JEVA: JOINT ENCODERS FOR VARIATIONAL ACTION WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning world models for robotic control typically requires large amounts of action-labeled data, which is expensive to collect, whereas action-free demonstrations such as video are abundant. Latent action models address this imbalance by inferring the actions that explain observed transitions, but training the inverse dynamics model (IDM) jointly with the forward model is unstable. Existing methods therefore fix the latent action space before training the world model, or add a separate inverse-dynamics warm-up stage. We introduce JEVA, a latent-action world model that co-trains an IDM and a forward world model end-to-end from scratch in a single stage, on a pool of trajectories in which only a small fraction of transitions carry action labels. JEVA treats the action between two visual states as a continuous Gaussian latent that can be planned over and decoded into continuous controls. Variance-reduced routing (VRR) is what makes single-stage training work. On labeled transitions, where the action decoder keeps posterior forward predictor on the posterior mean instead. This removes reparameterization noise from the predictor gradient at the cost of a small bias, and it also removes a penalty that otherwise drives latent actions toward collapse: with a warm-up stage and no VRR every latent-action dimension collapses. With 5% of episodes labeled, a policy trained in JEVA's latent action space matches or exceeds every baseline trained with 20% of labels on OGBench Cube-Double and Cube-Triple. Under CEM planning, JEVA leads the strongest world model baseline by 11-22 points on all three Cube tasks.

1 INTRODUCTION

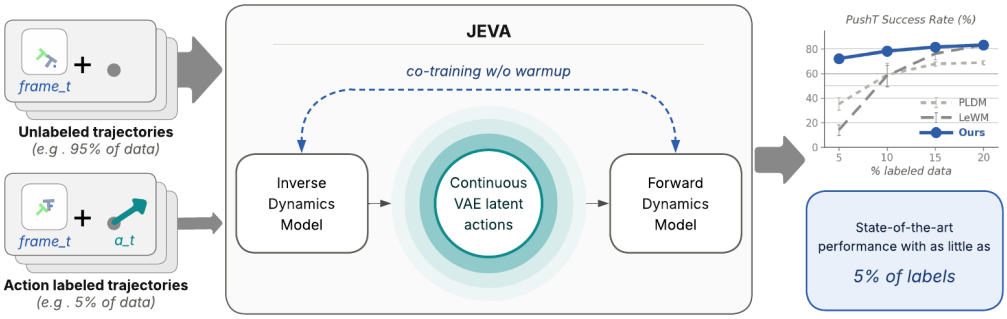

Figure 1: JEVA jointly optimizes an Inverse Dynamics Model alongside a Forward Dynamics Model using a continuous variational bottleneck. This end-to-end co-training bypasses the need for pre-training or warmup, extracting rich dynamics from abundant unlabeled video while relying on sparse labels (as little as 5%) purely to anchor the action space.

World models, which learn the dynamics of an environment so that an agent can plan by imagining the consequences of its actions, have emerged as a powerful foundation for robotic control. However, their scaling is bottlenecked by the need for expensive action-labeled data. Conversely, action-free demonstrations from videos are abundant and largely untapped (Seo et al., 2022). This asymmetry

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.

motivates a central question: *can we learn an accurate, controllable world model using mostly action-free data supplemented by sparse action labels?*

Latent action models (LAMs) offer a promising route. By treating the action connecting two observations as a latent variable, LAMs infer controls from visual transitions (Zhang et al., 2025) . This separates two learning signals that are usually entangled: an inverse dynamics model extracts *what action was taken*, while a forward model predicts *how the scene evolves* (Wang et al., 2020). In principle, the inverse model can be trained on unlabeled video, requiring only sparse labels to anchor the latent space to real controls.

In practice, two obstacles have limited this paradigm. First, jointly training an inverse and forward model from scratch is notoriously unstable: the latent action space can collapse, the forward model can shortcut predictions by leaking future information, . Prior approaches therefore rely on multi-stage pretraining or warmup schedules (Wang et al., 2026; Gao et al., 2025). Second, most methods fix the latent action space before labels are used and attach actions afterwards, so sparse labels cannot shape the space they ground.

We introduce JEVA, a latent-action world model that resolves both obstacles. JEVA formulates inverse dynamics as posterior inference in a conditional variational autoencoder, treating the action between two visual states as a continuous Gaussian latent variable. Unlike prior continuous latent-action models, which learn the latent action space before labels are used, JEVA grounds it with sparse action labels during training. This yields a fully differentiable, sampleable action space that can be optimized during planning and decoded directly into continuous robot controls (Losey et al., 2021). Crucially, JEVA co-trains its inverse and forward dynamics models end-to-end from scratch, with no pretraining or warmup schedules. Two design choices make this stable. First, a strictly causal inverse dynamics model with a low-dimensional variational bottleneck limits how much future information reaches the predictor. Second, and central to our approach, we introduce variance-reduced routing (VRR), which enables stable training on labeled and unlabeled data at once by conditioning the predictor on the posterior mean for labeled transitions.

Across five manipulation and control tasks, JEVA's CEM planner matches or outperforms every world-model baseline, and with only 5% of episodes labeled its learned policy matches or exceeds every baseline trained on 20% on Cube-Double and Cube-Triple. Overall,

1. We introduce variance-reduced routing (VRR), which conditions the forward predictor on the posterior mean for labeled transitions, analyze its bias–variance trade-off, and show that without it the latent action space collapses and unlabeled data does not help.
2. VRR enables co-training an inverse dynamics model and a forward world model end-to-end from scratch in a single stage, which outperforms warm-up variants of the same model.
3. We show across five manipulation and control tasks that JEVA outperforms or matches baselines under both CEM planning and behavior cloning, using a fraction of their labels: with only 5% of episodes labeled, its learned policy matches or exceeds every baseline trained on 20% on Cube-Double and Cube-Triple.

2 Related Work

World Models for Planning and Control. World models learn the dynamics of an environment so that an agent can plan by imagining the consequences of its actions. *Generative* world models predict future observations directly in pixel or observation space, which yields interpretable rollouts but expends capacity modeling task-irrelevant visual detail and couples planning quality to reconstruction fidelity. Conversely, *non-generative* Joint-Embedding Predictive Architectures (JEPAs) predict dynamics strictly in abstract latent spaces LeCun et al. (2022); Littwin et al. (2024), proving broadly effective across modalities, including images Assran et al. (2023); Balestriero & LeCun (2025a), video Bardes et al. (2024); Assran et al. (2025); Bardes et al. (2023), language Chen et al. (2025); Huang et al. (2025), audio Fei et al. (2023), 3D and cross-modal data Perera et al. (2025); Hu et al. (2024); Mo & Yun (2024), graphs Skenderi et al. (2023), and reinforcement learning and control Bagatella et al. (2025); Vujinovic & Kovacevic (2025); Ghaemi et al. (2025). Recent advances like LeWM Maes et al. (2026) achieve stable end-to-end latent prediction directly from raw pixels , avoiding complex heuristics. A growing body of work explores probabilistic predictive architectures Huang (2026). However, these models inherently lack grounded, controllable action interfaces

necessary for physical robot control. JEVA addresses precisely this gap by learning a latent action space that is simultaneously predictive of future latent states and grounded in physical robot actions.

