FUture MAtching: Improving Your Vision-Language-Action Models via Differentiable Latent World Model

Dohyun Lee Kinam Kim Jaegul Choo Minho Park
KAIST AI

Abstract

*Recent vision-language-action (VLA) models commonly use flow matching for continuous action generation, but its squared-error objective does not distinguish action errors with different physical consequences. We introduce **FU**ture **MA**tching (**FUMA**), an auxiliary objective that supervises actions through predicted future outcomes. Given the same observation, a separately trained, frozen latent world model predicts future features under the policy's one-step clean-action estimate and the demonstration action. FUMA matches these predictions, propagating gradients through the world model's action input to the policy. Locally, it penalizes deviations more strongly when they cause larger changes in predicted future features. FUMA complements each backbone's original objective without changing policy architecture or inference. Across four pretrained VLA backbones, FUMA improves average success rates on RoboCasa and LIBERO-Plus by 5.6 and 5.3 percentage points, respectively, and outperforms compared auxiliary objectives under the same fine-tuning protocol. Directional analyses show reduced errors along future-sensitive directions, which perturbation experiments associate with greater effects on grasp success.*

1. Introduction

Recent vision-language-action (VLA) models [1–3] commonly adopt flow matching [4] for continuous action generation. Conditioned on the current observation and language instruction, these models learn a velocity field that transports noise toward demonstration actions, providing a flexible way to capture multimodal action distributions. The standard flow-matching objective trains this field by minimizing the squared error between predicted velocities and demonstration-derived targets.

However, equal errors in action space can lead to substantially different outcomes in robotic manipulation. At a fixed flow time, the standard squared-error objective induces an isotropic penalty on errors in the corresponding one-step action estimate in the model's action coordinates. The physical effects of an action deviation depend on the current state and the direction of the deviation. In the Push-T [5] example shown in Fig. 1 (a), the red and blue actions have identical action-space errors, as illustrated in Fig. 1 (b), yet the red action produces a 28.1 times larger next-state error than the blue action, as shown in Fig. 1 (c). This suggests supervising action deviations according to their future consequences, in addition to their magnitude.

Future prediction provides a related source of supervision for policy learning. FLARE [6] aligns internal future-token representations with embeddings of future observations, while VLA-JEPA [7] learns latent-action representations through world modeling. These objectives incorporate future information into the representations used for action generation. We focus on a complementary form of supervision: *evaluating the control actions predicted by the policy through the futures they induce*. This requires an explicit connection from a predicted action to its future outcome, through which a discrepancy from the demonstration's outcome can guide action learning.

We introduce **FU**ture **MA**tching (**FUMA**), an auxiliary objective that aligns the predicted future outcomes of policy and demonstration actions. Directly comparing their outcomes in the environment would require repeated rollouts and would generally not provide action gradients for backpropagation. We therefore use a separately trained, differentiable latent world model that remains frozen during policy training. Given the same observation, the model predicts future features under a one-step clean-action estimate computed from the policy's predicted velocity and under the demonstration action. We minimize the discrepancy between these predictions, propagating gradients through the world model's action input to the policy. Using the demonstration-conditioned prediction as the reference makes the auxiliary loss vanish when the two actions coincide, even with an imperfect world model.

Locally, Future Matching induces an observation- and action-dependent geometry over action errors through the world model's sensitivity to its action input. Deviations along directions that cause larger

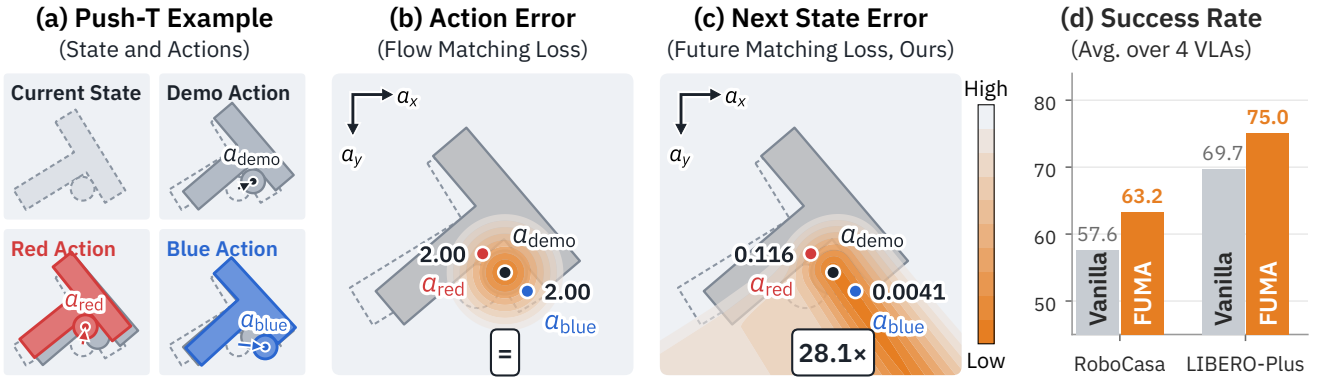


Figure 1. **Equal action errors, unequal consequences.** a_{red} and a_{blue} lie 20 px from a_{demo} and share one flow-matching loss (b), yet only a_{red} changes the T's outcome, a 28.1× larger next-state error (c). (d) FUMA raises four VLAs' average success on both benchmarks.

changes in predicted future features receive larger penalties. We add this objective to each backbone's original training losses, retaining direct supervision from demonstration actions while incorporating supervision from their predicted consequences. The world model is used only during training, leaving the policy architecture and inference procedure unchanged.

