LaLa-World: Scaling Latent World Models with Natural Language Actions

Anonymous authors
Paper under double-blind review

Abstract

This paper shows that JEPA-style latent world models (LWMs) can learn across diverse domains and data sources by adopting the language-conditioned interface of video generation models. We establish this approach as an effective and efficient basis for planning with imagination and downstream policy fine-tuning. To this end, we introduce **LaLa-World**, a **La**nguage-conditioned **La**tent **World** model that learns from simulation and robot trajectories together with broader image and video data. Two designs make this work. First, we describe both numerical actions and visual changes in language, enabling joint training of a single future predictor across heterogeneous datasets. Second, we build the predictor on a language model, allowing it to interpret language conditions while predicting future latent states. The resulting predictor achieves higher average MPC planning performance than existing LWMs trained separately for each domain or task. Adding image and video data further improves average planning performance and robustness to action phrasing. In a matched comparison with an RAEv2-based video generation model baseline, LaLa-World reaches comparable planning performance with $6\times$ less training cost and $13.9\times$ faster planning. Beyond planning, the pretrained LaLa-World serves as an initialization for LaLa-Policy, which jointly predicts actions and future visual representations. LaLa-Policy performs competitively with Cosmos Policy on LIBERO and RoboCasa, with up to $5.5\times$ faster inference.

1 Introduction

A central capability of an intelligent agent is to anticipate how the world will change in response to its actions. Before pushing a cup across a table, we know where it will end up and whether it might fall. World models capture this capability by predicting future states under candidate actions, providing a basis for planning or decision-making (Sutton, 1991; Ha & Schmidhuber, 2018; LeCun, 2022; Hafner et al., 2020). A promising approach to making this predictive capability computationally efficient is to model how visual representations change under actions, without learning to reconstruct images at the pixel level. This motivates our focus on JEPA-style latent world models (LWMs), such as V-JEPA 2-AC (Assran et al., 2025) and DINO-WM (Zhou et al., 2025).[1]

This efficiency of LWMs is particularly useful for planning with imagination in the latent space, which requires repeatedly evaluating candidate action sequences, by rolling out multiple candidate futures at low cost (Hansen et al., 2022). However, despite this advantage, the action-conditioned predictors in such LWMs have largely been trained on simulation or robotics datasets, where numerical actions have domain-specific dimensions and physical meanings. For example, $[2, 4]$ may encode horizontal and vertical displacement in one domain and $[4]$ force in another, complicating the learning of shared dynamics. Furthermore, image and video data, which provide a rich source of observations of the world, often lack these numerical action annotations, making them difficult to use for training action-conditioned future predictors. Therefore, scaling LWMs across diverse control actions and broader image and video data requires a shared interface that accommodates heterogeneous data.

We suggest that language can serve as the shared interface. We introduce LaLa-World, a language-conditioned LWM that learns from diverse control datasets and broader image and video data with a

[1] Throughout this paper, LWMs refer to JEPA-style world models trained without DreamerV3's reconstruction loss (Hafner et al., 2025a) or TD-MPC2's reward supervision (Hansen et al., 2024).

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

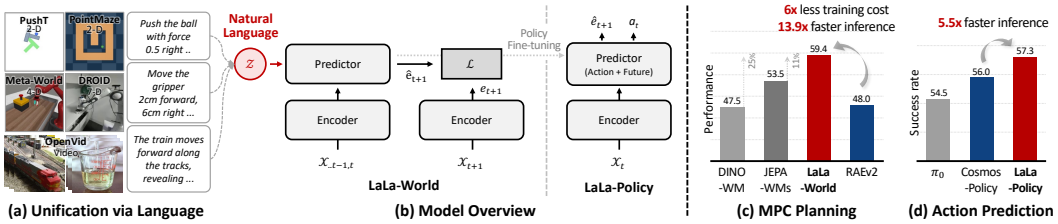

Figure 1: **Overview of LaLa-World. (a)** Language provides a shared interface for numerical actions and visual change, enabling joint training on control and video data. **(b)** A single predictor built on a language model uses encoded observations and language conditions to predict future visual representations. Fine-tuning the pretrained LaLa-World yields LaLa-Policy, which predicts actions as well as future visual representations. **(c)** The jointly trained predictor achieves higher average MPC planning performance than both separately trained LWM baselines and the generative world model baseline, RAEv2. **(d)** LaLa-Policy performs competitively with our adapted Cosmos Policy baseline on RoboCasa with 50 demonstrations per task, while reducing average inference time per trajectory by a factor of $5.5\times$.

single future predictor. Language conditioning already supports learning from diverse visual data in video generation (Wan et al., 2025; Agarwal et al., 2025). We use language as a shared interface for numerical actions and visual changes and can thereby scale LWM training with a single predictor across diverse contexts and actions.

As illustrated in Figure 1, LaLa-World has two main designs. First, we convert numerical actions in simulation or robotics datasets into natural language, allowing them to be expressed in the same form, such as `move 2 left and 4 up` and `push with force 2 downward`. We combine these control trajectories with image and video transitions described in language to help the predictor relate descriptions to visual changes across diverse scenes. Second, we equip the future predictor with language understanding. Built on a language model, it interprets language conditions while predicting future latent states. These designs allow one model to learn from different action spaces and data sources through a shared prediction objective. Figure 2 shows that a single LaLa-World predictor serves multiple domains and predicts future states under qualitative actions such as `move slightly left`.

Our experiments show that our jointly trained predictor achieves higher average MPC planning performance than existing LWM baselines trained separately for each domain. We also show that adding image and video data to training further improves planning performance and robustness to action phrasing. We compare LaLa-World with an RAEv2-based generative world model (Singh et al., 2026), whose predictor uses a diffusion Transformer (DiT). With the same language conditions, training data, and visual encoder, LaLa-World outperforms this RAEv2 baseline in planning with $6\times$ less training cost and $13.9\times$ faster inference.

Beyond planning, we investigate whether the pretrained LWM can serve as a useful initialization for policy fine-tuning. Cosmos Policy (Kim et al., 2026) demonstrates this use of a pretrained video model by adapting Cosmos-Predict2 (Ali et al., 2025) to predict robot actions together with future *frames*. Similarly, we fine-tune LaLa-World into LaLa-Policy for joint prediction of actions and future visual *representations*. On LIBERO and RoboCasa, LaLa-Policy performs competitively with Cosmos Policy while reducing average inference time per trajectory by a factor of up to 5.5. We hope these results draw greater attention to latent world models, showing that even without visually compelling outputs, they can offer practical advantages over generative world models in planning and downstream policy learning.

We summarize our contributions as follows:

- We introduce LaLa-World, a latent world model that uses language as a shared interface for joint training on control trajectories and broader image and video data.
- We show that a single LaLa-World predictor learns dynamics across domains and outperforms existing LWMs in MPC planning.
- We compare LaLa-World with an RAEv2-based generative world model under matched settings, demonstrating higher planning performance and greater training and inference efficiency.
- We fine-tune the pretrained LaLa-World into LaLa-Policy, which performs competitively with existing policy models while achieving faster inference than Cosmos Policy.

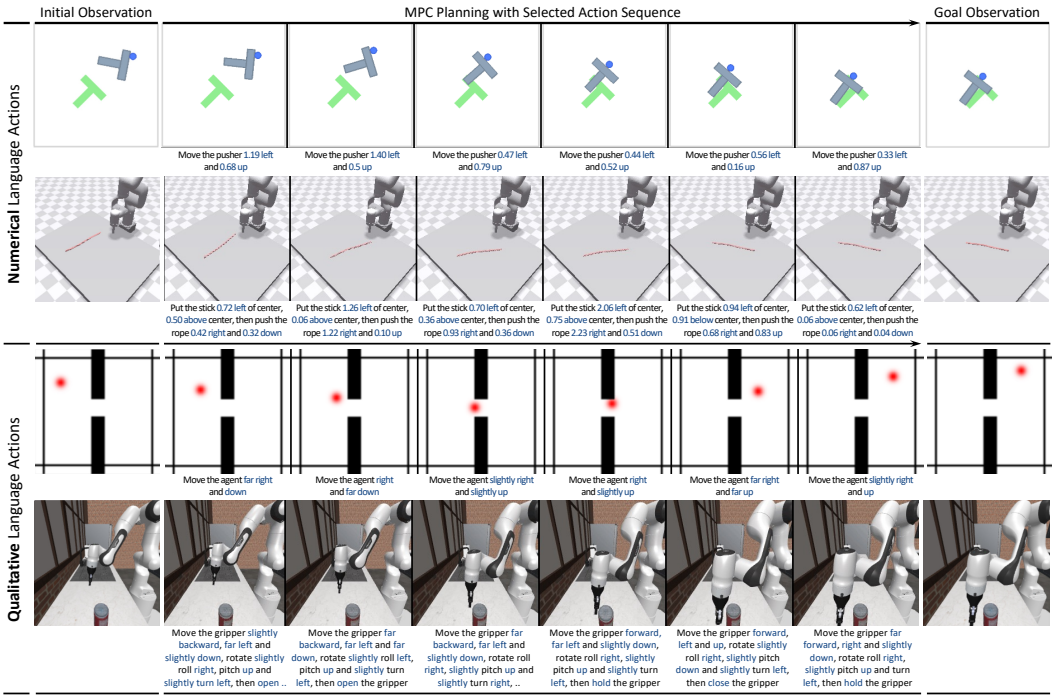

Figure 2: **MPC planning with language actions.** LaLa-World imagines future representations under candidate actions with MPC and selects the most promising action toward the goal. Because all actions share one language interface, a single LaLa-World predictor can plan across multiple domains, despite differences in action dimensionality and physical meaning. The same predictor also supports qualitative language actions without numerical values, as shown in the bottom two rows and detailed in Section D.4.3. The visualized observations are obtained by executing the selected actions in the simulator.

2 Background and Motivation

World models aim to predict how an environment evolves under a given condition (Ha & Schmidhuber, 2018; LeCun, 2022; Wan et al., 2025; Agarwal et al., 2025). One approach to making future prediction computationally efficient is to predict changes in latent representations. Latent World Models (LWMs), including V-JEPA 2-AC (Assran et al., 2025) and DINO-WM (Zhou et al., 2025), follow this approach. These models commonly use numerical actions in the following action-conditioned formulation:

$$e_t = E_\theta(x_t), \qquad \hat{e}_{t+1} = P_\phi\left(e_{t-w+1:t}, c_{t-w+1:t}\right), \quad c_t \in \mathbb{R}^k, \tag{1}$$

where x_t and c_t denote the observation and k-dimensional numerical action condition at time t, respectively, E_θ encodes observations into latent representations, and P_ϕ predicts the future latent $\hat{e}_{t+1}$ from a context window of length w. The action-conditioned predictors in these models are commonly trained on simulation or robotics trajectories and evaluated through planning in a domain-specific manner (Sobal et al., 2025; Maes et al., 2026a;b)

Video generation models (*i.e.*, generative world models) are trained to synthesize images or videos conditioned on language (Brooks et al., 2024; Wan et al., 2025; Agarwal et al., 2025). This language interface allows them to learn visual changes across domains and data sources. Through this training, these models acquire world knowledge that extends their capabilities beyond video generation. Their pretrained weights, for instance, serve as a prior for action prediction: Cosmos Policy fine-tunes Cosmos-Predict2 into a robot policy (Kim et al., 2026; Ali et al., 2025).

For LWMs, however, learning jointly from heterogeneous control datasets and broader visual data remains underexplored, as does their use in downstream policy learning. We therefore explore an approach that retains the efficiency of LWMs while adopting the flexible language interface of generative world models. The resulting model supports evaluation against existing LWMs in planning and against generative world models in both planning and policy fine-tuning.

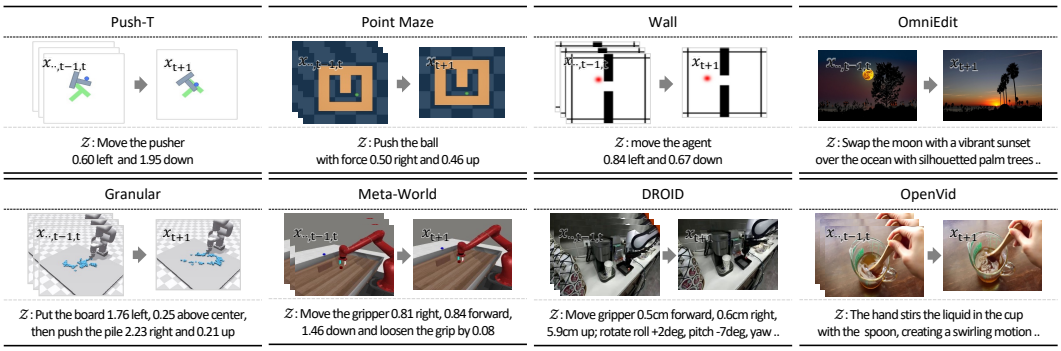

Figure 3: **Examples of language-conditioned transition data.** Each example consists of past observations, a language condition, and the next observation. Language conditions describe either numerical control actions from simulation and robotics or visual changes in images and videos. This shared interface enables training a single LWM across different action spaces and data sources.

3 LaLa-World

We introduce **LaLa-World**, a **La**nguage-conditioned **La**tent **World** model whose predictor takes a language condition z instead of the numerical action condition c in Equation (1). Two components are needed: transition data that describes the change from x_t to x_{t+1} in language, and a predictor that understands it. Section 3.1 covers the first, describing how we construct such language-conditioned data, and Section 3.2 covers the second with the architecture and training objective.

3.1 Language-Conditioned Transition Data

Language unifies the action spaces of multiple domains. Simulation and robotics datasets used to train LWMs record actions as numerical vectors. We convert these vectors into language conditions using fixed templates, as illustrated in Figure 3, retaining each domain's numerical values with their direction and magnitude. Language thus expresses actions across domains in a common form, allowing their control data to be pooled to train a single predictor. We apply this conversion to seven simulation and robotics datasets described in Section 4. The actions in these domains may differ in physical meaning and dimensionality. This raises the key question of whether a single future predictor can learn dynamics across these domains. Details of the dataset composition and conversion templates are provided in Section D.1.

Language enables action-conditioned training on broader image and video data. We also train the same future predictor on image and video transitions without numerical action annotations, using descriptions of visual change as conditions. Such data matter because the converted control datasets describe actions with fixed templates, which exposes the predictor to only a narrow range of language. We therefore supplement them with captioned videos from OpenVid (Nan et al., 2025) and image-editing pairs with instructions from OmniEdit (Wei et al., 2025) and ShareGPT (Chen et al., 2025b). In these data, the caption describing the change between frames or the editing instruction serves directly as the language condition, so no numerical action annotation is needed. The predictor thereby sees a wider range of language and visual change.

3.2 Model Architecture and Objective

As shown in Figure 1(a), LaLa-World follows the encoder-predictor framework of existing LWMs, with simple adaptations for language conditioning. Below, we describe the observation encoder (*i.e.*, vision encoder), transition predictor (*i.e.*, future predictor), and training objective.

Encoder. LWMs use an observation encoder to map each RGB frame x_t to a latent representation e_t (Assran et al., 2025; Terver et al., 2025). For this role, vision transformers pretrained with self-supervised objectives are commonly used, including DINO encoders (Oquab et al., 2024; Siméoni et al., 2025) and V-JEPA 2 encoder (Assran et al., 2025). These encoders are pretrained on large collections of unlabeled images and videos, providing visual representations for diverse environments (Zhou et al., 2025). In V-JEPA 2-AC, the encoder remains frozen during world

Method	Joint-training	Wall	PointMaze	Push-T	MetaWorld-R	MetaWorld-RW	Average
DreamerV3 (2025a)	✗	39.6 (5.0)	43.8 (5.1)	32.3 (4.8)	25.0 (6.2)	10.4 (4.4)	30.2 (2.3)
TD-MPC2 (2024)	✗	28.1 (4.6)	8.3 (2.8)	16.7 (3.8)	12.5 (4.8)	22.9 (6.1)	17.7 (2.0)
IRIS (2023)	✗	30.2 (4.7)	41.7 (5.0)	31.2 (4.7)	20.8 (5.9)	6.2 (3.5)	26.0 (2.2)
LeWorldModel (2026a)	✗	24.0 (4.4)	24.0 (4.4)	59.4 (5.0)	16.7 (5.4)	16.7 (5.4)	28.1 (2.2)
DINO-WM (2025)	✗	90.6 (3.0)	59.4 (5.0)	65.6 (4.8)	52.1 (7.2)	43.8 (7.2)	62.3 (2.5)
JEPA-WMs (2025)	✗	79.2 (4.1)	**60.4** (5.0)	68.8 (4.7)	66.7 (6.8)	47.9 (7.2)	64.6 (2.6)
LaLa-World	✓	**95.8** (2.0)	56.2 (5.1)	**81.2** (4.0)	**75.0** (6.2)	**52.1** (7.2)	**72.1** (2.3)

Method	Joint-training	Rope	Granular	RoboCasa-R	RoboCasa-P	DROID	Average
DINO-WM (2025)	✗	54.2 (10.2)	50.0 (14.4)	18.8 (6.9)	21.9 (7.3)	39.4 (3.8)	36.8 (4.1)
JEPA-WMs (2025)	✗	62.5 (9.9)	50.0 (14.4)	37.5 (8.6)	31.2 (8.2)	**43.0** (3.8)	44.8 (4.3)
V-JEPA 2-AC (2025)	✗	–	–	15.6 (6.4)	34.4 (8.4)	42.9 (2.5)	–
LaLa-World	✓	**66.7** (9.6)	**58.3** (14.2)	**40.6** (8.7)	**46.9** (8.8)	40.2 (5.2)	**50.5** (4.4)

Table 1: **MPC planning results.** We compare LaLa-World with existing world-model baselines across simulation and robot domains. LaLa-World achieves the highest average score in both tables. We report success rates (%), except for the DROID action score (Terver et al., 2025), with standard errors in parentheses. RoboCasa is evaluated zero-shot; -R, -RW, and -P denote Reach, Reach-Wall, and Place, respectively. Joint-training denotes a single predictor trained across domains. Additionally, scores in Figure 1(c) are averaged over the benchmarks used in the RAEv2 comparison (Section 4.2).

model training, which prevents encoder collapse and preserves its pretrained visual representations. Following JEPA-WMs (Terver et al., 2025), we use DINOv2-S/14 or DINOv3-L/16 as the frozen observation encoder. More details and ablations on vision encoders are provided in Section C.5.

Predictor. The predictor P_ϕ takes the visual context $e_{t-w+1:t}$ and the language condition z as input and predicts the next latent representation $\hat{e}_{t+1}$. Existing world models typically receive conditions in two ways: LWMs incorporate numerical actions through mechanisms such as AdaLN (Terver et al., 2025), feature conditioning (Zhou et al., 2025), and sequence conditioning (Nam et al., 2026), all of which offer no way to read text. Video generation models support language conditioning through a separate text encoder whose outputs condition the predictor (Wan et al., 2025), which adds parameters and text-encoding cost.

LaLa-World addresses both limitations by using a pretrained language model as the predictor itself: it reads the language condition as text and predicts future latent representations in the same network. Specifically, we use Qwen3.5-0.8B (Qwen Team, 2026) as our default predictor backbone. The language condition z is tokenized using the model's tokenizer and mapped to token embeddings, which are passed to the predictor together with the visual latents. To map the visual latents into the same embedding space, we use a two-layer MLP and add 3D positional embeddings indexed by time and 2D spatial location (Wang et al., 2024). The output $\hat{e}_{t+1}$ is then read out from learned query tokens (Pan et al., 2025). We train the predictor by minimizing the squared error between the predicted and target future representations: $\mathcal{L} = \|\hat{e}_{t+1} - e_{t+1}\|^2$.

Notably, we use proprioceptive state neither as an input nor as a prediction target. While proprioception can benefit LWMs tailored to a specific domain, it is defined in a domain-specific manner, which limits its applicability to a domain-agnostic world model. We further examine the predictor's backbone and architectural choices through ablations in Section C.3 and D.2, respectively, including BERT (Devlin et al., 2019) and Gemma (Gemma Team, 2025) as alternative backbones.

4 MPC Planning

Suppose we want to move a cup from our hand onto a table while avoiding nearby objects. We imagine the outcomes of candidate action sequences and select the one that brings the cup closest to the desired position (*i.e.*, planning with imagination). In robot planning, the world model provides these imagined outcomes: given an initial observation x_{init} and a goal observation x_{goal}, a planner selects the action sequence whose imagined outcome is closest to the goal. Following prior LWM studies (Zhou et al., 2025; Terver et al., 2025), we evaluate planning with model predictive control (MPC), using the Cross-Entropy Method (CEM) to optimize action sequences (Rubinstein, 1997).

