluate robot pretraining, objective supervision, and inference composition, complemented by policy and multimodal-output evaluations in simulation and on a physical robot.

2 Related Work

2.1 Large Diffusion Language Models

Most contemporary large language models [41–44] generate text autoregressively through left-to-right next-token prediction. Standard autoregressive decoding commits tokens in that order, with each prediction conditioned on the preceding sequence. Studies such as the Reversal Curse [45] have also documented order-sensitive generalization behavior in these models. For robot trajectories, flexible generation order offers an alternative way to condition and predict related visual and action variables.

Diffusion models learn to reverse a data-corruption process [46, 47]. Extensions to categorical variables use discrete transition processes [48, 49], allowing this formulation to operate over token vocabularies.

Early diffusion language models applied continuous diffusion to token embeddings for controllable or sequence-to-sequence generation [50, 51]. Later methods model diffusion directly over discrete vocabularies. Diffusion-BERT [52] uses an absorbing-state process based on token masking, while SEDD [53] learns the ratios needed to reverse a discrete diffusion process.

Masked diffusion uses [MASK] as an absorbing noise state: the forward process replaces clean tokens with masks, and the learned reverse process reconstructs them from visible context. Each denoising step predicts

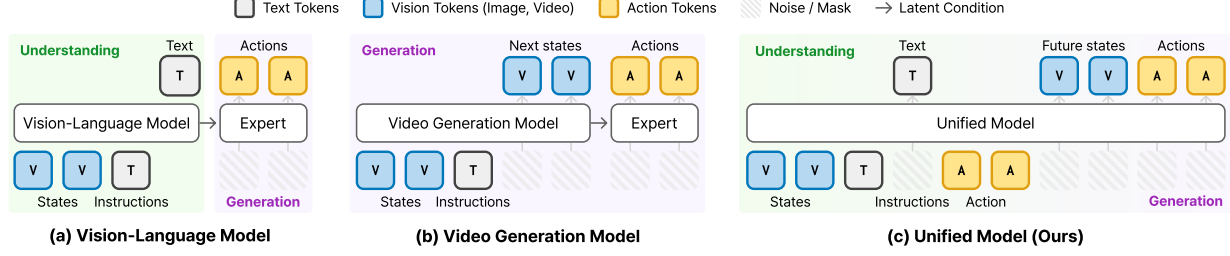


Figure 2 Representative architectures for connecting perception, prediction, and robot actions. (a) A VLM conditions a dedicated action expert, a common design in recent VLM-based VLAs; other VLM-based policies generate actions directly. (b) A video generator conditions an action expert, illustrating one architecture for video-based control. (c) Dynin-Robotics represents language, visual states, and actions within a shared masked-diffusion model, with objective-specific choices of visible context and prediction targets. Hatched spans denote masked or noisy targets, and arrows denote latent conditioning.

the currently masked positions in parallel. Depending on the decoding schedule, uncertain predictions can be remasked and revised, allowing iterative refinement under bidirectional context.

MDLM [54] develops a masked-diffusion language-modeling objective, and large-scale models including LLaDA [55] and Dream [56] apply masked diffusion to language understanding, reasoning, and instruction following. These models provide a token-generation interface that supports parallel prediction and flexible refinement schedules.

Quantizing visual observations and continuous controls extends this interface to robot trajectories. An action chunk can be predicted and refined using context across its dimensions and temporal positions. Recent policies apply masked diffusion to action decoding [16, 57] or jointly model actions with visual states and multimodal instructions [31, 34]. Our formulation uses the shared discrete sequence to express robot control, visual prediction, and trajectory understanding as conditional denoising queries.

2.2 Unified Multimodal Models

Multimodal understanding and generation can share a language backbone while using different output interfaces. DreamLLM [58] and SEED-X [59] connect multimodal language models with visual generators, while NeXT-GPT [60], CoDi-2 [61], and HyperCLOVAX-Omni [62] extend generation to additional visual and speech modalities. These designs combine shared language processing with modality-specific generation components.

Another line of work represents multiple modalities in a shared token space and models them using a common generative backbone. AnyGPT [63], Chameleon [64], Emu3 [65], and Janus-Pro [66] extend autoregressive language modeling to multimodal token sequences. Beyond autoregressive modeling, BAGEL [67] and Show-o2 [68] combine language reasoning with diffusion- or flow-based visual generation. In parallel, UniDisc [69], MMaDA [70], Fudoki [71], LaViDa-O [72], and Lumina-DiMOO [73] explore discrete diffusion [54, 55] over shared multimodal tokens. More recently, LLaDA2.0-Uni [74] combines a discrete diffusion language-model backbone with a diffusion decoder to support multimodal understanding, generation, and editing.

The shared-token approach provides a common representation for multimodal conditioning and prediction. Masked-diffusion models use visible tokens to condition target spans and refine those targets iteratively. Dynin-Omni [36] applies this approach to text, vision, and speech within a shared masked-denoising process. Dynin-Robotics extends the interface to robot trajectories, treating instructions, visual states, goals, and actions as context or targets for complementary prediction tasks.

2.3 Modeling Paradigms in Robot Learning

Robot-learning approaches differ in how they use language-vision representations, visual prediction, and action generation. Figure 2 illustrates representative VLM-based, video-based, and unified architectures. The

Table 1 Reported capabilities of robot-policy models. Modeling summarizes the architectural or generative formulation associated with each cited configuration. NSP and MoT denote next-scale prediction and Mixture-of-Transformers, respectively. Checkmarks and crosses indicate capabilities reported or not reported for the cited configuration, and an asterisk denotes configuration-dependent video input. Task Understanding follows the capability described in each source and encompasses a broader range of language-understanding tasks than Dynin-Robotics’ trajectory-to-instruction objective.

Model	Modeling	Video input	Goal-state prediction	World model	Task understanding	Goal-state guided policy	Candidate reranking
<i>VLM-Based Policies</i>							
RT-2 [1]	AR	✓*	✗	✗	✓	✗	✗
OpenVLA [2]	AR	✗	✗	✗	✗	✗	✗
OpenVLA-OFT [17]	AR	✓*	✗	✗	✗	✗	✗
SpatialVLA [13]	AR	✗	✗	✗	✗	✗	✗
CoT-VLA [14]	AR	✗	✓	✗	✗	✓	✗
CogVLA [19]	AR	✗	✗	✗	✗	✗	✗
π_0 [3]	Flow	✗	✗	✗	✗	✗	✗
π_0 -FAST [15]	AR	✗	✗	✗	✗	✗	✗
$\pi_{0.5}$ [4]	Flow	✗	✗	✗	✓	✗	✗
GR00T-N1, N1.6 [5]	Flow	✗	✗	✗	✗	✗	✗
F1 [75]	MoT + NSP	✗	✓	✗	✗	✓	✗
InternVLA-M1 [76]	AR	✗	✗	✗	✗	✗	✗
Discrete Diffusion VLA [16]	Masked Diff.	✗	✗	✗	✗	✗	✗
LLaDA-VLA [57]	Masked Diff.	✗	✗	✗	✗	✗	✗
X-VLA [6]	Flow	✗	✗	✗	✗	✗	✗
ABot-M0 [18]	DiT	✗	✗	✗	✗	✗	✗
NORA [77]	AR	✗	✗	✗	✗	✗	✗
RIPT-VLA [78]	AR	✗	✗	✗	✗	✗	✗
<i>World-/Video-Model-Based Policies</i>							
Video Policy [25]	Cont. Diff.	✗	✓	✗	✗	✓	✗
Cosmos Policy [22]	Cont. Diff.	✗	✓	✓	✗	✗	✓
Mimic-Video [24]	Flow	✗	✓	✗	✗	✓	✗
LingBot-VA [26]	AR Diff.	✓	✓	✓	✗	✓	✗
Fast-WAM [27]	Cont. Diff.	✗	✓	✗	✗	✗	✗
WorldVLA [30]	AR	✗	✓	✓	✗	✗	✗
RynnVLA-002 [33]	AR	✓	✓	✓	✗	✗	✗
UWM [23]	Cont. Diff.	✗	✓	✓	✗	✗	✗
<i>Unified Policies</i>							
UniVLA [29]	AR	✓	✓	✗	✗	✗	✗
UP-VLA [35]	AR	✗	✓	✗	✗	✗	✗
UD-VLA [31]	Masked Diff.	✗	✓	✗	✗	✓	✗
MMaDA-VLA [34]	Masked Diff.	✗	✓	✗	✗	✓	✗
dVLA [32]	Masked Diff.	✗	✗	✗	✗	✗	✗
Dynin-Robotics (Ours)	Masked Diff.	✓	✓	✓	✓	✓	✓

categories describe modeling emphasis, with several methods combining elements of more than one. Table 1 summarizes the cited configurations in terms of video context, visual prediction, task understanding, and inference-time action guidance.

VLM-based policies. Vision-language-model-based (VLM-based) policies adapt language–vision representations to robot control. RT-2 [1] and OpenVLA [2] represent actions as language-like tokens, whereas π_0 [3] and $\pi_{0.5}$ [4] use flow-matching action experts for continuous control. The expert-based architecture in Figure 2(a) is widely used in recent VLM-based VLAs; direct action-token generation provides another output interface. Other works improve spatial grounding, intermediate reasoning, adaptation efficiency, or preservation of pretrained VLM capabilities [13, 14, 17, 19]. Discrete Diffusion VLA [16] and LLaDA-VLA [57] apply discrete diffusion to action decoding, introducing parallel token refinement within VLM-based policies.

VGM-based policies. Video-generative-model-based (VGM-based) policies and related world-action models use visual prediction in policy learning. Cosmos Policy [22] and Mimic-Video [24] adapt pretrained video models for control, while LingBot-VA [26] jointly models frame prediction and policy execution. Fast-WAM [27] examines the roles of test-time future imagination and video co-training. The video-generator-plus-expert design in Figure 2(b) illustrates one way to connect visual generation to actions; the cited methods use different mechanisms to couple the two outputs.

Coupled and unified policies. Unified approaches combine action learning with visual prediction or understanding. WorldVLA [30], RynnVLA-002 [33], and UWM [23] integrate action prediction and world modeling, linking the world-/video-model-based and unified approaches summarized in Table 1. UD-VLA [31] and MMaDA-VLA [34] jointly denoise visual states and actions. UP-VLA [35] combines multimodal understanding, future visual prediction, and action learning, while dVLA [32] incorporates multimodal chain-of-thought prediction into action modeling.

These approaches connect visual prediction to action learning through different conditioning structures. In our formulation, action-conditioned WORLD MODELING predicts a next observation given an action, while GOAL-STATE PREDICTION predicts a terminal visual target from an observation and instruction. Dynin-Robotics combines these queries with POLICY and trajectory-to-instruction TASK UNDERSTANDING within one masked-diffusion backbone. Their shared interface supports goal conditioning, joint action–next-state denoising, and action-candidate evaluation through changes to the visible context and prediction targets.

3 Dynin-Robotics

Dynin-Robotics formulates robot learning as conditional masked denoising over shared trajectories containing visual observations, language instructions, robot actions, and optional next-state images, goal images, sensor streams, and metadata. An objective token specifies the visible context and target spans. Policy, World Modeling, Task Understanding, and Goal-State Prediction use different conditioning configurations of this representation. The same interface supports the six inference compositions in Section 3.4, including the default two-stage path that predicts a goal-state context before denoising an action chunk.

3.1 Backbone

Dynin-Robotics builds on Dynin-Omni [36], an omnimodal masked-diffusion model. Dynin-Omni represents text, images, sampled video frames, and speech as discrete tokens and processes their interleaved sequences with a single bidirectional Transformer, shared token embeddings, and a unified categorical prediction head. This architecture supports multimodal understanding and generation through the same backbone.

Each inherited modality uses a pretrained tokenizer: a language tokenizer for text, a shared MAGViT-v2 tokenizer [79] for images and sampled video frames, and the EMOVA-Speech-Tokenizer [80] for speech. The shared visual vocabulary represents both observed and generated images. The inherited text and visual tokenizers and their corresponding detokenizers remain frozen during robot training.

Dynin-Omni is trained to reconstruct masked tokens from visible multimodal context [36, 54, 55]. Changing the visible and target spans allows a modality to condition another prediction or become a generation target itself. Bidirectional attention conditions masked positions on the visible sequence, and iterative denoising reconstructs the targets. For robot trajectories, this interface supports predictions in several conditional directions, including actions from observations and instructions, visual outcomes from actions, and instructions from trajectory frames.

As illustrated in Figure 3, we retain the Dynin-Omni Transformer and text–visual tokenization interface and extend its discrete token space to robot actions. Speech tokens remain part of the inherited checkpoint, but speech-conditioned objectives are inactive in the reported robotics runs. The following sections define the robot token space and objective-specific conditioning layouts.

3.2 Robotic Token Space Extension

For notation, let $\mathcal{V}_{\text{text}}^+$ include lexical text tokens together with the objective, delimiter, padding, termination, and [MASK] control tokens allocated in the text-tokenizer range, and let M_S denote the number of sensor-type token blocks reserved by the configuration. The integrated Dynin-Robotics token space is

$$\mathcal{V}_{\text{DR}} = \mathcal{V}_{\text{text}}^+ \uplus \mathcal{V}_{\text{vision}} \uplus \mathcal{V}_{\text{speech}} \uplus \mathcal{V}_{\text{action}} \uplus \mathcal{V}_{\text{sensor}}, \quad \mathcal{V}_{\text{sensor}} = \biguplus_{m=1}^{M_S} \mathcal{V}_{S_m}. \quad (1)$$

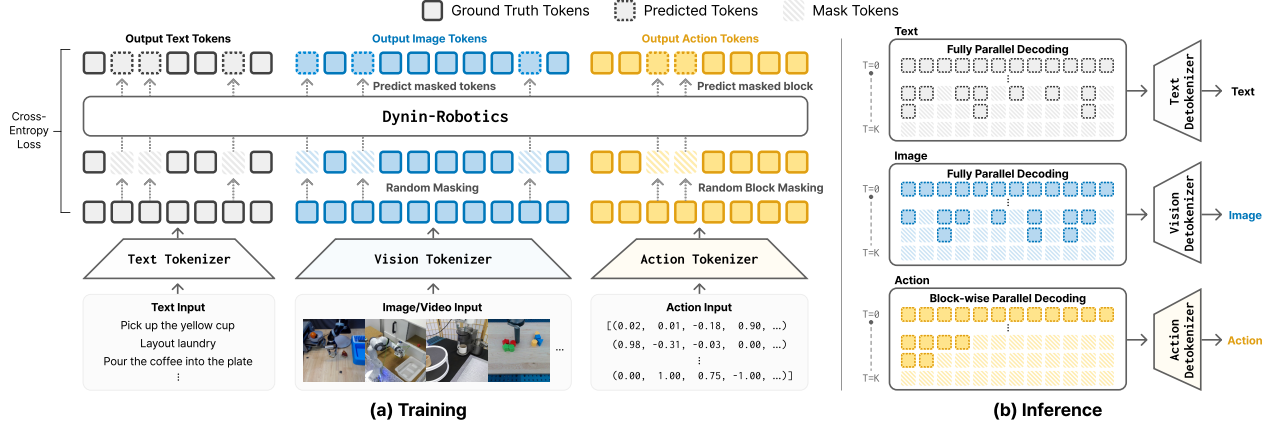


Figure 3 Shared masked-diffusion backbone and modality-specific token interfaces. (a) Text and visual observations are tokenized, and continuous actions are discretized through uniform quantization. For each training objective, the backbone reconstructs masked positions within the designated target spans while retaining the conditioning context. Text and visual targets use random masking, and action targets use block masking. (b) Inference iteratively refines masked targets, with parallel prediction at each denoising step and block-wise commitment for actions. Inverse quantization maps the resulting action tokens to continuous controls.

where $\uplus$ denotes non-overlapping token-ID blocks. The inherited speech block is present in the checkpoint but inactive in the reported robotics runs. The sets $\mathcal{V}_{\text{vision}}$ and $\mathcal{V}_{\text{action}}$ contain image/video-frame and robot-action tokens, respectively; when continuous sensor streams are configured, $\mathcal{V}_{S_m}$ denotes a reserved block for sensor type m .