We evaluate Future Matching on four pretrained VLA backbones, including $\pi_{0.5}$ [2], MolmoAct2 [8], X-VLA [9], and GR00T N1.6 [3], on RoboCasa [10] and LIBERO-Plus [11]. FUMA improves success rates for every backbone on both benchmarks, with average gains of 5.6 and 5.3 percentage points, respectively, over their original training objectives. It also outperforms the auxiliary objectives compared under the same fine-tuning protocol. Our analysis shows that FUMA reduces action errors along directions with greater predicted future sensitivity, and perturbation experiments show that deviations along these directions more strongly affect grasp success.

2. Related Work

Action Learning Objectives for VLA Models Vision-language-action models adapt pretrained visual and language representations for robot control [12, 13]. Generative policies model continuous action distributions using diffusion [5] or flow matching [1, 2, 14]. Action supervision has also evolved through compressed tokenization for autoregressive prediction [15] and hybrid objectives that combine discrete-action prediction with continuous flow matching [2, 8, 16]. Other approaches modify the relative importance of prediction errors. AttenA+ [17] derives per-timestep loss weights from demonstrated motion speed, using slower motion as a proxy for greater precision demands. FUMA adds an observation- and action-dependent penalty based on changes in predicted future features caused by deviations from the demonstration action.

Future Prediction for VLA Learning Future prediction has been integrated into policy learning through joint image or video prediction and action generation [18–20] and visual-subgoal generation for subsequent action prediction [21]. Recent work also shapes policy representations through future-feature prediction or world modeling with latent actions [6, 7, 22]. FUMA evaluates the future induced by the policy's explicit control-action estimate. The resulting discrepancy is backpropagated through the action input of a separately trained, frozen world model, directly supervising the action estimate while preserving the policy's action-generation architecture.

World Models for Action Supervision and Control Predictive models have been used to learn scores for candidate actions from the similarity between predicted and demonstrated future observations [23]. More broadly, learned dynamics support model-based control through predictions in image, state, or latent spaces [24–27]. V-JEPA 2-AC [28] searches for actions whose predicted future features approach a supplied goal representation. World models also enable reward-driven policy improvement through imagined trajectories in both online [29, 30] and offline reinforcement learning [31–33], while differentiable dynamics enable imitation learning through adversarial objectives and trajectory-matching losses [34, 35].

Unlike model-based RL, FUMA does not optimize rewards or value functions through imagined rollouts. Instead, it compares future features predicted under policy and demonstration actions by the same frozen latent world model, using the demonstration-

conditioned prediction as reference. This discrepancy directly augments the original imitation-learning objective, requiring neither additional environment interactions nor reward-based policy optimization. The world model is used only during training, leaving policy inference unchanged.

3. Method

Future Matching (FUMA) augments native policy training with supervision from the predicted consequences of actions. Fig. 2 compares FUMA with existing supervision.

3.1. Preliminaries

Notation. Let $\mathcal{D}$ denote a dataset of demonstration trajectories, with each training example containing a visual observation o, proprioceptive state s, language instruction ℓ, and demonstrated action chunk a. We represent a chunk of H actions, each with d_a coordinates, as a flattened vector $a \in \mathbb{R}^d$, where $d = Hd_a$. We denote the observed future at the prediction horizon by o' (possibly containing multiple observations) and the policy parameters by θ. Continuous flow time t is distinct from the robot's execution steps.

Native objectives and flow matching. We retain each backbone's native prediction parameterization and fine-tuning loss, denoted by $\mathcal{L}_{\text{Vanilla}}$, and refer to training with this objective alone as *Vanilla*. All losses are defined per training sample and averaged over demonstrations, sampled noise, and flow times. For velocity-based conditional flow matching [4], we use a noise-to-data convention, where $t = 0$ denotes noise and $t = 1$ a clean action. Given $\epsilon \sim \mathcal{N}(0, I)$ and $t \in [0, 1)$, the training interpolation is

$$a_t = (1 - t)\epsilon + ta. \tag{1}$$

The standard unweighted velocity-matching objective is

$$\mathcal{L}_{\text{FM}} = \|v_\theta(a_t, t; o, s, \ell) - (a - \epsilon)\|_2^2. \tag{2}$$

Velocity-based policies integrate the learned field from noise at inference, while each backbone retains its native sampling convention.

3.2. Future Matching

Frozen latent world model. As shown in Fig. 2(c,d), the world model consists of an observation encoder E producing $z = E(o)$ and an action-conditioned predictor $P_\phi(z, s, a) \in \mathbb{R}^M$ that predicts future features, where M is the number of scalar feature elements compared by the loss. We denote the complete mapping by $f_\phi(o, s, a) := P_\phi(E(o), s, a)$, including any required action-coordinate conversion and feature processing. We adapt V-JEPA 2-AC [28] using each benchmark's training demonstrations before policy fine-tuning, then freeze both E and P_ϕ.

Training-time action estimates. For velocity-prediction policies, we construct a one-step clean-action estimate at the sampled training time:

$$\hat{a}_\theta = a_t + (1 - t)v_\theta(a_t, t; o, s, \ell). \tag{3}$$

Policies that directly predict clean actions instead use their native estimates, allowing FUMA to construct its auxiliary loss without unrolling the inference sampler.

Demonstration-conditioned reference. As shown in Fig. 2(d), we evaluate the same frozen predictor under the estimated and demonstrated actions while sharing the current representation z and state s:

$$\begin{aligned} z'^{\pi} &= P_\phi(z, s, \hat{a}_\theta), \\ z'^{\text{ref}} &= P_\phi(z, s, a). \end{aligned} \tag{4}$$

We stop gradients through the reference and define the main GT Action objective (FUMA unless otherwise specified) as

$$\mathcal{L}_{\text{FUMA}} = \frac{1}{M} \left\| z'^{\pi} - \text{sg}[z'^{\text{ref}}] \right\|_2^2, \tag{5}$$

where sg denotes stop-gradient. Because both predictions use the same world model and current context, the loss and its action gradient vanish when $\hat{a}_\theta = a$.