Latent Action Models and Learning from Unlabeled Data. Learning control interfaces from action-free video typically involves inferring latent actions. Early works rely on non-differentiable discrete codes Bruce et al. (2024); Menapace et al. (2021), but continuous latent actions are better suited to fine-grained manipulation and gradient-based planning Liang et al. (2025); Kim et al. (2025; 2026), as AdaWorld Gao et al. (2025) and CLAM Liang et al. (2025) demonstrate. Existing methods differ mainly in when the latent action space is fixed and where action labels enter. Several freeze the latent space and then fit action heads post hoc Routray et al. (2026); Collins et al. (2025); Cui et al. (2024) or train the world model on top of it Gao et al. (2025); others train an imitation policy on the learned latents Daniel et al. (2026). Jointly training inverse and forward dynamics is notoriously unstable: CoLA-World Wang et al. (2026) requires an inverse-dynamics warm-up stage, while other methods rely on structural disentanglement Zhang et al. (2026a) or decoupled pretraining Zhang et al. (2026b) to prevent representation collapse. The closest prior work is CLAM Liang et al. (2025), which also learns continuous latent actions but has no variational bottleneck, grounds them with separate task-agnostic play data in a two-stage procedure, and learns fixed-goal policies. JEVA is, to our knowledge, the first method to co-train both components from scratch without any inverse-dynamics pretraining or multi-stage warmup - we show that this single-stage training outperforms warm-up variants of the same model (Table 1b).

3 JEVA: Joint Encoder for Variational Actions

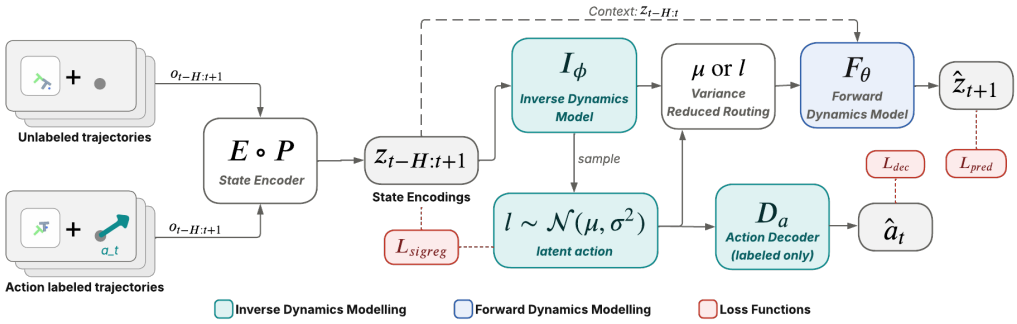

Figure 2: **JEVA architecture and gradient flow.** A shared state encoder maps visual observations into a latent space, where the Inverse Dynamics Model (IDM) infers a continuous Gaussian latent action. To stabilize training, a variance-reduced routing rule passes the deterministic posterior mean to the forward predictor for scarce labeled transitions, while preserving stochastic sampling for the abundant unlabeled transitions and the action decoder.

We introduce JEVA, a latent-action world model for robotic manipulation in the sparse-label regime, where image trajectories are abundant but only a small fraction of transitions are paired with ground-truth robot actions. Our goal is to co-train an inverse dynamics model and a forward world model from scratch, without requiring a separate inverse-dynamics pretraining stage or a multi-stage warmup schedule. Via co-training, JEVA learns a latent action space that is simultaneously useful for forward prediction and grounded in real robot controls.

JEVA treats the action between two visual states as a latent variable. It uses an inverse dynamics model to infer a distribution over this latent action from visual transitions, and a forward model to predict the next visual state from the current state and the inferred action, grounding the latent space with sparse supervision. On labeled transitions, an action decoder maps the latent action to the observed robot action. Thus the model learns from two sources of information: prediction from all trajectories and action grounding from the labeled subset.

The central difficulty is that joint training of an inverse dynamics model and a forward model from scratch is unstable. A continuous latent action space can collapse, the forward model can leak future information through the latent action, and once labels enter training, reparameterization noise enters the supervised prediction branch. JEVA addresses these with two design choices: a strictly causal VAE inverse dynamics model whose low-dimensional bottleneck limits future-information leakage

(Section 3.2), and, central to our approach, variance-reduced routing (VRR), which removes sampling noise from the labeled prediction branch and keeps the latent actions informative (Section 3.4).

JEVA consists of five components: an image encoder E, a shared projector P, a continuous VAE inverse dynamics model I_ϕ, a forward world model F_θ, and an action decoder D_a introduced below.

3.1 Latent Visual State Representation

Given an image observation o_t, we first encode it into a visual embedding using a ViT-Tiny backbone $h_t = E(o_t)$, $h_t \in \mathbb{R}^{192}$. We then apply a shared projector $P : \mathbb{R}^{192} \to \mathbb{R}^{192}$, implemented as a two-layer MLP with BatchNorm on the hidden activation, to obtain the latent visual state $z_t = P(h_t)$, $z_t \in \mathbb{R}^{192}$. The projected state z_t is consumed by both the inverse dynamics model and the forward world model. Sharing this representation couples latent action inference to predictive dynamics: the inverse model must produce actions that are useful for explaining transitions in the same latent space that the forward model predicts.

3.2 VAE Inverse Dynamics with Gaussian Latent Actions

We formulate inverse dynamics as posterior inference in a latent-variable model. For each transition, the unobserved variable is a latent action ℓ_t. The generative model first draws $\ell_t \sim p(\ell_t) = \mathcal{N}(0, I)$, and then uses this latent action to generate the next visual state,

$$z_{t+1} \sim p_\theta(z_{t+1} \mid z_{t-H+1:t}, \ell_{t-H+1:t}) = \mathcal{N}\left(F_\theta(z_{t-H+1:t}, \pi_F(\ell_{t-H+1:t})), \sigma_z^2 I\right). \tag{1}$$

For labeled transitions, the same latent action also generates the observed robot action,

$$a_t \sim p_\psi(a_t \mid \ell_t) = \mathcal{N}\left(D_a(\ell_t), \sigma_a^2 I\right). \tag{2}$$

Thus ℓ_t is shared by two likelihoods: a prediction likelihood available on every transition, and an action-grounding likelihood available only on labeled transitions.