Evaluation Setup. At each planning round, we sample candidate action sequences and, for each sequence, apply Equation (1) autoregressively for H steps and score it by the squared distance

between the final predicted latent state and the encoded goal:

$$\mathcal{C} = \|\hat{e}_{t+H} - e_{\text{goal}}\|^2, \quad e_{\text{goal}} = E_\theta(x_{\text{goal}}). \tag{2}$$

CEM repeatedly samples sequences and updates the sampling distribution using those with the lowest costs. For LaLa-World, we verbalize candidate actions using the training templates. In simulation, we execute the corresponding numerical actions from an initial segment of the selected sequence. For domains requiring re-planning, we obtain a new observation and repeat this process. Benchmark-specific settings and evaluation criteria are provided in Section D.4.

We use seven control datasets from DINO-WM and JEPA-WMs (Terver et al., 2025): six simulation datasets—Wall (Zhou et al., 2025), PointMaze (Fu et al., 2020), Push-T (Chi et al., 2023), MetaWorld (Yu et al., 2020), Rope and Granular (Zhang et al., 2024) and the real-robot dataset DROID (Khazatsky et al., 2024). We additionally evaluate on RoboCasa (Nasiriany et al., 2024), a domain excluded from training, to assess zero-shot transfer. In addition to existing LWM baselines including DINO-WM, we also train the world models of IRIS (Micheli et al., 2023), DreamerV3 (Hafner et al., 2025a), and TD-MPC2 (Hansen et al., 2024) on the same control datasets without reward or task information, which is the same setup for these baselines in DINO-WM. For a fair comparison, we train each baseline using its public implementation on the same data and evaluate all under the same planning settings, including the horizon H and none uses proprioceptive states.

Decoder. To visualize and interpret predicted futures, we build a 12-layer transformer decoder D that maps predicted latents to RGB images: $\hat{x}_{t+1} = D(\hat{e}_{t+1})$. We train the decoder with a combination of L_1 reconstruction and LPIPS perceptual losses (Zhang et al., 2018), following the decoder training recipe of RAEv2 (Singh et al., 2026). We note that the decoder is not used to train the world model.

4.1 Results and Analysis

A jointly trained LaLa-World predictor performs well across diverse domains. Table 1 reports MPC planning results across simulation and robot domains. While existing LWMs are typically trained within individual domains, LaLa-World trains its predictor jointly across all domains with different action spaces and physical dynamics. Despite these differences, LaLa-World achieves the highest average score in both benchmark groups. On the five simulation tasks in the upper table, LaLa-World reaches an average success rate of 72.1%, exceeding JEPA-WMs by 7.5 percentage points. The largest gain over the best baseline in this group is on Push-T, where success increases from 68.8% to 81.2%. LaLa-World also achieves the highest zero-shot success rates on both RoboCasa tasks, reaching 40.6% on Reach and 46.9% on Place.

Training on image and video data improves performance and robustness to phrasing. We evaluate the effect of training LaLa-World on image and video transitions and find two benefits. The results are shown in Figure 4. First, adding image and video transitions improves planning performance. Specifically, the average score increases from 59.3 with control trajectories alone to 61.6 with all data. Second, LaLa-World becomes more robust when the same actions are expressed in different words. To evaluate this, we paraphrase the action templates at evaluation, for example, replacing `move the gripper` with `shift the robot hand` and `move the agent` with `push the dot`. LaLa-World remains robust when trained on all data, including image and video transitions, whereas paraphrasing lowers the average score by 6.8 points with control trajectories alone. Further details and domain-level results are provided in Section C.7.

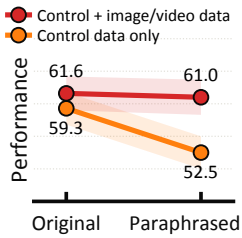

Fig. 4: Benefits of image /video data inclusion.

LaLa-World can predict the future from language without numbers. The language actions in our control trajectories retain numerical magnitudes. We next ask whether LaLa-World can predict outcomes from qualitative descriptions such as `move slightly down`. Using the action ranges observed in the training data, we define a conversion function that replaces each numerical magnitude with one of three qualitative levels: *slightly*, no modifier, or *far*. CEM samples numerical action sequences as before and passes them through this function to obtain the language conditions for LaLa-World. The bottom two rows of Figure 2 show successful planning examples in Wall and RoboCasa under these conditions. These examples indicate that the predictor can use qualitative descriptions to guide planning even when exact action magnitudes are unspecified. We provide more details on the evaluation setup in Section D.4.3 and additional results in Section C.6.

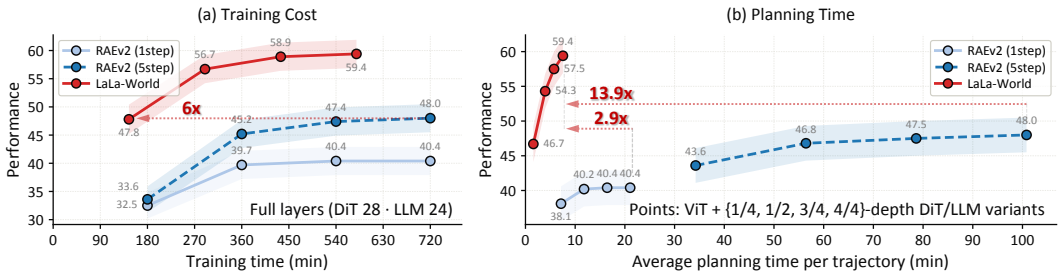

Figure 5: **Latent vs. generative world models for MPC planning under matched setups.** We match the vision encoder, language conditions, and training data to compare the two world models. (a) We track planning performance over training time for full-depth predictors under a 12-hour training budget. (b) We compare planning performance and average planning time per trajectory across four predictor depths, with all models trained on the same number of transition data. Overall, LaLa-World achieves higher planning performance than RAEv2 with lower training cost and planning time.

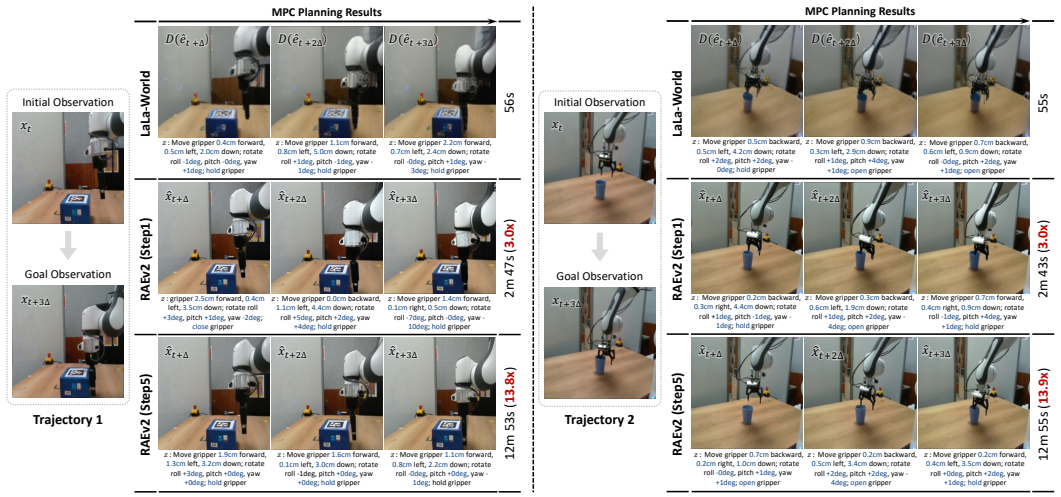

Figure 6: **Visualization of predicted future states during MPC planning on DROID.** We visualize decoded predictions from LaLa-World and RAEv2, using one or five denoising steps for RAEv2. Each example shows the initial and goal observations, the predicted rollout under the selected language actions, and planning time. Decoded images from LaLa-World may have lower visual quality because it is not trained for pixel reconstruction. Both models predict plausible future states, while LaLa-World requires less planning time.

4.2 Latent vs. Generative World Models

With language as the action interface, LaLa-World can be compared with generative world models trained on the same kinds of data. LWMs learn to match future visual latents, whereas generative world models are trained for pixel-level generation. We ask how these approaches compare in planning performance and computational cost. To the best of our knowledge, this is the first comparison of latent and generative world models in planning under matched conditions, as discussed in Section A.2.

Setup for a matched comparison. We use RAEv2 (Singh et al., 2026) as our generative world model baseline because it uses DINOv3-L/16 (Siméoni et al., 2025) rather than a VAE as its vision encoder. LaLa-World uses the same encoder in this comparison. Both models use language as the action interface and receive the same language conditions. We train both models on the same control trajectories and image and video transitions used to train LaLa-World. The full-depth predictors are comparable in size: LaLa-World's predictor and RAEv2's DiT (Peebles & Xie, 2023) each contain approximately 0.8B parameters. We initialize RAEv2 from its publicly released pretrained DiT weights and follow its training implementation, including flow matching (Lipman et al., 2023) and REPA (Yu et al., 2025). During planning, we use one or five denoising steps per prediction and compute the cost in Equation (2) between the resulting future representations and the encoded goal. Further details of RAEv2 are provided in Section D.5.

We compare the two world models in terms of training cost and planning time. For training cost, we track planning performance while training the full-depth predictors on eight H100 GPUs with a

Method	Pretrained weights	Spatial	Object	Goal	Long	Avg.
Dita (2025)	DINOv2[†24]+CLIP[†21]	97.4	94.8	93.2	83.6	92.3
π_0 (2025)	PaliGemma[†24]	96.8	98.8	95.8	85.2	94.2
UniVLA (2025)	Prismatic[†24]	96.5	96.8	95.6	92.0	95.2
$\pi_{0.5}$ (2025)	PaliGemma[†24]	**98.8**	98.2	**98.0**	92.4	96.9
OpenVLA-OFT (2025)	Prismatic[†24]	97.6	98.4	97.9	94.5	97.1
CogVLA (2025b)	Prismatic[†24]	98.6	98.8	96.6	**95.4**	97.4
Cosmos Policy (2026)	Random init.	94.7	98.9	96.3	88.6	94.6
Cosmos Policy (2026)	Cosmos-Predict2[†25]	97.6	99.8	97.9	94.0	97.3
LaLa-Policy	Random init.	94.4	99.6	95.2	87.2	94.1
LaLa-Policy	Qwen3.5[†26]	96.8	99.8	96.0	90.2	95.7
LaLa-Policy	LaLa-World	98.6	**100.0**	97.2	95.0	**97.7**

Table 2: **LIBERO policy results.** We compare LaLa-Policy with existing policies on LIBERO. Pretrained weights indicates the source model for policy fine-tuning, while Random init. denotes random initialization. This table and the adjacent table report success rates (SR, %).

Method	Pretrained weights	Demos	SR (%)
π_0 (2025)	PaliGemma[†24]	300	62.5
GR00T-N1.5 (2025)	Eagle 2.5[†25a]	300	64.1
Video Policy (2025)	Stable Video Diff.[†23a]	300	66.0
LaLa-Policy	Random init.	300	65.1
LaLa-Policy	LaLa-World	300	**66.4**
UVA (2025a)	MAR[†24]	50	50.0
$\pi_{0.5}$ (2025)	PaliGemma[†24]	50	54.5
MolmoAct2 (2026)	Molmo2-ER[†26]	50	54.3
Cosmos Policy (2026)	Random init.	50	53.8
Cosmos Policy (2026)	Cosmos-Predict2[†25]	50	56.0
LaLa-Policy	Random init.	50	55.5
LaLa-Policy	Qwen3.5[†26]	50	56.8
LaLa-Policy	LaLa-World	50	**57.3**

Table 3: **RoboCasa policy results.** We evaluate policy models on RoboCasa. Demos denotes the number of training trajectories.

12-hour budget. For planning time, we vary the number of predictor layers to obtain four model sizes for each approach, train each variant on 16M transitions in total, and measure average planning time per trajectory on a single H100 GPU.

LaLa-World outperforms the generative baseline with lower training and planning costs. As shown in Figure 5(a), LaLa-World reaches five-step RAEv2's 12-hour performance in two hours and later achieves a planning performance of 59.4%, exceeding RAEv2's best performance of 48.0%. For RAEv2, using five denoising steps per prediction improves performance but increases planning time. In Figure 5(b), full-depth LaLa-World outperforms every RAEv2 variant and plans 2.9× and 13.9× faster than full-depth RAEv2 with one and five denoising steps, respectively. [2] Together, these results show that a world model trained to match future visual latents can support effective and efficient planning without targeting the reconstruction of visual details. Domain-level results are provided in Section C.8. In addition, Figure 6 shows imagined DROID rollouts: RAEv2 generates plausible future observations even with one denoising step, while decoded LaLa-World predictions also depict plausible future states.

5 LaLa-Policy: From World Model To Action Predictor

We now ask whether this pretrained LaLa-World provides a useful initialization for policy fine-tuning. Cosmos Policy (Kim et al., 2026) takes this approach by fine-tuning the pretrained video generation model Cosmos-Predict2 (Ali et al., 2025) to jointly predict actions and future frames.

LaLa-Policy. Following Cosmos Policy's fine-tuning procedure, we fine-tune LaLa-World into LaLa-Policy using policy data to predict an action chunk together with a future latent representation. We add learned action queries to the predictor P_ϕ alongside its queries for future prediction. Given an observation x_t and a task instruction ℓ, the outputs at the action queries provide a sequence of conditioning features $h_{t:t+N-1}$ for an action generator A_ψ built on a diffusion transformer. We use the action-head architecture of Isaac-GR00T (Bjorck et al., 2025), which conditions action generation on vision-language features. Starting from Gaussian noise, A_ψ generates the action chunk through iterative denoising:

$$\begin{aligned}(\hat{e}_{t+N}, h_{t:t+N-1}) &= P_\phi\left(E_\theta(x_t), \ell\right), \\ \hat{a}_{t:t+N-1} &= A_\psi\left(\epsilon; h_{t:t+N-1}\right), \quad \epsilon \sim \mathcal{N}(0, I).\end{aligned} \tag{3}$$

Here, N is the number of actions in the chunk, and the future prediction targets $E_\theta(x_{t+N})$. Modeling an action distribution allows the policy to represent multiple action sequences for reaching the goal.

Policy fine-tuning setup. We compare LaLa-Policy with policies that predict actions alone (Intelligence et al., 2025; Li et al., 2025b) and models that jointly predict actions and future observations (Li et al., 2025a; Liang et al., 2025). For a fair comparison with the joint prediction baselines, we fine-tune Cosmos Policy (Kim et al., 2026) to predict only actions and future observations, without auxiliary future proprioception or value prediction. LaLa-Policy follows the same prediction setup in latent space. We fine-tune on LIBERO (Liu et al., 2023) and RoboCasa (Nasiriany et al., 2024), using demonstration trajectories composed of transitions (x_t, a_t, x_{t+1}) and paired with task instructions.

[2] Evaluating RAEv2 with 50 denoising steps would require approximately 25 days on 8 H100 GPUs.

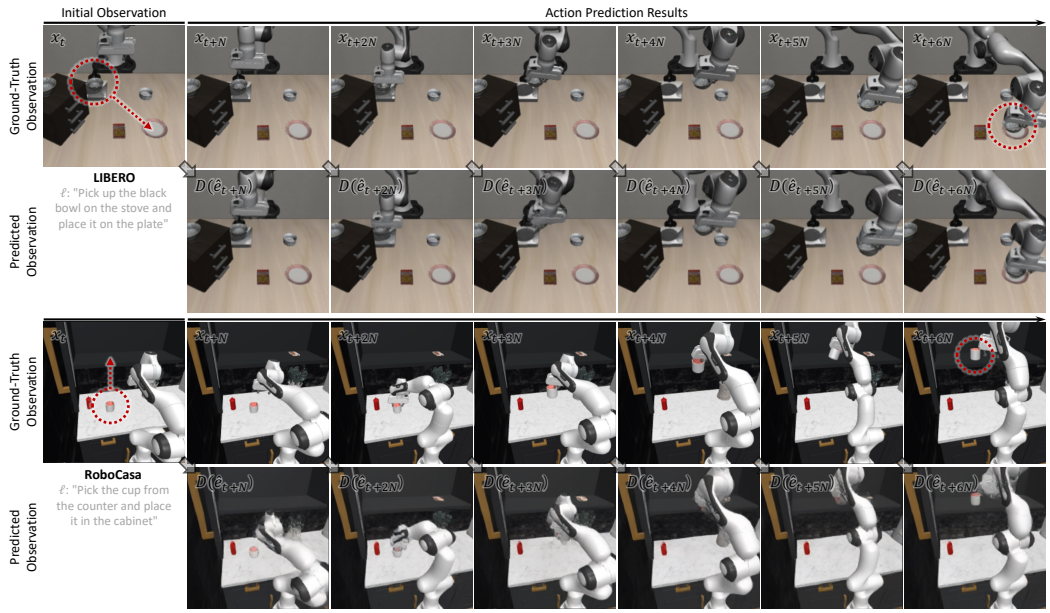

Figure 7: **Predicted future observations on LIBERO and RoboCasa.** Each example shows the initial observation and task instruction, followed by simulator observations during policy execution (upper row) and decoded future predictions (lower row). LaLa-Policy predicts $(x_t, \ell) \rightarrow (\hat{e}_{t+N}, \hat{a}_{t:t+N-1})$, and the simulator executes $\hat{a}_{t:t+N-1}$ to yield x_{t+N}; see Equation 3. Since LaLa-Policy is not trained for pixel reconstruction, decoded images may contain visual inaccuracies.

For RoboCasa, we consider two training sets: 50 human demonstrations or 300 demonstrations generated with MimicGen per task. We report success rates in simulation; dataset, training, and evaluation details are in Section D. Dataset, training, and evaluation details are provided in Section D.8.

LaLa-Policy achieves competitive and efficient action prediction via latent world modeling. Tables 2 and 3 show that LaLa-World can be fine-tuned into an effective policy, performing competitively with Cosmos Policy on both benchmarks. Notably, this performance comes with up to $5.5\times$ faster inference, measured by average inference time per trajectory (Table 4). Together with our planning results, these findings suggest that effective world models and action predictors can be learned without a pixel reconstruction objective. World modeling pretraining also benefits policy fine-tuning. Initializing the predictor from LaLa-World outperforms both random and Qwen3.5 initialization. This agrees with the finding of Cosmos Policy (Kim et al., 2026) that Cosmos-Predict2, pretrained on diverse videos, provides a useful prior for policy learning, and extends this observation to latent world models. To examine what the policy imagines, Figure 7 compares decoded future predictions with simulator observations during policy execution. The predictions follow robot motion and object state changes as objects are lifted and moved, showing that latent prediction can capture the manipulation dynamics.

	LIBERO	RoboCasa
Cosmos Policy	4.45 s	11.71 s
LaLa-Policy	**0.86** s ($\times$5.2)	**2.12** s ($\times$5.5)

Tab. 4: Avg. inference time per trajectory and speedup.

6 Conclusion

We presented LaLa-World, a latent world model that learns from heterogeneous control trajectories and broader image and video data through a shared language interface. This enables comparisons with existing LWMs in MPC planning and generative models in planning and policy fine-tuning. Despite sharing one predictor across domains, LaLa-World outperforms existing LWMs in average MPC planning performance. Under matched settings, including training data, it also surpasses the generative baseline in planning performance and efficiency. Following Cosmos Policy's use of world models pretrained on diverse data for policy learning, we fine-tuned LaLa-World into LaLa-Policy. It outperforms evaluated VLA baselines and achieves substantially faster inference than Cosmos Policy. These results demonstrate the value of latent world models for planning and policy learning without training to reconstruct visual details. We hope this work draws greater attention to latent world models, their scalability, and downstream uses. Limitations and future directions are in Section B.

AI USE STATEMENT

In this work, we used generative AI tools to assist with implementing methods specified by the authors and writing code for rule-based dataset cleaning and reformatting. We also used generative AI tools to suggest alternative phrasings. We also used Qwen3-VL-30B-A3B-Instruct to generate textual descriptions of changes between adjacent frames from OpenVid-1M as training annotations (Section D.1). Our use of generative AI for data synthesis was limited to these textual annotations.

We did not use generative AI tools to develop theoretical models or conceptual frameworks, formulate mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, conduct qualitative or thematic data analysis, or interpret results. Providing critical ingredients for mathematical proofs and assisting with proof writing were not applicable, as this work does not present formal mathematical proofs.

The authors manually reviewed the AI-assisted implementation and data-processing code. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The experimental setup is described in Section 4, Section 4.2, and Section 5. The implementation details are provided in Section D.

REFERENCES

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

Arslan Ali, Junjie Bai, Maciej Bala, Yogesh Balaji, Aaron Blakeman, Tiffany Cai, Jiaxin Cao, Tianshi Cao, Elizabeth Cha, Yu-Wei Chao, et al. World simulation with video foundation models for physical ai. *arXiv preprint arXiv:2511.00062*, 2025.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *CVPR*, 2023.