Observed-future reference ablation. The GT Latent variant in Fig. 2(c) instead uses the encoded observed future $z' = E(o')$ as its reference:

$$\mathcal{L}_{\text{GT Latent}} = \frac{1}{M} \left\| z'^{\pi} - \text{sg}[z'] \right\|_2^2. \tag{6}$$

Even when $\hat{a}_\theta = a$, world-model prediction errors can produce a nonzero residual relative to $E(o')$ and drive policy updates, whereas GT Action eliminates this residual at the demonstrated action. Both variants still depend on the model's response to action changes, and we compare them empirically in Section 4.4.

Policy optimization. We augment each backbone's native objective with FUMA:

$$\mathcal{L} = \mathcal{L}_{\text{Vanilla}} + \lambda \mathcal{L}_{\text{FUMA}}, \tag{7}$$

where $\lambda \geq 0$ controls the auxiliary loss weight. The native objective retains direct demonstration supervision,

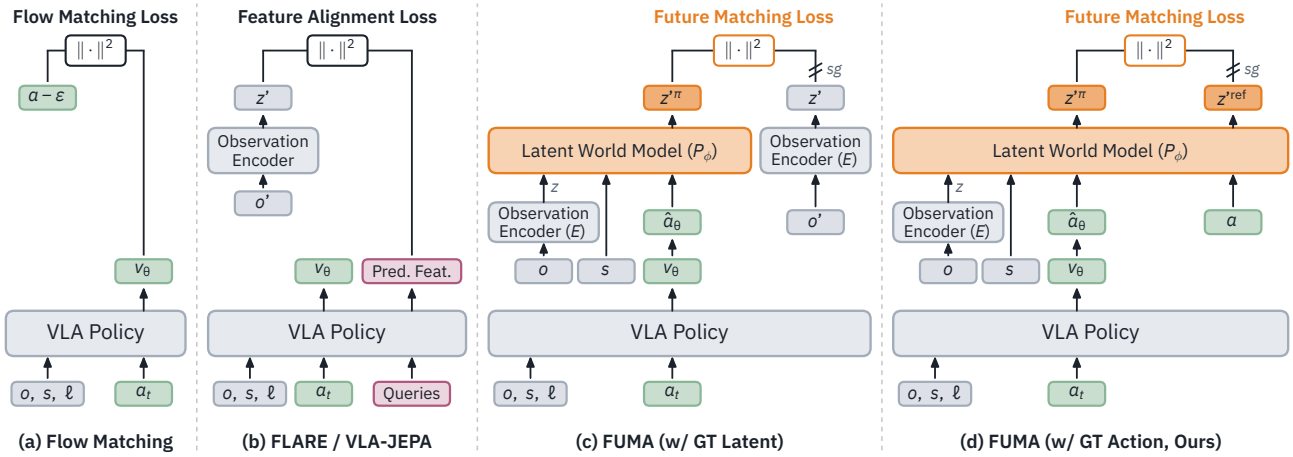


Figure 2. **Future Matching.** (a) Flow matching compares the predicted velocity with $a - \epsilon$. (b) Future-representation supervision compares policy-predicted features with encoded future observations [6, 7]. (c) The GT Latent ablation compares z'^{π} with the encoded future $z' = E(o')$. (d) FUMA (GT Action, ours) compares z'^{π} with the prediction z'^{ref} under the demonstration action. sg denotes stop-gradient.

while FUMA gradients propagate through the frozen predictor's action input and $\hat{a}_\theta$ to θ. Both future-matching variants retain the native objective, although its branches are omitted in panels (c,d) for clarity. Because the world model is used only during training, FUMA can be added to existing native objectives without modifying the policy architecture or inference procedure.

3.3. Context-Dependent Geometry of Action Errors

We analyze the main GT Action objective in Fig. 2(d) to understand how FUMA prioritizes different directions of action error. We show that, unlike standard flow matching, FUMA induces a context-dependent geometry that places greater emphasis on action directions to which the predicted future is most sensitive.

Standard flow-matching penalty. Let $\delta = \hat{a}_\theta - a$ denote the error of the one-step action estimate. Using Equations (1) and (3), the standard unweighted velocity-matching loss in (2) can be rewritten as

$$\mathcal{L}_{\text{FM}} = \|v_\theta - (a - \epsilon)\|_2^2 = \frac{\|\delta\|_2^2}{(1-t)^2}, \tag{8}$$

where we suppress the arguments of v_θ for clarity. At a fixed flow time, this objective penalizes errors of equal magnitude equally, regardless of their directions. Consequently, action errors with substantially different effects on the predicted future receive the same penalty.

Future-dependent action geometry. In contrast, FUMA measures action errors through their effects on predicted future features. For fixed (o, s, a) and a small deviation δ, a first-order expansion of the frozen world model gives $f_\phi(o, s, a+\delta) - f_\phi(o, s, a) \approx J_{o,s,a}\delta$, where $J_{o,s,a} = \partial f_\phi(o, s, a)/\partial a \in \mathbb{R}^{M\times d}$ is evaluated at the demonstrated action chunk. The local FUMA objective therefore becomes

$$\begin{aligned} \mathcal{L}_{\text{FUMA}} &\approx \frac{1}{M}\|J_{o,s,a}\delta\|_2^2 = \delta^\top G_{o,s,a}\delta, \\ G_{o,s,a} &= \frac{1}{M}J_{o,s,a}^\top J_{o,s,a} \in \mathbb{R}^{d\times d}. \end{aligned} \tag{9}$$

The positive-semidefinite matrix $G_{o,s,a}$ defines a local geometry over action errors, with penalties determined by their effects on predicted future features. Crucially, this geometry depends on the observation, state, and demonstrated action, allowing the same action perturbation to receive different penalties across contexts.