The posterior over latent actions is intractable, so we approximate it with the inverse dynamics model. The inverse dynamics model is strictly causal: it infers each latent action only from the two states it connects, and I_ϕ outputs a diagonal Gaussian variational posterior,

$$q_\phi(\ell_t \mid z_t, z_{t+1}) = \mathcal{N}(\mu_t, \mathrm{diag}(\sigma_t^2)), \qquad \mu_t, \log \sigma_t^2 = I_\phi(z_t, z_{t+1}). \tag{3}$$

We sample latent actions using the reparameterization trick, $\ell_t = \mu_t + \sigma_t \odot \epsilon_t$ with $\epsilon_t \sim \mathcal{N}(0, I)$ drawn independently for each transition. This is a standard conditional VAE (Doersch, 2016) for z_{t+1} given past history $z_{t-H+1:t}$, with Gaussian latent variables ℓ_t intended to capture latent actions. Continuous VAE-style latent actions also appear in AdaWorld Gao et al. (2025) and LPWM Daniel et al. (2026); JEVA differs in learning this space jointly with the world model and grounding it with sparse action labels during training. The Gaussian latent action posterior yielding a differentiable and sampleable action space that can be optimized during planning and decoded into continuous robot controls.

The low-dimensional latent action limits how much next-frame content the inverse model can transmit to the predictor. Two regularizers shape this space at different levels. Per transition, we apply a KL prior on the posterior,

$$\mathcal{L}_{\mathrm{KL}} = \sum_t \mathrm{KL}(q_\phi(\ell_t \mid z_t, z_{t+1}) \,\|\, \mathcal{N}(0, I)), \tag{4}$$

which keeps each posterior close to the prior but constrains posteriors one at a time. Across transitions, we apply SIGReg (Balestriero & LeCun, 2025b) to the sampled latent actions, driving their aggregate distribution toward an isotropic Gaussian. This keeps the latent-action space well scaled and isotropic, so that latent actions proposed near the prior at planning time land where the predictor and decoder were trained. Neither regularizer determines whether individual posteriors stay informative. The KL term pulls every posterior toward the prior, and a posterior equal to the prior still matches the aggregate target, so latent dimensions that are not yet useful for prediction or action grounding can collapse onto the prior (He et al., 2019; Razavi et al., 2019; Wang et al., 2021; Dai et al., 2020; Lucas et al., 2019). Keeping latent actions informative is the role of action grounding (Section 3.5) and of the routing rule in Section 3.4.

3.3 FORWARD WORLD MODEL

The forward model predicts the next latent visual state from a short history of states and latent actions. We use a 6-layer autoregressive Transformer with AdaLN-Zero conditioning blocks. Given history length $H = 3$, the predictor receives latent states $z_{t-H+1:t}$ and projected latent actions $\pi_F(\bar{\ell}_{t-H+1:t})$:

$$\hat{z}_{t+1} = F_\theta\left(z_{t-H+1:t}, \pi_F(\bar{\ell}_{t-H+1:t})\right). \tag{5}$$

Here $\bar{\ell}_t$ denotes the latent action we condition the predictor on. Section 3.4 defines this conditioning. Following the Gaussian form of Eq. 1, we train the predictor with a squared-error loss,

$$\mathcal{L}_{\text{pred}} = \sum_t \|\hat{z}_{t+1} - z_{t+1}\|_2^2. \tag{6}$$

This loss is computed on both labeled and unlabeled trajectories, allowing the world model to learn from abundant unlabeled video.

3.4 VARIANCE-REDUCED ROUTING: POSTERIOR-MEAN CONDITIONING

The central component of JEVA is Variance-Reduced Routing (VRR), a rule for which representative of the inverse-dynamics posterior conditions the forward predictor: a sample $\ell_t \sim q_\phi(\ell_t \mid z_t, z_{t+1})$ or the posterior mean μ_t. Each training step draws a labeled batch B_L and an unlabeled batch B_U, and VRR conditions the predictor on

$$\bar{\ell}_t = \begin{cases} \mu_t, & t \in B_L, \\ \ell_t, & t \in B_U. \end{cases} \tag{7}$$

A labeled transition drawn into B_U is treated as unlabeled. Conditioning on samples everywhere gives a Monte Carlo estimate of the posterior-predictive objective but injects reparameterization noise into the prediction pathway. VRR is a mean plug-in approximation that removes this noise on the labeled branch at the cost of a bias. The rest of this section characterizes that trade-off and shows why it favours the labeled branch.

For a transition t, the ideal prediction objective is the posterior-predictive risk

$$\mathcal{R}_t(\theta) = \mathbb{E}_{q_\phi}\left[\|F_\theta\left(z_{t-H+1:t}, \pi_F(\ell_{t-H+1:t})\right) - z_{t+1}\|_2^2\right]. \tag{8}$$

To compare gradient estimators for this objective, fix the labeled batch $B = B_L$ and let L collect the distinct latent actions entering its prediction windows. Under the variational posterior, $L \sim \mathcal{N}(\mu, \Sigma)$, where μ stacks the posterior means μ_t; because the reparameterization noise is drawn independently for each transition is drawn independently across time steps, q_ϕ factorizes over time steps and Σ is diagonal. Write $h_\theta(L) = \frac{1}{|B|}\sum_{t\in B}\|F_\theta(z_{t-H+1:t}, \pi_F(\ell_{t-H+1:t})) - z_{t+1}\|_2^2$ for the batch prediction loss, so that $\mathbb{E}[h_\theta(L) \mid B] = \frac{1}{|B|}\sum_{t\in B}\mathcal{R}_t(\theta)$, and treat the targets z_{t+1} as fixed within a training step. The target gradient is $G_\star = \mathbb{E}[\nabla_\theta h_\theta(L) \mid B]$. Conditional on B, the only remaining randomness is posterior sampling, and the error of any estimator G of $G_\star$ decomposes as

$$\mathbb{E}\left[\|G - G_\star\|^2 \mid B\right] = \|\mathbb{E}[G \mid B] - G_\star\|^2 + \operatorname{tr}\operatorname{Cov}(G \mid B). \tag{9}$$

The one-sample estimator $G_{\text{mc}} = \nabla_\theta h_\theta(L)$ is unbiased, so its error is pure variance, $\operatorname{tr}\operatorname{Cov}(G_{\text{mc}} \mid B)$. VRR instead evaluates the gradient at the posterior mean, $G_{\text{vrr}} = \nabla_\theta h_\theta(\mu)$. This estimator is deterministic given B, so its error is pure bias, $\|b\|^2$ with $b = G_{\text{vrr}} - G_\star$. VRR therefore has lower gradient error if and only if

$$\|b\|^2 < \operatorname{tr}\operatorname{Cov}(G_{\text{mc}} \mid B), \tag{10}$$

and neither estimator dominates in general. Classical Rao–Blackwellization (Casella & Robert, 1996; Ranganath et al., 2014; AUEB et al., 2015) removes sampling variance at no cost in bias by computing the conditional expectation exactly. For a nonlinear Transformer predictor that expectation is intractable, and VRR replaces it with a plug-in at the posterior mean, trading variance for bias.