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

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *CVPR*, 2025.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *TMLR*, 2024.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

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.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi0: A vision-language-action flow model for general robot control. In *RSS*, 2025.

Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, et al. Nvidia nemotron 3: Efficient and open intelligence. *arXiv preprint arXiv:2512.20856*, 2025.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023a.

Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *CVPR*, 2023b.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, et al. Video generation models as world simulators. OpenAI Technical Report, 2024. URL https://openai.com/research/video-generation-models-as-world-simulators.

Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *ICML*, 2024.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. In *RSS*, 2025.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

Guo Chen, Zhiqi Li, Shihao Wang, Jindong Jiang, Yicheng Liu, Lidong Lu, De-An Huang, Wonmin Byeon, Matthieu Le, Max Ehrlich, et al. Eagle 2.5: Boosting long-context post-training for frontier vision-language models. In *NeurIPS*, 2025a.

Junying Chen, Zhenyang Cai, Pengcheng Chen, Shunian Chen, Ke Ji, Xidong Wang, Yunjin Yang, and Benyou Wang. Sharegpt-4o-image: Aligning multimodal models with gpt-4o-level image generation. *arXiv preprint arXiv:2506.18095*, 2025b.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

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

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. In *NeurIPS*, 2023.

Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, et al. Molmoact2: Action reasoning models for real-world deployment. *arXiv preprint arXiv:2605.02881*, 2026.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *ICRA*, 2017.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Quentin Garrido, Mahmoud Assran, Nicolas Ballas, Adrien Bardes, Laurent Najman, and Yann LeCun. Learning and leveraging world models in visual representation learning. *arXiv preprint arXiv:2403.00504*, 2024.

Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *NeurIPS*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *ICML*, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *ICLR*, 2020.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with discrete world models. In *ICLR*, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 2025a.

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025b.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *ICML*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *ICLR*, 2024.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Learning massively multitask world models for continuous control. *arXiv preprint arXiv:2511.19584*, 2025.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. In *NeurIPS*, 2022.

Zhi Hou, Tianyi Zhang, Yuwen Xiong, Haonan Duan, Hengjun Pu, Ronglei Tong, Chengyang Zhao, Xizhou Zhu, Yu Qiao, Jifeng Dai, et al. Dita: Scaling diffusion transformer for generalist vision-language-action policy. In *ICCV*, 2025.

Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. GAIA-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. In *ICML*, 2024.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *RSS*, 2025.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. In *ICLR*, 2026.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Po-Chen Ko, Jiayuan Mao, Yilun Du, Shao-Hua Sun, and Joshua B Tenenbaum. Learning to act from actionless videos through dense correspondences. In *ICLR*, 2024.

Dan Kondratyuk, Lijun Yu, Xiuye Gu, José Lezama, Jonathan Huang, Grant Schindler, Rachel Hornung, Vighnesh Birodkar, Jimmy Yan, Ming-Chang Chiu, et al. Videopoet: A large language model for zero-shot video generation. In *ICML*, 2024.

Yann LeCun. A path towards autonomous machine intelligence. *OpenReview*, 2022.

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. In *RSS*, 2025a.

Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. In *NeurIPS*, 2024.

Wei Li, Renshan Zhang, Rui Shao, Jie He, and Liqiang Nie. Cogvla: Cognition-aligned vision-language-action model via instruction-driven routing & sparsification. In *NeurIPS*, 2025b.

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.

Jessy Lin, Yuqing Du, Olivia Watkins, Danijar Hafner, Pieter Abbeel, Dan Klein, and Anca Dragan. Learning to model the world with language. In *ICML*, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *ICLR*, 2023.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *NeurIPS*, 2023.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworldmodel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026a.

Lucas Maes, Quentin Le Lidec, Dan Haramati, Nassim Massaudi, Damien Scieur, Yann LeCun, and Randall Balestriero. stable-worldmodel-v1: Reproducible world modeling research and evaluation. *arXiv preprint arXiv:2602.08968*, 2026b.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *CoRL*, 2023.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *ICLR*, 2023.

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mahmoud Assran, Koustuv Sinha, Michael Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-jepa 2.1: Unlocking dense features in video self-supervised learning. *arXiv preprint arXiv:2603.14482*, 2026.

Heejeong Nam, Quentin Le Lidec, Lucas Maes, Yann LeCun, and Randall Balestriero. Causal-jepa: Learning world models through object-level latent masking. In *ICML*, 2026.

Kepan Nan, Rui Xie, Penghao Zhou, Tiehan Fan, Zhenheng Yang, Zhijie Chen, Xiang Li, Jian Yang, and Ying Tai. Openvid-1m: A large-scale high-quality dataset for text-to-video generation. In *ICLR*, 2025.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *RSS*, 2024.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *TMLR*, 2024.

Xichen Pan, Satya Narayan Shukla, Aashu Singh, Zhuokai Zhao, Shlok Kumar Mishra, Jialiang Wang, Zhiyang Xu, Jiuhai Chen, Kunpeng Li, Felix Juefei-Xu, Ji Hou, and Saining Xie. Transfer between modalities with metaqueries. *arXiv preprint arXiv:2504.06256*, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

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

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Reuven Y Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 1997.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 2020.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025.

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

Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. *arXiv preprint arXiv:2502.14819*, 2025.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163, 1991.

Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models? *arXiv preprint arXiv:2512.24497*, 2025.

Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai C Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *NeurIPS*, 2024.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. SigLIP 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. Phenaki: Variable length video generation from open domain textual description. In *ICLR*, 2023.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

Ying Wang, Oumayma Bounou, Yann LeCun, and Mengye Ren. AdaJEPA: An adaptive latent world model. *arXiv preprint arXiv:2606.32026*, 2026.

Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *NeurIPS*, 2015.

Cong Wei, Zheyang Xiong, Weiming Ren, Xeron Du, Ge Zhang, and Wenhu Chen. Omniedit: Building image editing generalist models through specialist supervision. In *ICLR*, 2025.

Sherry Yang, Yilun Du, Seyed Kamyar Seyed Ghasemipour, Jonathan Tompson, Leslie Pack Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. In *ICLR*, 2024.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *ICLR*, 2025.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *CoRL*, 2020.

Kaifeng Zhang, Baoyu Li, Kris Hauser, and Yunzhu Li. Adaptigraph: Material-adaptive graph-based neural dynamics for robotic manipulation. In *RSS*, 2024.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *ICLR*, 2026.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. In *ICML*, 2025.

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

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. In *RSS*, 2025.

Appendix

Table of Contents

A Related Work **17**

A.1 Latent World Models . . . 17

A.2 Video Generation Models . . . 17

A.3 Planning and Policy Learning with World Models . . . 18

A.4 Action Prediction Models . . . 18

A.5 Visual Representation Pretraining . . . 18

B Future Directions **18**

C Experiments and Analyses **19**

C.1 ImageNet Training and Evaluation of LaLa-World . . . 19

C.2 Communicable World Modeling . . . 20

C.3 LaLa-World Architecture Ablations . . . 21

C.4 Predictor Initialization and Joint Training . . . 22

C.5 LaLa-World Vision Encoder Ablations . . . 23

C.6 Additional Results on Planning with Qualitative Language Actions . . . 24

C.7 Additional Results on Action Phrasing Robustness . . . 24

C.8 Additional Results on Latent vs. Generative World Models . . . 25

C.9 Additional Qualitative Planning Results . . . 25

C.9.1 Planning with Numerical Language Actions . . . 25

C.9.2 Planning with Qualitative Language Actions . . . 25

C.10 Qualitative Policy Results . . . 25

D Implementation and Experimental Details **26**

D.1 LaLa-World Training Data Details . . . 26

D.2 LaLa-World Architecture Details . . . 27

D.3 LaLa-World Training Details . . . 27

D.4 MPC Planning and Evaluation Details . . . 28

D.4.1 Benchmark Settings and Evaluation Metrics . . . 28

D.4.2 Action Aggregation and Execution . . . 29

D.4.3 Qualitative Language Action Conversion . . . 29

D.5 RAEv2 Baseline Details . . . 29

D.6 LaLa-Policy Architecture Details . . . 30

D.7 Policy Model Training Data . . . 31

D.8 LaLa-Policy Fine-tuning Details . . . 31

D.9 Policy Model Evaluation Details . . . 31

A Related Work

We summarize in Figure A how LaLa-World and LaLa-Policy compare with representative world models and policy models.

(a) World models	DINO-WM, JEPA-WMs	Cosmos-Predict2	RAEv2-based WM	LaLa-World
Future modeling	Latent	Image, Video	Latent	**Latent**
Action / change conditioning	[0.7, 0.3, ..] Numerical actions	Natural language	Natural language	**Natural language**
Predictor training data	Domain-specific control data	Image, video data	Image, video data (with control data)	**Image, video data (with control data)**
Vision encoder	SSL-pretrained ViT	VAE	SSL-pretrained ViT	SSL-pretrained ViT

(b) Policy models	VLA: $\pi_{0.5}$, CogVLA	Cosmos Policy	LaLa-Policy
Policy initialization	Pretrained VLM	Cosmos-Predict2	**LaLa-World**
Action prediction	Action chunk	Action chunk	Action chunk
Joint future prediction	None	Future frames	**Latent**

Figure A: **Comparison with representative world models and policy models.** (a) LaLa-World uses natural language conditions, enabling training on control trajectories together with image and video transitions. The RAEv2-based WM is our generative world model baseline under matched settings in Section 4.2. (b) Fine-tuned from LaLa-World, LaLa-Policy jointly predicts actions and future latents, whereas VLAs predict actions alone and Cosmos Policy predicts the future with visual details.

A.1 Latent World Models

Latent world models predict future states in a learned representation space rather than in pixels. Recent work follows the joint-embedding predictive architecture (Assran et al., 2023; 2025). Existing studies focus on improving latent prediction for planning (Assran et al., 2025) or stabilizing training (Maes et al., 2026a). In the former direction, PLDM (Sobal et al., 2025) demonstrates reward-free planning with learned latent dynamics, while DINO-WM (Zhou et al., 2025) shows the effectiveness of using a frozen DINOv2 vision encoder. V-JEPA 2-AC (Assran et al., 2025) extends latent planning to real-robot manipulation, and JEPA-WMs (Terver et al., 2025) study design choices including multistep rollouts. An orthogonal direction addresses training stability: LeWorldModel (Maes et al., 2026a) trains the vision encoder and predictor end to end, using SIGReg to prevent representation collapse. AdaJEPA (Wang et al., 2026) further improves robustness by updating the predictor at test time. These methods typically train action-conditioned predictors within a single domain, limiting their development into General World Models. Although Newt (Hansen et al., 2025) learns across multiple domains, its training remains focused on control and game environments. Joint training of action-conditioned predictors on control trajectories and broader image and video transitions remains underexplored. We therefore use language to unify actions across control datasets and co-train them with diverse image and video transitions. Our use of language as the action interface differs from Dynalang (Lin et al., 2024), which incorporates language as observations or task descriptions.

A.2 Video Generation Models

Video generation models (*i.e.*, generative world models) predict future observations conditioned on language (Ho et al., 2022; Blattmann et al., 2023b; Villegas et al., 2023; Kondratyuk et al., 2024). Language conditioning lets them learn from web-scale video (Brooks et al., 2024; Wan et al., 2025), and the resulting models serve as interactive simulators for games (Bruce et al., 2024), driving (Hu et al., 2023), and robotics (Yang et al., 2024; Agarwal et al., 2025; Ali et al., 2025). While video

generation models can produce visually compelling content, using them for MPC planning is costly because evaluating candidate action sequences requires repeatedly synthesizing the future with visual details. V-JEPA 2-AC and V-JEPA 2.1 (Assran et al., 2025; Mur-Labadia et al., 2026) report faster planning than video generation baselines (Agarwal et al., 2025; Bar et al., 2025), but these comparisons differ in action interface, encoder, predictor size, and training data. RAE (Zheng et al., 2026; Singh et al., 2026) replaces the VAE encoder (Kingma & Welling, 2013) with a vision encoder pretrained with self-supervision, allowing a generative world model to share its encoder with an LWM. Using RAEv2 as a generative world model baseline, we compare it with LaLa-World under matched settings, including language conditioning.

A.3 Planning and Policy Learning with World Models

Planning with learned dynamics models has a long history (Sutton, 1991; Ha & Schmidhuber, 2018; Watter et al., 2015; Finn & Levine, 2017; Ebert et al., 2018). PlaNet (Hafner et al., 2019) plans with the Cross-Entropy Method (CEM) in a latent space trained with reconstruction and reward prediction. Dreamer (Hafner et al., 2020; 2021; 2025a;b) and IRIS (Micheli et al., 2023) instead train actor-critic policies on imagined rollouts. MuZero (Schrittwieser et al., 2020) and TD-MPC (Hansen et al., 2022; 2024) learn latent dynamics with reward and value prediction to support tree search and model predictive control, respectively. More recent work defines a world model more narrowly as a predictor of future states (Sobal et al., 2025; Assran et al., 2025; Maes et al., 2026a). Planning with them is performed at test time by sampling-based optimization toward a goal image (*i.e.*, MPC planning). Our experiments follow this reward-free and goal-conditioned protocol.

A.4 Action Prediction Models

Vision-language-action (VLA) models such as RT-2 (Brohan et al., 2023), OpenVLA (Kim et al., 2024), $\pi_{0.5}$ (Intelligence et al., 2025), UniVLA (Bu et al., 2025), and CogVLA (Li et al., 2025b) fine-tune vision-language models on large-scale robot demonstrations to map an observation and an instruction to actions. These models predict actions alone. When humans want to move a cup, however, they can imagine the action, how the cup will move, and where it will end up. Video-based policies such as UniPi (Du et al., 2023) and AVDC (Ko et al., 2024) first generate videos of task execution and then translate them into control actions. Unified video-action models (Li et al., 2025a; Zhu et al., 2025) instead predict actions and future frames jointly. Building on pretrained video models, Cosmos Policy (Kim et al., 2026) fine-tunes Cosmos-Predict2 (Ali et al., 2025) to jointly predict actions, future states, and values. LaLa-Policy follows this recipe but starts from a latent world model, predicting future representations rather than reconstructing visual details.

A.5 Visual Representation Pretraining

Self-supervised visual encoders provide the representations on which latent world models learn dynamics (Zhou et al., 2025; Assran et al., 2025). Contrastive and self-distillation methods (He et al., 2020; Chen et al., 2020; Caron et al., 2021; Oquab et al., 2024; Siméoni et al., 2025) learn invariances across augmented views, masked image modeling (He et al., 2022; Zhou et al., 2022) predicts masked patches, and JEPAs (Assran et al., 2023; Bardes et al., 2024) predict the representations of masked regions instead of their pixels. These encoders are conventionally evaluated by probing or fine-tuning on recognition tasks such as classification, segmentation, and depth estimation. V-JEPA 2 (Assran et al., 2025) introduced planning as a further evaluation: the pretrained encoder is frozen, an action-conditioned predictor is trained on top of it from robot trajectories, and the pair is used for MPC planning toward image goals. We follow the same evaluation protocol as V-JEPA 2, DINO-WM, and JEPA-WMs (Terver et al., 2025).

B Future Directions

LaLa-World shows that a shared language interface enables joint LWM training on heterogeneous control trajectories and broader image and video data, with benefits for planning and policy fine-tuning. These results suggest several directions for extending the current approach.

1. **Scaling data and model capacity.** With a 0.8B predictor, adding image and video data improves planning performance and robustness to action phrasing (Figure 4). However, our training data remain smaller in scale than those of video generation models such as Wan (Wan et al., 2025), and we have not established whether these gains continue at larger scales. Future work should increase dataset size and predictor capacity separately and together, measuring their effects on planning performance and downstream policy learning.
2. **Learning visual representations end to end.** We freeze the vision encoder to prevent representation collapse, but this also prevents its representations from adapting to future prediction. Better world models may require visual representations that capture information useful for anticipating future states. Jointly training the vision encoder and predictor while preventing collapse is therefore an important direction for future work.
3. **Extending prediction beyond one step.** Video generation models produce sequences of future frames, whereas our predictor is trained with supervision only on the next latent state. Whether latent world models can learn to generate similarly long sequences through autoregressive prediction remains an open question. Addressing this would require training on autoregressive rollouts with losses over multiple future states, rather than only the next state. We leave this investigation to future work because of the additional training cost, with particular interest in how prediction quality changes as the rollout length increases.
4. **Extending prediction horizons.** Following the evaluation protocols of JEPA-WMs and DINO-WM (Terver et al., 2025; Zhou et al., 2025), our MPC experiments use relatively short planning horizons of at most six steps (Table H). Evaluating longer horizons would help assess whether the model can anticipate distant consequences and support planning in real-world tasks that require such foresight. The multi-step supervision discussed above could support this extension.
5. **Evaluating on real robots.** Our experiments on LIBERO and RoboCasa show that initializing LaLa-Policy from LaLa-World outperforms existing VLAs, demonstrating the value of a pre-trained latent world model for policy fine-tuning. While these simulation experiments establish this benefit, extending the evaluation to real robots is an important next step.
6. **Best-of-N planning with LaLa-Policy.** LaLa-Policy achieves competitive action prediction without value supervision. Cosmos Policy (Kim et al., 2026) further learns from policy rollouts to predict future states and values for candidate action chunks, enabling best-of-N planning. We consider extending LaLa-Policy to such planning a valuable avenue for future work.

We hope these directions broaden the use of LWMs while retaining their efficiency in planning and policy learning.

C Experiments and Analyses

C.1 ImageNet Training and Evaluation of LaLa-World

Inspired by Image World Models (IWM) (Garrido et al., 2024), we use ImageNet (Deng et al., 2009) transformations to examine whether LaLa-World can predict visual changes specified in language.

Training. We construct transitions from a source image, a transformed target view, and a language description of the transformation (Figure B, left). Crops, shifts of the view, rotations, and horizontal flips become language actions, allowing static images to enter the same training format as control data. For example, a crop followed by a horizontal flip is paired with a sentence describing both operations. The predictor receives the source representation and this description, and learns to match the encoded target view. We train the predictor jointly on these pairs and other transition data.

Evaluation. For each of 200 ImageNet validation images, we construct a bank of 256 transformed views of that image, including photometric distractors with changes in color or brightness. Given the source and a geometric language action, the predictor must retrieve the corresponding target by the mean cosine similarity of aligned patch representations. We report mean reciprocal rank (MRR), averaging the reciprocal of the target's rank over queries. A target ranked first contributes 1, second $1/2$, and third $1/3$, so higher MRR indicates more accurate retrieval. Because all candidates show the same underlying image, retrieval requires distinguishing the requested transformation.

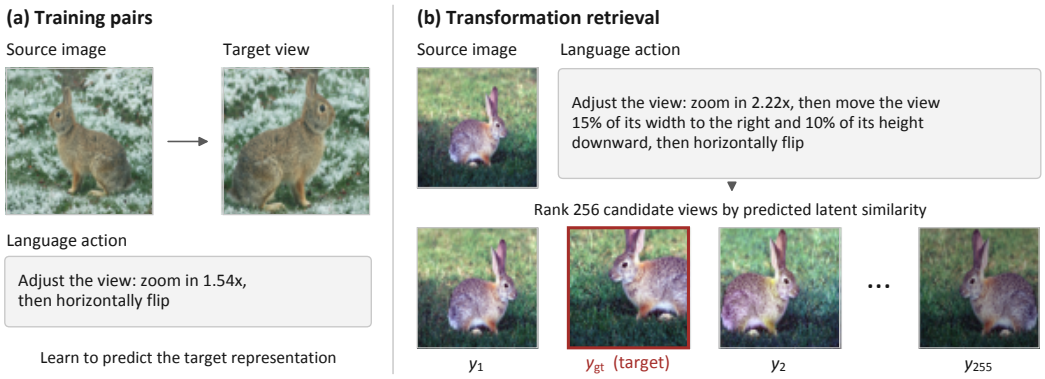

Figure B: **ImageNet training pairs and transformation retrieval.** Left: a source image and a transformed view form a training pair, with the transformation described in language. Right: the predicted representation is used to rank 256 transformed views of the same validation image; a subset is illustrated. The outlined view is the target specified by the language action.

Method	MRR ↑
IWM (default)†	0.79
IWM (strong jitter)†	0.85
LaLa-World	
Source latent	0.024
Predicted latents, empty language	0.026
Predicted latents, language action	**0.856**

Table A: **ImageNet transformation retrieval.** We report mean reciprocal rank (MRR) over 256 transformed views of each source image. The LaLa-World rows compare retrieval using the source latent with predicted latents conditioned on empty language or a language action. †Published IWM results (Garrido et al., 2024) use different training and transformation settings.

Controls. The source latent is the encoded input image, used directly for retrieval. The predicted latents are obtained from LaLa-World with either empty language (`""`) or the language action.