For objective o , a robot example follows the typed serialization template

$$\mathbf{x}^{(o)} = [o; \mathbf{x}^V; \mathbf{x}^T; \mathbf{x}^A; \mathbf{x}_{t+1}^V; \mathbf{x}_g^V; \mathbf{x}^S; \mathbf{x}^M], \quad (2)$$

where $\mathbf{x}^V$ denotes observation or trajectory-frame tokens, $\mathbf{x}^T$ denotes instruction or textual target tokens, $\mathbf{x}^A$ denotes action tokens, $\mathbf{x}_{t+1}^V$ denotes optional immediately subsequent visual-state tokens, $\mathbf{x}_g^V$ denotes optional terminal or successful goal-state tokens, $\mathbf{x}^S$ denotes one or more optional typed sensor spans, and $\mathbf{x}^M$ denotes optional textual metadata encoded and delimited within $\mathcal{V}_{\text{text}}^+$. The active spans, visible context, targets, and masking pattern are selected for each objective. Next-state and goal-state spans are included when used by that objective or inference composition, as illustrated in the objective-layout figure below.

3.2.1 Actions

For an action chunk $\mathbf{a}_{t:t+H-1} \in \mathbb{R}^{H \times d_a}$, the reported Dynin-Robotics model uses fixed, deterministic, per-dimension uniform binning; no learned action tokenizer is used. Source-specific preprocessing first maps native controls to the common action interface described in Section 4.1. For channel j , robust lower and upper statistics ℓ_j^A and u_j^A are computed from the corresponding training data. The Stage-1 OXE preprocessing uses the per-source 1st and 99th percentiles for the six motion channels. Let $\epsilon > 0$ be a numerical floor for degenerate normalization ranges, and let v_{start}^A denote the first global token ID of the reserved action-token block. The action is normalized, quantized into B_A active bins, and mapped into the reserved action-token block as

$$\begin{aligned} \bar{a}_{t+h,j} &= \text{clip} \left(2 \frac{a_{t+h,j} - \ell_j^A}{\max(u_j^A - \ell_j^A, \epsilon)} - 1, -1, 1 \right), \\ b_{t+h,j}^A &= \text{clip} \left(\left\lfloor \frac{B_A(\bar{a}_{t+h,j} + 1)}{2} \right\rfloor, 0, B_A - 1 \right), \\ x_{t+h,j}^A &= v_{\text{start}}^A + b_{t+h,j}^A. \end{aligned} \quad (3)$$

Here, $h \in \{0, \dots, H-1\}$ and $j \in \{1, \dots, d_a\}$. The Hd_a action tokens are serialized in time-major order. The six motion channels in the reported 7-DoF interface use source- or benchmark-specific robust normalization, while the gripper is retained as an absolute channel rather than accumulated as a delta. Constant or unused motion dimensions are set to zero before tokenization.

After denoising, each predicted bin is mapped to its normalized bin center and then returned to continuous action space:

$$\begin{aligned}\hat{a}_{t+h,j} &= 2 \frac{\hat{b}_{t+h,j}^A + \frac{1}{2}}{B_A} - 1, \\ \hat{a}_{t+h,j} &= \ell_j^A + \frac{\hat{a}_{t+h,j} + 1}{2} (u_j^A - \ell_j^A).\end{aligned}$$

The released Stage-1 checkpoint uses $B_A = 256$, whereas Stage 2 selects a task-specific active bin count within the same reserved action block; Section 5.7 evaluates this choice. All reported experiments use uniform binning.

3.2.2 Optional Sensor Token Interface

The same deterministic discretization interface can extend to aligned continuous robot-sensor streams. For sensor type m with a window $\mathbf{s}^{(m)} \in \mathbb{R}^{H_m \times d_m}$, preprocessing synchronizes or resamples the stream to the observation timestamps before tokenization. Every channel is then calibrated with configuration-specific robust lower and upper statistics $\ell_j^{(m)}$ and $u_j^{(m)}$, computed from the corresponding training split; the optional force/torque preprocessing path uses channel-wise 1st and 99th percentiles:

$$\bar{s}_{t+h,j}^{(m)} = \text{clip} \left(2 \frac{s_{t+h,j}^{(m)} - \ell_j^{(m)}}{\max(u_j^{(m)} - \ell_j^{(m)}, \epsilon)} - 1, -1, 1 \right). \quad (4)$$

Here, $h \in \{0, \dots, H_m - 1\}$. The normalized value is uniformly quantized into B_{S_m} bins and mapped to the non-overlapping reserved token block $\mathcal{V}_{S_m}$, whose first global token ID is denoted by $v_{\text{start}}^{S_m}$:

$$\begin{aligned}b_{t+h,j}^{(m)} &= \text{clip} \left(\left\lfloor \frac{B_{S_m} (\bar{s}_{t+h,j}^{(m)} + 1)}{2} \right\rfloor, 0, B_{S_m} - 1 \right), \\ x_{t+h,j}^{S_m} &= v_{\text{start}}^{S_m} + b_{t+h,j}^{(m)}.\end{aligned} \quad (5)$$

Within each typed sensor span, tokens are serialized in time-major order over the $H_m d_m$ values. Sensor-specific channel statistics absorb differences in unit, range, and scale, while H_m , d_m , the typed delimiters, and the fixed channel order encode temporal extent, dimensionality, and sensor identity. Multiple streams can be represented by concatenating non-overlapping typed visible spans without changing the masked-diffusion objective; when a stream is unavailable, its optional span is omitted. This interface can accommodate calibrated vector-valued signals such as force/torque, proprioception, and tactile measurements.

The sensor interface reserves capacity for optional sensor conditioning. Sensor-token training is inactive in the released 220k Stage-1 checkpoint, and the reported Stage-2 experiments use no sensor-conditioned policy decoding. Sensor detokenization and sensor-generation objectives are also inactive in the reported setup.

3.3 Unified Objective Training

Dynin-Robotics constructs objective-specific examples from a common corpus of robot trajectories. Let $\mathcal{O} = \{\mathcal{O}_{\text{PO}}, \mathcal{O}_{\text{WM}}, \mathcal{O}_{\text{TU}}, \mathcal{O}_{\text{GP}}\}$ denote Policy, World Modeling, Task Understanding, and Goal-State Prediction. For objective o , let $\mathcal{C}_o$ be the visible conditioning positions and $\mathcal{I}_o$ the target positions in $\mathbf{x}^{(o)}$. The model learns the conditional target distribution

$$p_\theta(\mathbf{x}_{\mathcal{I}_o}^{(o)} \mid \mathbf{x}_{\mathcal{C}_o}^{(o)}, o)$$

using the objective-specific prompt layouts in Figure 4.

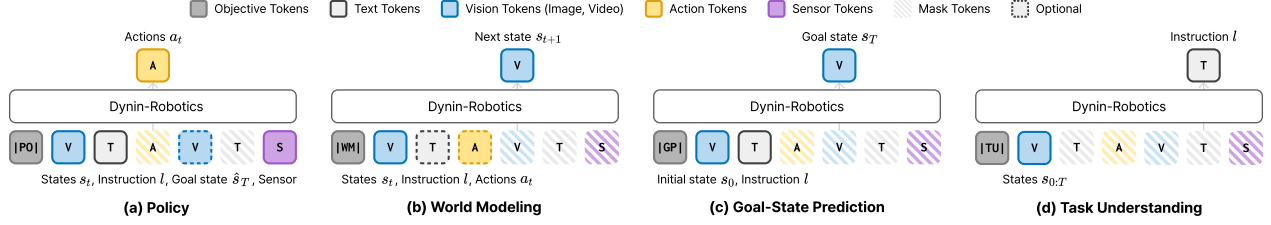


Figure 4 Objective-specific observed-target layouts used in unified training. (a) Policy predicts action tokens from states, an instruction, and optional goal-state and sensor context. (b) World Modeling predicts next-state visual tokens conditioned on states, the instruction, and actions. (c) Goal-State Prediction predicts a terminal goal state from the initial state and instruction. (d) Task Understanding reconstructs the task instruction from trajectory states. Solid tokens are visible context, hatched tokens are masked prediction targets, and dotted spans are optional.

The Policy objective predicts an action chunk from the current visual state and instruction, with goal-state and sensor context included when those spans are available. Goal-State Prediction predicts a terminal visual state corresponding to the instructed goal from the initial observation and instruction. World Modeling predicts the immediately subsequent visual state s_{t+1} conditioned on the current visual context s_t , instruction, and corresponding action chunk. Throughout this work, World Modeling denotes this action-conditioned next-state visual-token prediction task. Task Understanding reconstructs the instruction from sampled trajectory frames. The objectives thus associate action chunks with local visual transitions, terminal outcomes, and task descriptions within the same trajectory representation.

For each sampled objective, corruption is applied only inside the selected target span or spans, while conditioning tokens remain visible. Text and visual targets use random token masking, whereas action targets use contiguous block masking over the time-major action sequence. For an objective- o micro-batch, let N_o denote the number of examples. For example b , let $\mathcal{I}_{o,b}$ denote its target positions and $\tilde{\mathbf{x}}_b^{(o)}$ the corrupted input sequence after target-only masking. The sampled mask probability ρ_b masks at least one target token in example b ; unmasked target tokens are copied into the input and excluded from supervision. Let $L_{o,b} = |\mathcal{I}_{o,b}|$ be the full target-span length and

$$\mathcal{M}_{o,b} = \{i \in \mathcal{I}_{o,b} : \tilde{x}_{b,i}^{(o)} = [\text{MASK}]\}$$

be the masked target positions. Before cross-entropy is computed, the unified prediction head is sliced to the vocabulary of the target modality. The target-only masked-diffusion loss is

$$\mathcal{L}_{\text{MDM}}(o) = \mathbb{E} \left[-\frac{1}{N_o} \sum_{b=1}^{N_o} \frac{1}{\rho_b L_{o,b}} \times \sum_{i \in \mathcal{M}_{o,b}} \log p_{\theta}^{(v_o)}(x_{b,i}^{(o)} | \tilde{\mathbf{x}}_b^{(o)}, o) \right], \quad (6)$$

where v_o denotes the target-modality vocabulary. Inverse mask-probability weighting makes the estimator comparable across sampled mask rates, while normalization by the full target length prevents examples with more masked positions from receiving systematically larger weight. For the Stage-1 Policy objective, let N_{PO} denote the number of Policy examples in the micro-batch, and define the normalized center of action bin r as $\gamma_r = 2(r + \frac{1}{2})/B_A - 1$. The expected normalized action and Policy-only auxiliary are

$$\begin{aligned} \hat{a}_{b,i} &= \sum_{r=0}^{B_A-1} p_{\theta,i}^{(A)}(r | \tilde{\mathbf{x}}_b^{(\text{PO})}, o_{\text{PO}}) \gamma_r, \\ \mathcal{L}_{\text{bin}} &= \frac{1}{N_{\text{PO}}} \sum_{b=1}^{N_{\text{PO}}} \frac{1}{|\mathcal{M}_b^A|} \sum_{i \in \mathcal{M}_b^A} |\hat{a}_{b,i} - \bar{a}_{b,i}|. \end{aligned} \quad (7)$$

where $\mathcal{M}_b^A$ contains the masked, unpadded action positions and $\bar{a}_{b,i}$ is the corresponding flattened normalized action target. Section 4.2 specifies the Stage-1 coefficient applied to this Policy-only auxiliary.

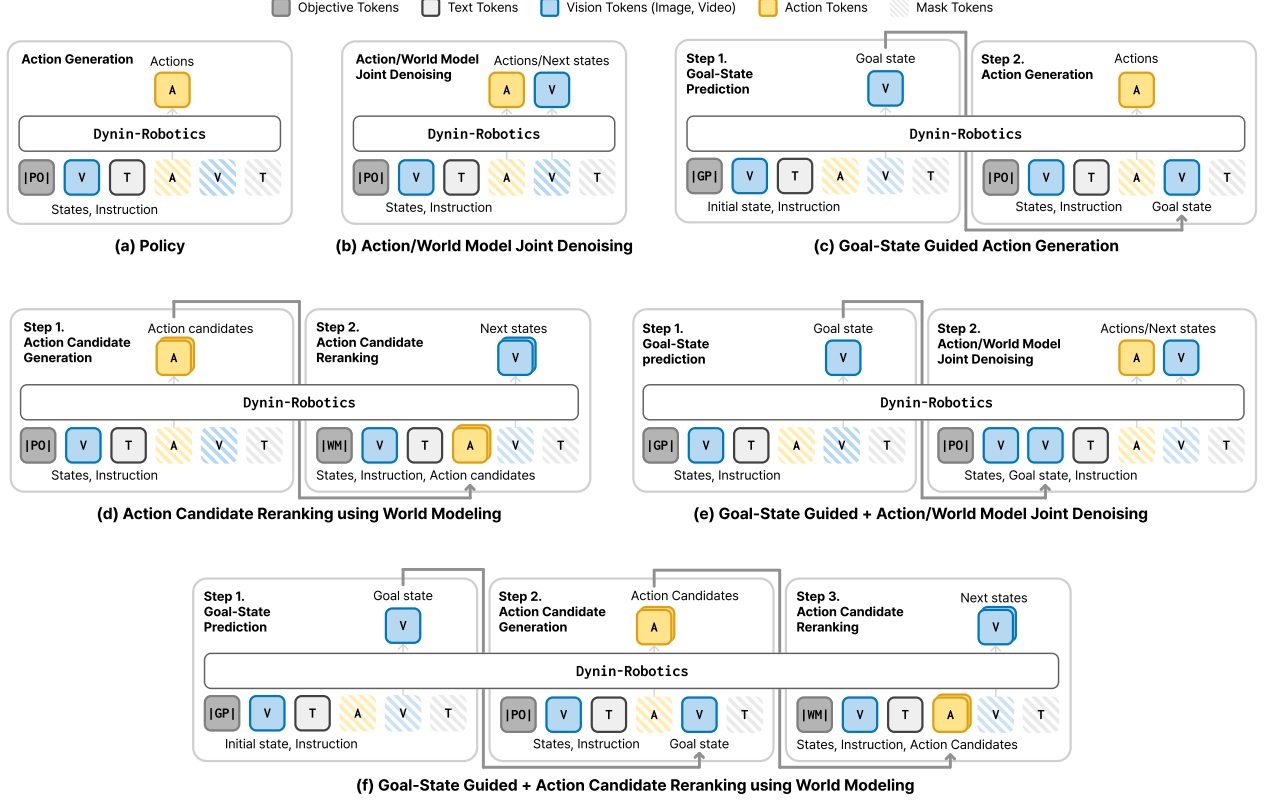


Figure 5 Six inference compositions from the shared denoising interface: (a) action-only Policy; (b) joint action/next-state denoising; (c) the default two-stage goal-guided Policy; (d) world-model action-candidate reranking; (e) goal-guided joint denoising; and (f) goal-guided candidate reranking. Hatched spans are denoising targets, and solid spans are visible context.

At training micro-step n , let $\mathcal{A}_n$ denote the scheduled objective multiset and let $\mathcal{L}_{\text{MDM}}^{(n)}(o)$ denote the corresponding micro-batch estimate. The common objective-mixture loss is

$$\mathcal{L}_{\text{mix}}^{(n)} = \frac{1}{|\mathcal{A}_n|} \sum_{o \in \mathcal{A}_n} \lambda_o \mathcal{L}_{\text{MDM}}^{(n)}(o). \quad (8)$$

Here, repeated entries in $\mathcal{A}_n$ determine objective exposure, whereas λ_o controls loss scaling. The Stage-1 and Stage-2 objectives in Section 4 specialize this expression with their respective schedules, weights, and auxiliary losses.

Training proceeds in two stages. Stage 1 performs unified continual pretraining on large-scale multimodal robot trajectories. Stage 2 post-trains the same token interface and masked-denoising objective on benchmark-specific robot data. Dataset mixtures, action-tokenization settings, objective schedules, and optimization details are reported in Section 4.

3.4 Unified Multi-Stage Inference

Figure 5 summarizes the six inference compositions. For individual objective queries, Dynin-Robotics uses the same objective-conditioned prompts at inference as in training; the joint modes combine trained target spans through an explicitly identified inference-time composite mask. Given visible context and an objective token, each target span is initialized with [MASK] tokens and decoded by reverse masked diffusion. We use $k \in \{K, \dots, 0\}$ exclusively for the denoising index: $k = K$ denotes a fully masked target and $k = 0$ a fully decoded target. We use ℓ to index the language instruction, so $\mathbf{x}_\ell^T$ denotes its text-token span. At each step,

the model predicts all currently masked target positions, commits high-confidence tokens, and refines the remaining low-confidence positions. Visual targets are decoded in parallel, whereas action targets are refined in contiguous blocks over the time-major action sequence.