Since $\mu_t = \mathbb{E}_{q_\phi}[\ell_t]$, the plug-in is motivated by

$$\mathbb{E}_{q_\phi}\left[F_\theta\left(z_{t-H+1:t}, \pi_F(\ell_{t-H+1:t})\right)\right] \approx F_\theta\left(z_{t-H+1:t}, \pi_F(\mu_{t-H+1:t})\right), \tag{11}$$

which is exact when $F_\theta \circ \pi_F$ is affine in the latent actions. For a smooth nonlinear predictor, a second-order expansion around the posterior mean gives, for each output coordinate k,

$$\begin{aligned}\mathbb{E}_{q_\phi}\left[F_{\theta,k}\left(z_{t-H+1:t}, \pi_F(\ell_{t-H+1:t})\right)\right] = F_{\theta,k}\left(z_{t-H+1:t}, \pi_F(\mu_{t-H+1:t})\right) \\ + \tfrac{1}{2}\operatorname{tr}\left(H_{k,t}\,\Sigma_{t-H+1:t}\right) + o\left(\|\Sigma_{t-H+1:t}\|\right),\end{aligned} \tag{12}$$

where $H_{k,t}$ is the Hessian of the k-th output with respect to the latent-action window and $\Sigma_{t-H+1:t}$ is the posterior covariance over that window. This bounds the plug-in error for the *predicted mean* only. It is motivation rather than a guarantee, because training uses the loss, and the loss carries an additional term that is first order in the posterior covariance. The affine case makes this explicit: for $f_\theta(L) = AL + c$ with target y,

$$\mathbb{E}\left[\|f_\theta(L) - y\|_2^2 \mid B\right] = \|f_\theta(\mu) - y\|_2^2 + \operatorname{tr}\left(A\Sigma A^\top\right), \tag{13}$$

so even when the mean prediction is exact, the plug-in drops a term whose gradient with respect to A, namely $2A\Sigma$, is nonzero whenever the posterior has nonzero width. The VRR gradient is therefore biased even for an affine predictor. Appendix A makes this trade-off precise. When the posterior covariance scales as τ^2, the squared bias of VRR is $O(\tau^4)$ while the variance of the sampled estimator is $O(\tau^2)$, so VRR has lower gradient error once posteriors are tight (Proposition 1). The same expansion shows what sampling costs when posteriors are wide: it adds a penalty $\operatorname{tr}(J\Sigma J^\top)$, with J the predictor's Jacobian with respect to the latent actions, which pushes the predictor to ignore ℓ in proportion to posterior width. VRR removes this penalty on the labeled branch; consistent with this, with a warm-up stage and no VRR every latent-action dimension collapses (Table 1b).

Labeled-only routing follows from this analysis. The action decoder always consumes samples, $\hat{a}_t = D_a(\ell_t)$, forcing posterior samples, not just the mean, to correspond to executable actions; a wide posterior then decodes to noisy actions against a fixed ground truth, so labeled posteriors stay tight against only a weak KL pull ($\beta = 0.01$), the regime of Proposition 1. Because VRR changes only the predictor's input, the loss that tightens the posterior and the branch that exploits the tightness are distinct: on the labeled branch, the prediction loss reaches the inverse model only through μ_t, while σ_t is shaped by the decoder, KL, and SIGReg terms. Unlabeled posteriors stay wide, since many latent actions explain the same transition and no decoder loss tightens them, so Proposition 1 does not favour the mean there; the predictor must also remain accurate away from μ_t because CEM queries it there.

3.5 Sparse Action Grounding

For labeled transitions, we ground latent actions with a lightweight action decoder D_a, implemented as a two-layer MLP $\hat{a}_t = D_a(\ell_t) \in \mathbb{R}^{bd_a}$, where b=5 and d_a is the action dimention of the task. With a fixed-variance Gaussian action likelihood, the negative log-likelihood reduces to

$$\mathcal{L}_{\text{dec}} = \sum_{t \in \mathcal{D}_L} \|\hat{a}_t - a_t\|_2^2 . \tag{14}$$

This sparse decoder loss anchors the latent action space to real controls. This is different from behavior cloning, in that the decoder loss is not the only learning signal: the same latent actions are also shaped by forward prediction on both labeled and unlabeled trajectories.

3.6 Joint Objective

JEVA is trained end-to-end from scratch with a single objective,

$$\mathcal{L} = \mathcal{L}_{\text{pred}} + \lambda_{\text{dec}}\mathcal{L}_{\text{dec}} + \beta\mathcal{L}_{\text{KL}} + \lambda_{\text{sig}}\mathcal{L}_{\text{sigreg}}. \tag{15}$$

The prediction loss $\mathcal{L}_{\text{pred}}$ is computed on all transitions. The decoder loss $\mathcal{L}_{\text{dec}}$ is computed only on labeled transitions. The KL $\mathcal{L}_{\text{KL}}$ regularizes the Gaussian latent-action posterior while SIGReg matches the aggregate distribution of sampled latent actions to an isotropic Gaussian. We additionally apply SIGReg to the projected visual states, to keep the representations well conditioned.

$$\mathcal{L}_{\text{sigreg}} = \text{SIGReg}(z) + \text{SigReg}(\ell_{\text{t}}), \tag{16}$$

This objective couples three signals: prediction learns visual dynamics from all trajectories, decoding grounds latent actions where labels are available, and the VAE regularizer preserves a compact sampleable action space. VRR controls how these signals interact: it removes sampling noise from the labeled prediction branch while preserving stochastic learning on unlabeled data.

4 PLANNING IN THE LATENT ACTION SPACE

At inference time, JEVA selects latent actions ℓ_t rather than robot actions, and decodes each one into an executable action chunk $\hat{a}_t = D_a(\ell_t)$ with the frozen action decoder. We present two ways of choosing ℓ_t: optimization-based planning with the cross-entropy method (Section 4.1), and a goal-conditioned behavior-cloning policy in the latent action space that amortizes the cost of planning into a single forward pass (Section 4.2).

4.1 PLANNING WITH CEM

Given the current observation history, we encode observations into projected states and optimize candidate latent-action sequences using the cross-entropy method (CEM). The CEM distribution is initialized with the empirical mean latent action from the training set.

For each candidate sequence $\ell_{t:t+K-1}$, the forward model rolls out future latent states autoregressively:

$$\hat{z}_{t+k+1} = F_\theta\left(\hat{z}_{t+k-H+1:t+k}, \pi_F(\ell_{t+k-H+1:t+k})\right). \tag{17}$$

The best candidate sequence is selected according to the task objective, and the first latent action is decoded into a robot action through D_a. Planning is then repeated in a receding-horizon manner.

4.2 GOAL-CONDITIONED LATENT-ACTION POLICY

Given the trained encoder E , projector P and the frozen action decoder D_a, we train a remaining horizon conditioned behavior cloning policy following Nguyen et al. (2026). However, we apply the behavior cloning strategy directly onto the latent action space instead of the raw action space:

$$g_\xi(z_t,\, z_g,\, \tau) \;\longmapsto\; \ell \in \mathbb{R}^{d_\ell}, \tag{18}$$

where $z_t = P(E(o_t))$ is the current latent state, $z_g = P(E(o_g))$ the goal latent state, and τ the number of steps remaining to the goal.