Identifying important action directions. To understand which directions FUMA emphasizes, consider the eigendecomposition of $G_{o,s,a}$. Let (γ_k, q_k) denote its eigenvalue–eigenvector pairs, ordered such that $\gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_d \geq 0$, with orthonormal eigenvectors q_k. The local FUMA penalty decomposes as

$$\delta^\top G_{o,s,a}\delta = \sum_{k=1}^{d} \gamma_k (q_k^\top \delta)^2. \tag{10}$$

Each eigenvector q_k represents a direction over the full action chunk, while its eigenvalue γ_k quantifies the sensitivity of predicted future features to perturbations along that direction. In particular, the principal eigenvector q_1, corresponding to the largest eigenvalue γ_1,

identifies the action direction to which the predicted future is most sensitive. *FUMA therefore assigns its strongest local penalty to errors along this direction, while weighting other directions according to their respective sensitivities.* Because these directions depend on (o, s, a), FUMA adaptively emphasizes different action errors depending on the current context. These sensitivities characterize the learned world model; Section 4.3 examines how they relate to physical execution.

Complementarity with native supervision. The directional interpretation of FUMA is local, and the frozen world model may provide less reliable supervision for actions far from the demonstration distribution. The native objective complements FUMA by directly penalizing action deviations, while FUMA places greater emphasis on future-sensitive directions. This directional supervision emerges from the nonlinear feature-matching loss without explicitly computing the Jacobian or geometry matrix.

4. Experiments

We evaluate whether FUMA improves VLA policy performance across diverse settings and investigate the roles of its geometric supervision and training design. We first describe the setup, then address the following questions:

RQ1: Does FUMA consistently improve performance across VLA backbones, simulation benchmarks, and real-world tasks? (Section 4.2)

RQ2: Does the geometry matrix guide learning toward task-critical action directions? (Section 4.3)

RQ3: How do the geometry matrix and supervision target affect performance? (Section 4.4)

RQ4: How do the Vanilla loss and FUMA weight λ affect training and performance? (Section 4.5)

4.1. Experimental Setup

Tasks and Datasets. We evaluate FUMA on RoboCasa [10] and LIBERO-Plus [11]. For RoboCasa, we train on official human demonstrations covering 24 atomic kitchen-manipulation tasks and evaluate 50 rollouts per task, totaling 1,200 trials. For LIBERO-Plus, we train on demonstrations from the 40 original LIBERO [36] tasks across the Spatial, Object, Goal, and Long suites. Evaluation uses 10,030 perturbed task instances, with one rollout per instance. We additionally evaluate four real-world manipulation tasks (Fig. 3): placing a banana in a bowl, stacking cubes, building a bridge, and peg insertion. We collect 20 teleoperated demonstrations per task and evaluate each policy with 10 rollouts per task. We report success rates across all benchmarks and real-world tasks.

Baselines. We compare FUMA with FLARE [6] and VLA-JEPA [7], as both use future-state information to supervise policy learning. We also include AttenA+ [17], which weights supervision using a prior based on demonstration action velocities, as another approach that assigns unequal importance to action errors. To evaluate the applicability of these methods across VLA backbones, we experiment with four pretrained models: X-VLA [9], MolmoAct2 [8], $\pi_{0.5}$ [2], and GR00T N1.6 [3]. For each backbone, Vanilla uses its original training objective. We implement the additional objectives as originally formulated.

Implementation Details. For each backbone and benchmark, we first perform a 1k-step warm-up using the native objective to obtain a common starting checkpoint. All methods then train on the same data for 10k steps on RoboCasa [10] or 20k steps on LIBERO-Plus [11], while real-world experiments follow the RoboCasa training budget. We follow each backbone's official fine-tuning recipe and use $\lambda = 1$ for FUMA unless otherwise specified. For the latent world model, we fine-tune V-JEPA 2-AC (ViT-g/16) [28] on benchmark demonstrations for 20k steps and freeze it during policy training. For the RGB world-model comparison in Section 4.4, we instead fine-tune Wan2.2-TI2V-5B [37] with the same training budget. Additional hyperparameters and implementation details are provided in the supplementary material.

4.2. Main Results

Simulation Experiments As shown in Tab. 1, FUMA achieves the highest success rate for all four VLA backbones on both benchmarks. Average success rates improve over the original objectives by 5.6 points on RoboCasa and 5.3 points on LIBERO-Plus. FUMA supervises generated actions through their predicted future consequences using a frozen world model, requiring no architectural changes or additional trainable policy modules. Its gains on MolmoAct2 [8] also show compatibility with objectives combining flow matching and autoregressive action supervision.

However, the baselines do not show consistent improvements across backbones and benchmarks. Their average gains reach at most 1.2 points on RoboCasa, and none improves the average LIBERO-Plus score. FLARE and VLA-JEPA [6, 7] require architectural changes and additional learnable tokens. Optimizing these new token parameters alongside the pretrained policy adds an adaptation burden that may limit their gains under our training setup. AttenA+ [17]

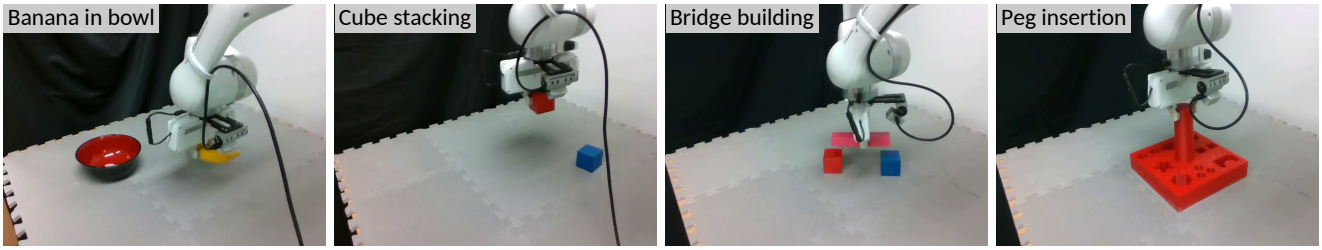


Figure 3. **Real-world tasks.** Representative third-person camera frames from the teleoperated demonstrations of the four tasks.