Results. Table A shows that LaLa-World reaches 0.856 MRR with language actions, while using the source latent directly or predicting with empty language gives 0.024 and 0.026, respectively. This gap indicates that the predictor uses language to model the specified visual change, extending the action interface to transformations of static images.

C.2 Communicable World Modeling

CEM-based MPC requires thousands of imagined action sequences, which limits its practicality even when each world-model rollout is inexpensive. A natural direction is to let a capable LLM act as the decision-maker, using the world model to evaluate its proposed actions. Since LaLa-World reads actions as language, it provides an interface for this interaction. We therefore ask whether an LLM can replace CEM with fewer imaginations, and whether this reduced search budget still yields successful plans.

Planning procedure. We call this setting *communicable world modeling*, in which an LLM actor and LaLa-World exchange language actions and their evaluation costs (Figure C). The LLM receives the initial and goal observations and proposes N candidate action sequences using the numerical language of the training templates, such as `move 0.6 left and 1.7 down`. LaLa-World imagines each sequence and returns the cost in Equation (2); a lower cost indicates a predicted outcome closer to the goal. The LLM uses these costs to revise its proposals. After K rounds, we select the lowest-cost sequence for execution and replan when required by the benchmark. During this search, the costs are the only feedback: the LLM sees neither the imagined states nor privileged simulator states.

Evaluation. We use Claude Opus 5 and GPT-6 Astra as actors and compare their results with the CEM baseline from Table 1 in Table B. We count one imagined candidate sequence as one rollout and report the total per trajectory, including replanning. Success rates measure task completion in simulation, while DROID uses the score defined in Equation (5).

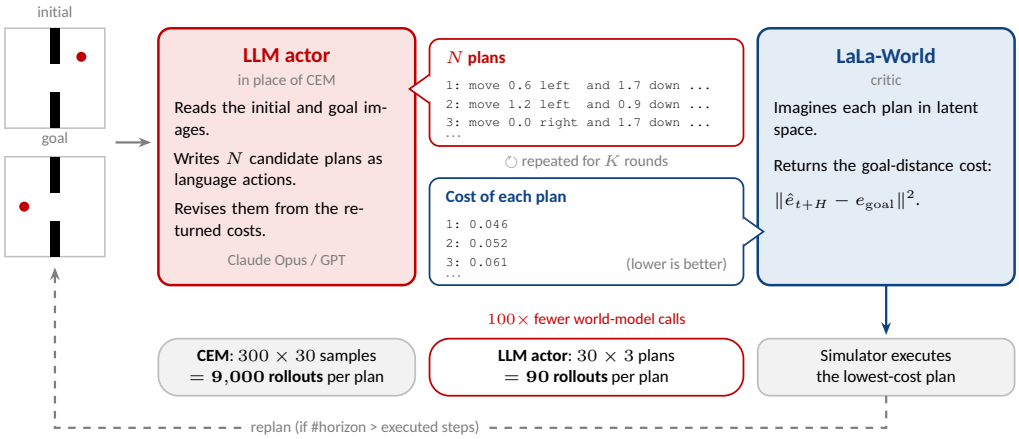

Figure C: **Communicable world modeling.** An LLM takes the place of CEM and talks to LaLa-World in language. It reads the initial and goal observations, writes N candidate plans as language actions (top message), and receives only the goal-distance cost of each plan (bottom message), which is the same cost that CEM minimizes. After K rounds, the lowest-cost plan is executed. The LLM needs two orders of magnitude fewer world-model rollouts than CEM.

Planner	Wall	PointMaze	RoboCasa-R	DROID
CEM	95.8 (9,000)	**56.2** (9,000)	40.6 (54,000)	40.2 (4,500)
Claude Opus 5	**100.0** (30×3=90)	31.2 (30×3=90)	**53.1** (10×2×4=80)	36.2 (20×2=40)
GPT-6 Astra	91.3 (8×1=8)	26.1 (8×3=24)	17.4 (8×1×12=96)	**44.7** (16×3=48)

Table B: **MPC planning with an LLM actor in place of CEM.** We report success rates (%), except for the DROID action score, with the number of world-model rollouts per trajectory in parentheses. For the LLM actors, the rollout count is written as $N \times K$, e.g., $30 \times 3 = 90$, where $N = 30$ is the number of candidate action sequences proposed per round and $K = 3$ is the number of rounds; for RoboCasa-R, the third factor is the number of replanning steps per trajectory. The LLM actors use 8–90 rollouts per trajectory, compared with 4,500–54,000 for CEM.

Results. Table B shows that the LLM actors use 8–96 rollouts per trajectory, compared with 4,500–54,000 for CEM. Claude Opus 5 reaches 100.0% success on Wall with 90 rollouts and 53.1% on RoboCasa-R with 80 rollouts. Figure D shows successful simulation episodes. These results demonstrate that an LLM can plan through the language interface of LaLa-World with a small imagination budget, while leaving room to improve performance on tasks requiring precise control.

C.3 LaLa-World Architecture Ablations

We examine how training data and predictor design affect the use of language conditions, beyond what validation loss alone reveals. A predictor may fit transitions using visual regularities or numerical patterns without responding to the meaning of action words.

Language sensitivity. For a held-out transition, we measure prediction accuracy relative to retaining the current state:

$$S(z) = 1 - \frac{\|\hat{e}_{t+1}(z) - e_{t+1}\|_2}{\|e_t - e_{t+1}\|_2}, \tag{4}$$

where $\hat{e}_{t+1}(z)$ is the prediction conditioned on z. The score is 1 for an exact prediction, 0 for returning e_t, and negative when the prediction is farther from the target than e_t.

We keep the target fixed and vary the condition: numerical language, qualitative language without numbers, reversed direction words in numerical language, and swapped magnitude words in qualitative language (Figure E(a)). Qualitative descriptions preserve direction but coarsen magnitude, so some loss of accuracy is expected. By contrast, following reversed direction words should move the prediction away from the original target and can make the score negative. Magnitude swaps preserve direction but change the amount of motion, so we expect a smaller drop. Panel (b) illustrates this expected pattern.

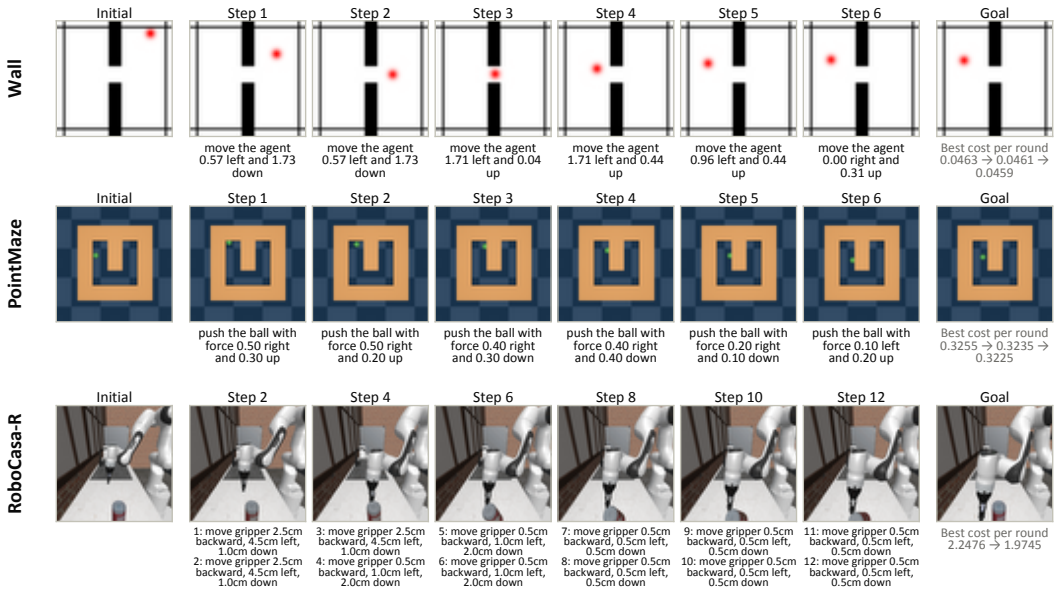

Figure D: **Qualitative Results of Communicable World Modeling with an LLM Actor.** Each row shows the initial and goal observations and the simulator observations after executing the lowest-cost sequence written by the LLM, with its language action below each frame. Below the goal observation, we list the cost of the best sequence at each round, which decreases as the LLM revises its plans from the returned costs.

Training data. With control data alone, the score drops when numerical magnitudes are replaced by qualitative descriptions (Figure E(c)). Adding image and video transitions narrows this gap and makes the direction-swap score more negative. These results suggest that broader transition data help the predictor interpret qualitative magnitudes and respond to direction words.

Language backbone. We train BERT-0.3B (Devlin et al., 2019), Gemma-0.3B (Gemma Team, 2025), Nemotron-1B (Blakeman et al., 2025), and Qwen3.5-0.8B (Qwen Team, 2026) on the same data. Qwen3.5 most closely follows the expected sensitivity pattern, while Gemma shows a similar pattern with lower scores under the original actions (Figure E(d)). Nemotron reaches a validation loss close to Qwen3.5 (Figure F(a)), yet its score barely changes when direction words are reversed. Thus, low prediction loss alone does not establish that a predictor follows action semantics. Pretrained Qwen3.5 also reaches a lower loss earlier in training than its randomly initialized counterpart, although their final losses are similar.

Positional encoding and readout. We compare the positional encoding and readout designs shown in Figure G. We measure language reliance as $\Delta S = S(z) - S(\varnothing)$, the score drop when the condition is replaced by empty text. A larger gap indicates that language contributes more to prediction accuracy. The 3D input positional embeddings index time and spatial location, preserving the structure of the visual tokens. In Figure F(b), 3D input positional embeddings with learned query tokens yield the highest reliance among the tested designs. Reading predictions from the last frame's state tokens instead of learned queries reduces reliance under both 3D embeddings and MRoPE (Wang et al., 2024). These results support our default predictor, which combines Qwen3.5 with 3D input positional embeddings and learned queries.

C.4 Predictor Initialization and Joint Training

We examine how predictor initialization and training scope affect MPC planning (Table C). All variants use the same predictor architecture and planning settings (Section 4). The jointly trained predictors use the full data mixture, including control trajectories and image and video transitions. For comparison, we train separate predictors from pretrained weights on each domain's control transitions, using the DROID-trained predictor for zero-shot evaluation on RoboCasa.

With pretrained initialization, joint training achieves an average score of 60.2, compared with 54.5 for single-domain training. Random initialization lowers the jointly trained predictor's average to 55.6.

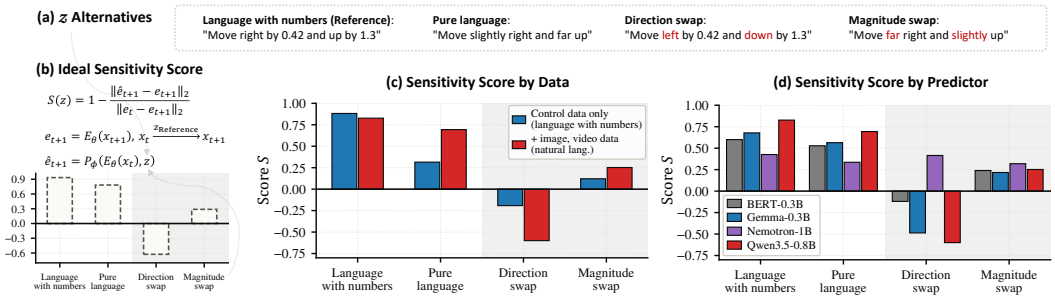

Figure E: **Language sensitivity across training data and predictor backbones.** (a) Language conditions used to evaluate the same target transition; red words mark semantic swaps. (b) Expected score pattern for a predictor that follows language. (c) Effect of adding image and video transitions to control data. (d) Comparison of language model backbones.

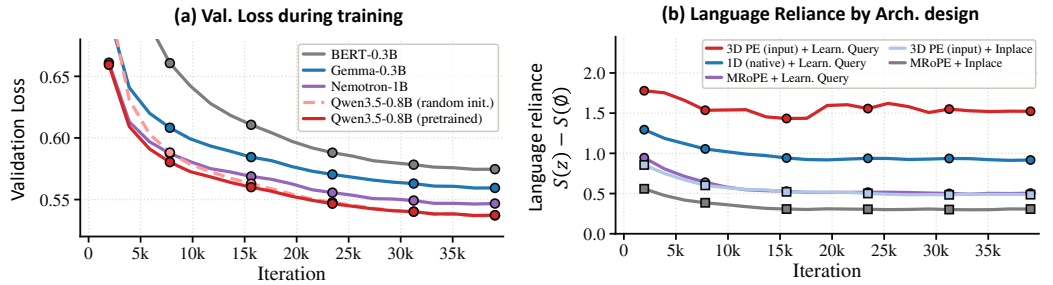

Figure F: **Predictor backbone, initialization, and architecture ablations.** (a) Validation loss for language model backbones, including pretrained and randomly initialized Qwen3.5. (b) Language reliance $S(z) - S(\varnothing)$ for positional encoding and readout variants. 3D input positional embeddings with learned queries yield the highest reliance.

The two initializations reach similar final validation losses (Figure F(a)), suggesting that prediction loss alone does not fully capture planning performance.

C.5 LaLa-World Vision Encoder Ablations

Setup. JEPA-WMs (Terver et al., 2025) shows that the preferred vision encoder for planning varies across domains. We examine this dependence in LaLa-World by comparing frozen DINOv2-S/14 (Oquab et al., 2024), DINOv3-L/16 (Siméoni et al., 2025), SigLIP2-So400m/14 (Tschannen et al., 2025), and DINOv2-S/14 + DINOv3-L/16. For each encoder configuration, we train a separate predictor on all training data, including control trajectories and image and video transitions. For DINOv2-S/14 + DINOv3-L/16, one predictor receives both representations and predicts both as future targets. We also report JEPA-WMs with each DINO encoder. Table D presents results on all ten benchmarks and their average score. The comparison in Table 1 allows the vision encoder to vary across benchmarks, as does JEPA-WMs (Terver et al., 2025); we include those reported scores as reference rows. DINO-WM in Table 1 likewise uses the better-performing vision encoder for each benchmark, while V-JEPA 2-AC retains the V-JEPA 2 encoder used in the original method.

Results. Our encoder ablations suggest that DINOv2-S/14 and DINOv3-L/16 provide complementary information for planning, motivating further work on combining encoders (Tong et al., 2024) and learning representations tailored to latent world modeling. The preferred DINO encoder varies across benchmarks for both methods. For LaLa-World, DINOv2-S/14 performs better on Push-T and Granular, while DINOv3-L/16 performs better on PointMaze and both MetaWorld tasks. Using DINOv2-S/14 + DINOv3-L/16 with one predictor increases the average to 63.0, although improvements are not uniform across benchmarks.

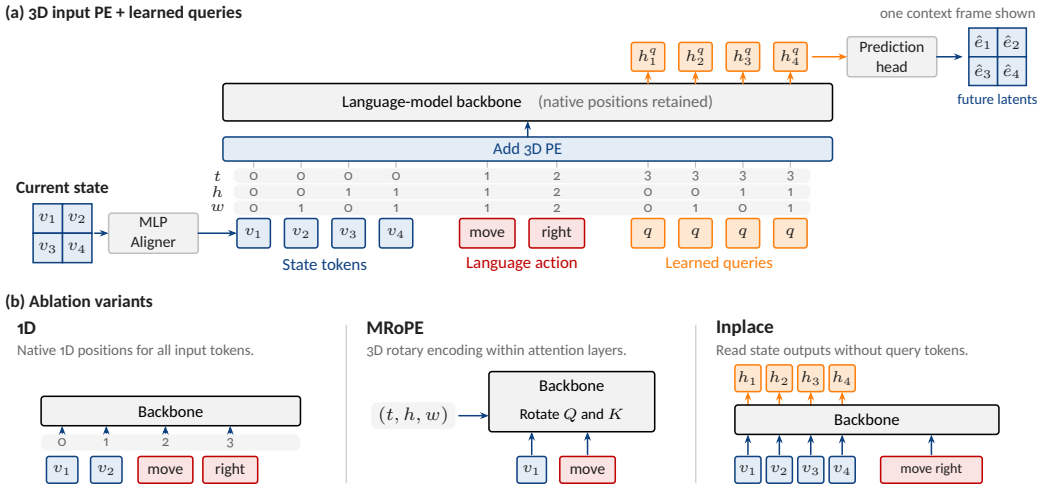

Figure G: **Predictor positional encoding and readout.** (a) Projected state patches, language tokens and copies of one shared learned query vector receive additive 3D positional embeddings. The prediction head maps query outputs to future patch representations. (b) Native 1D uses sequence indices for visual and text tokens, handled by the backbone's positional mechanism. MRoPE uses 3D coordinates to rotate attention queries and keys inside the backbone. Inplace readout uses state-token hidden states.

Initialization	Training scope	Wall	PointMaze	Push-T	DROID	RoboCasa-R	RoboCasa-P	Average
Pretrained LLM	Single-domain	**97.9** (1.5)	**66.7** (4.8)	70.8 (4.6)	38.3 (4.5)	21.9 (7.3)	31.2 (8.2)	54.5 (2.3)
Random init.	Joint	**97.9** (1.5)	50.0 (5.1)	79.2 (4.1)	31.6 (3.8)	31.2 (8.2)	43.8 (8.8)	55.6 (2.4)
Pretrained	Joint	95.8 (2.0)	56.2 (5.1)	**81.2** (4.0)	**40.2** (5.2)	**40.6** (8.7)	**46.9** (8.8)	**60.2** (2.5)

Table C: **LaLa-World Predictor initialization and joint training.** We compare pretrained and random initialization, and single-domain and joint training. We report success rates (%), except for the DROID action score, with standard errors in parentheses. Average is the unweighted mean of the six benchmark scores.

C.6 Additional Results on Planning with Qualitative Language Actions

We compare numerical and qualitative language actions to evaluate whether LaLa-World can plan from descriptions that omit exact action magnitudes. The action conversion and planning procedure are detailed in Section D.4.3.

Table E shows that qualitative language actions support planning in several domains. LaLa-World achieves success rates of 83.3% on Wall, 70.8% on MetaWorld-R, and 60.4% on MetaWorld-RW. Success rates decrease on some tasks, as expected when many distinct numerical magnitudes are mapped to only three qualitative levels. Even with this reduced information, it is still surprising that LaLa-World can use descriptions of direction and relative magnitude to guide planning without receiving exact numerical action values.

C.7 Additional Results on Action Phrasing Robustness

We provide the domain-level results underlying Figure 4 and the corresponding action template paraphrases. We compare predictors trained on control trajectories alone and on all data, including image and video transitions. At evaluation, we apply the substitutions in Table F (top) to the language conditions supplied to the predictor, preserving the numerical action values and directions. For each predictor, we keep the model weights and planning settings unchanged. Table F shows that paraphrasing lowers scores in all five domains when training uses control trajectories alone, reducing the average from 59.3 to 52.5. These results suggest that image and video transitions reduce the predictor's dependence on the wording of the action templates.

Vision encoder	Wall	PointMaze	Push-T	MetaW.-R	MetaW.-RW	Rope	Granular	RoboC.-R	RoboC.-P	DROID	Average
JEPA-WMs											
DINOv2-S/14	**79.2**	**60.4**	**68.8**	20.8	4.2	54.2	**50.0**	**37.5**	**31.2**	40.1	44.6
DINOv3-L/16	76.0	56.2	47.9	**66.7**	**47.9**	**62.5**	41.7	21.9	28.1	**43.0**	**49.2**
Table 1 (reference)	79.2	60.4	68.8	66.7	47.9	62.5	50.0	37.5	31.2	43.0	54.7
LaLa-World											
DINOv2-S/14	95.8	45.8	**81.2**	8.3	4.2	58.3	58.3	40.6	43.8	40.2	47.7
DINOv3-L/16	94.8	56.2	52.1	75.0	52.1	**66.7**	50.0	12.5	**46.9**	30.3	53.7
SigLIP2-So400m/14	60.4	37.5	66.7	33.3	8.3	45.8	41.7	9.4	34.4	35.5	37.3
DINOv2 + DINOv3	**96.9**	**59.4**	80.2	**77.1**	**54.2**	**66.7**	**66.7**	**43.8**	43.8	**41.5**	**63.0**
Table 1 (reference)	95.8	56.2	81.2	75.0	52.1	66.7	58.3	40.6	46.9	40.2	61.3

Table D: **Vision encoder ablations.** We compare frozen vision encoders for JEPA-WMs and LaLa-World. DINOv2 + DINOv3 uses both DINOv2-S/14 and DINOv3-L/16. We report success rates (%), except for the DROID action score. Average is the unweighted mean of the ten benchmark scores. Bold denotes the highest score among encoder configurations within each method; reference rows reproduce the main planning results and are excluded from this comparison.