Control. Given the current observation o_t and goal image o_g, the policy produces a latent action $\ell_t = g_\xi\big(z_t,\, z_g,\, \tau_t\big)$, which is decoded into an executable action chunk $\hat{a}_t = D_a(\ell_t)$. Control is receding-horizon: after executing the chunk we re-encode the new observation, decrement the remaining horizon $\tau_{t+1} = \max(1,\, \tau_t - 1)$, and re-query g_ξ. A single forward pass through E, P, g_ξ, and D_a produces the action; no search or forward rollout is used.

Training. The policy is trained by hindsight relabeling on the demonstrations, with the encoder, projector, inverse dynamics model, and action decoder held fixed. For a transition at time t we sample a future offset $\Delta \sim \mathcal{U}[\Delta_{\min}, \Delta_{\max}]$, set the goal latent to the future state $z_g = z_{t+\Delta}$ and the horizon to $\tau = \Delta$, and optimize behavior cloning through the frozen decoder:

$$\mathcal{L}(\xi) \;=\; \big\|\, D_a\big(g_\xi(z_t, z_{t+\Delta}, \Delta)\big) - a_t \,\big\|_2^2, \tag{19}$$

where $a_t \in \mathbb{R}^{b\, d_a}$ is the ground-truth action chunk. Training over a range of offsets Δ lets the horizon input τ select progressively nearer sub-goals as the agent approaches the target.

5 RESULTS

We test three claims: (i) with few action labels, action-free transitions let JEVA learn better policies than methods trained on labeled data alone; (ii) this advantage holds whether actions come from test-time search (CEM planning) or from a learned policy; and (iii) the unlabeled pool helps only when the latent action is regularized and the labeled branch uses variance-reduced routing.

5.1 EXPERIMENTAL SETUP

We evaluate on five tasks with 224×224 RGB observations: PushT (Zhou et al., 2025), Reacher (Tassa et al., 2018), and OGBench Cube-Single, -Double, and -Triple (Park et al., 2025),

which require moving one to three cubes to goal positions (details in Appendix H). From each dataset we label a fraction $\rho \in \{5, 10, 15, 20\}\%$ of episodes to form $\mathcal{D}_L$; the unlabeled set $\mathcal{D}_U$ contains the observations of all episodes with actions removed. We compare against latent world models planned with CEM (PLDM, LeWM, DINO-WM), goal-conditioned imitation and offline-RL policies (GCBC, GCIVL), and a goal-conditioned BC head on LeWM's latent space (LeWM-BC) (Nguyen et al., 2026), which separates the contribution of the representation from that of policy extraction. Baselines cannot consume action-free transitions, so they train on $\mathcal{D}_L$ alone at the same labeling budget. We report success rates over three seeds of 100 episodes each.

5.2 Comparison Against Baselines

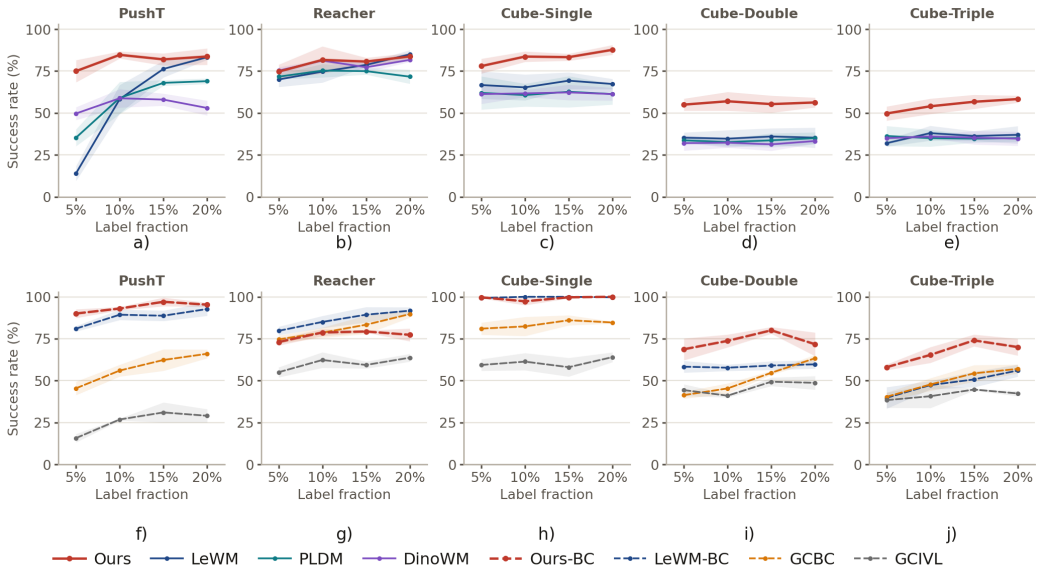

Figure 3: Success rates using CEM planner (a-e) and behavior cloning methods (f-j) across five different tasks

We report success rates for all methods in Fig. 3. Numerical success rates across all datasets are reported in Appendix C.

JEVA matches the baselines with a fraction of the labels. With 5% labels, JEVA-BC reaches 68.7% on Cube-Double and 58.0% on Cube-Triple, matching or exceeding every baseline trained with 20% labels (best: GCBC, 63.3% and 57.0%). On PushT, JEVA-BC with 10% labels (93.0%) matches the best 20% baseline (LeWM-BC, 92.7%). JEVA-BC is the best learned policy at every label fraction on PushT, Cube-Double and Cube-Triple. On Cube-Single, both world-model BC policies saturate ($\geq$ 97%).

The gain holds for both planning and learned policies. Within each group, JEVA-BC leads LeWM-BC, which trains the same BC architecture on LeWM's latent space, by 2.6–9.0 points on PushT, 10.4–21.0 on Cube-Double and 14.0–23.3 on Cube-Triple. The same holds under CEM planning: JEVA beats the strongest world-model baseline by 11–20 points on Cube-Single, 19–22 on Cube-Double and 13–21 on Cube-Triple. On PushT the CEM margin is largest with few labels (23.7 points at 5%, 26.4 at 10%) and closes by 20%, where LeWM matches JEVA (83.3 vs. 83.7).

BC outperforms CEM planning on the same latent. For both JEVA and LeWM, the BC policy beats CEM planning in the same latent space on PushT and all three Cube tasks. On the Cube tasks the CEM baselines stay at 31–38% regardless of label fraction, while LeWM-BC reaches 58–60% on Cube-Double from the same LeWM latent. The flat CEM curves therefore reflect a limit of planning, not of the representation.