Action-only and goal-state-guided action decoding. The action-only policy mode denoises an action chunk from the current observation and instruction:

$$\hat{\mathbf{x}}_{t:t+H-1}^{A,(0)} \sim p_{\theta}(\mathbf{x}_{t:t+H-1}^A \mid \mathbf{x}_t^V, \mathbf{x}_{\ell}^T, o_{\text{PO}}).$$

When goal-state guidance is enabled, Dynin-Robotics first predicts a goal-state context from the initial observation and language instruction:

$$\hat{\mathbf{x}}_g^V \sim p_{\theta}(\mathbf{x}_g^V \mid \mathbf{x}_0^V, \mathbf{x}_{\ell}^T, o_{\text{GP}}), \quad (9)$$

and then provides the predicted goal-state tokens as visible context for Policy decoding:

$$\hat{\mathbf{x}}_{t:t+H-1}^{A,(0)} \sim p_{\theta}(\mathbf{x}_{t:t+H-1}^A \mid \mathbf{x}_t^V, \mathbf{x}_{\ell}^T, \hat{\mathbf{x}}_g^V, o_{\text{PO}}). \quad (10)$$

In either mode, the generated action tokens are mapped from their predicted bins to continuous controls using the inverse quantization defined in Section 3.2.1, and are executed with receding-horizon replanning. The goal-state-guided two-stage path is the default inference configuration, whereas action-only decoding is retained as an ablation.

Joint action-world-model denoising. The shared token interface also supports an inference-time composite mask in which the action span and action-conditioned next-state visual span are initialized together and iteratively refined under the same visible context. Let $\mathcal{I}_{\text{PO}} \equiv \mathcal{I}_{o_{\text{PO}}}$ and $\mathcal{I}_{\text{WM}} \equiv \mathcal{I}_{o_{\text{WM}}}$ denote the Policy and World Modeling target positions, respectively, within this composite layout. With $\mathcal{I}_{\text{J}} = \mathcal{I}_{\text{PO}} \cup \mathcal{I}_{\text{WM}}$, let $\tilde{\mathbf{x}}_{\text{J}}^{(K)}$ denote the inference-only composite sequence containing the shared visible context, the trained Policy and World Modeling objective markers, and fully masked action and next-state target spans. The joint decoding is

$$\left(\hat{\mathbf{x}}_{t:t+H-1}^{A,(0)}, \hat{\mathbf{x}}_{t+1}^{V,(0)} \right) = \text{Decode}_{\theta} \left(\tilde{\mathbf{x}}_{\text{J}}^{(K)}; \mathcal{I}_{\text{PO}}, \mathcal{I}_{\text{WM}} \right). \quad (11)$$

The optional goal-state span $\hat{\mathbf{x}}_g^V$ is included in the visible context of $\tilde{\mathbf{x}}_{\text{J}}^{(K)}$ only in the goal-guided variant. This inference-only composite reuses the trained objective markers, prediction head, and model parameters within one query. At each denoising step, the shared head is restricted to the action vocabulary on $\mathcal{I}_{\text{PO}}$ and to the visual vocabulary on $\mathcal{I}_{\text{WM}}$.

World-model-based candidate reranking. Dynin-Robotics uses the World Modeling objective to evaluate candidate actions at inference. Starting from the action-only or goal-guided Policy output $\hat{\mathbf{x}}^{A,(0)}$, we form a local candidate set $\mathcal{C}$ by replacing low-confidence action tokens with high-probability alternatives under the original Policy logits. The shared model scores these candidates, and an acceptance rule determines whether to replace the original output.

Before scoring, we construct one reference visual-token span $\mathbf{y}$ and hold it fixed across all candidates. In the goal-guided filtering mode, $\mathbf{y} = \hat{\mathbf{x}}_g^V$. In a next-state filtering mode, $\mathbf{y}$ is predicted once from the base Policy output $\mathbf{c}^0$ and then reused as the reference for every candidate. The World Modeling energy is the mean negative log-likelihood assigned to this common reference:

$$E_{\text{WM}}(\mathbf{c}) = -\frac{1}{|\mathcal{Y}|} \sum_{j \in \mathcal{Y}} \log p_{\theta}(y_j \mid \mathbf{x}_t^V, \mathbf{x}_{\ell}^T, \mathbf{c}, o_{\text{WM}}), \quad (12)$$

where $\mathcal{Y}$ is the common set of reference visual-target positions. Every candidate is scored against the same reference $\mathbf{y}$.

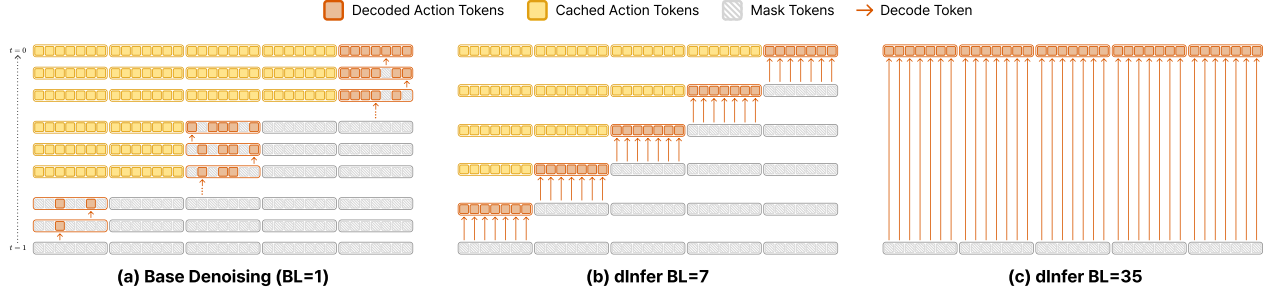


Figure 6 Schematic token-commitment patterns for a five-step, 7-DoF action chunk serialized as 35 time-major tokens. Block lengths of 1, 7, and 35 correspond to individual tokens, one action timestep, and the full chunk, respectively. Orange tokens are newly committed, yellow tokens are retained from earlier steps, and hatched tokens remain masked. Confidence-based commitment determines which positions require further refinement. The schematics illustrate block granularity; realized backbone evaluations are reported in Table 14.

For action position i , let k_i denote the denoising step at which the candidate alternatives for that position are collected, and let $\tilde{\mathbf{x}}^{(\text{PO}, k_i)}$ denote the corresponding partially denoised Policy input. We define the frozen base-Policy distribution

$$\pi_i^{\text{base}}(v) = p_{\theta, i}^{(A)}\left(v \mid \tilde{\mathbf{x}}^{(\text{PO}, k_i)}, \mathbf{x}_t^V, \mathbf{x}_\ell^T, [\tilde{\mathbf{x}}_g^V], o_{\text{PO}}\right). \quad (13)$$

The distributions π_i^{base} are computed once and reused for all candidates. A Policy prior penalizes candidates with low probability under these distributions:

$$E_{\text{PO}}(\mathbf{c}) = -\frac{1}{L_A} \sum_{i=1}^{L_A} \log \pi_i^{\text{base}}(c_i), \quad (14)$$

$$J_{\text{rank}}(\mathbf{c}) = E_{\text{WM}}(\mathbf{c}) + \alpha_{\text{prior}} E_{\text{PO}}(\mathbf{c})$$

where $L_A = Hd_a$, $\alpha_{\text{prior}} \geq 0$ is the Policy-prior weight, and the bracketed goal-state context is optional. We select $\mathbf{c}^* = \arg \min_{\mathbf{c} \in \mathcal{C}} J_{\text{rank}}(\mathbf{c})$, but accept it only when it improves both the world-model energy by a margin δ and the combined ranking score:

$$\begin{aligned} E_{\text{WM}}(\mathbf{c}^0) - E_{\text{WM}}(\mathbf{c}^*) &\geq \delta, \\ J_{\text{rank}}(\mathbf{c}^0) - J_{\text{rank}}(\mathbf{c}^*) &> 0 \end{aligned} \quad (15)$$

where $\mathbf{c}^0 = \hat{\mathbf{x}}^{A, (0)}$ is the original Policy output. If either condition is not satisfied, $\mathbf{c}^0$ is executed. The World Modeling query therefore supplies an acceptance criterion for local alternatives to the Policy output. We evaluate this branch as an ablation.

These compositions support test-time scaling through additional prediction and evaluation queries to the shared model. Goal guidance allocates computation to a visual target before action decoding, joint denoising refines action and next-state spans together, and filtering allocates computation to candidate evaluation. Combining these operations provides several ways to trade inference computation for policy performance.

3.5 Acceleration

Dynin-Robotics predicts all masked action positions in parallel during each backbone evaluation. Confidence-based commitment determines which predictions become part of the decoded sequence and which positions receive further refinement. Figure 6 illustrates a five-step, 7-DoF chunk serialized as 35 time-major tokens. A block length of 7 spans one action timestep, while a block length of 35 spans the full chunk. The configured block length and the realized number of backbone evaluations are distinct quantities; Section 5.7.2 evaluates their latency and control trade-offs.

The accelerated implementation reuses computation for the fixed visual–language–goal context across denoising steps, following diffusion-LLM caching strategies [37, 81]. This reuse is approximate because changes to target tokens affect the bidirectional backbone. Candidate scoring can also be batched when actions share the

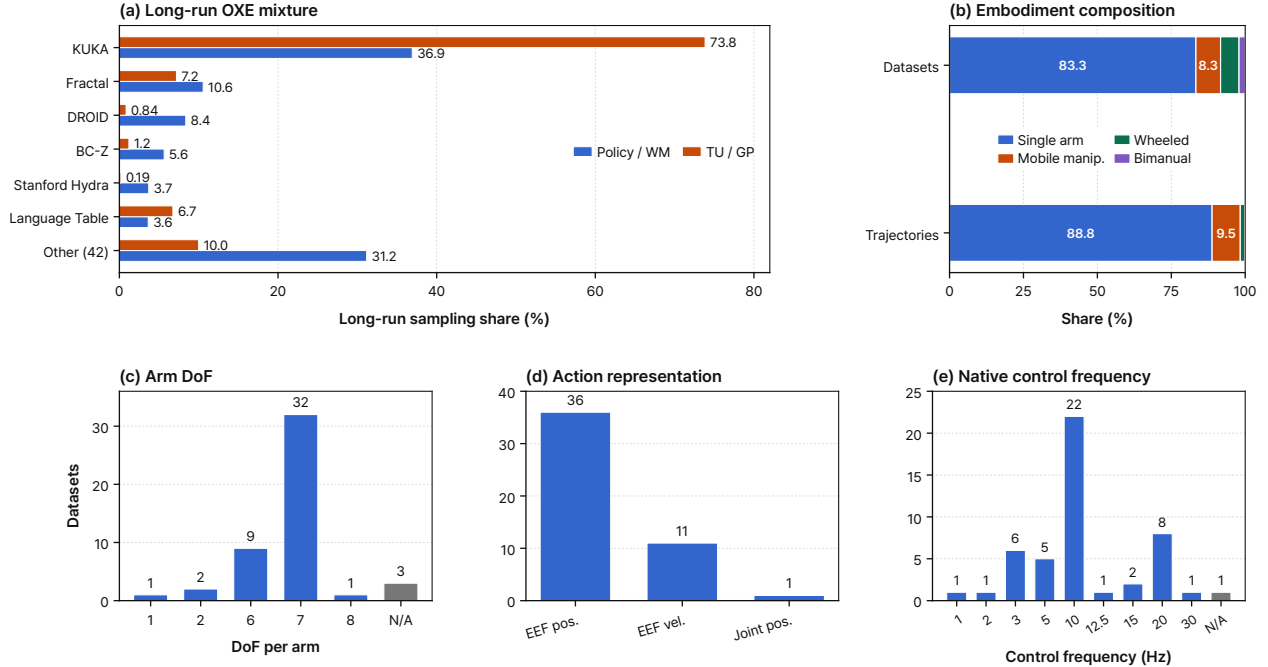


Figure 7 Composition of the 48-dataset Stage-1 Open X-Embodiment mixture. (a) Long-run sampling mass averaged across partitions and virtual epochs; Policy and World Modeling are weighted by transitions, whereas Task Understanding and Goal-State Prediction are weighted by trajectories. (b) Embodiment shares under dataset- and trajectory-count denominators. (c-e) Dataset counts by arm DoF, native action representation, and control frequency. Panels use different denominators; all sources map to the common 7D action interface.

observation, instruction, target-span length, and decoding schedule. The base and dInfer [37] decoders use confidence-based commitment schedules. The online reinforcement-learning stage in Section 4.3.3 replaces that schedule with a learned count-and-position remasking controller while keeping the shared token predictor and modality tokenizers frozen. These implementation choices target the latency of model-side action generation.

4 Training Recipe

4.1 Training Data

Continual-pretraining data. Our Stage-1 mixture contains 48 Open X-Embodiment (OXE) datasets [82], comprising 1,332,985 trajectories and 65,217,081 transitions in the runtime training split. The collection spans single-arm, mobile-manipulator, wheeled, and bimanual source embodiments, with heterogeneous kinematics and control rates (Figure 7). Before mixing, source-specific transforms map native controls to a common 7D end-effector interface containing 3D translation, 3D rotation, and gripper control. The six motion channels are normalized with per-source robust quantiles, the gripper remains an absolute channel, and constant or unused motion dimensions are set to zero before action tokenization. The active bimanual source is reduced to the same 7D action view. Source embodiment DoF describes the diversity of the collected data; the model’s action output uses the common 7D interface.

The optional force/torque preprocessing path implements the sensor interface described in Section 3.2.2. The released 220k Stage-1 run uses only the OXE mixture above, with no active force/torque or force-policy batches; its corpus statistics and evaluation therefore cover that mixture.

Post-training data. Stage 2 uses separate domain-specific runs for the four standard LIBERO suites [39], the SelectFruit and InsertFlower tasks of VLABench [28], and the real-world Franka manipulation tasks described in Section 5.1. Each run has its own training data and objective schedule. Domain-specific adapters change

camera mappings, trajectory indexing, action statistics, and optional textual metadata while preserving the shared objective-conditioned token interface. DROID results use the Stage-1 checkpoint on the DROID validation set without additional DROID-specific post-training.

4.2 Continual Pretraining

Starting from the Dynin-Omni initialization, we continually pretrain the shared masked-diffusion Transformer on robot sequences while keeping the inherited text and visual tokenizers frozen. Following Section 3.3, let $\mathcal{O} = \{o_{\text{PO}}, o_{\text{WM}}, o_{\text{TU}}, o_{\text{GP}}\}$ denote Policy, World Modeling, Task Understanding, and Goal-State Prediction. Every objective uses the target-only masked-diffusion loss $\mathcal{L}_{\text{MDM}}(o)$ defined there. We reserve k for the denoising index and use n for the training micro-step. Let $\mathcal{L}_{\text{MDM}}^{(n)}(o)$ and $\mathcal{L}_{\text{bin}}^{(n)}$ denote the corresponding micro-batch estimates at training micro-step n . A deterministic schedule forms an objective multiset $\mathcal{A}_n$, which may contain repeated Policy requests. The Stage-1 micro-step loss is

$$\mathcal{L}_{\text{Stage-1}}^{(n)} = \frac{1}{|\mathcal{A}_n|} \sum_{o \in \mathcal{A}_n} \lambda_o \left[\mathcal{L}_{\text{MDM}}^{(n)}(o) + \mathbb{I}[o = o_{\text{PO}}] \cdot \beta \mathcal{L}_{\text{bin}}^{(n)} \right]. \quad (16)$$

Here, $\mathbb{I}[\cdot]$ is the indicator function and $\beta = 0.2$. Schedule multiplicity determines long-run objective exposure, whereas λ_o controls loss scaling within a micro-step. These two quantities are varied independently. The expected-bin MAE auxiliary $\mathcal{L}_{\text{bin}}^{(n)}$ is applied only to the Stage-1 Policy objective.

Balanced distributed rotation. For objective o , source i has effective sampling mass $\mu_i^{(o)} = w_i N_i^{(o)}$, where w_i is its mixture weight and $N_i^{(o)}$ counts transitions for Policy or World Modeling and trajectories for Task Understanding or Goal-State Prediction. Large sources are first divided into virtual partitions, producing 99 partition records from the 48 base datasets. The records are balanced into eight source groups. In the reported 32-rank run, ranks are assigned modulo eight to eight data lanes, with four ranks per lane. At virtual epoch e , each lane visits the next group in a lane-specific cyclic order; ranks within the lane consume disjoint, epoch-permuted record shards before trajectory decoding and augmentation. Thus, every lane covers all eight balanced groups over eight virtual epochs while avoiding redundant host-side preprocessing.

Training proceeds in two consecutive segments. The first segment performs 21.8k optimizer updates with equal per-objective loss weights. The second segment loads only the model weights from that checkpoint, reinitializes the optimizer and scheduler, and performs a further 5.8k updates with a more policy-centered schedule and

$$(\lambda_{\text{PO}}, \lambda_{\text{WM}}, \lambda_{\text{TU}}, \lambda_{\text{GP}}) = (1, 0.2, 0.05, 0.2).$$

Both segments use AdamW with a peak learning rate of 2×10^{-5} , $(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay 0.01, cosine decay, 4k warm-up updates, gradient accumulation of eight, gradient clipping at 1.0, and BF16 training with TF32 enabled. The two segments total 27.6k optimizer updates, or 220.8k accumulated micro-steps; the released checkpoint name reflects this micro-step count.