Language action	Wall	PointMaze	Push-T	MetaWorld-R	MetaWorld-RW	RoboCasa-R	RoboCasa-P
Qualitative	83.3 (3.8)	47.9 (5.1)	4.2 (2.0)	70.8 (6.6)	60.4 (7.1)	28.1 (7.9)	25.0 (7.7)
Numerical	95.8 (2.0)	56.2 (5.1)	81.2 (4.0)	75.0 (6.2)	52.1 (7.2)	40.6 (8.7)	46.9 (8.8)

Table E: **MPC planning with numerical and qualitative language actions.** Qualitative language actions replace numerical magnitudes with *slightly*, no modifier, or *far*. Lower success rates are expected when exact magnitudes are omitted, yet it is still interesting that LaLa-World can plan successfully in several domains from qualitative language actions. Only the language conditions supplied to the predictor change; CEM samples numerical actions, and the simulator executes the selected actions in their original numerical form.

C.8 Additional Results on Latent vs. Generative World Models

Figure H breaks down the planning results in Figure 5 by domain. We use the full-depth LLM predictor for LaLa-World and the full-depth DiT for both RAEv2 variants. LaLa-World outperforms both RAEv2 variants on six of the seven benchmarks and matches five-step RAEv2 on MetaWorld-R. Its average score is 59.4, compared with 40.4 and 48.0 for one- and five-step RAEv2, respectively. These results follow the trend in Section 4.2, showing that the planning advantage of LaLa-World extends across domains.

C.9 Additional Qualitative Planning Results

C.9.1 Planning with Numerical Language Actions

We provide additional visualizations of LaLa-World with numerical language actions on Wall, PointMaze, Push-T, MetaWorld, DROID, and RoboCasa Reach and Place in Figure M, Figure N, Figure O, Figure P, Figure Q, Figure R, and Figure S. The figures show initial and goal observations alongside decoded future predictions, with the language actions below the frames; Wall also includes PCA visualizations of the predicted latents.

C.9.2 Planning with Qualitative Language Actions

We provide additional planning examples with qualitative language actions on Wall, PointMaze, Push-T, MetaWorld, and RoboCasa Reach and Place in Figure T, Figure U, Figure V, Figure W, Figure X, and Figure Y. Each example shows simulator observations during MPC execution, with qualitative action descriptions below the frames; action conversion and execution follow Section D.4.3.

C.10 Qualitative Policy Results

We provide additional LaLa-Policy rollouts on LIBERO Spatial (Figure Z, Figure AA), Object (Figure AB, Figure AC), Goal (Figure AD, Figure AE), and Long (Figure AF, Figure AG). RoboCasa examples cover coffee tasks (Figure AH, Figure AI), opening doors and drawers (Figure AJ,

Phrasing	Wall	PointMaze	MetaWorld-R	RoboCasa-R / DROID
Original	move the agent	push the ball with force	move the gripper	move gripper, rotate roll
Rephrased	push the dot	push the sphere	shift the robot hand	move robot hand, turn roll

Method	Wall	PointMaze	MetaWorld-R	RoboCasa-R	DROID	Average
Control data only	96.9 (1.8)	64.6 (4.9)	68.8 (6.7)	25.0 (7.7)	41.5 (4.6)	59.3 (2.5)
+ Paraphrasing	87.5 (3.4)	56.2 (5.1)	60.4 (7.1)	18.8 (6.9)	39.8 (5.0)	52.5 (2.5)
All data	95.8 (2.0)	56.2 (5.1)	75.0 (6.2)	40.6 (8.7)	40.2 (5.2)	61.6 (2.6)
+ Paraphrasing	92.7 (2.7)	57.3 (5.0)	72.9 (6.4)	43.8 (8.8)	38.3 (5.3)	61.0 (2.7)

Table F: **Robustness to action phrasing.** We report planning results with original and paraphrased action templates (bottom) and the substitutions used at evaluation (top). All data combines control trajectories with image and video transitions. We report success rates (%), except for the DROID action score, with standard errors in parentheses.

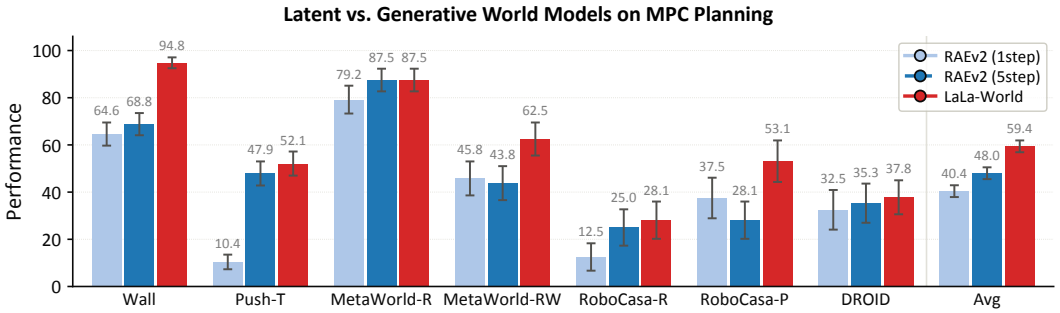

Figure H: **Domain-level comparison of latent and generative world models for MPC planning.** We compare LaLa-World with RAEv2 using one and five denoising steps under the matched setup in Section 4.2. We report success rates (%) for simulation benchmarks and the offline action score for DROID.

Figure AK), object transfer (Figure AL, Figure AM), and stove control (Figure AN). For each camera view, simulator observations (upper row) are paired with decoded future predictions (lower row) during policy execution.

D Implementation and Experimental Details

D.1 LaLa-World Training Data Details

Table G reports the training data size for each dataset. We combine these datasets and sample in proportion to their sizes. We reserve approximately 2% of episodes or editing pairs for validation and keep sequences from the same source video in one split.

Simulation data. We use the six simulation datasets from DINO-WM and JEPA-WMs (Zhou et al., 2025; Terver et al., 2025) listed in Section 4. For Wall, PointMaze, Push-T, and MetaWorld, each transition spans five simulator steps; for Rope and Granular, it spans one push. We enumerate overlapping transitions within each episode and pair each with both numerical and qualitative action descriptions. Action aggregation is described in Section D.4.2.

DROID. We use the LeRobot release of DROID (Khazatsky et al., 2024), retaining one external camera view and every fourth frame of the 15-fps recordings.[3] We derive action descriptions from changes in end-effector position, orientation, and gripper state between retained frames. Our DROID subset contains 1.5 million target transitions, approximately 20% of the original dataset's transitions. This keeps DROID from dominating the mixture under a fixed training budget. The same subset is used for both action description formats.

Image and video data. From OpenVid-1M (Nan et al., 2025), we select clips by duration, motion, and aesthetic scores and extract ten-frame sequences at 0.5-second intervals. Qwen3-VL-30B-A3B-Instruct describes the visible change between each adjacent frame pair to provide its language

[3] https://huggingface.co/datasets/lerobot/droid_1.0.1

Dataset	Training data size
Wall	86,296
PointMaze	186,675
Push-T	2,199,488
MetaWorld	1,162,310
Rope	18,525
Granular	18,582
DROID	1,500,000
OpenVid-1M	1,023,363
OmniEdit	295,353
ShareGPT-4o-Image	45,613

Table G: **Data used to train LaLa-World.** We count target transitions for control and video data and editing pairs for OmniEdit and ShareGPT-4o-Image. Counts exclude validation data and count each control transition once across its two action description formats.

condition. We also use source–edited image pairs and their instructions from OmniEdit (Wei et al., 2025) and ShareGPT-4o-Image (Chen et al., 2025b).

Further DROID-focused post-training, as in V-JEPA 2-AC (Assran et al., 2025), may improve robot planning; we leave this to future work.

D.2 LaLa-World Architecture Details

Each example contains observed context frames, language conditions for the corresponding transitions, and one target frame. In multi-frame training, we sample the context length uniformly from one to three frames, subject to the available episode history; image pairs provide one context frame. A two-layer MLP projects each frame's patch representations into the language model's embedding space. As illustrated in Figure G(a), we append one query token per target patch after the context and language tokens. The queries are copies of a shared learned vector, with their target locations specified by 3D positional embeddings. These embeddings encode a temporal index and the row and column of each spatial patch. Context patches share a temporal index within each frame, while queries receive the target frame's temporal index and spatial coordinates. We concatenate sinusoidal encodings for the three axes, multiply them by a learnable scale, and add them to the input embeddings while retaining the backbone's native positional encoding. The appended queries receive the preceding visual and language context through the LLM backbone. A two-layer MLP prediction head maps their output hidden states back to the vision encoder's feature dimension, producing the future patch grid in one forward pass. Both projection MLPs use GELU between their linear layers. The target frame is used only to provide supervision for these predictions.

D.3 LaLa-World Training Details

We keep the observation encoder frozen and jointly train the language-model backbone, visual projection layers, and learned query tokens. The same encoder provides the context and target representations. We optimize the prediction objective in Section 3.2. We use AdamW with an effective global batch size of 512, $(\beta_1, \beta_2) = (0.9, 0.98)$, and weight decay of 0.05. The learning rate increases linearly from 1×10^{-5} to 2×10^{-4} over the first 1,000 optimizer steps, followed by a cosine decay schedule with a minimum learning rate of 1×10^{-6}. We clip the gradient norm at 1.0 and use gradient accumulation to maintain the global batch size. We train for 20 epochs using bfloat16 mixed precision on eight NVIDIA H100 80GB GPUs. We define each epoch as approximately one million training examples sampled from the training pool.

Benchmark	Evaluation trajectories	Trajectory length	Planning horizon H	Executed steps per plan	Substeps per step	CEM settings	Evaluation criterion
Wall	96	6	6	6	5	$30 \times 300 \times 10$	Position error < 4.5
PointMaze	96	6	6	6	5	$30 \times 300 \times 10$	Position error < 0.5
Push-T	96	6	6	6	5	$30 \times 300 \times 10$	Position error < 20 and angle error $< 20°$
MetaWorld-R	48	20	6	3	5	$15 \times 300 \times 10$	Success at any step
MetaWorld-RW	48	20	6	3	5	$15 \times 300 \times 10$	Success at any step
Rope	24	5	5	1	1	$30 \times 300 \times 30$	Chamfer distance < 0.675
Granular	12	5	5	1	1	$30 \times 300 \times 30$	Chamfer distance < 0.323
DROID	64	3	3	–	1	$15 \times 300 \times 10$	Action Score
RoboCasa-R	32	12	3	1	5	$15 \times 300 \times 10$	Subtask success at final step
RoboCasa-P	32	12	3	1	5	$15 \times 300 \times 10$	Subtask success at final step

Table H: **MPC planning settings and evaluation criteria.** One step corresponds to one world-model transition, with the listed number of action updates (substeps). CEM settings are iterations $\times$ candidate sequences per iteration $\times$ retained sequences. For benchmarks requiring replanning, we execute the listed number of steps, observe the environment, and re-optimize the action sequence with CEM. Position and angle errors are measured at the final step, using L_2 distance for position. DROID is evaluated offline without simulator execution.

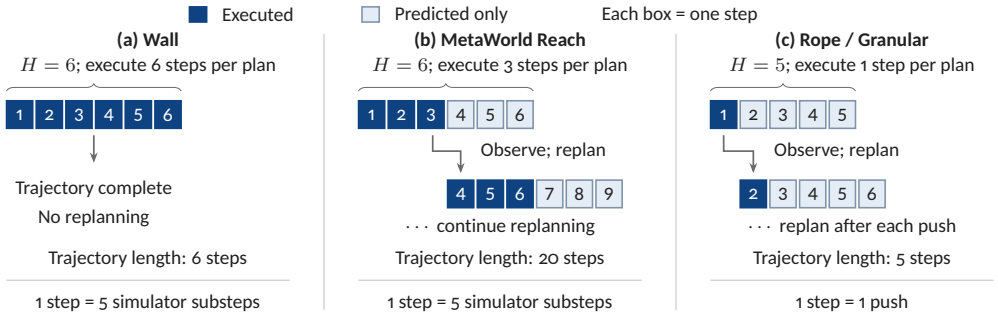

Figure I: **MPC planning horizons and action execution.** One step corresponds to one world-model transition, with the listed number of action updates (substeps). For benchmarks requiring replanning, we obtain a new observation after executing the indicated prefix and optimize a new sequence with CEM.

D.4 MPC Planning and Evaluation Details

D.4.1 Benchmark Settings and Evaluation Metrics

We follow the planning and evaluation protocols of JEPA-WMs (Terver et al., 2025) and DINO-WM (Zhou et al., 2025), with our own success criteria for Rope and Granular. As in Section 4, we evaluate planning with model predictive control (MPC), using the cross-entropy method (CEM) to optimize action sequences. Table H summarizes the MPC settings, CEM parameters, and evaluation criteria for each benchmark. One planning step corresponds to one transition predicted by the world model. In simulation, each action update within a step is called a substep. The planning horizon H specifies how many steps the model predicts ahead during each candidate rollout, while trajectory length gives the total step budget.

In simulation, we execute an initial segment of the sequence optimized by CEM, whose length is specified by executed steps. For benchmarks requiring replanning, we follow the MPC loop: after executing this segment, we obtain a new observation and re-optimize the action sequence with CEM. Figure I illustrates these settings. Wall executes all six planned steps in one round, whereas MetaWorld Reach executes three of six and Rope and Granular execute one of five before replanning.

For DROID, we evaluate planned actions offline without simulator execution. We compute the endpoint position error $\mathrm{error}_{\mathrm{xyz}}$ in meters from the cumulative planned displacement and report the Action Score:

$$\text{Action Score} = \max\left(0,\ 800\,(0.1 - \mathrm{error}_{\mathrm{xyz}})\right). \tag{5}$$

For zero-shot planning on RoboCasa (Nasiriany et al., 2024), we reuse DROID's action sampling statistics because the two domains share the same action definitions and dimensionality.

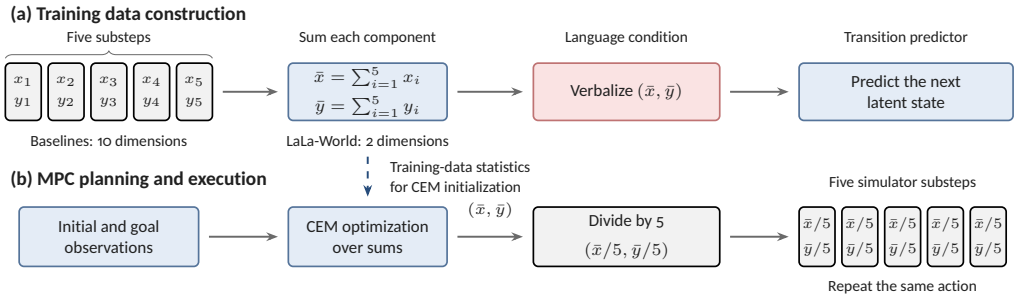

Figure J: **Substep aggregation for language conditioning and action execution.** (a) Summing across substeps reduces the action dimension per step from 10 to 2 for language conditioning. (b) Within MPC, CEM optimizes these sums, which are distributed equally across five substeps for execution. This discards within-step variation. Yet, it is notable that LaLa-World achieves accurate prediction and high planning success rates.

D.4.2 Action Aggregation and Execution

The baselines represent each planning step by concatenating the actions across its substeps. For example, five substeps with two action components, x and y, yield a 10-dimensional vector. Verbalizing every component would produce a long language condition. We therefore sum each component across substeps when constructing the training data and verbalize only these sums. In this example, the language condition describes two values, $\bar{x} = \sum_{i=1}^{5} x_i$ and $\bar{y} = \sum_{i=1}^{5} y_i$, as shown in Figure J(a).

Within MPC, CEM optimizes action sequences in this aggregated action space, with its initial sampling statistics computed from the aggregated training data. Each sampled action is verbalized using the training templates to condition the predictor. To execute a selected action, we divide each optimized sum by the number of substeps and repeat the resulting action at every substep. For the five-substep example, we apply $(\bar{x}/5, \bar{y}/5)$ five times, as illustrated in Figure J(b). This preserves the component sums but discards variation within a step, which can lead to differences between predicted and executed transitions. It is notable that LaLa-World achieves accurate future prediction and high planning success rates despite this loss of information.

D.4.3 Qualitative Language Action Conversion

Figure K illustrates the conversion rule and the planning procedure.

Action conversion. Numerical language actions preserve each action component's direction and magnitude. To construct qualitative descriptions, we define a conversion rule for each component and direction using the action range observed in the corresponding training data. We divide the interval from zero to the maximum observed magnitude into three equal-width bins, represented by *slightly*, no modifier, and *far*, respectively. The direction remains unchanged. For example, when the maximum leftward magnitude is 6, `move 0.4 left` becomes `move slightly left`, and `move 4.5 left` becomes `move far left`.

MPC planning procedure. At each MPC planning round, CEM samples numerical action sequences as in the main experiments. Before each sequence is evaluated by LaLa-World, the conversion rule maps its actions to qualitative descriptions. These descriptions condition the latent rollout, which CEM scores using the same goal-distance cost in Equation (2). The simulator executes the selected actions using their original numerical values, and re-planning follows the same MPC procedure. We keep the trained predictor and all other planning settings unchanged.

D.5 RAEv2 Baseline Details

Model and initialization. We initialize the generative world model baseline's DiT from the released RAEv2 checkpoint pretrained for image generation on ImageNet.[4] Its frozen DINOv3-L/16 encoder

[4] https://huggingface.co/nyu-visionx/RAEv2-models/tree/main/stage2/imagenet/dinov3l-k7

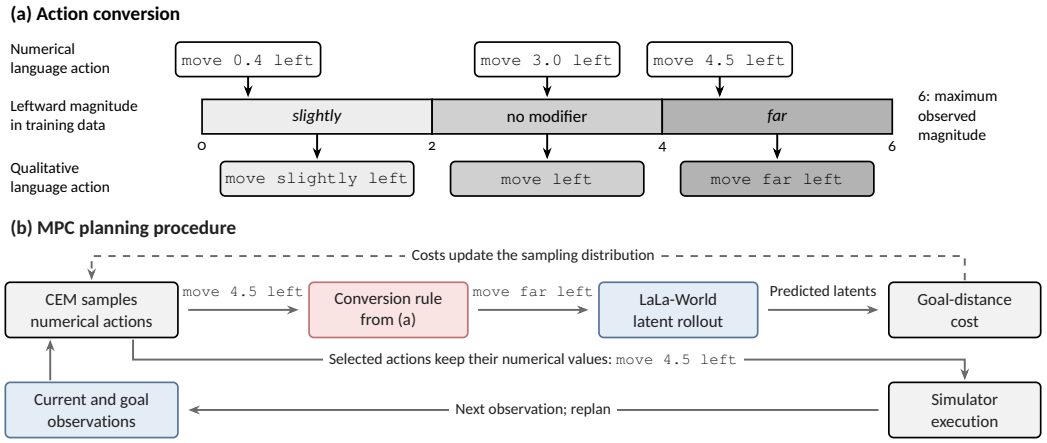

Figure K: **MPC planning with qualitative language actions.** (a) Each magnitude maps to one of three equal-width bins, and the direction is kept. (b) LaLa-World evaluates CEM samples through their qualitative descriptions, while the simulator executes the selected actions with their numerical values. Thus, LaLa-World plans without exact action magnitudes.

aggregates features from seven layers to form the visual representation $e_t = E_\theta(x_t)$. A frozen Qwen3-0.6B encodes the language condition z into text embeddings $T(z)$.

Training. For each transition (x_t, z, x_{t+1}), we encode the future observation as $e_{t+1} = E_\theta(x_{t+1})$ and interpolate it with Gaussian noise for flow matching (Singh et al., 2026):

$$e_{t+1}^{(\tau)} = (1-\tau)e_{t+1} + \tau\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \quad \tau \in [0, 1]. \tag{6}$$

Here, t indexes environment time and τ denotes the noise level. The DiT receives $e_{t+1}^{(\tau)}$, τ, the current observation representation e_t, and the text embeddings $T(z)$. We train it with flow matching and a representation alignment (REPA) loss (Yu et al., 2025) that aligns intermediate DiT features with DINOv2-B/14 features of the future observation.

MPC planning. We use the same MPC procedure as LaLa-World, with CEM optimizing candidate action sequences. Starting from Gaussian noise, the DiT produces a future representation $\hat{e}_{t+1}$ through denoising, conditioned on the current observation and language action. We evaluate RAEv2 with one and five denoising steps per prediction. CEM scores candidate action sequences using the mean squared error between the final predicted representation $\hat{e}_{t+H}$ and the encoded goal $E_\theta(x_{\text{goal}})$, following Equation (2). The goal is encoded with the same DINOv3-L/16 feature aggregation as the training targets. Pixel decoding is not used to compute the planning cost.