Table 1. **Main results.** Success rates (%) of four VLAs on RoboCasa (top) and LIBERO-Plus (bottom). Per-backbone results report mean ± std over three seeds. Vanilla: original objective only. Numbers in parentheses indicate differences in mean success rate relative to Vanilla.

Method	X-VLA [9]	MolmoAct2 [8]	$\pi_{0.5}$ [2]	GR00T N1.6 [3]	Average
RoboCasa [10]					
Vanilla	40.8±1.7	61.0±1.4	60.0±1.7	68.7±0.8	57.6±1.4
FLARE [6]	39.0±1.1 (−1.8)	58.6±1.1 (−2.4)	61.7±1.4 (+1.7)	70.2±1.2 (+1.5)	57.4±1.2 (−0.2)
VLA-JEPA [7]	43.3±0.6 (+2.5)	57.1±0.7 (−3.9)	65.0±1.3 (+5.0)	69.8±0.7 (+1.1)	58.8±0.8 (+1.2)
AttenA+ [17]	44.5±1.8 (+3.7)	58.6±1.3 (−2.4)	62.0±0.7 (+2.0)	68.6±2.1 (−0.1)	58.4±1.5 (+0.8)
FUMA (Ours)	**47.7**±0.7 **(+6.9)**	**64.5**±0.6 **(+3.5)**	**68.4**±1.4 **(+8.4)**	**72.1**±1.5 **(+3.4)**	**63.2**±1.1 **(+5.6)**
LIBERO-Plus [11]					
Vanilla	52.1±0.3	79.8±0.6	75.7±0.2	71.4±0.5	69.7±0.4
FLARE [6]	49.2±0.8 (−2.9)	80.6±0.2 (+0.8)	75.1±0.6 (−0.6)	70.7±0.1 (−0.7)	68.9±0.4 (−0.8)
VLA-JEPA [7]	51.3±0.9 (−0.8)	80.7±0.5 (+0.9)	75.6±0.5 (−0.1)	71.4±0.7 (0.0)	69.7±0.7 (0.0)
AttenA+ [17]	52.5±0.8 (+0.4)	80.6±0.3 (+0.8)	74.1±0.7 (−1.6)	70.1±0.8 (−1.3)	69.3±0.7 (−0.4)
FUMA (Ours)	**62.7**±0.7 **(+10.6)**	**82.9**±0.5 **(+3.1)**	**80.6**±0.6 **(+4.9)**	**73.9**±0.7 **(+2.5)**	**75.0**±0.6 **(+5.3)**

Table 2. **Real-world results.** Successful rollouts out of 10 on each real-world task (Fig. 3). Avg. reports the mean success rate (%).

Method	Easy	Medium		Hard	Avg. (%)
	Banana	Cube	Bridge	Peg	
$\pi_{0.5}$ [2]					
Vanilla	7/10	3/10	2/10	0/10	30.0
FLARE [6]	7/10	4/10	3/10	0/10	35.0
VLA-JEPA [7]	8/10	3/10	4/10	2/10	42.5
AttenA+ [17]	7/10	3/10	3/10	2/10	37.5
FUMA (Ours)	9/10	8/10	8/10	4/10	72.5
GR00T N1.6 [3]					
Vanilla	7/10	2/10	2/10	0/10	27.5
FLARE [6]	6/10	3/10	3/10	0/10	30.0
VLA-JEPA [7]	7/10	4/10	4/10	1/10	40.0
AttenA+ [17]	7/10	3/10	3/10	1/10	35.0
FUMA (Ours)	8/10	7/10	6/10	3/10	60.0

avoids additional trainable parameters but assigns temporal importance using demonstration motion speed. These results suggest that the velocity-based importance prior is not consistently effective across the evaluated backbones and benchmarks.

Real-world Experiments Most methods perform well on placing a banana in a bowl (Tab. 2). Cube stacking and bridge building require precise object placement, where small action differences can lead to substantially different outcomes. FUMA outperforms the comparison methods on both tasks. FUMA also achieves the highest success rate on peg insertion, although its gain over the strongest comparison method is modest, which may reflect limitations of the learned world model in capturing subtle changes in alignment and contact that determine insertion success.

4.3. Geometry Matrix Analysis

Objectives. In this section, we analyze the geometry matrix G through its eigenvectors q_k and corresponding eigenvalues γ_k in (10). Each q_k defines an action direction, and a larger γ_k indicates greater sensitivity of the predicted future features to changes along that direction. We examine whether FUMA reduces errors more strongly along directions with larger eigenvalues and whether these directions matter for task success. We conduct these analyses using $\pi_{0.5}$ policies on RoboCasa. Each action chunk consists of 32 seven-dimensional actions. Flattening the chunk yields a 224-dimensional vector, so $G \in \mathbb{R}^{224\times224}$.