4.3 Post-Training

Stage 2 adapts the continual-pretraining checkpoint separately to each downstream domain d . It preserves the shared token vocabulary, objective-conditioned prompt layouts, and masked-denoising interface, while recomputing action-normalization statistics and selecting an active action-bin count for the target domain. Objective schedules and loss weights are checkpoint-specific: a run may use Policy-only supervision or a unified four-objective multiset when the corresponding targets are available. Similarly, B_A is selected per downstream setting within the reserved action block. Section 5.7 evaluates $B_A \in \{16, 32, 64\}$ for the VLABench Stage-2 study and selects $B_A = 32$ for that comparison; the released Stage-1 checkpoint instead uses $B_A = 256$.

4.3.1 Trajectory Metadata Conditioning

For Stage-2 Policy examples with metadata enabled, we augment the instruction with demonstration-side text fields describing a normalized local action-smoothness proxy and trajectory length. Let $\{\mathbf{u}_h\}_{h=0}^4$, $\mathbf{u}_h \in \mathbb{R}^6$,

denote the first six normalized arm channels of a five-step action chunk; the discontinuous gripper channel is excluded. We reconstruct a normalized arm path as

$$\begin{aligned} \mathbf{y}_h &= \sum_{r=0}^h \mathbf{u}_r && \text{for delta actions,} \\ \mathbf{y}_h &= \mathbf{u}_h && \text{for absolute actions.} \end{aligned}$$

For control interval Δt , the two available third finite differences and their mean norm are

$$\mathbf{j}_h = \frac{\mathbf{y}_h - 3\mathbf{y}_{h-1} + 3\mathbf{y}_{h-2} - \mathbf{y}_{h-3}}{\Delta t^3}, \quad h \in \{3, 4\}, \quad J_{\text{chunk}} = \frac{1}{2} \sum_{h=3}^4 \|\mathbf{j}_h\|_2. \quad (17)$$

We use $\Delta t = 0.05$ s for LIBERO and VLABench and approximately 0.0667 s for the 15-Hz real-world setting. Because the calculation is performed in normalized action coordinates and contains only two jerk vectors, J_{chunk} is a local action-chunk proxy rather than physical Cartesian jerk in SI units.

Let τ denote the complete successful demonstration trajectory and let $L_\tau = |\tau|$ denote its length in environment steps. The visible text context serializes the instruction together with the fields **Action-Chunk Jerk** (J_{chunk}), **Trajectory Steps** (L_τ), and **Success** before the target action span. The success field is fixed to **true** for the successful demonstrations in the training source. These quantities use the existing text-token interface. At rollout time, the prompt instead supplies fixed values: unless overridden, **Action-Chunk Jerk** is set to 300 and **Trajectory Steps** to 200, together with the desired **Success** value. This conditioning uses no ground-truth future actions or realized episode lengths.

4.3.2 Supervised Fine-Tuning

SFT reuses the target-only masked-diffusion loss defined in Section 3.3. Let $\mathcal{A}_n^{(d)}$ denote the active-objective multiset scheduled at Stage-2 micro-step n for domain d , and let $\lambda_{d,o} \geq 0$ denote the domain- and objective-specific loss weight. For each scheduled request $o \in \mathcal{A}_n^{(d)}$, its conditioning span remains visible, only a sampled subset of its target span is corrupted, and logits are restricted to the target-modality vocabulary. Let $\mathcal{L}_{\text{MDM}}^{(n,d)}(o)$ denote the corresponding micro-batch estimate of $\mathcal{L}_{\text{MDM}}(o)$. The Stage-2 micro-step loss is

$$\mathcal{L}_{\text{SFT}}^{(n,d)} = \frac{1}{|\mathcal{A}_n^{(d)}|} \sum_{o \in \mathcal{A}_n^{(d)}} \lambda_{d,o} \mathcal{L}_{\text{MDM}}^{(n,d)}(o). \quad (18)$$

For a Policy-only run, $\mathcal{A}_n^{(d)}$ contains only o_{PO} ; unified runs retain the same backbone and denoising interface while changing visible context, target modality, schedule exposure, and $\lambda_{d,o}$. Stage 2 uses categorical masked-denoising without the Stage-1 expected-bin MAE auxiliary unless explicitly stated otherwise.

Let N_d denote the number of Stage-2 micro-steps for domain d , let $\mathcal{B}_n^{(d)}$ denote the sampled domain-specific micro-batch at step n , and let ξ_n collect masking and corruption randomness. Initializing θ from the Stage-1 model weights, we optimize

$$\theta_d^* = \arg \min_{\theta} \frac{1}{N_d} \sum_{n=1}^{N_d} \mathbb{E}_{\mathcal{B}_n^{(d)}, \xi_n} \left[\mathcal{L}_{\text{SFT}}^{(n,d)} \left(\theta; \mathcal{B}_n^{(d)}, \xi_n \right) \right]. \quad (19)$$

The objective multiset $\mathcal{A}_n^{(d)}$ remains deterministically specified by the checkpoint-specific schedule.

4.3.3 Online Reinforcement-Learning Fine-Tuning

Supervised fine-tuning reconstructs demonstrated action tokens but does not directly optimize online task return or the number of backbone evaluations used during iterative decoding. We therefore introduce a separate online reinforcement-learning stage for the remasking schedule. Prior work has treated diffusion

denoising as a sequential decision process [83, 84], optimized diffusion initialization or step allocation [85, 86], and applied reinforcement learning to discrete-diffusion VLAs [87]. To isolate schedule adaptation, the Dynin-Robotics backbone p_θ and all modality tokenizers remain frozen; only a lightweight remasking controller π_ϕ and its value estimator are trainable.

Remasking decision process. At environment step t and inner denoising step k , the controller observes

$$\zeta_{t,k} = \left(\mathbf{x}_t^V, \mathbf{x}_\ell^T, \mathbf{x}_t^M, \widehat{\mathbf{x}}_g^V, o_{\text{PO}}, \mathbf{x}_{t:t+H-1}^{A,(k)} \right), \quad (20)$$

where the metadata span $\mathbf{x}_t^M$ and goal-state span $\widehat{\mathbf{x}}_g^V$ are omitted when their corresponding conditioning paths are disabled. For metadata-conditioned online rollouts, $\mathbf{x}_t^M$ uses the fixed placeholder fields defined in Section 4.3.1 and remains visible throughout the inner denoising process. Decoding begins from a fully masked action span of length $L_A = Hd_a$. Let

$$\mathcal{M}_{t,k} = \left\{ i \in \{1, \dots, L_A\} : x_{t,i}^{A,(k)} = [\text{MASK}] \right\}$$

denote the currently masked positions. The frozen backbone proposes tentative values $\bar{x}_{t,i}^{A,(k-1)}$ for all $i \in \mathcal{M}_{t,k}$ in parallel and exposes their hidden states and categorical distributions to the controller.

Count-and-position controller. Let $\mathbf{h}_{\theta,t,k,i}$ denote the frozen backbone hidden state at action-token position i , and let $p_{\theta,t,k,i}(v)$ denote its categorical distribution over active action-token values $v \in \mathcal{V}_{\text{action}}$ at denoising step k . For each tentative prediction, the controller constructs

$$\begin{aligned} \mathbf{f}_{t,k,i} &= [\mathbf{h}_{\theta,t,k,i}; \max_v p_{\theta,t,k,i}(v); \mathbf{H}(p_{\theta,t,k,i}); \\ &\quad p_{\theta,t,k,i}^{(1)} - p_{\theta,t,k,i}^{(2)}; k/K], \end{aligned} \quad (21)$$

where $\mathbf{H}$ is categorical entropy and $p^{(1)}$ and $p^{(2)}$ are the two largest token probabilities. A position head assigns the score $z_{t,k,i} = g_\phi^{\text{pos}}(\mathbf{f}_{t,k,i})$. We collect these scores as $\mathbf{z}_{t,k} = (z_{t,k,i})_{i \in \mathcal{M}_{t,k}}$. To avoid an exponentially large subset action space, a count head first samples the number of positions to remask, $c_{t,k} \in \{0, \dots, |\mathcal{M}_{t,k}| - 1\}$, and the position head then samples an ordered set $\mathbf{r}_{t,k}$ of that size without replacement according to the scores. With controller action $u_{t,k} = (c_{t,k}, \mathbf{r}_{t,k})$, its likelihood factorizes as

$$\begin{aligned} \log \pi_\phi(u_{t,k} \mid \zeta_{t,k}) &= \log \pi_\phi^{\text{cnt}}(c_{t,k} \mid \zeta_{t,k}) \\ &\quad + \log \pi_\phi^{\text{pos}}(\mathbf{r}_{t,k} \mid c_{t,k}, \mathbf{z}_{t,k}). \end{aligned} \quad (22)$$

The strict upper bound on $c_{t,k}$ guarantees that at least one new action token is accepted at every inner step; $c_{t,k} = 0$ accepts all tentative predictions and terminates denoising. For $\mathcal{R}_{t,k}$ equal to the positions in $\mathbf{r}_{t,k}$, the next token state is

$$x_{t,i}^{A,(k-1)} = \begin{cases} [\text{MASK}], & i \in \mathcal{R}_{t,k}, \\ \bar{x}_{t,i}^{A,(k-1)}, & i \in \mathcal{M}_{t,k} \setminus \mathcal{R}_{t,k}, \\ x_{t,i}^{A,(k)}, & i \notin \mathcal{M}_{t,k}. \end{cases} \quad (23)$$

When the inner process terminates, inverse action quantization maps $\mathbf{x}_{t:t+H-1}^{A,(0)}$ to continuous controls, which are executed with receding-horizon replanning.

Efficiency-aware online objective. The controller trades environment return against iterative-decoding cost. Let T_{env} denote the environment rollout horizon, $\gamma \in [0, 1]$ the discount factor, and s_t the environment state at step t . Let $\widehat{\mathbf{a}}_t$ denote the control actually executed after decoding and inverse quantization, and let $r_{\text{env}}(s_t, \widehat{\mathbf{a}}_t, s_{t+1})$ denote the resulting environment reward. If $K_t \leq K$ is the number of backbone evaluations used to decode the action chunk at environment step t , the objective is

$$J_{\text{RL}}(\phi) = \mathbb{E}_{\pi_\phi} \left[\sum_{t=0}^{T_{\text{env}}-1} \gamma^t \left(r_{\text{env}}(s_t, \widehat{\mathbf{a}}_t, s_{t+1}) - \eta_{\text{den}} K_t - \eta_{\text{step}} \right) \right], \quad (24)$$

where the expectation covers controller actions and environment transitions, η_{den} penalizes additional refinement, and η_{step} discourages unnecessarily long episodes. We maximize $J_{\text{RL}}(\phi)$ with respect to the controller parameters ϕ . We optimize the controller and value estimator with proximal policy optimization (PPO) [88] while keeping p_θ frozen. At inference, the count head selects the most likely $c_{t,k}$, and the $c_{t,k}$ positions with the largest remasking scores are reconsidered. This learned controller replaces only the refinement schedule; it does not modify the shared token predictor or modality tokenizers. The present ablations do not isolate the controller’s contribution from the other post-training and decoding choices.

5 Evaluation

5.1 Evaluation Setup

Our evaluation examines how the shared trajectory formulation performs as a robot policy, as a predictor of visual states and task descriptions, and as an interface for composing predictions during control. We first assess policy success on standard tasks, distribution shifts, and physical manipulation. We then evaluate the model’s direct outputs and use ablations to distinguish the effects of robot pretraining, Stage-2 objective mixtures, and inference composition. Finally, we measure the accuracy–latency trade-offs introduced by accelerated decoding and the sensitivity to action discretization and auxiliary-loss scaling.

Unless stated otherwise, the policy receives RGB observations and a natural-language instruction and produces continuous controls by denoising and dequantizing action tokens. Each downstream domain uses a separately adapted checkpoint, allowing us to evaluate the shared modeling and adaptation approach across domains. Task success rate is the primary policy metric. On VLABench, in-distribution (ID) success denotes Track 1 performance, out-of-distribution (OOD) success averages Tracks 3 and 4, and the gap is ID – OOD. We interpret this gap together with absolute success, since low performance in both conditions can also produce a small gap.

Cross-model policy tables combine cited and reproduced results obtained with model-specific training recipes; within-model ablations examine the effects of individual design choices or training recipes. We treat numerical differences as descriptive and report variability where available. For checkpoints using the demonstration metadata in Section 4.3.1, rollout prompts contain the specified fixed placeholder values.

LIBERO. LIBERO [39] comprises four suites covering spatial relations, object-centric manipulation, goal-conditioned interaction, and long-horizon composition. We report success on each suite and their macro-average to assess standard-task policy performance. Section 5.6 separately examines how the objective mixture affects downstream behavior. The reported LIBERO result uses the two-stage goal-guided inference path, variant (c) in Section 5.6.3.

LIBERO-Plus. LIBERO-Plus [40] evaluates seven perturbation families: Camera, Robot, Language, Light, Background, Noise, and Layout. We apply the LIBERO-trained checkpoint with the same inference path and no LIBERO-Plus training or adaptation. Here, “zero-shot” denotes transfer from LIBERO to these perturbed evaluation conditions. We report each family’s success rate and their macro-average to characterize which changes are accommodated by the learned policy and which remain difficult.

VLABench. We study SelectFruit and InsertFlower from VLABench [28]. SelectFruit requires selecting a target across object and layout variation, while InsertFlower requires grasping, alignment, and insertion. Models are trained with direct Track 1 instructions and evaluated with those instructions, commonsense descriptions from Track 3, and indirect semantic paraphrases from Track 4. Figure 8 illustrates the two tasks and their instruction formats. The 5k–30k post-training checkpoints in Section 5.2.3 characterize learning behavior. The ablations and sensitivity studies in Sections 5.6 and 5.7 instead compare 70k-step Stage-2 endpoints. These experiments focus on instruction shifts within the two selected tasks.

Real-world experiments. We evaluate a Franka Research 3 robot in one workspace under four conditions drawn from three task families. Fruit PnP requires selecting an instructed fruit and placing it at a target.

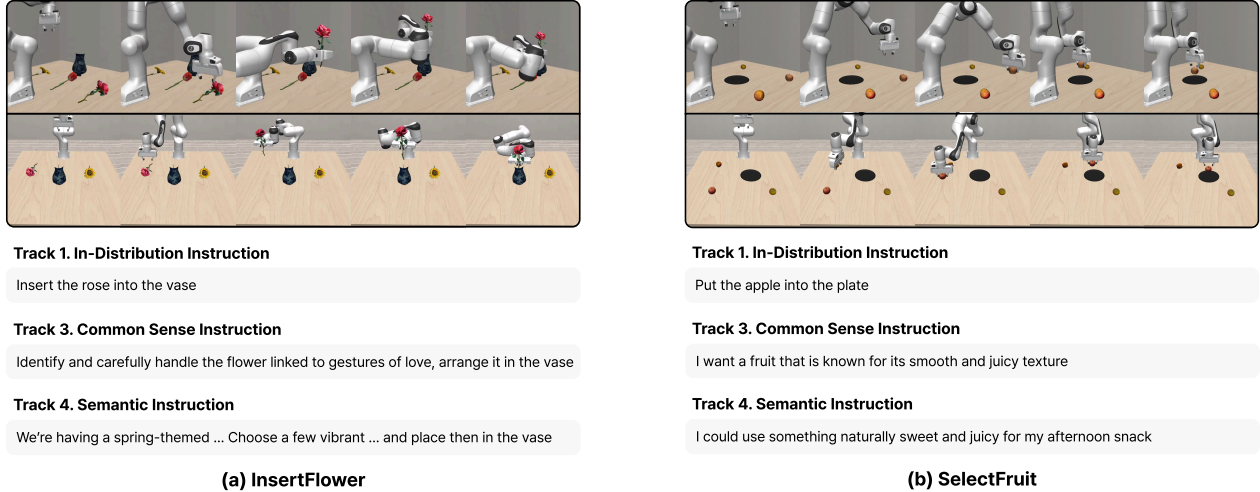


Figure 8 VLABench diagnostic tasks and instruction tracks. Example rollouts and language variants are shown for (a) INSERTFLOWER, a contact-rich insertion task requiring precise grasp–alignment–insertion control, and (b) SELECTFRUIT, an object-grounding task requiring adaptation to varying object identities and layouts. Track 1 provides direct in-distribution instructions, Track 3 uses commonsense descriptions, and Track 4 uses indirect semantic paraphrases.