Visualization. For Figure 6, RAEv2's predicted representations are mapped to pixels with its released DINOv3-L-K7 decoder,[5] while LaLa-World uses the decoder described in Section 4.

D.6 LaLa-Policy Architecture Details

As illustrated in Figure L, LaLa-Policy extends the predictor of LaLa-World with proprioceptive input and action queries. Following previous VLAs that use SigLIP-based visual encoders (Black et al., 2025; Kim et al., 2025; Li et al., 2025b), we use SigLIP2-So400M/14 at 224×224 resolution and the 24-layer Qwen3.5-0.8B as the predictor backbone. Each camera view is encoded independently, and a shared two-layer MLP projects its patch representations into the backbone's embedding space. We add learned view embeddings and map the current proprioceptive state to one token through a separate MLP.

The input sequence contains visual tokens, the proprioceptive token, task instruction tokens, action queries, and future queries, in this order. We use $N = 16$ action queries for LIBERO and $N = 32$ for RoboCasa, with one future query per spatial patch in each camera view. A two-layer MLP maps

[5] https://huggingface.co/nyu-visionx/RAEv2-models/tree/main/stage1/general/dinov3l-k7

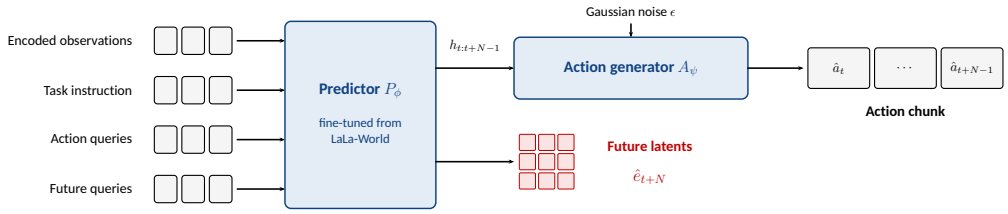

Figure L: **LaLa-Policy architecture.** The predictor, fine-tuned from LaLa-World, receives encoded observations, a task instruction, and learned action and future queries. It predicts future visual latents and provides conditioning features for an action generator, which produces an action chunk from Gaussian noise. Proprio and intermediate projections are omitted for clarity.

the future-query outputs to the encoder's feature dimension to predict the representations at $t + N$. The action generator follows Isaac-GR00T (Bjorck et al., 2025), using 16 DiT blocks of width 1,536 with alternating cross-attention and self-attention. It generates the action chunk from Gaussian noise, conditioned on the predictor's full sequence of output hidden states and the current proprioceptive state.

D.7 Policy Model Training Data

We describe the datasets used to fine-tune LaLa-Policy in Section 5. The episode counts below refer to the data before the training/validation split.

LIBERO. We use the Spatial, Object, Goal, and Long suites of LIBERO (Liu et al., 2023), comprising 40 language-conditioned manipulation tasks. The original data contain 50 human demonstrations per task, for a total of 2,000 episodes. We adopt the replayed datasets released by Cosmos Policy (Kim et al., 2026).[6] Its preprocessing removes no-op transitions and filters unsuccessful replays from the policy demonstration set. The resulting set contains 1,700 successful episodes: 430, 456, 438, and 376 from Spatial, Object, Goal, and Long, respectively. We use these episodes for policy training and the full set of 2,000 replays, including failures, for action-conditioned future prediction.

RoboCasa. We use 24 kitchen manipulation tasks from RoboCasa (Nasiriany et al., 2024) under two data settings. For the setting with approximately 50 human demonstrations per task, we use the Cosmos Policy release.[7] It provides 1,199 successful episodes for policy training and 1,291 replay episodes, including 92 failures, for action-conditioned future prediction. For the 300-demonstration setting, we use RoboCasa's released data generated with MimicGen (Mandlekar et al., 2023).

D.8 LaLa-Policy Fine-tuning Details

We initialize the predictor from pretrained LaLa-World and jointly train it with the newly added policy modules, while keeping the observation encoder frozen. The action generator is trained from scratch. We minimize a flow-matching velocity loss on policy examples and a mean squared error loss on future representations for both types of examples. Both loss coefficients are set to 1.

We use AdamW with an effective global batch size of 256, $(\beta_1, \beta_2) = (0.9, 0.98)$, and weight decay of 0.05. The learning rate increases linearly from 1×10^{-6} to 1×10^{-4} over the first 1,000 optimizer steps, then follows a cosine decay to 1×10^{-6}. We clip the gradient norm at 1.0 and use bfloat16 mixed precision on eight NVIDIA H100 80GB GPUs, with gradient accumulation to maintain the global batch size. We train for 8 epochs on LIBERO, 12 epochs on RoboCasa with approximately 50 human demonstrations per task, and 3 epochs on RoboCasa with 300 MimicGen demonstrations per task.

D.9 Policy Model Evaluation Details

We evaluate LaLa-Policy through direct interaction with the LIBERO (Liu et al., 2023) and RoboCasa (Nasiriany et al., 2024) simulators, following the evaluation procedure of Cosmos Policy (Kim

[6] https://huggingface.co/datasets/nvidia/LIBERO-Cosmos-Policy
[7] https://huggingface.co/datasets/nvidia/RoboCasa-Cosmos-Policy

et al., 2026). The policy receives the task instruction, current proprioceptive state, and camera observations. LIBERO provides an external camera view and a wrist view, while RoboCasa provides left and right external views and a wrist view.

After 10 initial no-op steps, the policy predicts an action chunk. On LIBERO, the chunk contains 16 actions, all of which are executed before the next query. On RoboCasa, the chunk contains 32 actions, of which we execute the first 16. Each subsequent query uses the latest simulator observations. The predicted future representations are not used for action selection during evaluation.

We run 50 episodes for each of the 40 tasks across LIBERO's Spatial, Object, Goal, and Long suites. For RoboCasa, we run 50 episodes for each of 24 tasks, distributed equally across five test scenes with held-out object instances from split B. The step budgets follow the Cosmos Policy evaluation code: 220, 280, 300, and 520 for the four LIBERO suites, and 300–1,000 depending on the RoboCasa task. An episode ends upon task success or exhaustion of its step budget. We report success rates averaged across suites for LIBERO and across tasks for RoboCasa.

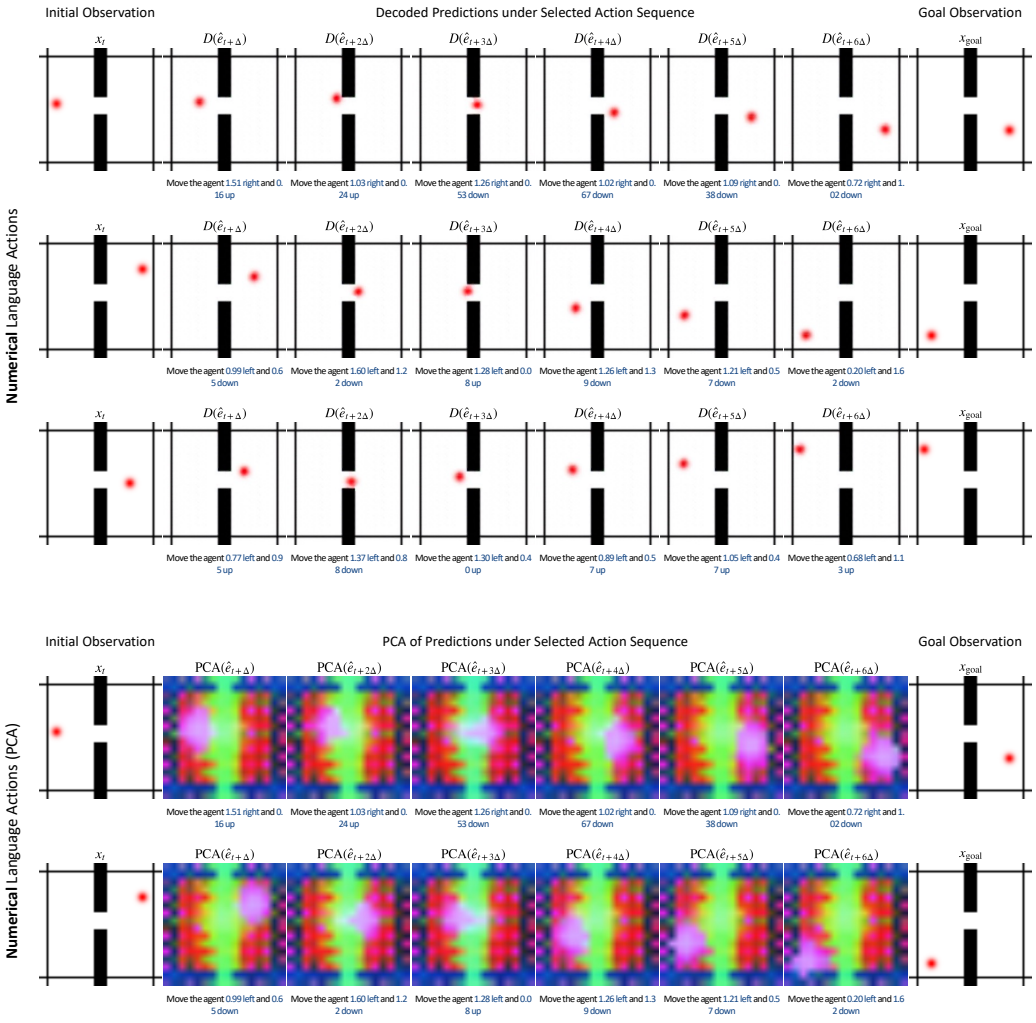

Figure M: **Numerical language actions on Wall.** Initial and goal observations bracket decoded future predictions (top) and PCA visualizations of predicted latents (bottom), with the selected language actions below each frame.

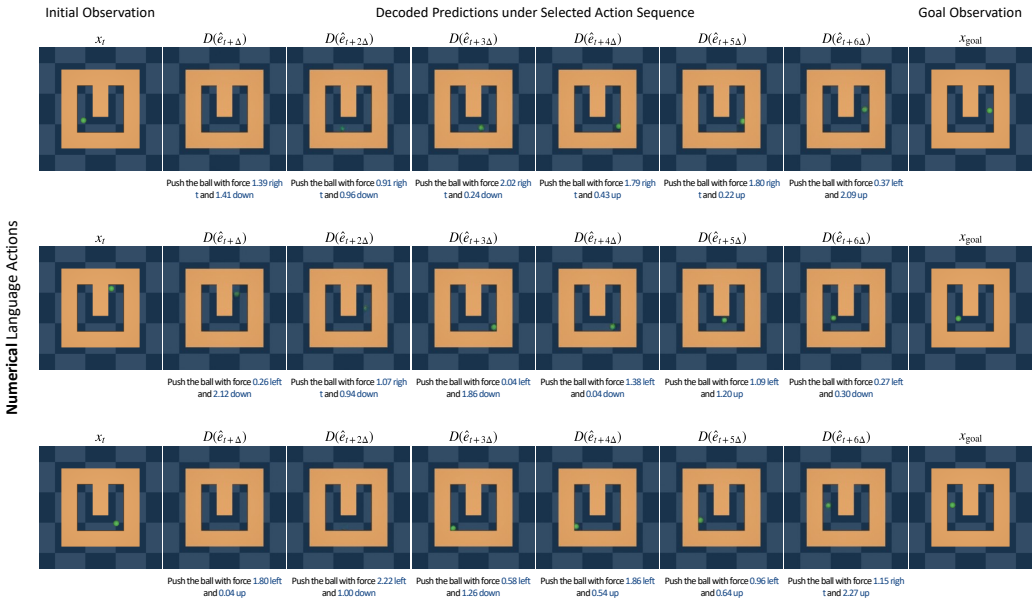

Figure N: **Numerical language actions on PointMaze.** Each row shows initial and goal observations and decoded future predictions under the selected actions, which specify pushing forces and directions.

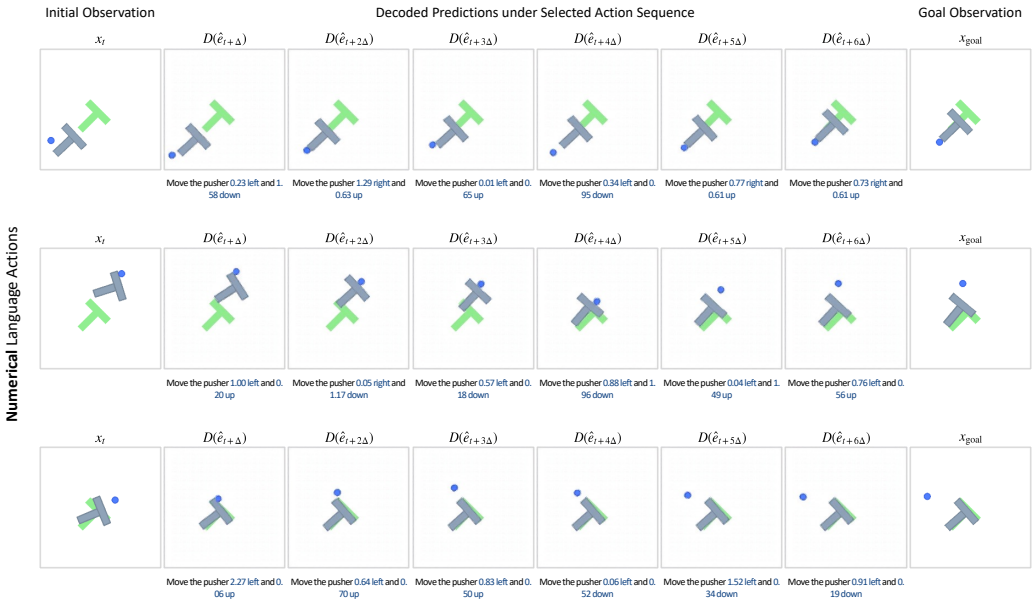

Figure O: **Numerical language actions on Push-T.** Initial and goal observations bracket decoded future predictions under the selected pusher movements, with numerical displacements described below each frame.

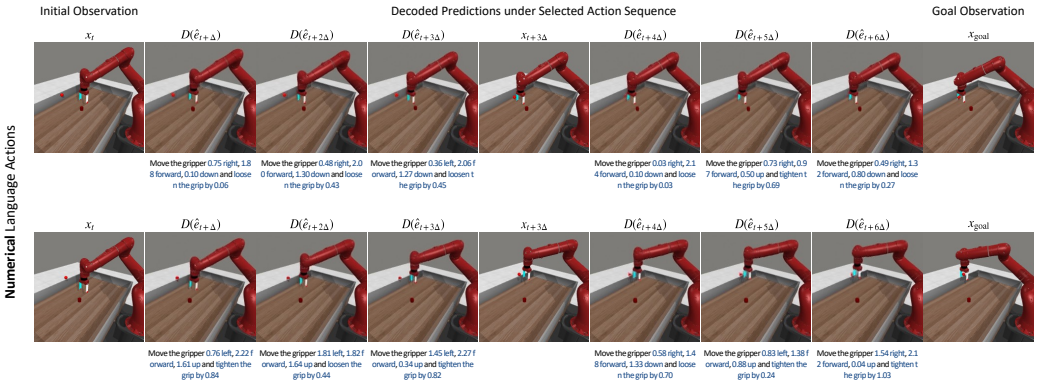

Figure P: **Numerical language actions on MetaWorld.** Decoded predictions are shown before and after replanning from the simulator observation $x_{t+3\Delta}$, alongside initial and goal observations.

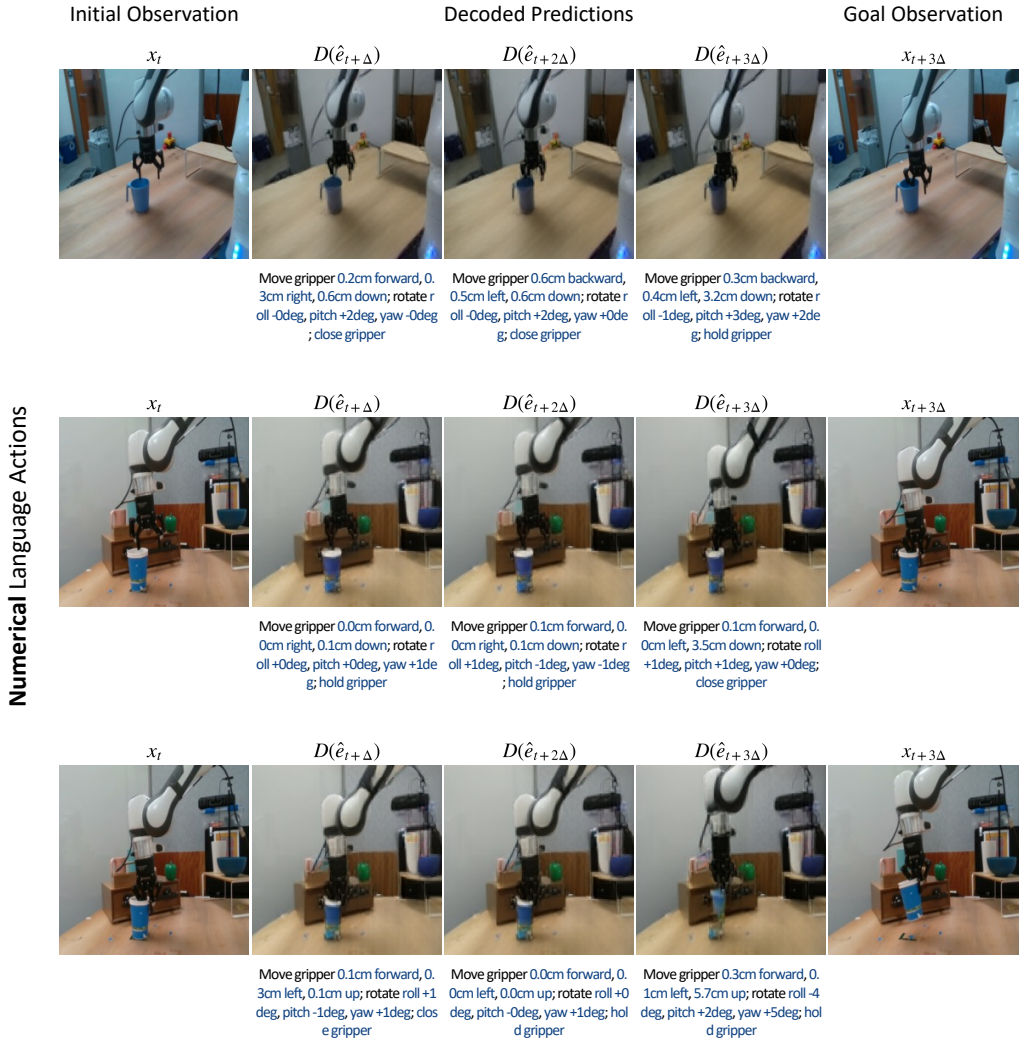

Figure Q: **Numerical language actions on DROID.** Each row shows an initial observation, three decoded future predictions, and the goal observation, with language actions below.

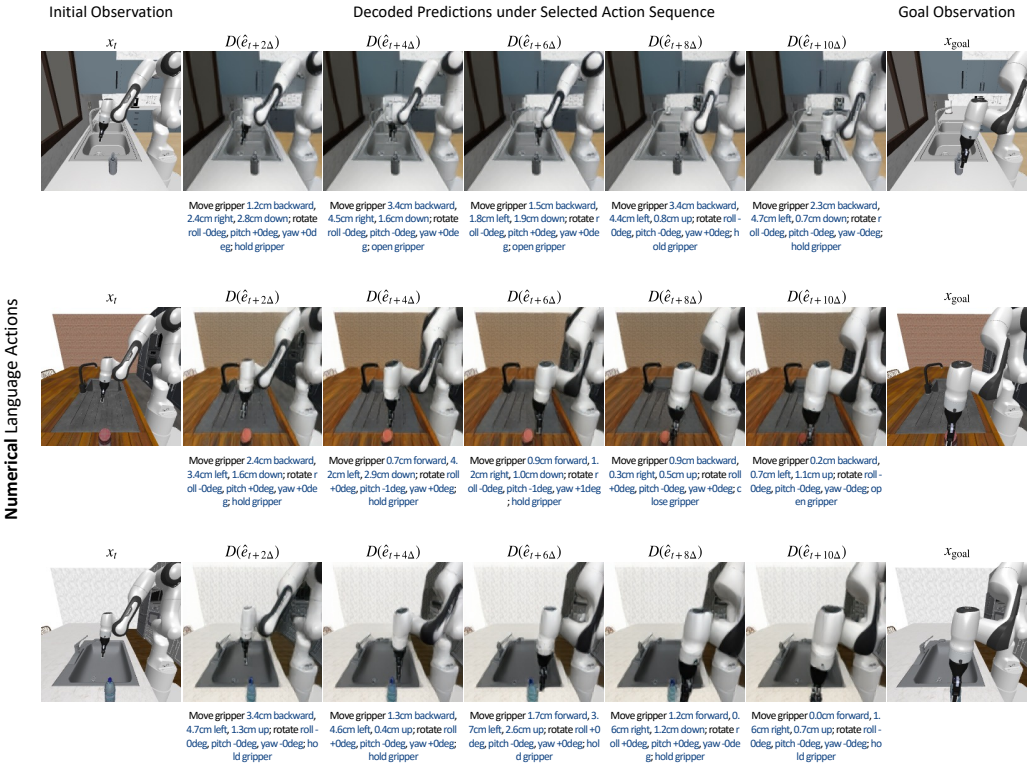

Figure R: **Numerical language actions on RoboCasa Reach.** Decoded future predictions under selected gripper actions are shown between initial and goal observations. RoboCasa is excluded from world-model training.