Where Does FUMA Reduce Action Errors? We compare Vanilla and FUMA over 3,000 training states from 24 tasks. At each state, we generate action chunks from both policies using the same eight noise seeds. We measure action errors along the eigenvectors of $G_{o,s,a}$, evaluated at the demonstration action of each state: the squared projections $(q_k^\top \delta)^2$, with $\delta = \hat{a}_\theta - a$, are averaged over the sampled chunks, then over the directions within each rank interval, and finally over states. Fig. 4(a) shows that FUMA achieves lower errors than Vanilla along eigenvectors with larger eigenvalues, but higher errors along those with smaller eigenvalues. Despite these larger errors in less sensitive directions, FUMA improves the RoboCasa success rate over Vanilla by 8.4 points (Tab. 1). These results suggest that prioritizing optimization along directions with larger eigenvalues can improve task performance.

Do These Directions Matter for Task Success? We perturb demonstration actions along selected eigenvectors at 125 pre-grasp states from eight pick-and-place tasks and execute the perturbed action chunks in the simulator. The nominal ($\times 1$) perturbation magnitude is the Vanilla policy's median action-error magnitude at each state, applied in both signs of q_k before action clipping. We measure grasp success, which requires two-finger contact with the target object within the chunk, and the final end-effector position error relative to the demonstration execution. Fig. 4(b) shows that perturbations along eigenvectors with larger eigenvalues produce lower grasp success and larger position errors at the same nominal perturbation scale. Directions with smaller eigenvalues have less impact across the tested scales. Fig. 4(c) illustrates this difference: a perturbation along q_1 causes the gripper to miss the tomato, whereas a perturbation with eight times the nominal magnitude along q_{210}, a low-sensitivity direction with a small gripper component, still allows a successful grasp. These results indicate that the sensitivities captured by G are associated with meaningful differences in physical execution.

4.4. Geometry Matrix Ablation

Setup. We vary the geometry matrix G in (9) to examine how its structure and the world model's representation affect policy performance. Tab. 3 reports results on RoboCasa for the four VLA backbones, with each variant retaining the backbone's original training objective. The identity variant ($G{=}I$) uses the MSE of the one-step action estimate, corresponding to the identity quadratic form in (8). The fixed-average variant ($G{=}\bar{G}$) averages the geometry matrices computed at the demonstration action chunks of 3,000 training states (125 per task). The resulting matrix remains fixed and is shared across all training samples. The RGB variant ($G{=}G_{\mathrm{RGB}}$) computes future discrepancies using an RGB world model [37] instead of a latent world model [28]. FUMA (w/ GT Latent) targets features encoded from observed future frames (Fig. 2(c)). FUMA (w/ GT Action, Ours) instead targets the same world model's future predictions under demonstration actions (Fig. 2(d)).

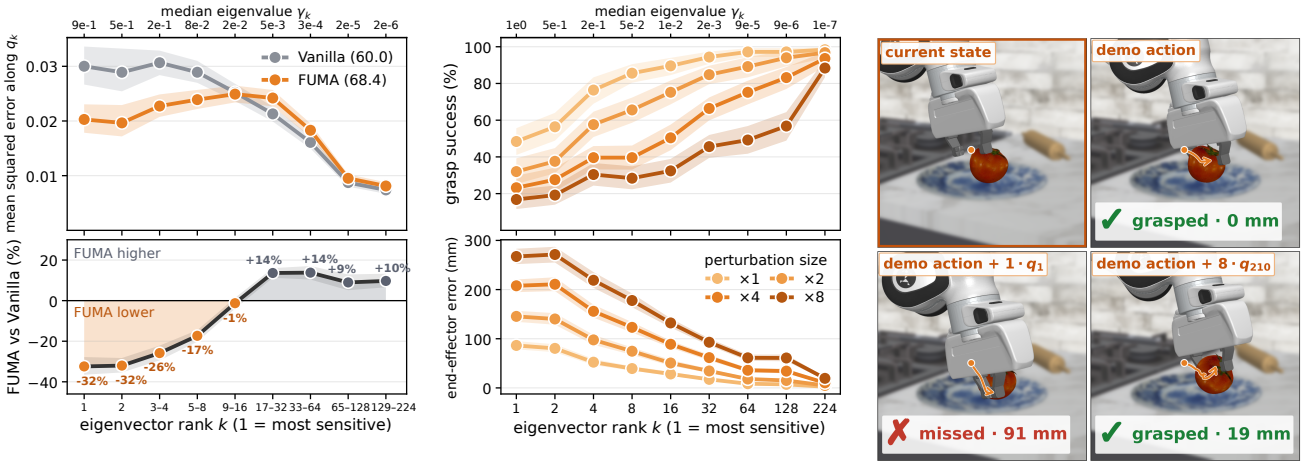
Identity and Fixed-Average Geometry Matrices. The identity variant remains close to Vanilla, with a 0.3-point decrease in average success rate. Like flow matching, it treats all action directions equally and provides limited additional supervision beyond the original objective. In contrast, the fixed-average matrix improves the average success rate by 3.4 points over Vanilla, with gains across all four backbones. Thus, even a geometry matrix shared across samples provides useful supervision. The matrix is computed once with the world model before policy training and kept fixed, so policy training requires no world-model forward or backward passes. The fixed-average matrix can therefore serve as a practical alternative.

RGB versus Latent World Models. The RGB world model improves over Vanilla but underperforms our latent world model. One possible explanation is that pixel-space discrepancies emphasize surface appearance, providing a less focused learning signal for meaningful state changes. These results support using latent features for future supervision in our setting.

GT Latent versus GT Action. FUMA with GT Action outperforms GT Latent. A learned world model can produce features that differ from the encoded observed future even when conditioned on the demonstration action. Consequently, the GT Latent variant can incur a nonzero future loss even when the policy predicts the demonstration action exactly. The GT Action variant instead compares both actions through the same world model, so the future loss vanishes when the actions coincide. This avoids penalizing the policy for world-model prediction error.