Cube Sort requires placing colored cubes into matching boxes. Cube Stack requires a stable four-cube stack in arbitrary order, while Color-Ordered Stack additionally requires an instruction-specified color sequence. Success is the fraction of rollouts satisfying the corresponding terminal condition, and the overall score is the macro-average across these four conditions. This study assesses execution of the adapted policy on the reported physical setup.

5.2 Policy Evaluation

5.2.1 LIBERO

Dynin-Robotics achieves a 98.1% macro-average on LIBERO (Table 2), matching X-VLA and placing 0.5 percentage points below ABot-M0. Several methods obtain closely grouped scores near saturation, positioning Dynin-Robotics among the stronger policies in this comparison.

The suite profile reveals where performance remains uneven. Dynin-Robotics reaches 98.9% on Spatial and 99.8% on Object, compared with 97.8% on Goal and 95.8% on Long. Long-horizon tasks retain the most room for improvement among the four suites. This performance establishes that the shared trajectory model supports high policy success alongside its visual and language output interfaces.

5.2.2 LIBERO-Plus

Transfer to perturbed conditions reveals a less uniform performance profile. Dynin-Robotics obtains a 73.0% macro-average in Table 3, above the listed OpenVLA-OFT and RIPT-VLA results and below ABot-M0 at 81.6%. Language perturbations retain an 85.0% success rate, whereas Camera and Robot yield 59.8% and 48.2%, respectively. The high clean-benchmark score therefore coexists with substantial sensitivity to some of the evaluated changes.

The perturbation families also contribute unevenly to the difference from ABot-M0. Camera is one of Dynin-Robotics’ lowest-scoring categories, yet its gap to ABot-M0 is only 0.6 percentage points. The gaps are larger for Robot and Light, at 19.7 and 12.7 points. The family-level results thus distinguish conditions that are difficult for both models from those where Dynin-Robotics falls further behind, identifying priorities for improving transfer.

Table 2 Success rates (%) on the four LIBERO task suites. Methods are grouped into continuous-diffusion, vision-language, video-generation, and unified policy models. Average is the macro-average across the four suites. The best result in each suite and the overall average is underlined.

Models	Spatial	Object	Goal	Long	Average $\uparrow$
<i>Continuous Diffusion Models</i>					
Diffusion Policy [89]	78.5	87.5	73.5	64.8	76.1
<i>Vision-Language Models</i>					
OpenVLA [2]	84.7	88.4	79.2	53.7	76.5
SpatialVLA [13]	88.2	89.9	78.6	55.5	78.1
CoT-VLA [14]	87.5	91.6	87.6	69.0	83.9
π_0 -FAST [15]	96.4	96.8	88.6	60.2	85.5
GR00T-N1 [5]	94.4	97.6	93.0	90.6	93.9
π_0 [3]	98.0	96.8	94.4	88.4	94.4
F1 [75]	98.2	97.8	95.4	91.3	95.7
InternVLA-M1 [76]	98.0	99.0	93.8	92.6	95.9
Discrete Diffusion VLA [16]	97.2	98.6	97.4	92.0	96.3
$\pi_{0.5}$ [4]	98.8	98.2	98.0	92.4	96.9
GR00T-N1.6 [5]	97.7	98.5	97.5	94.4	97.0
OpenVLA-OFT [17]	97.6	98.4	97.9	94.5	97.1
X-VLA [6]	98.2	98.6	97.8	97.6	98.1
ABot-M0 [18]	98.8	99.8	<u>99.0</u>	96.6	<u>98.6</u>
<i>Video Generation Models</i>					
Video Policy [25]	-	-	-	94.0	-
Cosmos Policy [22]	98.1	<u>100.0</u>	98.2	97.6	98.5
CogVLA [19]	98.6	98.8	96.6	95.4	97.4
Mimic-Video [24]	96.2	98.3	96.2	-	-
LingBot-VA [26]	98.5	99.6	97.2	<u>98.5</u>	98.5
Fast-WAM [27]	98.2	<u>100.0</u>	97.0	95.2	97.6
<i>Unified Models</i>					
dVLA [32]	97.4	97.9	98.2	92.2	96.4
WorldVLA [30]	87.6	96.2	83.4	60.0	81.8
RynnVLA-002 [33]	<u>99.0</u>	99.8	96.4	94.4	97.4
UD-VLA [31]	94.1	95.7	91.2	89.6	92.7
MMaDA-VLA [34]	98.8	99.8	98.0	95.2	98.0
Dynin-Robotics (Ours)	98.9	99.8	97.8	95.8	98.1

5.2.3 VLABench

The two-task diagnostic examines how policy performance varies with the manipulation task and instruction form. In Figure 9, $\pi_{0.5}$ performs better than Mimic-Video on SelectFruit across the displayed tracks. On InsertFlower, Mimic-Video is particularly competitive under the indirect Track 4 instructions, while the direct-instruction curves show a different ordering. Dynin-Robotics substantially exceeds Mimic-Video on SelectFruit and remains competitive on InsertFlower, with its relative position varying across checkpoints and tracks. These observations reveal distinct task and instruction profiles among the evaluated policies.

The curves track performance across post-training checkpoints. Together with the absolute success rates, they show why a small separation between instruction tracks is insufficient to characterize robustness: a policy can perform similarly across tracks while solving few tasks. The 70k-step ablations below therefore compare absolute ID and OOD success alongside their gap.

Instruction randomization further distinguishes the two reference policies. In Table 4, replacing the instruction with a length-matched random string reduces $\pi_{0.5}$'s success by 16 and 30 percentage points on InsertFlower and SelectFruit. Mimic-Video instead changes by +10 and +2 points, with progress scores showing the same qualitative contrast. Thus, $\pi_{0.5}$ is more sensitive to this text intervention on the two tasks, while Mimic-Video retains its success under instruction randomization. Together with the checkpoint profiles, this observation motivates evaluating task success and instruction sensitivity as complementary aspects of policy behavior.

Table 3 Zero-shot transfer from LIBERO to the seven LIBERO-Plus perturbation categories without additional LIBERO-Plus training. Entries report success rates (%) under camera, robot, language, lighting, background, noise, and layout perturbations. Average is recomputed as the unweighted mean of the seven displayed category scores. Underlining marks the highest value in each column.

Models	Camera	Robot	Language	Light	Background	Noise	Layout	Average $\uparrow$
OpenVLA [2]	0.8	3.5	23.0	8.1	34.8	15.2	28.5	16.3
OpenVLA-OFT [17]	56.4	31.9	79.5	88.7	93.3	75.8	74.2	71.4
OpenVLA-OFT _w [17]	10.4	38.7	70.5	76.8	<u>93.6</u>	49.9	69.9	58.5
OpenVLA-OFT _m [17]	55.6	21.7	81.0	92.7	91.0	78.6	68.7	69.9
NORA [77]	2.2	37.0	65.1	45.7	58.6	12.8	62.1	40.5
UniVLA [29]	1.8	46.2	69.6	69.0	81.0	21.2	31.9	45.8
π_0 [3]	13.8	6.0	58.8	85.0	81.4	79.0	68.9	56.1
π_0 -FAST [15]	<u>65.1</u>	21.6	61.0	73.2	73.2	74.4	68.8	62.5
RIPT-VLA [78]	55.2	31.2	77.6	88.4	91.6	73.5	74.2	70.2
ABot-M0 [18]	60.4	<u>67.9</u>	<u>86.4</u>	<u>96.2</u>	91.6	<u>86.4</u>	<u>82.6</u>	<u>81.6</u>
WorldVLA [30]	0.1	27.9	41.6	43.7	17.1	10.9	38.0	25.6
Dynin-Robotics (Ours)	59.8	48.2	85.0	83.5	84.6	78.2	71.8	73.0

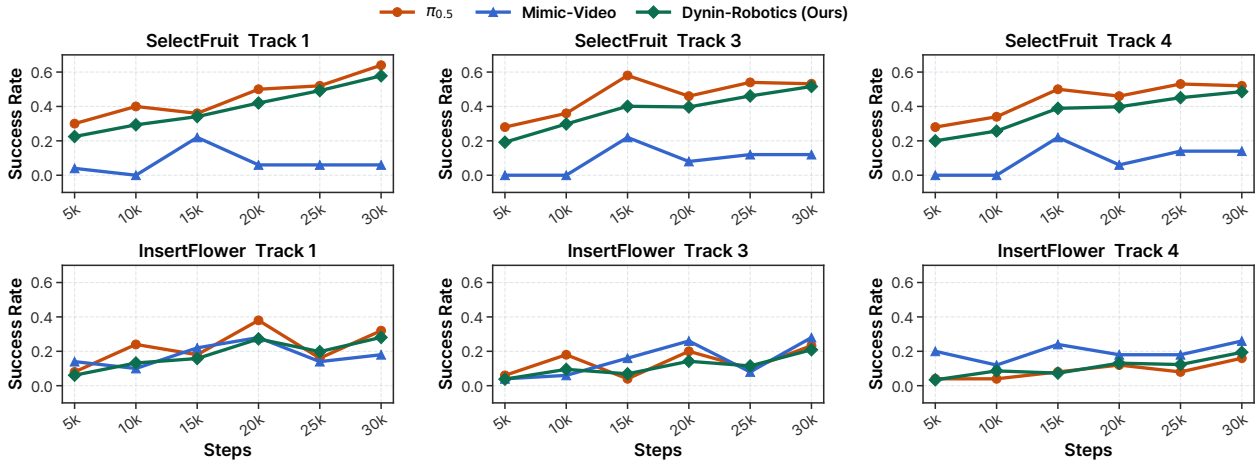


Figure 9 Intermediate VLABench checkpoint performance from 5k to 30k post-training steps for $\pi_{0.5}$, Mimic-Video, and Dynin-Robotics. The top row shows SelectFruit and the bottom row InsertFlower; curves compare direct, commonsense, and indirect instructions from Tracks 1, 3, and 4. These diagnostics are distinct from the 70k endpoints used in the ablation and sensitivity studies.

Table 4 Instruction sensitivity on VLABench. Success rate and progress score are measured with the original instruction and a length-matched random string under the same visual observation and rollout settings. Parentheses show the signed change (random minus original).

Model	Task	Success Rate $\uparrow$		Progress Score $\uparrow$	
		Original	Random	Original	Random
Mimic-Video [24]	INSERTFLOWER	0.18	0.28 (+0.10)	0.58	0.63 (+0.05)
	SELECTFRUIT	0.08	0.10 (+0.02)	0.53	0.53 (0.00)
$\pi_{0.5}$ [4]	INSERTFLOWER	0.32	0.16 (-0.16)	0.65	0.57 (-0.08)
	SELECTFRUIT	0.64	0.34 (-0.30)	0.82	0.53 (-0.29)

5.2.4 Real-World Manipulation

The physical evaluation covers object selection, sorting, and stacking on the Franka Research 3 platform. Figure 10 illustrates the task families, including the additional order constraint in Color-Ordered Stack. The four conditions allow us to examine how task requirements affect the comparative performance of the adapted policies.

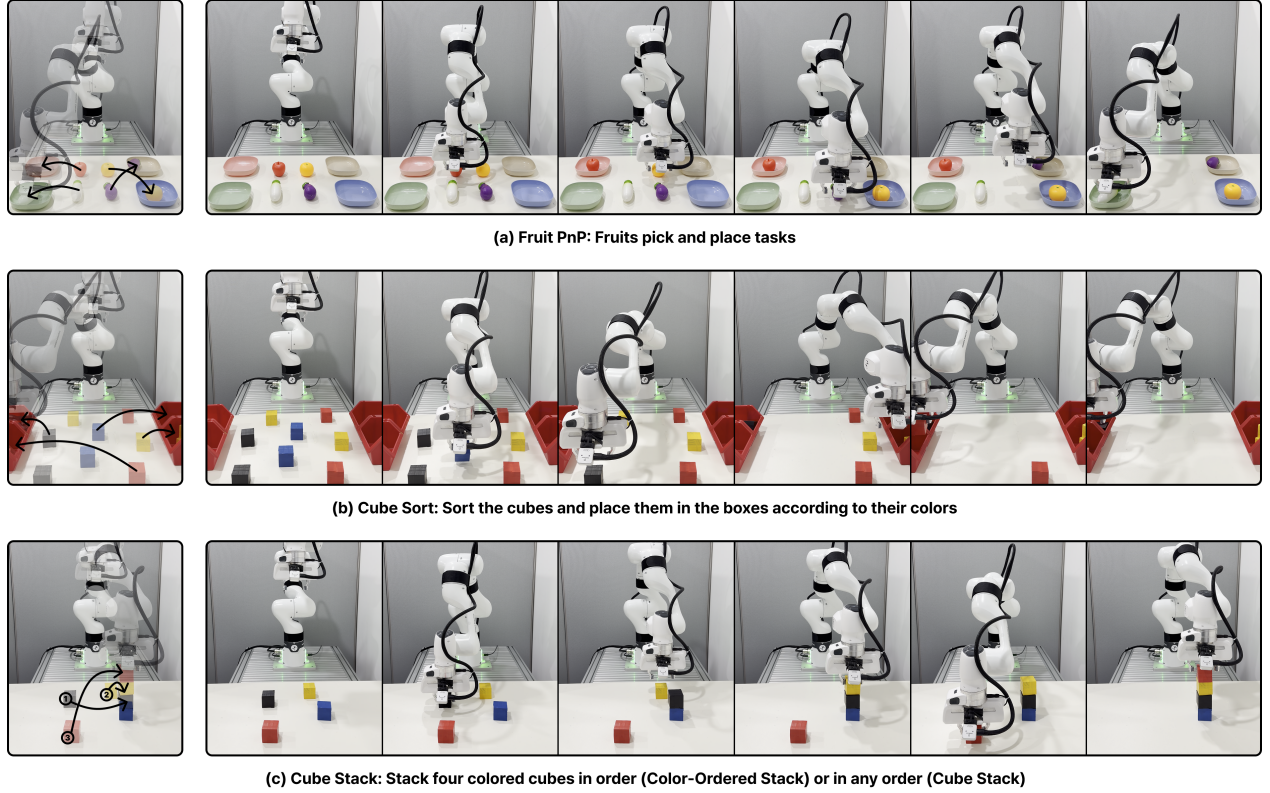


Figure 10 Real-world manipulation tasks on the Franka Research 3 platform. (a) FRUIT PNP selects an instructed fruit and places it at the target location. (b) CUBE SORT places colored cubes into color-matched boxes. (c) CUBE STACK builds a four-cube stack either in arbitrary order or according to an instruction-specified color sequence, denoted COLOR-ORDERED STACK.

Table 5 Real-world manipulation success rates (%) on a Franka Research 3 robot. Fruit PnP, Cube Sort, Cube Stack, and Color-Ordered Stack define four evaluation conditions, with the latter two differing in whether the stacking order is specified by the instruction. Average is the unweighted mean across the four conditions. Underlining marks the highest value in each column.

Method	FRUIT PNP	CUBE SORT	CUBE STACK	COLOR-ORDERED STACK	Average(%) $\uparrow$
<i>Vision-Language Models</i>					
$\pi_{0.5}$ [4]	96.5	83.5	63.5	64.0	76.9
GR00T-N1.6 [5]	91.0	78.0	59.5	63.5	73.0
<i>Video-Generation Models</i>					
Cosmos Policy [22]	<u>98.0</u>	<u>87.0</u>	<u>67.5</u>	42.0	73.6
Mimic-Video [24]	96.5	84.5	64.0	39.5	71.1
<i>Unified Models</i>					
Dynin-Robotics (Ours)	97.5	85.5	62.0	<u>68.5</u>	<u>78.4</u>

Dynin-Robotics achieves the highest reported macro-average, 78.4%, and the highest Color-Ordered Stack success, 68.5%, in Table 5. Relative to $\pi_{0.5}$, its condition-wise differences are +1.0, +2.0, -1.5, and +4.5 percentage points for Fruit PnP, Cube Sort, Cube Stack, and Color-Ordered Stack, respectively. Its 1.5-point average advantage is therefore unevenly distributed, with the largest positive difference occurring when the stacking order is specified by language.

Cosmos Policy obtains the strongest results on Fruit PnP, Cube Sort, and unconstrained Cube Stack, whereas Dynin-Robotics leads on Color-Ordered Stack. The condition-level results therefore reveal different strengths beneath the overall ranking. Dynin-Robotics’ clearest comparative advantage occurs in stacking with an

Table 6 Visual and offline action prediction on the DROID validation set. Adjacent-future image predictions are evaluated with PSNR, SSIM, and LPIPS, and normalized action predictions with accuracy at tolerance 0.1 and mean absolute error. Models use their respective prediction interfaces; Dynin-Robotics uses action-conditioned World Modeling. Its results are obtained from the Stage-1 checkpoint without additional DROID-specific post-training, with DROID included in the Stage-1 mixture. Underlining marks the best values among learned models, excluding the Last Frame and Random Frame references. Dashes indicate metrics not reported for the corresponding row.