Cont.	Unsup.	VAE	LA-Reg	VRR	SR (%)
✓	×	×	×	×	49.70±5.50
✓	✓	×	×	×	46.67±1.15
✓	✓	✓	×	×	51.00±6.03
✓	✓	✓	✓	×	61.00±4.40
✓	✓	✓	✓	✓	**75.0±6.6**

(a) Each additional component of JEVA's latent-action bottleneck improves performance. **Cont.**: continuous bottleneck; **Unsup.**: unsupervised (action-free) pool; **VAE**: variational bottleneck; **LA-Reg**: regularization on the latent-action space; **VRR**: variance-reduced routing.

Training	Act. Ground.	VRR	Active dims	SR (%)
Ours	✓	✓	7	**75.0**±6.6
Ours	✓	×	6	61.00±4.40
Warm-up	×	✓	6	68.6
Warm-up	✓	✓	4	48.33±1.27
Warm-up	× / ✓	×	**0**	collapsed

(b) Effect of a warm-up stage. Single-stage training outperforms every warm-up variant. **Act. Ground.**: action grounding; **VRR**: variance-reduced routing; **Active dims**: latent-action dimensions still active after training, which collapse to **0** under a warm-up without VRR regardless of grounding.

Table 1: Ablations on PushT (5% data).

Fine-grained control: Reacher. On Reacher, JEVA-BC (73–79%) trails LeWM-BC and GCBC (91.7% and 89.7% at 20% labels). Reacher requires precise per-step joint deltas, which the JEVA latent does not capture: it encodes coarse arm pose well (linear probe $R^2 \approx 0.90$) but not the fine motion between consecutive states. JEVA-BC must decode actions from this latent, whereas LeWM-BC and GCBC regress raw actions directly. JEVA's CEM planner remains competitive; we hypothesize that search, which scores many candidate rollouts and executes the best, tolerates decoder imprecision better than a single amortized prediction. This reflects a limitation of latent-action learning when per-step motion lies below the resolution of the learned encoding.

Component ablation. Table 1a adds components cumulatively on PushT with 5% labels, planning with CEM. Neither the unlabeled pool alone ($49.7 \rightarrow 46.7$) nor a variational bottleneck on top of it (51.0) improves on labeled-only training. The table shows that both regularizing the latent action space and using VRR are necessary for best performance. VRR feeds the predictor the posterior mean μ_t on the labeled branch instead of a reparameterized sample, so the action-grounding gradient is free of sampling noise. The unlabeled branch keeps sampling, so the KL term stays active. Unlabeled data therefore helps only when the right bottleneck surrounds it.

Joint vs. two-stage training. Could the IDM and predictor instead be pre-trained on $\mathcal{D}_U$ and grounded afterwards? Table 1b crosses two warm-up choices, action grounding and VRR. The second stage is always full JEVA co-training with data, architecture and compute held fixed (PushT, 5% labels). Without VRR, the latent action collapses during warm-up (no active dimensions), with or without grounding. With VRR, warm-up without grounding reaches 68.6%, below single-stage training (75.0%), and warm-up with grounding drops to 48.3%. We hypothesize that early grounding hurts performance since it commits the latent action space to the small labeled set before $\mathcal{D}_U$ has shaped it. Across all configurations, success tracks the number of active latent-action dimensions.

6 Conclusion

We presented JEVA, a latent-action world model that co-trains inverse and forward dynamics from scratch on sparsely labeled data. Variance-reduced routing, together with a regularized continuous variational bottleneck, makes the action-free pool useful: without them, unlabeled transitions do not improve on labeled-only training, and VRR is needed for the best performance. With this, JEVA matches or outperforms every world-model baseline planned with CEM on all five tasks, with the largest gains at low label fractions. The advantage carries over to learned policies: JEVA-BC is the best learned policy on PushT and all three Cube tasks, and with only 5% labels it matches or exceeds every baseline trained with 20% on Cube-Double and Cube-Triple.

Limitations. Latent-action learning assumes that actions are identifiable from consecutive observations. On Reacher, where control depends on precise per-step joint deltas, the learned latent encodes arm pose but not the fine motion between states, so JEVA-BC trails policies that regress raw actions directly, while CEM planning with the same decoder remains competitive. Autoregressive rollouts accumulate per-step prediction errors, which restricts the effective planning horizon. Tasks are specified by goal frames in the agent's native observation space; supporting language instructions or sparse rewards requires alternative goal encodings.

AI USE STATEMENT

AI assistants were used to help run experiments, write code, and polish the writing of this manuscript. All AI-assisted outputs, including code, experimental results, and text, were manually reviewed and verified by the authors, who take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

We will release our code, training configurations, and model checkpoints upon acceptance.

ACKNOWLEDGMENTS

If a paper is accepted, the final camera-ready version will (and probably should) include acknowledgments. All acknowledgments go at the end of the paper, including thanks to reviewers who gave useful comments, to colleagues who contributed to the ideas, and to funding agencies and corporate sponsors that provided financial support.

REFERENCES

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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15619–15629, 2023.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Titsias RC AUEB, Miguel Lázaro-Gredilla, et al. Local expectation gradients for black box variational inference. *Advances in neural information processing systems*, 28, 2015.

Marco Bagatella, Matteo Pirotta, Ahmed Touati, Alessandro Lazaric, and Andrea Tirinzoni. Td-jepa: Latent-predictive representations for zero-shot reinforcement learning. *arXiv preprint arXiv:2510.00739*, 2025.

Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025a.

Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics, 2025b. URL https://arxiv.org/abs/2511.08544.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. V-jepa: Latent video prediction for visual representation learning. 2023.

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. *arXiv preprint arXiv:2404.08471*, 2024.

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments, 2024. URL https://arxiv.org/abs/2402.15391.

George Casella and Christian P Robert. Rao-blackwellisation of sampling schemes. *Biometrika*, 83 (1):81–94, 1996.

Delong Chen, Mustafa Shukor, Theo Moutakanni, Willy Chung, Jade Yu, Tejaswi Kasarla, Yejin Bang, Allen Bolourchi, Yann LeCun, and Pascale Fung. Vl-jepa: Joint embedding predictive architecture for vision-language. *arXiv preprint arXiv:2512.10942*, 2025.

Jeremy A. Collins, Loránd Cheng, Kunal Aneja, Albert Wilcox, Benjamin Joffe, and Animesh Garg. Amplify: Actionless motion priors for robot learning from videos, 2025. URL https://arxiv.org/abs/2506.14198.

Zichen Jeff Cui, Hengkai Pan, Aadhithya Iyer, Siddhant Haldar, and Lerrel Pinto. Dynamo: In-domain dynamics pretraining for visuo-motor control, 2024. URL https://arxiv.org/abs/2409.12192.

Bin Dai, Ziyu Wang, and David Wipf. The usual suspects? reassessing blame for vae posterior collapse. In *International conference on machine learning*, pp. 2313–2322. PMLR, 2020.