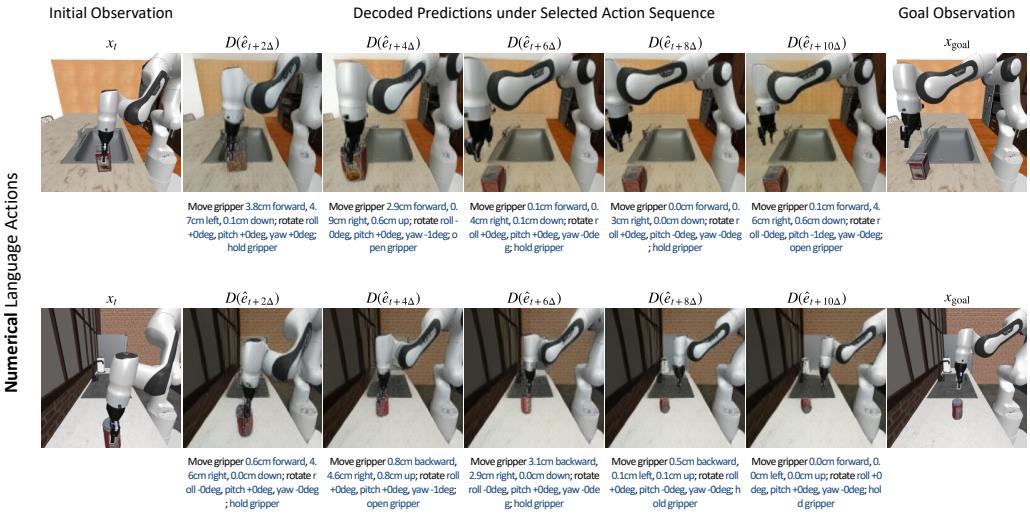

Figure S: **Numerical language actions on RoboCasa Place.** Initial and goal observations bracket decoded future predictions during object placement, with the selected gripper actions below each frame.

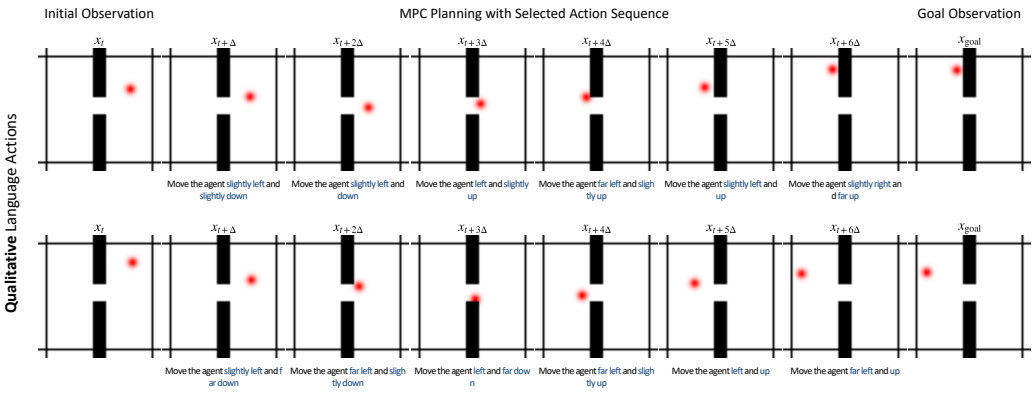

Figure T: **Qualitative language actions on Wall.** Simulator observations show the agent passing through the opening toward the goal. Actions describe directions and relative movement magnitudes.

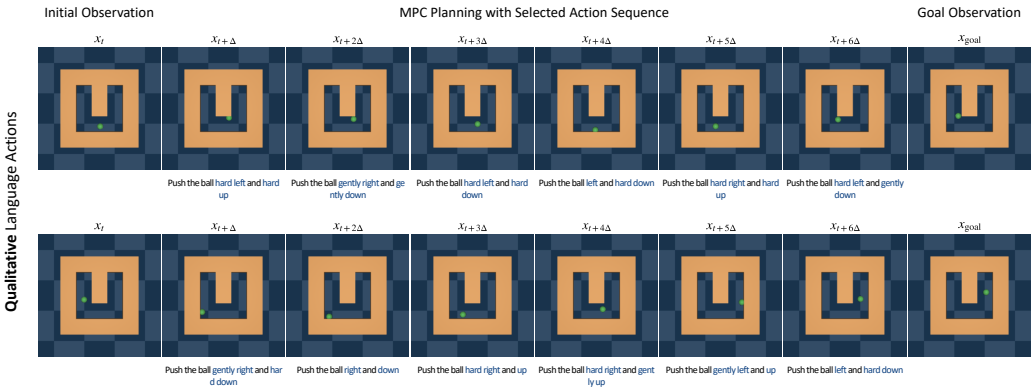

Figure U: **Qualitative language actions on PointMaze.** Simulator observations show the ball moving through the maze under MPC, with pushing directions and qualitative force descriptions below the frames.

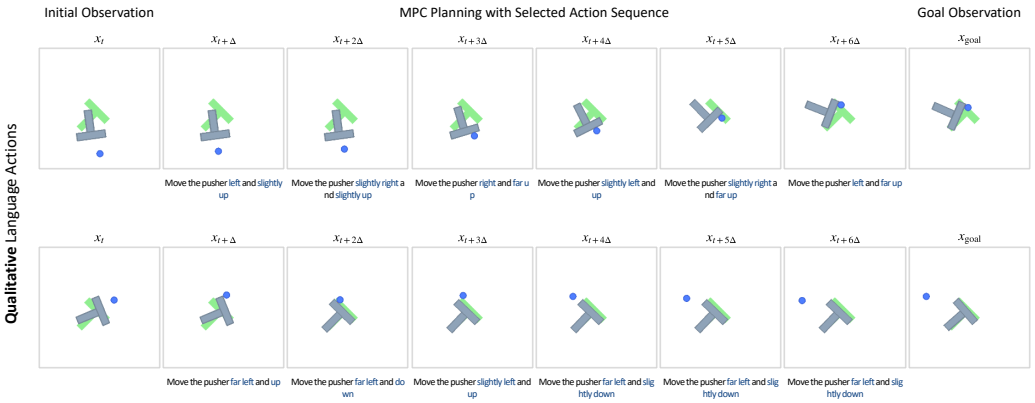

Figure V: **Qualitative language actions on Push-T.** Initial and goal observations bracket simulator states during MPC, with the pusher's direction and relative displacement described below each frame.

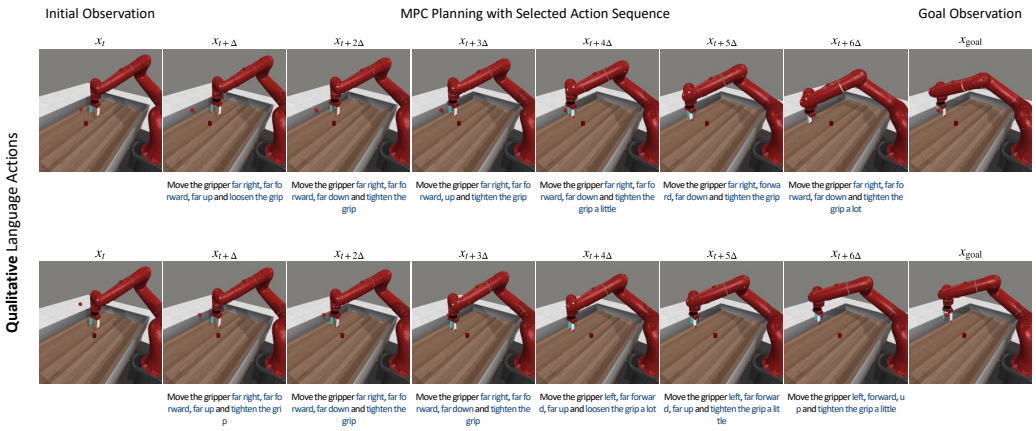

Figure W: **Qualitative language actions on MetaWorld.** Simulator observations show reaching trajectories planned using descriptions of gripper motion and grip adjustment without exact numerical magnitudes.

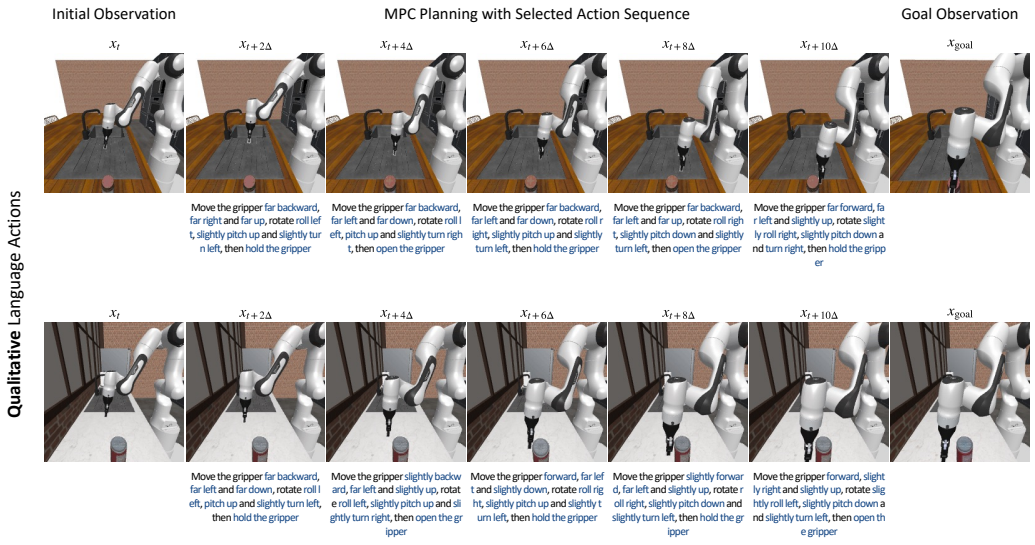

Figure X: **Qualitative language actions on RoboCasa Reach.** Simulator observations show zero-shot MPC execution toward the goal, conditioned on qualitative descriptions of gripper translation, rotation, and opening or closing.

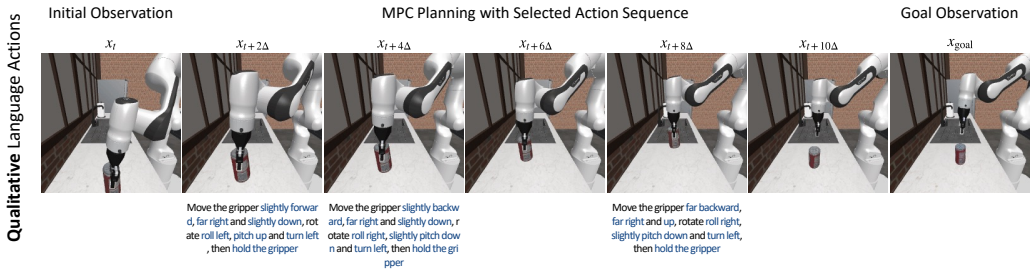

Figure Y: **Qualitative language actions on RoboCasa Place.** Simulator observations show an object-placement trajectory under MPC, with qualitative gripper actions and the goal observation for reference.

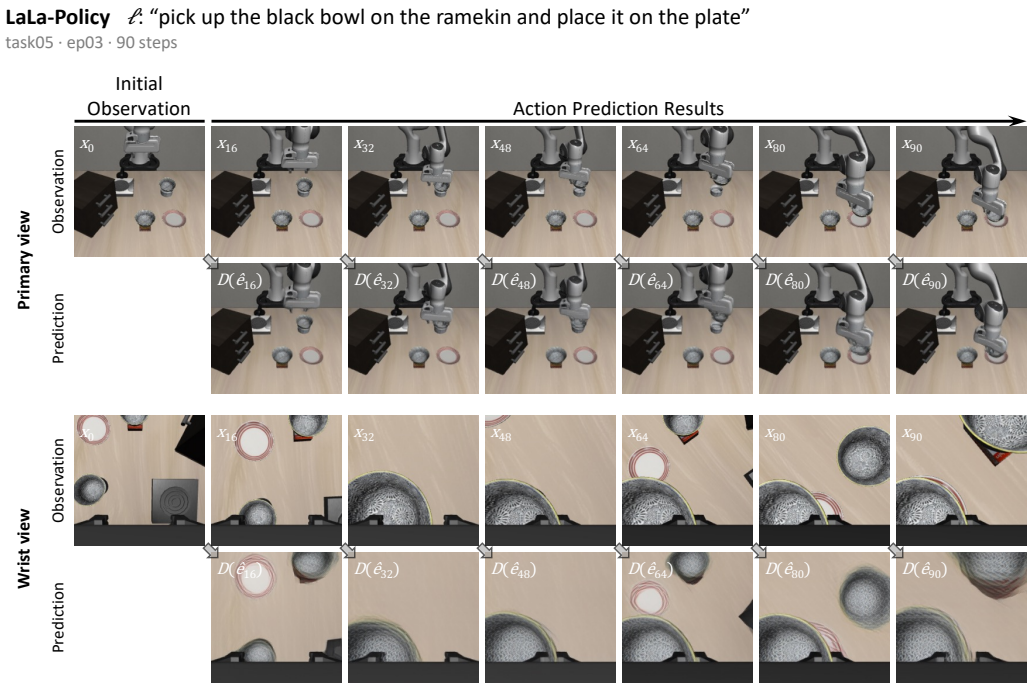

Figure Z: **LIBERO Spatial: bowl on a ramekin.** LaLa-Policy moves the bowl from the ramekin onto the plate. Primary and wrist views compare simulator observations (upper rows) with decoded future predictions (lower rows).

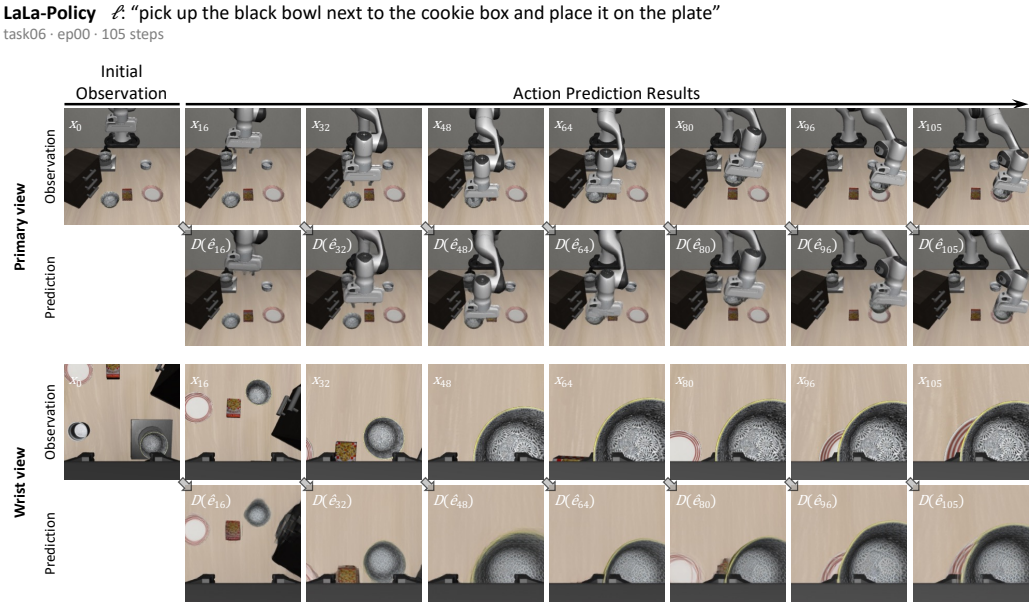

Figure AA: **LIBERO Spatial: bowl next to a cookie box.** LaLa-Policy moves the specified bowl onto the plate. Simulator observations and decoded future predictions are paired for the primary and wrist views.

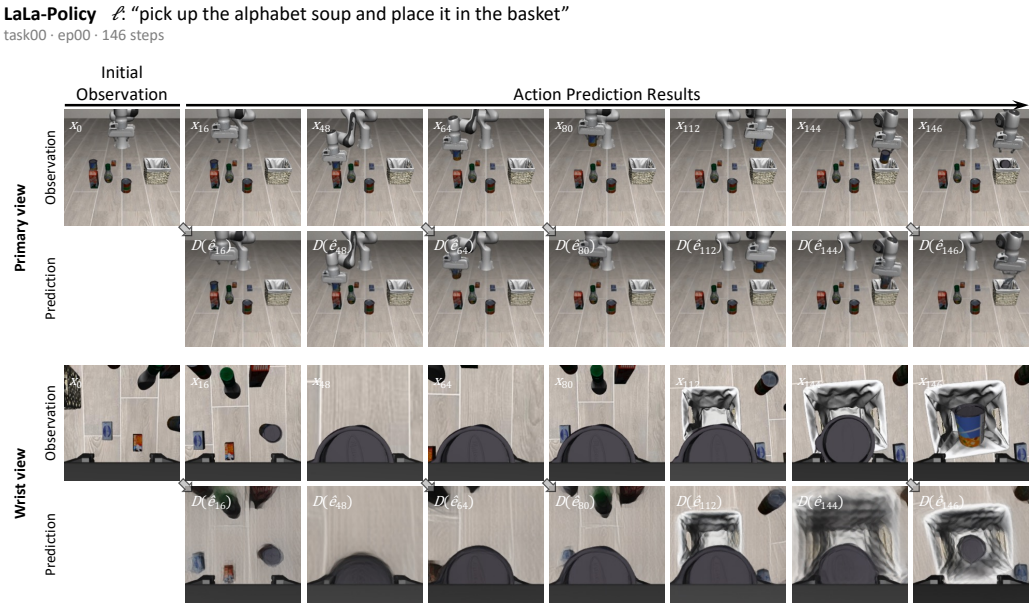

Figure AB: **LIBERO Object: alphabet soup.** LaLa-Policy picks up the alphabet soup and places it in the basket. Each camera view pairs observed states with decoded future predictions.

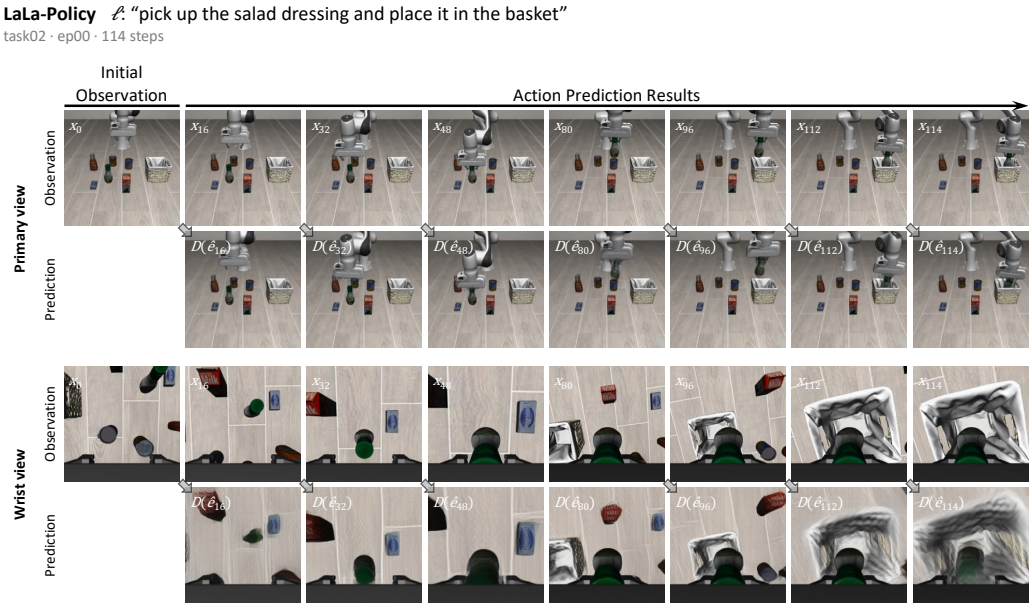

Figure AC: **LIBERO Object: salad dressing.** LaLa-Policy transfers the salad dressing to the basket. Primary and wrist views show simulator observations above the corresponding decoded future predictions.