Model	World Modeling			Policy	
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	Action Acc@0.1 $\uparrow$	Action MAE $\downarrow$
<i>Baselines</i>					
Last Frame	31.07	0.953	0.013	–	–
Random Frame	8.94	0.155	0.665	–	–
<i>Unified VLA</i>					
UP-VLA	11.19	0.290	0.491	0.2952	0.2492
MMaDA-VLA	17.72	0.533	<u>0.232</u>	0.3204	0.3068
Dynin-Robotics (Ours)	<u>20.79</u>	<u>0.573</u>	0.299	<u>0.3327</u>	<u>0.2109</u>

instruction-specified order, a task that combines object manipulation with a sequence constraint.

5.3 World Modeling and Goal-State Prediction

Direct output evaluation examines the predictions provided by the shared trajectory interface. World Modeling predicts the immediately subsequent visual state from the current context and corresponding action; Goal-State Prediction generates a terminal task state from the initial observation and instruction. All quantitative and qualitative Dynin-Robotics outputs in this section use the Stage-1 checkpoint on the DROID validation set without additional DROID-specific post-training. DROID [38] is included in the Stage-1 OXE mixture.

We measure reference-image similarity using peak signal-to-noise ratio (PSNR), structural similarity (SSIM), and learned perceptual image patch similarity (LPIPS); higher PSNR and SSIM and lower LPIPS indicate better agreement. These metrics characterize image reconstruction, while the inference ablations evaluate the use of predictions during control. Their interpretation depends on the prediction task: adjacent frames can share substantial static content, and distinct terminal configurations can satisfy the same instruction. Cross-model rows use each model’s reported prediction interface, providing a comparison of output quality across those interfaces.

5.3.1 Comparison with Unified Policies

Frame copying provides a strong reference for adjacent-future prediction in Table 6. Copying the last frame yields 31.07 PSNR, 0.953 SSIM, and 0.013 LPIPS, compared with Dynin-Robotics at 20.79, 0.573, and 0.299. The generated predictions therefore fall below persistence on all three whole-image metrics. Because these scores aggregate static scene content and changing regions, evaluating action-dependent changes remains a distinct requirement for assessing the learned transition model.

Among the learned predictors in the table, Dynin-Robotics has the highest PSNR and SSIM, while MMaDA-VLA has the lowest LPIPS. The same Dynin-Robotics checkpoint also achieves the highest reported offline action accuracy at tolerance 0.1 (0.3327) and the lowest normalized action MAE (0.2109). These results show complementary strengths across output metrics within a model that supports both visual and action prediction.

Figure 11 illustrates next-state outputs generated from robot observations and action conditioning across several scenes. The examples show the visual content produced by this interface; the quantitative comparison above measures agreement with recorded futures. Section 5.6.3 evaluates how composing these predictions with action generation affects policy success.

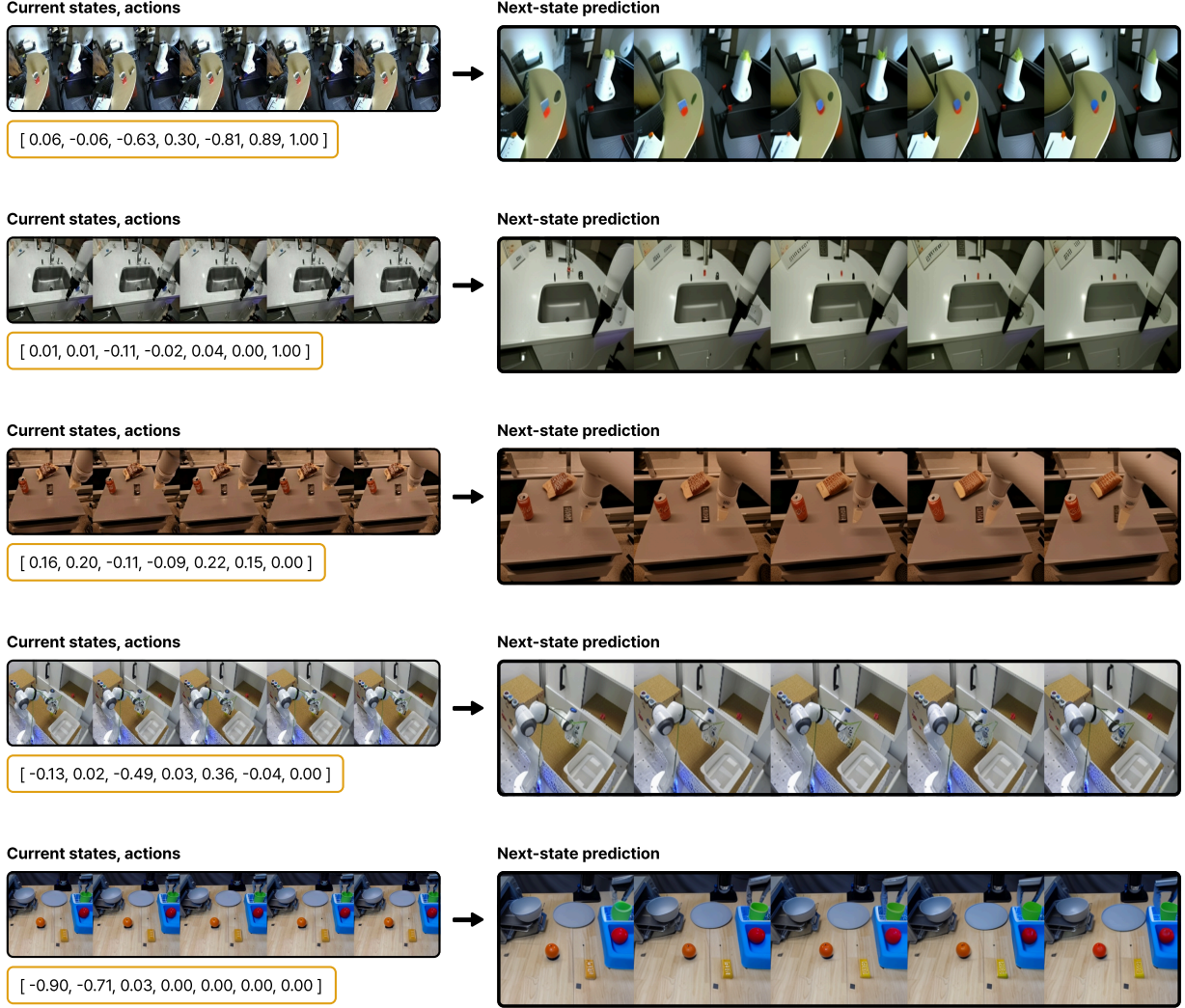


Figure 11 Qualitative action-conditioned next-observation prediction on the DROID validation set using the Stage-1 checkpoint without additional DROID-specific post-training. Each example shows the visual observations and action context on the left and the predicted adjacent-future observations on the right.

5.3.2 Comparison with Image-Generation Models

Comparison with image-generation models shows that the relative performance depends on the prediction task and metric. In Table 7, Dynin-Robotics has the highest World Modeling PSNR, whereas BAGEL has higher SSIM and lower LPIPS. For Goal-State Prediction, BAGEL leads on all three metrics and Dynin-Robotics ranks second among the listed models. Dynin-Robotics is therefore strongest in adjacent-future pixel reconstruction within this comparison, with greater room for improvement in terminal-state generation.

The two prediction settings also play different roles during control: the next-state interface represents a local transition, while the goal interface supplies a visual target for task completion. For goal prediction, reference-image agreement captures only one aspect of output quality, since alternative successful terminal configurations can differ from the recorded image. Task-condition satisfaction and the effect on policy success provide complementary criteria for evaluating generated goals.

Figure 12 shows selected initial observations, instructions, and generated goal images for tasks involving object placement and scene rearrangement. These examples illustrate the terminal-state predictions available for goal conditioning. The inference-composition study evaluates their use by comparing goal-conditioned and

Table 7 Visual prediction quality on the DROID validation set for adjacent-future observations and terminal goal states. Predictions are evaluated with PSNR, SSIM, and LPIPS using each model’s prediction interface. Dynin-Robotics uses the Stage-1 checkpoint without additional DROID-specific post-training. Arrows indicate metric direction, and underlining marks the best value in each column.

Model	World Modeling			Goal-State Prediction		
	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$
BAGEL	19.3063	<u>0.6405</u>	<u>0.1568</u>	<u>13.4755</u>	<u>0.5005</u>	<u>0.3626</u>
MMaDA	8.8326	0.2120	0.7532	8.6710	0.2058	0.7630
Show-o2	8.7519	0.2098	0.7220	8.7544	0.1850	0.7275
Dynin-Robotics (Ours)	<u>20.7941</u>	0.5731	0.2994	10.7098	0.2825	0.4715

action-only decoding on the same checkpoint.

5.4 Qualitative Task Understanding

Task Understanding queries the shared backbone in the trajectory-to-language direction: sampled trajectory frames provide context for generating a task description. Figure 13 presents selected outputs from the Stage-1 checkpoint on DROID validation trajectories without additional DROID-specific post-training. The descriptions express action verbs such as picking, placing, folding, and sliding; object attributes such as color; and spatial relations involving bowls, drawers, and counters. One example combines retrieval from a drawer with placement on a counter, describing a sequence that spans multiple local actions.

The outputs vary in specificity, from referring to a purple object to naming a can and its source and destination. This variation illustrates the granularity of the generated descriptions. Quantitative evaluation against reference instructions remains a next step for assessing the accuracy of trajectory-to-language prediction.

The cumulative objective study in Section 5.6 evaluates Task Understanding from the perspective of policy adaptation. Adding it to the Stage-2 mixture accompanies higher downstream success under the coupled decoder, connecting trajectory-language supervision to the performance of the overall training recipe.

5.5 Acceleration

We evaluate the throughput and prediction fidelity of the accelerated action-decoding implementation built on dInfer [37]. The measured implementation combines modality-aware denoising, confidence-based block commitment, compiled execution, CUDA Graph replay, and loop unrolling. Profiling uses NVIDIA B200 hardware, batch size 1, a 2,048-token context, and a 35-token action chunk corresponding to five 7-DoF actions. Internal Dynin-Robotics fidelity measurements use a commitment threshold of 0.9.

Effective TPS measures action-output throughput in 7-DoF action-token units, while query latency measures generation of a complete five-action chunk. These are model-side action-decoding measurements. Goal generation, sensing, communication, environment stepping, and the execution and replanning schedule are outside their scope. BL7 and BL35 identify the configured token-block lengths; the realized number of backbone evaluations per chunk is measured separately below. For continuous-output baselines, effective TPS provides a common action-output unit rather than a literal count of discrete tokens emitted by their native interfaces.

The cross-model profiling results in Table 8 provide context for the internal acceleration, while Table 9 compares decoder variants on the same Dynin-Robotics model. BL7 increases throughput from 9.221 to 91.236 effective TPS, a 9.89 $\times$ speedup. BL35 reaches 268.834 TPS, or 29.15 $\times$ the base throughput. Thus, increasing the configured block size together with the optimized execution path substantially reduces the cost of producing an action chunk.

The reconstruction metrics show small changes in output fidelity across the decoder variants. BL7 has a token accuracy of 99.8214%, compared with 99.7545% for the base decoder, while its action MAE is slightly higher (0.021383 versus 0.021317). BL35 lowers token accuracy to 99.4643% and raises MAE to 0.021468.

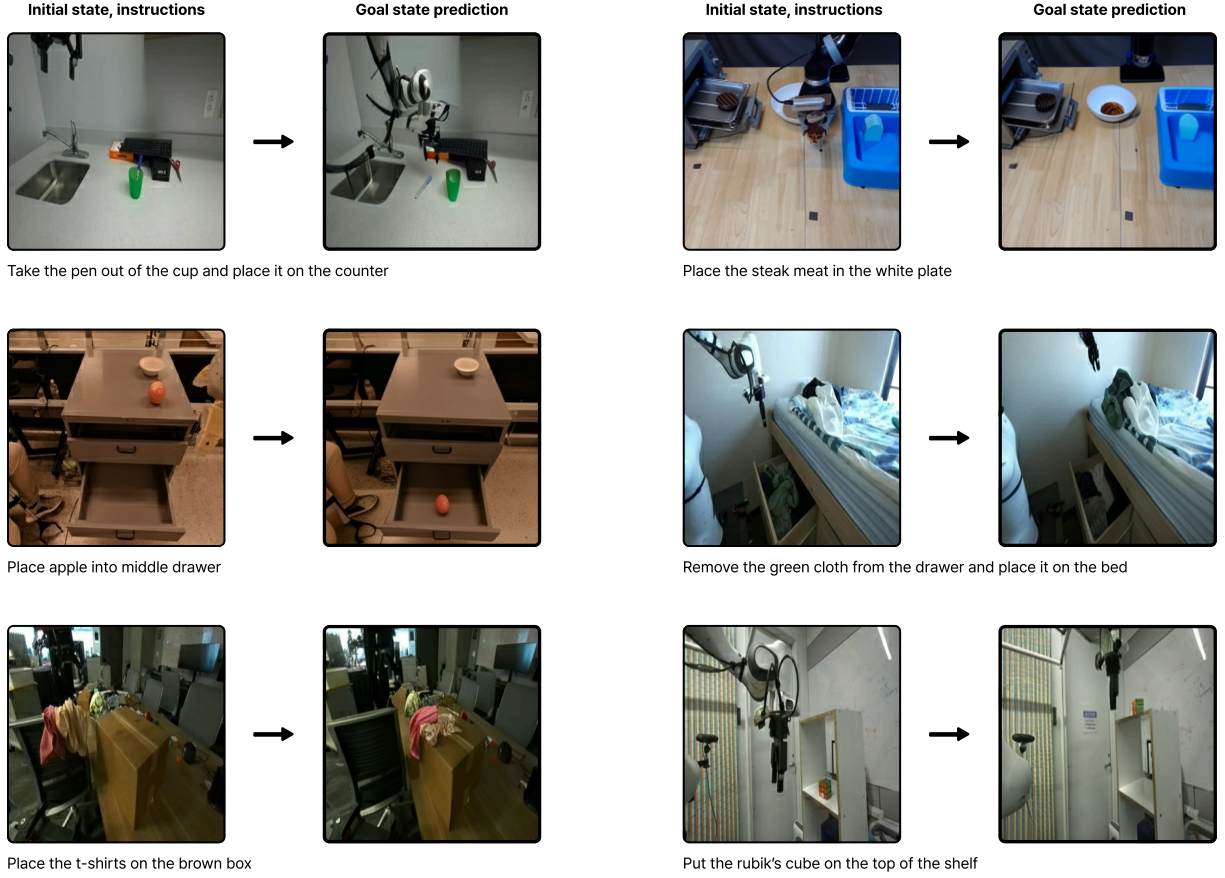


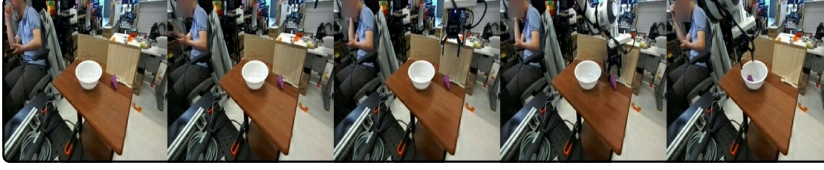
Figure 12 Qualitative Goal-State Prediction on the DROID validation set using the Stage-1 checkpoint without additional DROID-specific post-training. Each example pairs an initial observation and task instruction with the predicted terminal visual state, shown on the right.

Table 8 Model-side action-decoding throughput under the common single-B200 profiling protocol. Effective TPS reports scalar action-output units per second, with seven units per action. For discrete-token decoders, TPF is the total number of finalized action tokens divided by the total number of backbone evaluations; dashes indicate that this measure is not applicable to the continuous-output interface. Dynin-Robotics TPF values are derived from the reported mean evaluations per 35-token chunk in Table 14. BL7 and BL35 denote configured block lengths. Parentheses report speedup over the base Dynin-Robotics decoder. Timing excludes goal generation, sensing, communication, and robot execution. The highest throughput is underlined.