Tal Daniel, Carl Qi, Dan Haramati, Amir Zadeh, Chuan Li, Aviv Tamar, Deepak Pathak, and David Held. Latent particle world models: Self-supervised object-centric stochastic dynamics modeling, 2026. URL https://arxiv.org/abs/2603.04553.

Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

Zhengcong Fei, Mingyuan Fan, and Junshi Huang. A-jepa: Joint-embedding predictive architecture can listen. *arXiv preprint arXiv:2311.15830*, 2023.

Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. Adaworld: Learning adaptable world models with latent actions, 2025. URL https://arxiv.org/abs/2503.18938.

Hafez Ghaemi, Eilif Benjamin Muller, and Shahab Bakhtiari. seq-JEPA: Autoregressive predictive learning of invariant-equivariant world models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=GKt3VRaCU1.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv preprint arXiv:1901.05534*, 2019.

Naiwen Hu, Haozhe Cheng, Yifan Xie, Shiqi Li, and Jihua Zhu. 3d-jepa: A joint embedding predictive architecture for 3d self-supervised representation learning. *arXiv preprint arXiv:2409.15803*, 2024.

Hai Huang, Yann LeCun, and Randall Balestriero. Llm-jepa: Large language models meet joint embedding predictive architectures. *arXiv preprint arXiv:2509.14252*, 2025.

Yongchao Huang. Vjepa: Variational joint embedding predictive architectures as probabilistic world models. *arXiv preprint arXiv:2601.14354*, 2026.

Hanjung Kim, Jaehyun Kang, Hyolim Kang, Meedeum Cho, Seon Joo Kim, and Youngwoon Lee. Uniskill: Imitating human videos via cross-embodiment skill representations, 2025. URL https://arxiv.org/abs/2505.08787.

Hanjung Kim, Lerrel Pinto, and Seon Joo Kim. Hierarchical latent action model, 2026. URL https://arxiv.org/abs/2603.05815.

Yann LeCun et al. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

Anthony Liang, Pavel Czempin, Matthew Hong, Yutai Zhou, Erdem Biyik, and Stephen Tu. Clam: Continuous latent action models for robot learning from unlabeled demonstrations, 2025. URL https://arxiv.org/abs/2505.04999.

Etai Littwin, Omid Saremi, Madhu Advani, Vimal Thilak, Preetum Nakkiran, Chen Huang, and Joshua Susskind. How jepa avoids noisy features: The implicit bias of deep linear self distillation networks. *Advances in Neural Information Processing Systems*, 37:91300–91336, 2024.

Dylan P. Losey, Hong Jun Jeon, Mengxi Li, Krishnan Srinivasan, Ajay Mandlekar, Animesh Garg, Jeannette Bohg, and Dorsa Sadigh. Learning latent actions to control assistive robots, 2021. URL https://arxiv.org/abs/2107.02907.

James Lucas, George Tucker, Roger B Grosse, and Mohammad Norouzi. Don't blame the elbo! a linear vae perspective on posterior collapse. *Advances in Neural Information Processing Systems*, 32, 2019.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels, 2026. URL https://arxiv.org/abs/2603.19312.

Willi Menapace, Stéphane Lathuilière, Sergey Tulyakov, Aliaksandr Siarohin, and Elisa Ricci. Playable video generation, 2021. URL https://arxiv.org/abs/2101.12195.

Shentong Mo and Sukmin Yun. Dmt-jepa: Discriminative masked targets for joint-embedding predictive architecture. *arXiv preprint arXiv:2405.17995*, 2024.

Hoang Nguyen, Xiaohao Xu, and Xiaonan Huang. Latent geometry beyond search: Amortizing planning in world models, 2026. URL https://arxiv.org/abs/2605.08732.

Seohong Park, Zichen Wang, Yuke Zhu, Hao Li, and Chelsea Finn. OGBench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations (ICLR)*, 2025.

Avishka Perera, Kumal Hewagamage, Saeedha Nazar, Kavishka Abeywardana, Hasitha Gallella, Ranga Rodrigo, and Mohamed Afham. Crossjepa: Cross-modal joint-embedding predictive architecture for efficient 3d representation learning from 2d images. *arXiv preprint arXiv:2511.18424*, 2025.

Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial intelligence and statistics*, pp. 814–822. PMLR, 2014.

Ali Razavi, Aäron van den Oord, Ben Poole, and Oriol Vinyals. Preventing posterior collapse with delta-vaes. *arXiv preprint arXiv:1901.03416*, 2019.

Sandeep Routray, Hengkai Pan, Unnat Jain, Shikhar Bahl, and Deepak Pathak. Vipra: Video prediction for robot actions, 2026. URL https://arxiv.org/abs/2511.07732.

Younggyo Seo, Kimin Lee, Stephen James, and Pieter Abbeel. Reinforcement learning with action-free pre-training from videos, 2022. URL https://arxiv.org/abs/2203.13880.

Geri Skenderi, Hang Li, Jiliang Tang, and Marco Cristani. Graph-level representation learning with joint-embedding predictive architectures. *arXiv preprint arXiv:2309.16014*, 2023.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite, 2018. URL https://arxiv.org/abs/1801.00690.

Aleksandar Vujinovic and Aleksandar Kovacevic. Act-jepa: Novel joint-embedding predictive architecture for efficient policy representation learning. *arXiv preprint arXiv:2501.14622*, 2025.

Jianren Wang, Yujie Lu, and Hang Zhao. Cloud: Contrastive learning of unsupervised dynamics, 2020. URL https://arxiv.org/abs/2010.12488.

Yixin Wang, David Blei, and John P Cunningham. Posterior collapse and latent variable non-identifiability. *Advances in neural information processing systems*, 34:5443–5455, 2021.

Yucen Wang, Fengming Zhang, De-Chuan Zhan, Li Zhao, Kaixin Wang, and Jiang Bian. Co-evolving latent action world models, 2026. URL https://arxiv.org/abs/2510.26433.

Chuheng Zhang, Tim Pearce, Pushi Zhang, Kaixin Wang, Xiaoyu Chen, Wei Shen, Li Zhao, and Jiang Bian. What do latent action models actually learn?, 2025. URL https://arxiv.org/abs/2506.15691.

Tianqiu Zhang, Muyang Lyu, Yufan Zhang, Fang Fang, and Si Wu. Dila: Disentangled latent action world models, 2026a. URL https://arxiv.org/abs/2605.15725.

Wenyao Zhang, Bozhou Zhang, Zekun Qi, Wenjun Zeng, Xin Jin, and Li Zhang. Disentangled robot learning via separate forward and inverse dynamics pretraining, 2026b. URL https://arxiv.org/abs/2604.16391.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning, 2025. URL https://arxiv.org/abs/2411.04983.