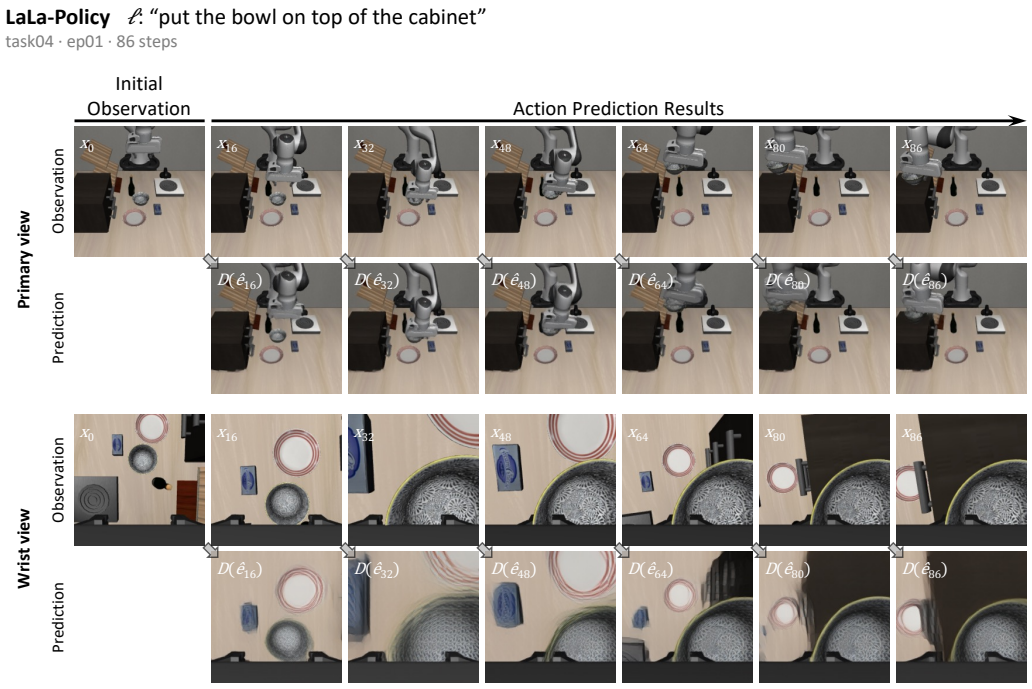

Figure AD: **LIBERO Goal: bowl on the cabinet.** LaLa-Policy places the bowl on top of the cabinet. Primary and wrist views compare observed states with decoded future predictions throughout execution.

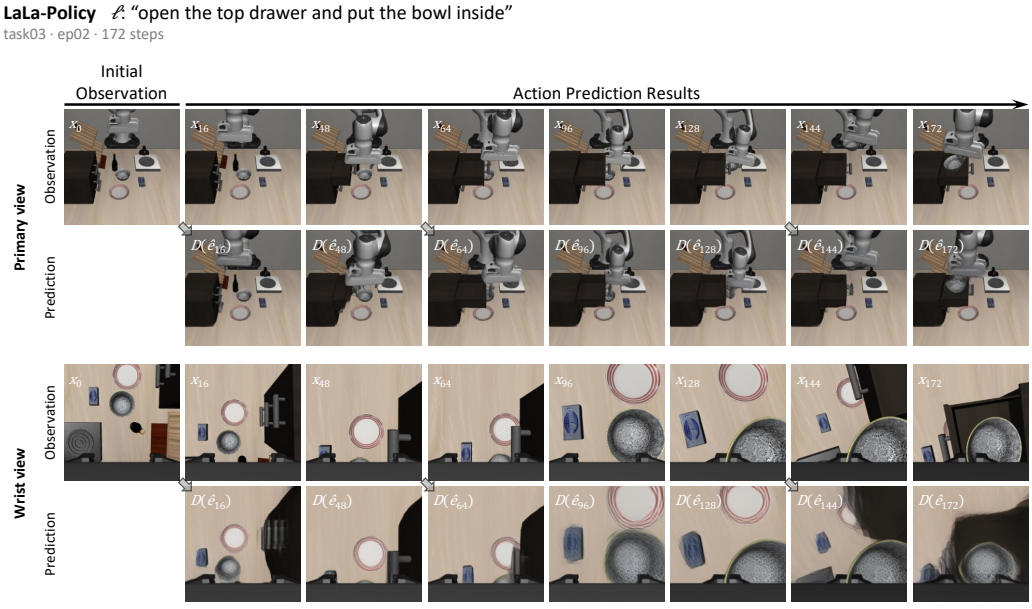

Figure AE: **LIBERO Goal: bowl inside the drawer.** LaLa-Policy opens the top drawer and places the bowl inside. Both camera views pair simulator observations with decoded future predictions.

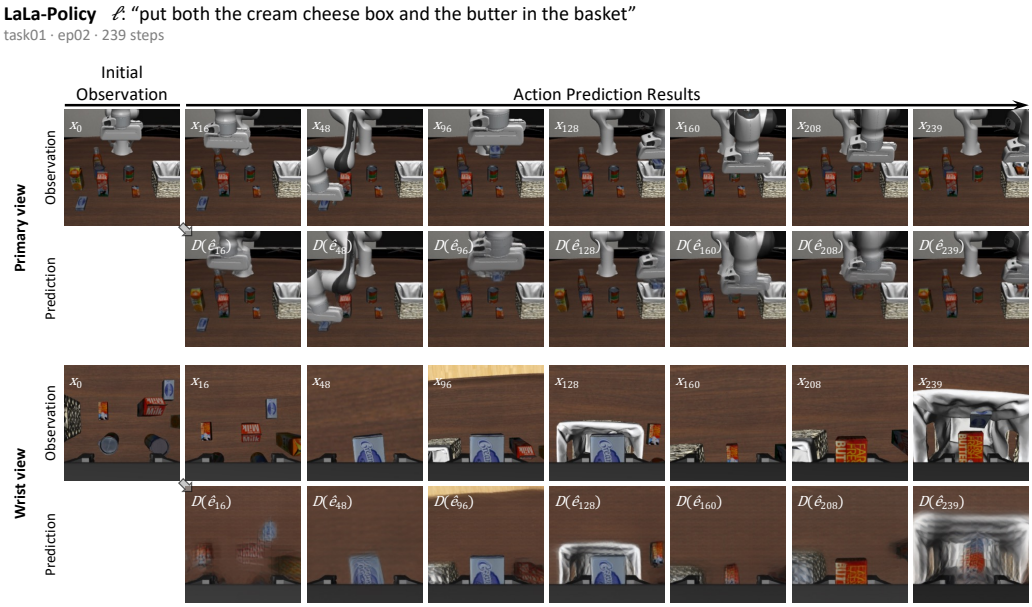

Figure AF: **LIBERO Long: placing two objects in a basket.** LaLa-Policy transfers the cream cheese box and butter to the basket. Observations and decoded future predictions are shown for both camera views.

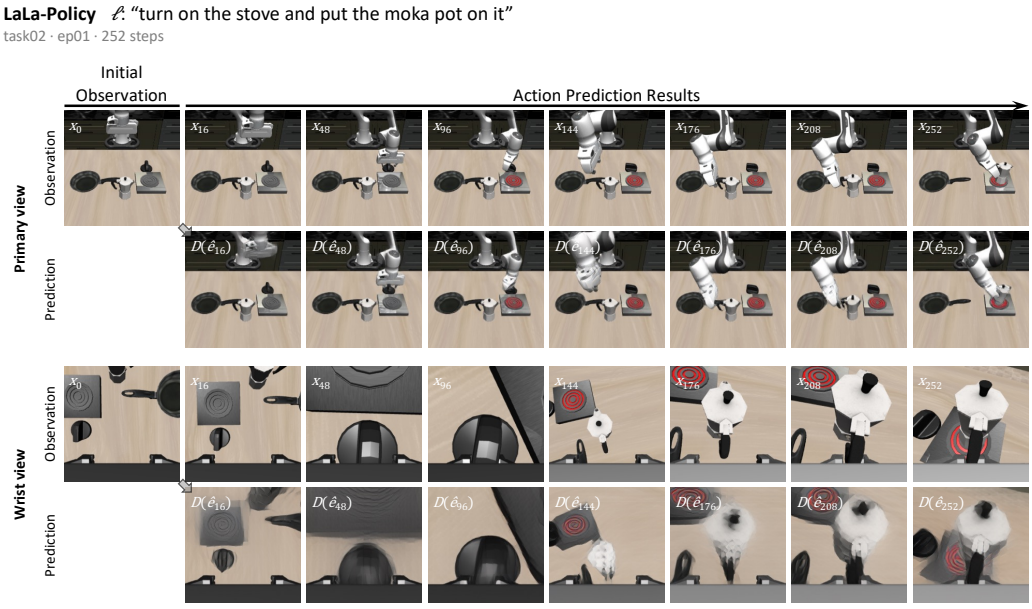

Figure AG: **LIBERO Long: stove and moka pot.** LaLa-Policy turns on the stove and places the moka pot on it. Primary and wrist views compare simulator observations with decoded future predictions.

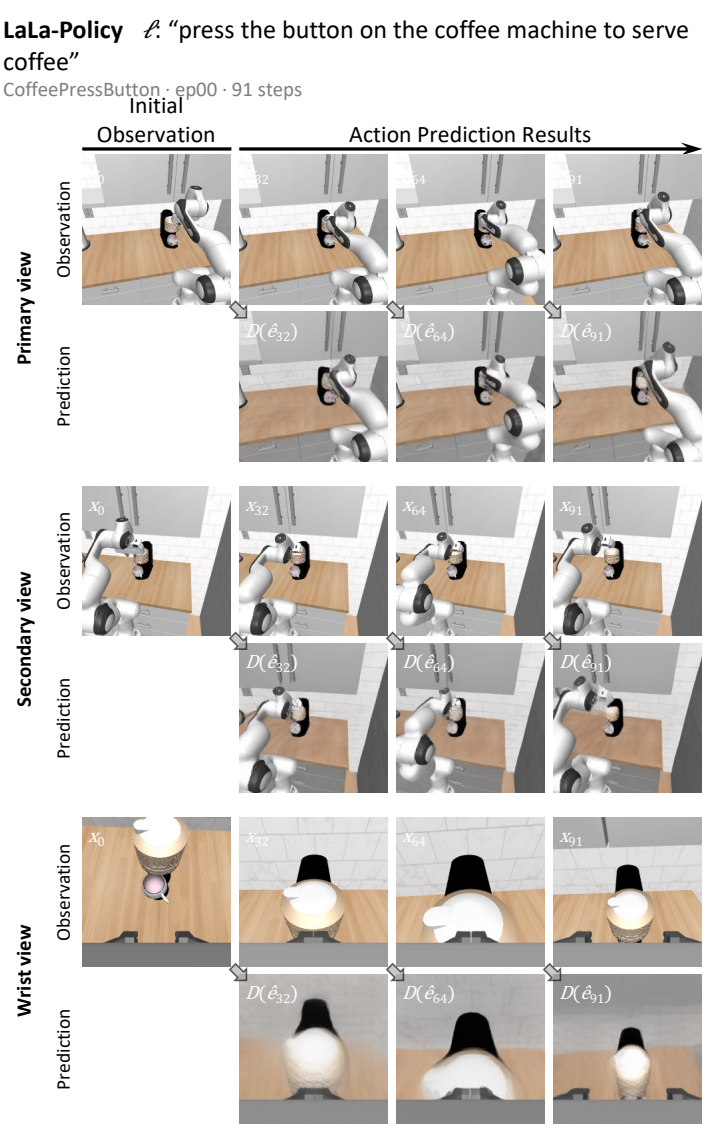

Figure AH: **RoboCasa: pressing the coffee-machine button.** Simulator observations (upper rows) and decoded future predictions (lower rows) show LaLa-Policy pressing the button, from two external views and the wrist view.



Figure AI: **RoboCasa: serving a mug.** LaLa-Policy moves the mug from beneath the coffee-machine dispenser to the counter. Each of the three views compares observed states with decoded future predictions.

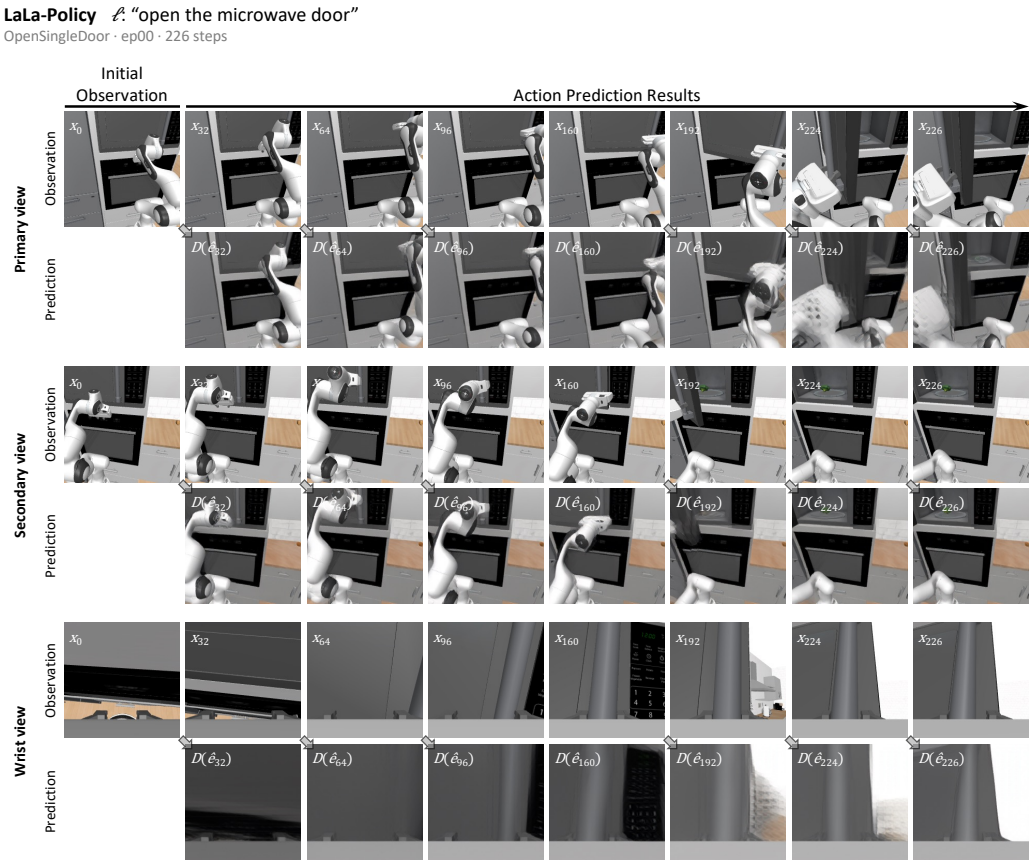

Figure AJ: **RoboCasa: opening a microwave.** Simulator observations and decoded future predictions show LaLa-Policy opening the microwave door, from the primary, secondary, and wrist views.



Figure AK: **RoboCasa: opening a drawer.** LaLa-Policy opens the left drawer. The three camera views pair simulator observations with decoded future predictions during reaching and drawer opening.



Figure AL: **RoboCasa: counter-to-cabinet transfer.** LaLa-Policy picks up the bar from the counter and places it in the cabinet. Each camera view compares observed states with decoded future predictions.

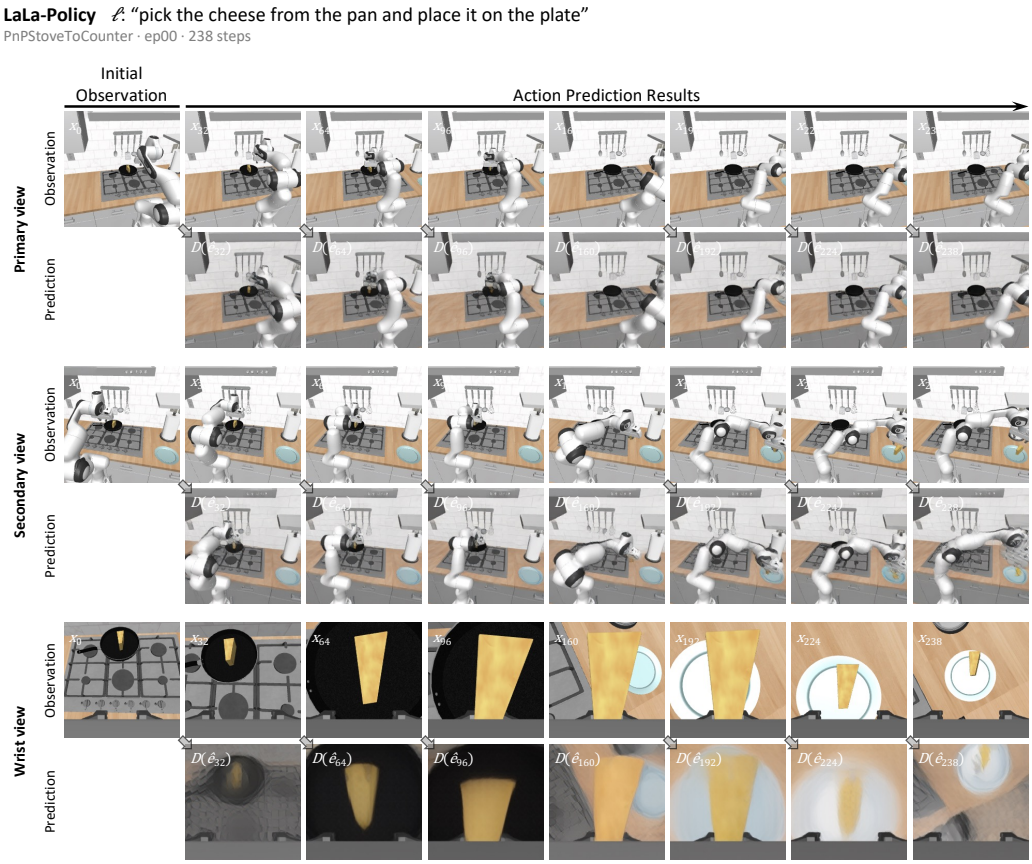

Figure AM: **RoboCasa: pan-to-plate transfer.** LaLa-Policy moves the cheese from the pan onto the plate. Observations and decoded future predictions are shown for two external views and the wrist view.

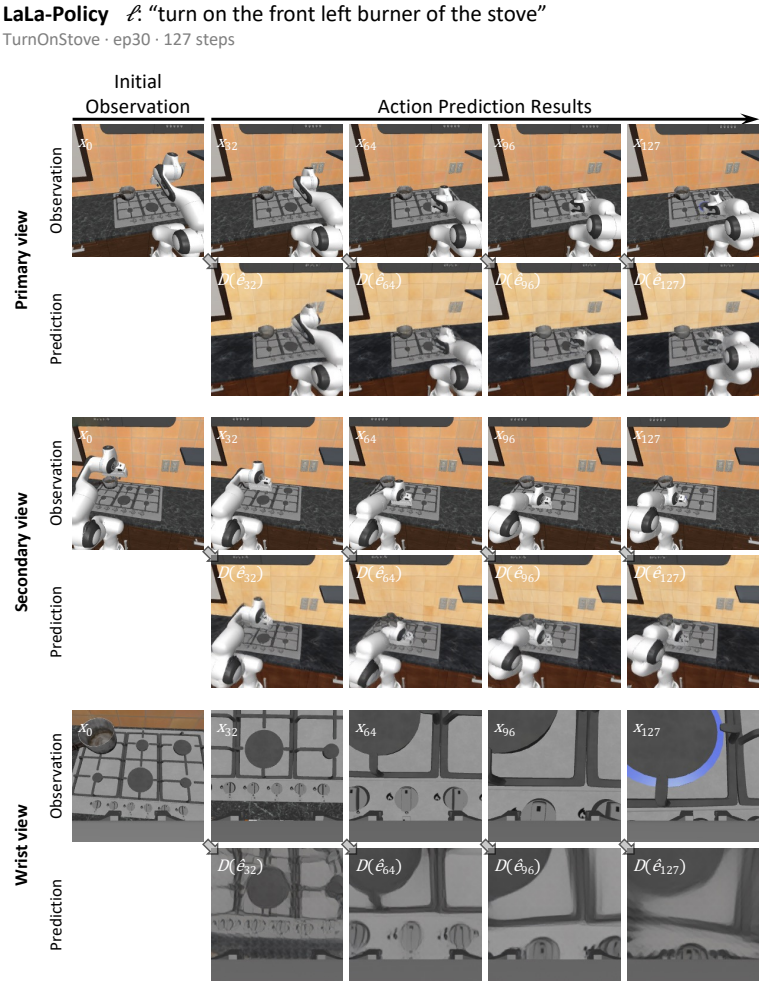

Figure AN: **RoboCasa: turning on a stove burner.** Simulator observations and decoded future predictions show LaLa-Policy turning on the front-left burner, from the primary, secondary, and wrist views.