4.5. Training Objective Analysis

Role of the Vanilla Objective. Tab. 3 shows that FUMA alone achieves performance comparable to Vanilla, with average success rates of 58.6 and 57.6, respectively. Combining the two objectives outperforms either objective alone across all four backbones for every evaluated value of λ. These results support retaining the Vanilla objective when applying FUMA. One possible explanation is that FUMA relies on a learned latent world model, whose prediction errors can affect



(a) Action error by sensitivity rank (b) Grasp outcome under perturbation along q_k (c) Example pre-grasp state

Figure 4. **Geometry matrix analysis** ($\pi_{0.5}$, RoboCasa). Eigenvectors q_k of G are ordered by decreasing eigenvalue γ_k. (a) Mean squared action error along each q_k (top) and its change from Vanilla to FUMA (bottom): FUMA lowers the error along high-eigenvalue directions. (b) Perturbing demonstration actions along high-eigenvalue directions lowers grasp success and raises end-effector (EEF) error the most. (c) Executions perturbed along q_1 (×1) and q_{210} (×8); labels: final EEF error (mm). Bands: 95% bootstrap intervals.

Table 3. **Ablation studies.** Success rates (%) of four VLAs on RoboCasa. Subscripts denote the change relative to the base model. Gray entries are fixed to our default setting. (a) Variants of $\boldsymbol{G}$. (b) Training objectives.

Method	λ	X-VLA [9]	MolmoAct2 [8]	$\pi_{0.5}$ [2]	GR00T N1.6 [3]	Average
Vanilla	–	40.8	61.0	60.0	68.7	57.6
(*a*) *Variants of G*						
Identity matrix	1.0	40.8 (0.0)	59.3 (−1.7)	60.5 (+0.5)	68.9 (+0.2)	57.3 (−0.3)
Fixed average matrix	1.0	45.5 (+4.7)	64.1 (+3.1)	64.5 (+4.5)	70.0 (+1.3)	61.0 (+3.4)
RGB world model	1.0	44.2 (+3.4)	62.8 (+1.8)	64.0 (+4.0)	71.2 (+2.5)	60.5 (+2.9)
FUMA w/ GT latent	1.0	45.8 (+5.0)	62.3 (+1.3)	66.0 (+6.0)	71.2 (+2.5)	61.3 (+3.7)
(*b*) *Training objectives*						
$\mathcal{L}_{\text{FUMA}}$ only	–	43.9 (+3.1)	58.5 (−2.5)	64.3 (+4.3)	67.9 (−0.8)	58.6 (+1.0)
$\mathcal{L}_{\text{Vanilla}} + \lambda\mathcal{L}_{\text{FUMA}}$	0.5	47.2 (+6.4)	61.4 (+0.4)	66.6 (+6.6)	69.5 (+0.8)	61.1 (+3.5)
	2.0	47.5 (+6.7)	64.8 (+3.8)	66.5 (+6.5)	71.9 (+3.2)	62.6 (+5.0)
	4.0	46.9 (+6.1)	**66.1** (+5.1)	67.3 (+7.3)	72.0 (+3.3)	63.0 (+5.4)
FUMA w/ GT action (Ours)	1.0	**48.3** (+7.5)	64.5 (+3.5)	**68.4** (+8.4)	**72.1** (+3.4)	**63.3** (+5.7)

the supervision it provides. This limitation may be more pronounced for policy-generated actions outside the world model's training distribution. For example, early in training, the policy may generate actions far from demonstration actions, where the world model's less reliable supervision may slow policy improvement. This may explain why FUMA alone improves more slowly at first in Fig. 5, although it eventually surpasses the Vanilla objective alone. The flow-matching term provides a direct anchor to demonstration actions, helping reduce reliance on potentially inaccurate world-model supervision during early training. Combining the two objectives avoids this initial delay and achieves higher final performance.

Effect of the Loss Weight. All evaluated values of $\lambda \in \{0.5, 1, 2, 4\}$ improve performance over Vanilla across all four backbones. Average success rates range from 61.1 to 63.3, with no consistent trend as λ varies. Thus, when the Vanilla objective is retained, FUMA provides gains across the tested range of loss weights without requiring precise tuning. We use $\lambda = 1$, which achieves the highest average success rate, as the default setting.

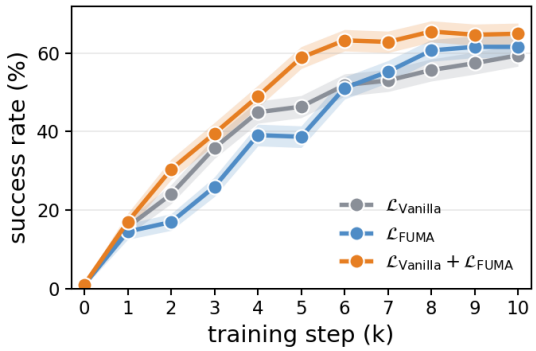


Figure 5. **Success rate over training** ($\pi_{0.5}$, RoboCasa). Policies are trained from the pretrained checkpoint for 10k steps with the Vanilla objective only, FUMA only, or both, and each 1k-step snapshot is evaluated on 24 tasks × 50 episodes. Shaded bands indicate binomial 95% intervals of a single 1,200-episode.

5. Conclusion

We introduced Future Matching (FUMA), an auxiliary objective that supervises VLA policies through the predicted future consequences of their actions. Using a frozen differentiable latent world model, FUMA provides context-dependent supervision that places greater emphasis on action errors along future-sensitive directions. Across four VLA backbones, FUMA consistently improves performance on RoboCasa and LIBERO-Plus, with average success-rate gains of 5.6 and 5.3 percentage points, respectively, and further demonstrates improvements on real-world manipulation tasks. Our geometric analysis shows that FUMA reduces errors along directions that have greater effects on physical execution. These results highlight the value of complementing conventional action supervision with supervision from predicted future outcomes.