Model	Effective TPS $\uparrow$	TPF
<i>Vision-Language Models</i>		
OpenVLA-OFT [17]	19.114	—
$\pi_{0.5}$ [4]	7.351	—
<i>Masked-Diffusion Models</i>		
LLaDA-VLA [57]	2.079	1.00
MMaDA-VLA [34]	1.827	1.00
<i>Unified Models</i>		
Dynin-Robotics	9.221	1.00
Dynin-Robotics-dInfer-BL7	91.236 (9.89 $\times$)	6.81
Dynin-Robotics-dInfer-BL35	<u>268.834</u> (29.15 $\times$)	20.23

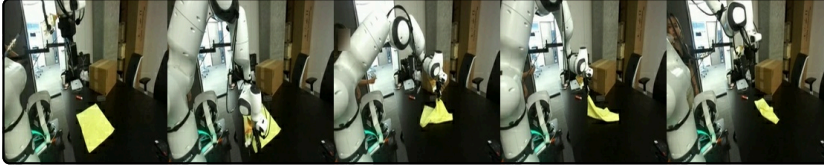
Section 5.7.2 complements these offline measurements with query latency, denoising work, and LIBERO-Goal rollout success.

Task video frames



→ Task instruction prediction
Put the purple object in the bowl

Task video frames



→ Task instruction prediction
Fold the cloth on the table

Task video frames



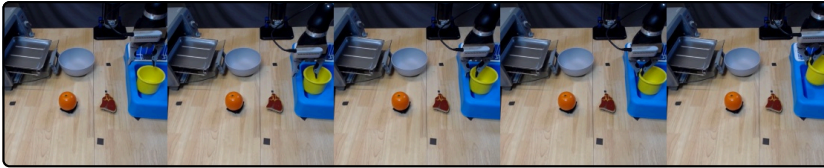
→ Task instruction prediction
Pick pepsi can from middle drawer
and place on counter

Task video frames



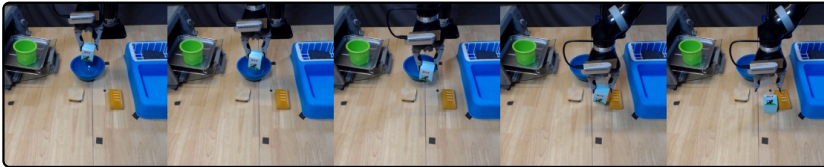
→ Task instruction prediction
Grasp the door handle,
slide the door to the left

Task video frames



→ Task instruction prediction
Pick up the yellow cup

Task video frames



→ Task instruction prediction
Place the milk dairy on the table

Figure 13 Qualitative Task Understanding on the DROID validation set using the Stage-1 checkpoint without additional DROID-specific post-training. Sampled trajectory frames are shown on the left, with the generated task descriptions on the right. Underlining highlights object references and spatial expressions in the model outputs.

5.6 Ablation Study

The ablations examine three interventions: changing the initialization before Stage 2, changing the Stage-2 objective mixture, and changing inference composition after training. The training comparisons use the same

Table 9 Block-parallel decoding throughput and reconstruction fidelity on LIBERO-Goal. All rows use the common one-B200 profiling protocol; parentheses report speedup over the Base decoder. Reconstruction metrics use held-out action chunks, and the best value in each metric is underlined.

Configuration	Effective TPS $\uparrow$	Token Acc (%) $\uparrow$	Action MAE $\downarrow$
Base	9.221	99.7545	<u>0.021317</u>
dInfer-BL7	91.236 (9.89 $\times$)	<u>99.8214</u>	0.021383
dInfer-BL35	<u>268.834</u> (29.15 $\times$)	99.4643	0.021468

reported 70k-step VLABench Stage-2 budget. In the objective-mixture comparison, inference variant (e) is fixed while supervision changes; in the inference comparison, the checkpoint is fixed while the decoding composition changes. This design evaluates training recipes under a common decoder and inference choices with a common model. We report Track 1 success as ID, the average of Tracks 3 and 4 as OOD, and (ID – OOD) as the gap.

5.6.1 Continual-Pretraining Ablation

Table 10 Effect of OXE continual pretraining on VLABench adaptation with a fixed 70k-step Stage-2 budget. Both variants build on Dynin-Omni; Stage 2 starts either directly from Dynin-Omni or from the checkpoint obtained after 220.8k OXE training micro-steps. ID denotes Track 1 success, and OOD is the mean success across Tracks 3 and 4. The higher success rate in each column is underlined.

Training variant	Stage 1	Stage 2	ID (%) $\uparrow$	OOD (%) $\uparrow$
Without pretraining	$\times$	$\checkmark$	3.26	0.70
Dynin-Robotics	$\checkmark$	$\checkmark$	<u>49.61</u>	<u>47.28</u>

Robot continual pretraining substantially improves adaptation within the reported Stage-2 budget. In Table 10, initialization from Dynin-Omni without OXE continual pretraining reaches 3.26% ID and 0.70% OOD success, compared with 49.61% and 47.28% after initialization from the 220.8k-micro-step OXE checkpoint. Both pipelines start from the same omnimodal foundation, and the comparison measures the benefit of the additional robot-pretraining stage for downstream adaptation.

The improvement spans direct and shifted instructions. Under this recipe, 70k Stage-2 steps from the Dynin-Omni initialization do not recover the performance of the robot-pretrained model. The result identifies robot continual pretraining as a major contributor to downstream success within this fixed adaptation budget.

5.6.2 Training-Objective Ablation

Table 11 Cumulative Stage-2 objective mixtures under the 70k-step VLABench adaptation protocol. All variants start from the same Stage-1 checkpoint and are evaluated with goal-guided joint action–next-state denoising, variant (e). Exposure ratios specify the relative frequency of the listed objectives in the training schedule, separately from their loss weights. Each row changes the active objectives and their exposure allocation. ID denotes Track 1 success, OOD averages Tracks 3 and 4, and Gap is ID minus OOD in percentage points. Underlining marks the highest success rates and the smallest gap.

Training variant	Objectives	Exposure ratio	ID (%) $\uparrow$	OOD (%) $\uparrow$	Gap (%) $\downarrow$
Policy Only	$\{o_{PO}\}$	1.0	47.15	33.88	13.27
+ World Modeling	$\{o_{PO}, o_{WM}\}$	0.8 : 0.2	46.01	39.54	6.47
+ Task Understanding	$\{o_{PO}, o_{WM}, o_{TU}\}$	0.75 : 0.2 : 0.05	49.38	45.29	4.09
+ Goal-State Prediction (All)	$\{o_{PO}, o_{WM}, o_{TU}, o_{GP}\}$	0.65 : 0.2 : 0.05 : 0.10	<u>49.61</u>	<u>47.28</u>	<u>2.33</u>

The Stage-2 mixture primarily changes performance under shifted instructions. Table 11 compares cumulative objective sets after 70k Stage-2 steps, evaluating every row with goal-guided joint action–next-state denoising, variant (e). Relative to Stage-2 Policy-only training, the full mixture increases ID success from 47.15% to 49.61% and OOD success from 33.88% to 47.28%. The resulting improvements are +2.46 and +13.40

percentage points, respectively, and the gap decreases from 13.27 to 2.33 points. Here, the smaller gap accompanies substantially higher OOD success rather than a loss of ID performance.

The cumulative increments are not uniform. Adding World Modeling increases OOD success by 5.66 points while reducing ID by 1.14 points. Adding Task Understanding then raises both scores, and adding Goal-State Prediction provides a further 1.99-point OOD increase with a smaller 0.23-point ID increase. These observations support the full mixture as a stronger Stage-2 recipe for the evaluated instruction shifts. The exposure schedule changes alongside the objective set, with Policy exposure falling from 1.0 to 0.65, so each increment reflects both the added targets and a reallocation of supervision.

All variants inherit the Stage-1 checkpoint and use a decoder that consumes both goal and next-state predictions. The objective mixture can therefore improve the predictions used for action generation as well as the action predictor itself. The observed gain reflects the combined effect of adapting these interfaces under the shared inference procedure.

5.6.3 Inference-Strategy Ablation

Table 12 Inference compositions on VLABench using the same post-trained checkpoint. The variants allocate inference computation to action decoding, goal prediction, joint action–next-state refinement, and world-model-based candidate reranking. Variant (c) is the default path used for the reported LIBERO evaluations. ID denotes Track 1 success, OOD averages Tracks 3 and 4, and Gap is ID minus OOD in percentage points. Effective TPS follows the model-side timing scope defined in Section 5.5. Underlining marks the highest success rates and throughput and the smallest gap.

Inference variant	ID (%) $\uparrow$	OOD (%) $\uparrow$	Gap (%p) $\downarrow$	Effective TPS $\uparrow$
(a) Action-Only Policy	45.8	41.4	4.4	<u>9.238</u>
(b) Action/World-Model Joint Denoising	46.4	40.5	5.9	8.805
(c) Goal-State-Guided Policy (default)	45.6	38.2	7.4	9.208
(d) World-Model Filtering	46.3	42.0	4.3	4.904
(e) Goal-Guided Joint Action/World Denoising	<u>49.6</u>	<u>47.3</u>	<u>2.3</u>	<u>8.709</u>
(f) Goal-Guided World-Model Filtering	48.9	<u>47.7</u>	<u>1.2</u>	4.828

Inference composition determines whether the checkpoint’s auxiliary outputs improve policy success. Table 12 holds the post-trained checkpoint fixed. Action-only decoding (a) obtains 45.8% ID and 41.4% OOD success. Goal-guided Policy (c), the default path used for the reported LIBERO evaluations, obtains 45.6% and 38.2%; its OOD result is 3.2 percentage points below action-only decoding. Joint action–next-state denoising without goal guidance (b) also has a lower OOD score, 40.5%. Neither auxiliary path in isolation improves OOD success in this comparison.

Combining goal guidance with joint denoising (e) increases ID and OOD success to 49.6% and 47.3%, gains of 3.8 and 5.9 percentage points over action-only decoding. Goal guidance thus has different effects in the two decoding settings: it reduces OOD success when added to Policy decoding but improves it when added to joint action–next-state denoising. This pattern supports composing the goal and next-state interfaces together for the tested instruction shifts. Although (c) is used in the reported LIBERO benchmark protocol, the VLABench comparison favors (e) and (f) in absolute success.

The compositions allocate additional inference computation in different ways and yield distinct performance–cost trade-offs. Variant (e) reduces effective throughput from 9.238 TPS for action-only decoding to 8.709 TPS, a 5.7% decrease. Goal-guided world-model filtering (f) achieves the highest reported OOD success, 47.7%, at 4.828 TPS. Compared with (e), this is a 0.4-point OOD increase, a 0.7-point ID decrease, and approximately 44.6% lower throughput. These variants instantiate test-time scaling through additional visual prediction and candidate evaluation. The comparison shows that the return on additional computation depends on its allocation: the combined goal-guided joint decoder improves success at a modest throughput cost, while filtering incurs a larger cost for a smaller additional OOD gain.

5.7 Hyperparameter Sensitivity

We examine three operational choices: the resolution of action discretization, the granularity of token commitment during denoising, and the relative strength of auxiliary losses. The action-bin and loss-scale

comparisons use the 70k-step VLABench Stage-2 protocol, while decoding is evaluated on LIBERO-Goal. These studies identify operating points and characterize performance sensitivity within the tested settings.

5.7.1 Action Token Binning

Table 13 Action-bin sensitivity under the 70k-step VLABench Stage-2 protocol. Only the active Stage-2 bin count B_A changes; Action MAE uses normalized action space, and ID/OOD are success rates. The Stage-1 initialization uses 256 bins. The best value in each metric is underlined.

B_A	Action MAE ↓	Validation Token Acc ↑	ID (%) ↑	OOD (%) ↑
16	0.09	0.75	27.84	25.05
32	<u>0.02</u>	<u>0.89</u>	<u>49.61</u>	<u>47.28</u>
64	0.05	0.58	32.90	29.86

Finer action discretization does not monotonically improve downstream performance. In Table 13, 32 bins yield the lowest normalized action MAE (0.02) and the highest rollout success: 49.61% ID and 47.28% OOD. Both 16 and 64 bins increase MAE and reduce rollout success. We therefore use 32 bins as the operating point for this VLABench Stage-2 comparison.

Continuous-action error and rollout success provide a common basis for comparing bin counts. Exact token accuracy uses a different number of categories at each resolution and therefore measures a different classification problem. The Stage-1 model uses 256 bins, so selecting a downstream bin count also changes the mapping between action-token indices and continuous values during adaptation. The sweep evaluates these discretization choices within the robot-pretrained adaptation pipeline, where the intermediate resolution of 32 bins yields the lowest normalized action error and highest rollout success.

5.7.2 Denoising and Block-Parallel Decoding

The decoding study tests how reductions in sequential denoising work translate into latency and policy success. For the fixed 35-token chunk, BL7 configures blocks at the granularity of one 7-DoF action, while BL35 spans all five actions. The block length is a decoding setting; the measured number of backbone evaluations captures the realized refinement work.

Table 14 Denoising and control trade-offs on LIBERO-Goal under the common one-B200 profiling protocol. Evals./chunk is the mean number of backbone evaluations; amortized latency assumes execution of all five predicted actions. Success is the mean $\pm$ standard deviation, and the best value in each metric is underlined.

Configuration	Evals./chunk ↓	Query (ms) ↓	Amort. (ms/action) ↓	Success (%) ↑
Dynin-Robotics-base	35.00	3,795.7	759.1	97.8 $\pm$ 0.4
Dynin-Robotics-dInfer-BL7	5.14	383.6	76.7	97.7 $\pm$ 0.2
Dynin-Robotics-dInfer-BL35	<u>1.73</u>	<u>130.2</u>	<u>26.0</u>	97.2 $\pm$ 0.4

Table 14 connects the offline comparison in Table 9 to LIBERO-Goal rollouts. BL7 reduces the mean backbone evaluations per chunk from 35.00 to 5.14 and query latency from 3,795.7 to 383.6 ms. Its mean success rate is 97.7%, compared with 97.8% for the base decoder. BL35 further reduces the means to 1.73 evaluations and 130.2 ms, with a success rate of 97.2%. The observed cost reductions therefore accompany a 0.1-point success decrease for BL7 and a 0.6-point decrease for BL35. The table reports the variability of the success measurements alongside these means.

The latency figures distinguish production of an action chunk from responding to a new observation. Dividing query latency by five gives amortized costs of 76.7 and 26.0 ms per action for BL7 and BL35, assuming that all five actions are executed before replanning. With one-step replanning, each decision instead incurs the full chunk-generation latency. These measurements consequently support faster model-side action production, while the achievable feedback rate depends on how the decoder is integrated with sensing and execution. Among the tested settings, BL7 offers a smaller observed success change and BL35 a shorter query latency.

5.7.3 Auxiliary-Objective Loss Scale

Table 15 Auxiliary-loss-scale sensitivity under the 70k-step VLABench Stage-2 protocol. The Policy, World Modeling, Task Understanding, and Goal-State Prediction exposure ratio is fixed at 0.65 : 0.20 : 0.05 : 0.10 with $\lambda_{PO} = 1$; α scales the three auxiliary losses jointly. The best value in each metric is underlined.

α	$(\lambda_{PO}, \lambda_{WM}, \lambda_{TU}, \lambda_{GP})$	ID (%) $\uparrow$	OOD (%) $\uparrow$	Gap (%p) $\downarrow$
0.5 $\times$	(1, 0.5, 0.5, 0.5)	49.42	45.91	3.51
1.0 $\times$	(1, 1, 1, 1)	<u>49.61</u>	<u>47.28</u>	<u>2.33</u>
2.0 $\times$	(1, 2, 2, 2)	38.05	36.70	<u>1.35</u>

The loss-scale comparison separates objective exposure from the strength of the corresponding updates. In Table 15, the Policy, World Modeling, Task Understanding, and Goal-State Prediction exposure ratio remains 0.65 : 0.20 : 0.05 : 0.10, Policy loss weight is fixed at one, and α scales the three auxiliary losses together. Equal loss weights ($\alpha = 1$) yield the highest observed ID and OOD success, 49.61% and 47.28%. Halving the auxiliary weights gives similar ID success (49.42%) and a lower OOD score (45.91%), whereas doubling them reduces both scores substantially, to 38.05% and 36.70%.

The largest performance change occurs when auxiliary losses are upweighted to $\alpha = 2$. Although this setting has the smallest ID–OOD gap, 1.35 points, it also has the lowest success in both conditions. Equal weighting provides the best observed balance of ID and OOD success among the tested settings, with a smaller performance difference from $\alpha = 0.5$ than from $\alpha = 2$. This pattern indicates that the training recipe is more sensitive to increasing auxiliary weights above one than to halving them.

6 Conclusion

We presented Dynin-Robotics, a unified vision-language-action model built on Dynin-Omni that expresses robot control, visual prediction, and trajectory understanding through a shared masked-diffusion formulation. Representing language, visual states, goals, and actions in a common trajectory sequence allows the model to learn complementary prediction tasks and reuse its visual predictions during action generation and selection. The shared interface connects multi-objective robot learning with inference procedures that allocate computation to goal prediction, joint refinement, and candidate evaluation.