Limitations. FUMA introduces additional training cost due to the separately trained world model and backpropagation through its action input. Although we use latent future representations and one-step action estimates to reduce computation, policy training on $\pi_{0.5}$ still takes approximately 3.3× longer than Vanilla. Nevertheless, FUMA requires no additional demonstrations or changes to the policy architecture and inference procedure. Our primary objective is to improve imitation learning under the same demonstration data and environment setup, rather than to minimize the computational cost of each training step. The results demonstrate that future-aware auxiliary supervision can improve behavioral cloning performance without requiring additional demonstration data.

LLM Usage Statement

We used ChatGPT and Claude Code to assist with manuscript drafting and revision, experimental work, figure preparation, research ideation, and the identification and review of related literature. The authors take full responsibility for the content of this paper, including all scientific claims, experimental results, and references.

References

[1] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[2] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A Vision-Language-Action Model with Open-World Generalization. In *Proceedings of the 9th Conference on Robot Learning*, pages 17–40. PMLR, 2025.

[3] NVIDIA GEAR Team. GR00T N1.6: An improved open foundation model for generalist humanoid robots. https://research.nvidia.com/labs/gear/gr00t-n1_6/, 2025.

[4] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

[5] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[6] Ruijie Zheng, Jing Wang, Scott Reed, Johan Bjorck, Yu Fang, Fengyuan Hu, Joel Jang, Kaushil Kundalia, Zongyu Lin, Loic Magne, Avnish Narayan, You Liang Tan, Guanzhi Wang, Qi Wang, Jiannan Xiang, Yinzhen Xu, Seonghyeon Ye, Jan Kautz, Furong Huang, Yuke Zhu, and Linxi Fan. FLARE: Robot learning with implicit world modeling. *arXiv preprint arXiv:2505.15659*, 2025.

[7] Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. VLA-JEPA: Enhancing vision-language-action model with latent world model. *arXiv preprint arXiv:2602.10098*, 2026.

[8] Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, Shanli Xing, Jaemin Cho, Jae Sung Park, Ainaz Eftekhar, Peter Sushko, Karen Farley, Angad Wadhwa, Cole Harrison, Winson Han, Ying-Chun Lee, Eli VanderBilt, Rose Hendrix, Suveen Ellawela, Lucas Ngoo, Joyce Chai, Zhongzheng Ren, Ali Farhadi, Dieter Fox, and Ranjay Krishna. MolmoAct2: Action reasoning models for real-world deployment. *arXiv preprint arXiv:2605.02881*, 2026.

[9] Jinliang Zheng, Jianxiong Li, Zhihao Wang, et al. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, 2026.

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

[11] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: A progressive robustness benchmark for visual-language-action models. In *CVPR*, 2026.

[12] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[13] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[14] NVIDIA. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[15] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

[16] Danny Driess, Jost Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *Advances in Neural Information Processing Systems*, 38:102867–102888, 2026.

[17] Daojie Peng, Fulong Ma, Jiahang Cao, Qiang Zhang, Xupeng Xie, Jian Guo, Ping Luo, Andrew F. Luo, Boyu Zhou, and Jun Ma. AttenA+: Rectifying action inequality in robotic foundation models. *arXiv preprint arXiv:2605.13548*, 2026.

[18] Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. *arXiv preprint arXiv:2312.13139*, 2023.

[19] Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, Hanbo Zhang, and Minzhao Zhu. GR-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.

[20] Yanjiang Guo, Yucheng Hu, Jianke Zhang, Yen-Jen Wang, Xiaoyu Chen, Chaochao Lu, and Jianyu Chen. Prediction with action: Visual policy learning via joint denoising process. *Advances in Neural Information Processing Systems*, 37:112386–112410, 2024.

[21] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Ming-Yu Liu, Donglai Xiang, Gordon Wetzstein, and Tsung-Yi Lin. CoT-VLA: Visual chain-of-thought reasoning for vision-language-action models. *arXiv preprint arXiv:2503.22020*, 2025.

[22] Wei Li, Peijin Jia, Yuan Ma, Xuefeng Jiang, Titong Jiang, Sheng Sun, Yujian Li, Xin Wen, Han Hong, Zhikang Liu, Bailin Li, and Kun Zhan. FoMoVLA: Bridging visual foresight and motion guidance for vision-language-action models. *arXiv preprint arXiv:2607.14739*, 2026.

[23] Alan Wu, AJ Piergiovanni, and Michael S Ryoo. Model-based behavioral cloning with future image similarity learning. In *Conference on Robot Learning*, pages 1062–1077. PMLR, 2020.

[24] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. *arXiv preprint arXiv:1610.00696*, 2017.

[25] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a

handful of trials using probabilistic dynamics models. *arXiv preprint arXiv:1805.12114*, 2018.

[26] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2019.

[27] Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2024.

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

[29] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

[30] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2024.

[31] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in neural information processing systems*, 33:14129–14142, 2020.

[32] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34: 28954–28967, 2021.

[33] Akshay L Chandra, Iman Nematollahi, Chenguang Huang, Tim Welschehold, Wolfram Burgard, and Abhinav Valada. DiWA: Diffusion policy adaptation with world models. *arXiv preprint arXiv:2508.03645*, 2025.

[34] Nir Baram, Oron Anschel, and Shie Mannor. Model-based adversarial imitation learning. *arXiv preprint arXiv:1612.02179*, 2016.

[35] Siwei Chen, Xiao Ma, and Zhongwen Xu. Imitation learning via differentiable physics. *arXiv preprint arXiv:2206.04873*, 2022.

[36] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

[37] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.