On two VLABench tasks, robot continual pretraining improves adaptation within a fixed Stage-2 step budget, and the full objective mixture yields higher shifted-instruction success than Policy-only post-training under the same coupled decoder. With the checkpoint fixed, combining goal guidance with joint action–next-state denoising improves shifted-instruction success over action-only decoding, whereas either component alone yields lower shifted-instruction success than action-only decoding. The benefit of using visual predictions therefore depends on how they are composed during inference. Evaluations on LIBERO, zero-shot LIBERO-Plus, and a Franka Research 3 platform show competitive policy performance across the reported settings. An optimized block-parallel implementation also provides up to a 29.2 $\times$ speedup in model-side action decoding over the base implementation under the profiling protocol.

The evaluation covers adjacent-future action-conditioned World Modeling and terminal Goal-State Prediction. Frame copying remains stronger on the reported whole-image metrics for adjacent-future prediction, motivating direct evaluation of action-dependent changes and their relationship to control performance. Task Understanding is evaluated qualitatively, and the physical study covers one robot platform and workspace. Extending visual dynamics evaluation to longer horizons, quantifying trajectory-to-language accuracy, and studying additional embodiments and sensor modalities are natural next steps for the shared trajectory formulation.

References

- [1] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR, 2023.
- [2] 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 The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 2025.
- [3] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael 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, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0 : A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.
- [4] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 17–40. PMLR, 2025.
- [5] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.
- [6] Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025.
- [7] Chris L Baker, Rebecca Saxe, and Joshua B Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009.
- [8] Daniel M Wolpert, Zoubin Ghahramani, and Michael I Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995.
- [9] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2786–2793. IEEE, 2017.
- [10] Markku Suomalainen, Yiannis Karayiannidis, and Ville Kyrki. A survey of robot manipulation in contact. *Robotics and Autonomous Systems*, 156:104224, 2022.
- [11] Soujanya Poria, Navonil Majumder, Chia-Yu Hung, Amir Ali Bagherzadeh, Chuan Li, Kenneth Kwok, Ziwei Wang, Cheston Tan, Jiajun Wu, and David Hsu. 10 open challenges steering the future of vision-language-action models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 39771–39779, 2026.
- [12] Kento Kawaharazuka, Jihoon Oh, Jun Yamada, Ingmar Posner, and Yuke Zhu. Vision-language-action models for robotics: A review towards real-world applications. *IEEE Access*, 2025.
- [13] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

- [14] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1702–1713, 2025.
- [15] 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. *arXiv preprint arXiv:2501.09747*, 2025.
- [16] Zhixuan Liang, Yizhuo Li, Tianshuo Yang, Chengyue Wu, Sitong Mao, Tian Nian, Liuaio Pei, Shunbo Zhou, Xiaokang Yang, Jiangmiao Pang, et al. Discrete diffusion vla: Bringing discrete diffusion to action decoding in vision-language-action policies. *arXiv preprint arXiv:2508.20072*, 2025.
- [17] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.
- [18] Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, Junjin Xiao, Haoyun Liu, Ronghan Chen, Yuzhi Chen, Dongjie Huo, et al. Abot-m0: Vla foundation model for robotic manipulation with action manifold learning. *arXiv preprint arXiv:2602.11236*, 2026.
- [19] Wei Li, Renshan Zhang, Rui Shao, Jie He, and Liqiang Nie. Cogvla: Cognition-aligned vision-language-action model via instruction-driven routing & sparsification. *arXiv preprint arXiv:2508.21046*, 2025.
- [20] Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. Vla-jepa: Enhancing vision-language-action model with latent world model. *arXiv preprint arXiv:2602.10098*, 2026.
- [21] Haonan Chen, Jingxiang Guo, Bangjun Wang, Tianrui Zhang, Xuchuan Huang, Boren Zheng, Yiwen Hou, Chenrui Tie, Jiajun Deng, and Lin Shao. Goal-vla: Image-generative vlms as object-centric world models empowering zero-shot robot manipulation. *arXiv preprint arXiv:2506.23919*, 2025.
- [22] Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.
- [23] Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified World Models: Coupling Video and Action Diffusion for Pretraining on Large Robotic Datasets, May 2025. *arXiv:2504.02792 [cs]*.
- [24] Jonas Pai, Liam Achenbach, Victoriano Montesinos, Benedek Forrai, Oier Mees, and Elvis Nava. mimic-video: Video-action models for generalizable robot control beyond vlms. *arXiv preprint arXiv:2512.15692*, 2025.
- [25] Junbang Liang, Pavel Tokmakov, Ruoshi Liu, Sruthi Sudhakar, Paarth Shah, Rares Ambrus, and Carl Vondrick. Video generators are robot policies. *arXiv preprint arXiv:2508.00795*, 2025.
- [26] Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.
- [27] Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.
- [28] Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, et al. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11142–11152, 2025.
- [29] Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified Vision-Language-Action Model, June 2025. *arXiv:2506.19850 [cs]*.
- [30] Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.
- [31] Jiayi Chen, Wenxuan Song, Pengxiang Ding, Ziyang Zhou, Han Zhao, Feilong Tang, Donglin Wang, and Haoang Li. Unified diffusion vla: Vision-language-action model via joint discrete denoising diffusion process. *arXiv preprint arXiv:2511.01718*, 2025.

- [32] Junjie Wen, Minjie Zhu, Jiaming Liu, Zhiyuan Liu, Yicun Yang, Linfeng Zhang, Shanghang Zhang, Yichen Zhu, and Yi Xu. dvla: Diffusion vision-language-action model with multimodal chain-of-thought. [arXiv preprint arXiv:2509.25681](#), 2025.
- [33] Jun Cen, Siteng Huang, Yuqian Yuan, Kehan Li, Hangjie Yuan, Chaohui Yu, Yuming Jiang, Jiayan Guo, Xin Li, Hao Luo, et al. Rynnvla-002: A unified vision-language-action and world model. [arXiv preprint arXiv:2511.17502](#), 2025.
- [34] Yang Liu, Pengxiang Ding, Tengyue Jiang, Xudong Wang, Wenxuan Song, Minghui Lin, Han Zhao, Hongyin Zhang, Zifeng Zhuang, Wei Zhao, et al. Mmada-vla: Large diffusion vision-language-action model with unified multi-modal instruction and generation. [arXiv preprint arXiv:2603.25406](#), 2026.
- [35] Jianke Zhang, Yanjiang Guo, Yucheng Hu, Xiaoyu Chen, Xiang Zhu, and Jianyu Chen. UP-VLA: A Unified Understanding and Prediction Model for Embodied Agent, June 2025. [arXiv:2501.18867 \[cs\]](#).
- [36] Jaeik Kim, Woojin Kim, Jihwan Hong, Yejoon Lee, Sieun Hyeon, Mintaek Lim, Yunseok Han, Dogeun Kim, Hoeun Lee, Hyunggeun Kim, et al. Dynin-omni: Omnimodal unified large diffusion language model. [arXiv preprint arXiv:2604.00007](#), 2026.
- [37] Yuxin Ma, Lun Du, Lanning Wei, Kun Chen, Qian Xu, Kangyu Wang, Guofeng Feng, Guoshan Lu, Lin Liu, Xiaojing Qi, Xinyuan Zhang, Zhen Tao, Haibo Feng, Ziyun Jiang, Ying Xu, Zenan Huang, Yihong Zhuang, Haokai Xu, Jiaqi Hu, Zhenzhong Lan, Junbo Zhao, Jianguo Li, and Da Zheng. dinfer: An efficient inference framework for diffusion language models, 2025.
- [38] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, et al. DROID: A large-scale in-the-wild robot manipulation dataset. [arXiv preprint arXiv:2403.12945](#), 2024.
- [39] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. [Advances in Neural Information Processing Systems](#), 36:44776–44791, 2023.
- [40] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. [arXiv preprint arXiv:2510.13626](#), 2025.
- [41] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. [arXiv preprint arXiv:2303.08774](#), 2023.
- [42] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. [arXiv preprint arXiv:2407.21783](#), 2024.
- [43] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. [arXiv preprint arXiv:2412.15115](#), 2024.
- [44] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. [arXiv preprint arXiv:2412.19437](#), 2024.
- [45] Lukas Berglund, Meg Tong, Maximilian Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: LLMs trained on “a is b” fail to learn “b is a”. In [The Twelfth International Conference on Learning Representations](#), 2024.
- [46] Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. [arXiv preprint arXiv:1503.03585](#), 2015.
- [47] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. [arXiv preprint arXiv:2006.11239](#), 2020.
- [48] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. [Advances in Neural Information Processing Systems](#), 34:12454–12465, 2021.
- [49] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. [Advances in Neural Information Processing Systems](#), 34:17981–17993, 2021.

- [50] Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. Diffusion-LM Improves Controllable Text Generation. In *Proc. of Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.
- [51] Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. DiffuSeq: Sequence to Sequence Text Generation with Diffusion Models. In *Proc. of Int’l Conf. on Learning Representations (ICLR)*, 2022.
- [52] Zhengfu He, Tianxiang Sun, Qiong Tang, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. DiffusionBERT: Improving Generative Masked Language Models with Diffusion Models. In *Proc. of Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.
- [53] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proc. of Int’l Conf. on Machine Learning (ICML)*, 2024.
- [54] Subham Sekhar Sahoo, Marianne Arriola, Aaron Gokaslan, Edgar Mariano Marroquin, Alexander M Rush, Yair Schiff, Justin T Chiu, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Proc. of Neural Information Processing Systems (NeurIPS)*, 2024.
- [55] Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.
- [56] Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models, 2025.
- [57] Yuqing Wen, Hebei Li, Kefan Gu, Yucheng Zhao, Tiancai Wang, and Xiaoyan Sun. Llada-vla: Vision language diffusion action models. *arXiv preprint arXiv:2509.06932*, 2025.
- [58] Runpei Dong, Chunrui Han, Yuang Peng, Zekun Qi, Zheng Ge, Jinrong Yang, Liang Zhao, Jianjian Sun, Hongyu Zhou, Haoran Wei, et al. Dreamllm: Synergistic multimodal comprehension and creation. *arXiv preprint arXiv:2309.11499*, 2023.
- [59] Yuying Ge, Sijie Zhao, Jinguo Zhu, Yixiao Ge, Kun Yi, Lin Song, Chen Li, Xiaohan Ding, and Ying Shan. Seed-x: Multimodal models with unified multi-granularity comprehension and generation. *arXiv preprint arXiv:2404.14396*, 2024.
- [60] Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. NExT-GPT: Any-to-any multimodal LLM. In *Proceedings of the International Conference on Machine Learning*, pages 53366–53397, 2024.
- [61] Zineng Tang, Ziyi Yang, Mahmoud Khademi, Yang Liu, Chenguang Zhu, and Mohit Bansal. Codi-2: In-context interleaved and interactive any-to-any generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 27425–27434, June 2024.
- [62] NAVER Cloud HyperCLOVA X Team. Hyperclova x 8b omni. *arXiv preprint arXiv:2601.01792*, 2026.
- [63] Jun Zhan, Junqi Dai, Jiasheng Ye, Yunhua Zhou, Dong Zhang, Zhigeng Liu, Xin Zhang, Ruibin Yuan, Ge Zhang, Linyang Li, Hang Yan, Jie Fu, Tao Gui, Tianxiang Sun, Yu-Gang Jiang, and Xipeng Qiu. AnyGPT: Unified multimodal LLM with discrete sequence modeling. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9637–9662, Bangkok, Thailand, August 2024. Association for Computational Linguistics.
- [64] Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models, 2024. URL <https://arxiv.org/abs/2405.09818>, 9(8), 2024.
- [65] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiyang Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024.
- [66] Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. Janus-pro: Unified multimodal understanding and generation with data and model scaling. *arXiv preprint arXiv:2501.17811*, 2025.
- [67] Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.
- [68] Jinheng Xie, Zhenheng Yang, and Mike Zheng Shou. Show-o2: Improved native unified multimodal models. *arXiv preprint arXiv:2506.15564*, 2025.

- [69] Alexander Swerdlow, Mihir Prabhudesai, Siddharth Gandhi, Deepak Pathak, and Katerina Fragkiadaki. Unified multimodal discrete diffusion, 2025.
- [70] Ling Yang, Ye Tian, Bowen Li, Xincheng Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. MMaDA: Multimodal large diffusion language models. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025.
- [71] Jin Wang, Yao Lai, Aoxue Li, Shifeng Zhang, Jiacheng Sun, Ning Kang, Chengyue Wu, Zhenguo Li, and Ping Luo. Fudoki: Discrete flow-based unified understanding and generation via kinetic-optimal velocities. arXiv preprint arXiv:2505.20147, 2025.
- [72] Shufan Li, Jiuxiang Gu, Kangning Liu, Zhe Lin, Zijun Wei, Aditya Grover, and Jason Kuen. Lavid-a: Elastic large masked diffusion models for unified multimodal understanding and generation. arXiv preprint arXiv:2509.19244, 2025.
- [73] Yi Xin, Qi Qin, Siqi Luo, Kaiwen Zhu, Juncheng Yan, Yan Tai, Jiayi Lei, Yuewen Cao, Keqi Wang, Yibin Wang, Jinbin Bai, Qian Yu, Dengyang Jiang, Yuandong Pu, Haoxing Chen, Le Zhuo, Junjun He, Gen Luo, Tianbin Li, Ming Hu, Jin Ye, Shenglong Ye, Bo Zhang, Chang Xu, Wenhai Wang, Hongsheng Li, Guangtao Zhai, Tianfan Xue, Bin Fu, Xiaohong Liu, Yu Qiao, and Yihao Liu. Lumina-dimoo: An omni diffusion large language model for multi-modal generation and understanding, 2025.
- [74] Inclusion AI, Tiwei Bie, Haoxing Chen, Tiejun Chen, Zhenglin Cheng, Long Cui, Kai Gan, Zhicheng Huang, Zhenzhong Lan, Haoquan Li, Jianguo Li, Tao Lin, Qi Qin, Hongjun Wang, Xiaomei Wang, Haoyuan Wu, Yi Xin, and Junbo Zhao. Llada2.0-uni: Unifying multimodal understanding and generation with diffusion large language model, 2026.
- [75] Qi Lv, Weijie Kong, Hao Li, Jia Zeng, Zherui Qiu, Delin Qu, Haoming Song, Qizhi Chen, Xiang Deng, and Jiangmiao Pang. F1: A vision-language-action model bridging understanding and generation to actions. arXiv preprint arXiv:2509.06951, 2025.
- [76] InternVLA-M1 Contributors. Internvla-m1: A spatially guided vision-language-action framework for generalist robot policy. arXiv preprint arXiv:2510.13778, 2025.
- [77] Chia-Yu Hung, Qi Sun, Pengfei Hong, Amir Zadeh, Chuan Li, U Tan, Navonil Majumder, Soujanya Poria, et al. Nora: A small open-sourced generalist vision language action model for embodied tasks. arXiv preprint arXiv:2504.19854, 2025.
- [78] Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models. arXiv preprint arXiv:2505.17016, 2025.
- [79] Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, et al. Magvit: Masked generative video transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10459–10469, 2023.
- [80] Kai Chen, Yunhao Gou, Runhui Huang, Zhili Liu, Daxin Tan, Jing Xu, Chunwei Wang, Yi Zhu, Yihan Zeng, Kuo Yang, et al. Emova: Empowering language models to see, hear and speak with vivid emotions. arXiv preprint arXiv:2409.18042, 2024.
- [81] Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. In The Fourteenth International Conference on Learning Representations, 2026.
- [82] Open X-Embodiment Collaboration. Open x-embodiment: Robotic learning datasets and RT-X models. arXiv preprint arXiv:2310.08864, 2023.
- [83] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In The Twelfth International Conference on Learning Representations, 2024.
- [84] Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In The Thirteenth International Conference on Learning Representations, 2025.
- [85] Andrew Wagenmaker, Yunchu Zhang, Mitsuhiko Nakamoto, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, Proceedings of The 9th Conference on Robot Learning, volume 305 of Proceedings of Machine Learning Research, pages 258–282. PMLR, 2025.

- [86] Shu-Ang Yu, Feng Gao, Yi Wu, Chao Yu, and Yu Wang. D3P: Dynamic denoising diffusion policy via reinforcement learning, 2025.
- [87] Yuhao Wu, Yitian Liu, Weijie Shen, Mishuo Han, Wenjie Xu, Haotian Liang, Zhongshan Liu, Yinan Mao, Lei Xu, Xinpeng Guan, Ru Ying, Ran Zheng, Wei Sui, Xiaokang Yang, Wenbo Ding, and Yao Mu. dVLA-RL: Reinforcement learning over denoising trajectories for discrete diffusion vision-language-action models, 2026.
- [88] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.
- [89] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy, 2023.