A VRR THEORY

Proposition 1 (VRR in the small-posterior regime)**.** *Fix a labeled batch B and let $L_\tau \sim \mathcal{N}(\mu, \tau^2\Sigma_0)$ for a fixed covariance Σ_0. Assume $g(L) = \nabla_\theta h_\theta(L)$ is three times continuously differentiable in L, with derivatives bounded by a polynomial in $\|L - \mu\|$, and let $M = \nabla_L g(\mu) = \nabla_\theta \nabla_L h_\theta(\mu)$. Then*

$$\|b\|^2 = O(\tau^4), \qquad \operatorname{tr}\operatorname{Cov}(G_{\mathrm{mc}} \mid B) = \tau^2 \operatorname{tr}\left(M\Sigma_0 M^\top\right) + O(\tau^4). \tag{20}$$

If $\operatorname{tr}(M\Sigma_0 M^\top) > 0$, there exists $\tau^\star > 0$ such that VRR has strictly lower conditional gradient error than G_{mc} for all $\tau < \tau^\star$.

The analysis concerns the predictor gradient ∇_θ only; the posterior parameters μ, Σ are produced by the encoder, projector, and inverse model, which do not depend on θ, and are treated as fixed. The proof (Appendix B) expands g in L around μ; odd Gaussian moments vanish, so the bias is $b = -\frac{\tau^2}{2}\nabla_\theta \operatorname{tr}\left(\nabla_L^2 h_\theta(\mu)\,\Sigma_0\right) + O(\tau^3)$. The crossover depends on batch size. Transitions from different trajectories have independent latent actions, so the variance term averages down as $1/N$, with N the number of independent trajectories in B, while the bias does not; $\tau^\star$ therefore scales as $N^{-1/2}$.

Why the bias helps: VRR removes a collapse-inducing penalty. Proposition 1 covers tight posteriors, but VRR's clearest empirical effect is on collapse: with a warm-up stage and no VRR, every latent-action dimension collapses, with or without action grounding, and without warm-up VRR still keeps one more dimension active (Table 1b). The same expansion suggests why. For a transition with Jacobian $J_t = \partial F_\theta / \partial \ell_{t-H+1:t}$ and residual $r_t = F_\theta(\cdot, \pi_F(\mu_{t-H+1:t})) - z_{t+1}$, both evaluated at the posterior mean, the sampled loss satisfies

$$\begin{aligned}\mathbb{E}_{q_\phi}\left\|F_\theta(\cdot, \pi_F(\ell_{t-H+1:t})) - z_{t+1}\right\|_2^2 = \|r_t\|_2^2 + \operatorname{tr}\left(J_t \Sigma_{t-H+1:t} J_t^\top\right) \\ + \sum_k r_{t,k} \operatorname{tr}\left(H_{k,t}\Sigma_{t-H+1:t}\right) + o\left(\|\Sigma_{t-H+1:t}\|\right),\end{aligned} \tag{21}$$

of which Eq. (13) is the affine special case. To leading order in Σ, the VRR bias is minus the θ-gradient of the two correction terms, averaged over the batch. The second vanishes as the residual goes to zero; the first is a contractive penalty on the predictor's sensitivity to latent actions, weighted by posterior width. Sampling therefore does more than add noise: it trains the predictor to ignore ℓ in proportion to posterior uncertainty. We read this as feeding a collapse spiral. A predictor that ignores ℓ gives the inverse model no reason to keep the posterior informative, the KL term pulls it toward the prior, the posterior widens, the penalty grows, and latent dimensions become uninformative. SIGReg (Section 3.2) does not remove the force behind this spiral: it fixes where latent actions live in aggregate, and a collapsed posterior still matches its target. VRR instead removes this force on the labeled branch, where latent actions must stay informative to be decoded. The penalty is largest when posteriors are wide; once the labeled posteriors tighten, Proposition 1 applies. On this reading, VRR's bias is not merely bounded: it removes a term that drives collapse.

B PROOF OF PROPOSITION 1

Setup. Fix the labeled batch B and the parameters θ, and treat the targets z_{t+1} as fixed. Write $g(L) = \nabla_\theta h_\theta(L) \in \mathbb{R}^p$ and $\delta = L_\tau - \mu = \tau\,\Sigma_0^{1/2}\xi$ with $\xi \sim \mathcal{N}(0, I)$, so that

$$G_{\mathrm{mc}} = g(\mu + \delta), \qquad G_{\mathrm{vrr}} = g(\mu), \qquad G_\star = \mathbb{E}\left[g(\mu + \delta)\right].$$

By assumption there are constants $C, m \geq 0$ with $\|\nabla_L^j g(\mu + s)\| \leq C(1 + \|s\|^m)$ for $j = 1, 2, 3$ and all s. Since δ is Gaussian with covariance $\tau^2\Sigma_0$, for every $k \geq 0$ there is a constant c_k such that, for all $\tau \leq 1$,

$$\mathbb{E}\left[\|\delta\|^k \left(1 + \|\delta\|^m\right)^2\right] \leq c_k \tau^k. \tag{22}$$

Let $M = \nabla_L g(\mu) \in \mathbb{R}^{p \times n}$ and define the quadratic map $Q(\delta)_i = \delta^\top \nabla_L^2 g_i(\mu)\,\delta$. Taylor's theorem with integral remainder gives

$$g(\mu + \delta) = g(\mu) + M\delta + \tfrac{1}{2}Q(\delta) + R(\delta), \qquad \|R(\delta)\| \leq \tfrac{C}{6}\|\delta\|^3\left(1 + \|\delta\|^m\right), \tag{23}$$

$$g(\mu + \delta) = g(\mu) + M\delta + S(\delta), \qquad \|S(\delta)\| \leq \tfrac{C}{2}\|\delta\|^2\left(1 + \|\delta\|^m\right), \tag{24}$$

with $S(\delta) = \frac{1}{2}Q(\delta) + R(\delta)$.

Step 1: bias. Taking expectations in Eq. (23) and using $\mathbb{E}[\delta] = 0$ and $\mathbb{E}[Q(\delta)]_i = \operatorname{tr}\left(\nabla_L^2 g_i(\mu)\operatorname{Cov}(\delta)\right)$,

$$b = G_{\mathrm{vrr}} - G_\star = -\frac{\tau^2}{2}\, q - \mathbb{E}\left[R(\delta)\right], \qquad q_i = \operatorname{tr}\left(\nabla_L^2 g_i(\mu)\,\Sigma_0\right).$$

By Eq. (22), $\|\mathbb{E}[R(\delta)]\| \le c\,\tau^3$, so $\|b\| \le \frac{\tau^2}{2}\|q\| + c\,\tau^3$ and $\|b\|^2 \le K_b\tau^4$ for $\tau \le 1$. Since $\nabla_L^2 g_i = \partial_{\theta_i}\nabla_L^2 h_\theta$, we have $q = \nabla_\theta \operatorname{tr}\left(\nabla_L^2 h_\theta(\mu)\,\Sigma_0\right)$, which is the leading term stated in the main text.

Step 2: variance. By Eq. (24) and $\mathbb{E}[\delta] = 0$, $G_{\mathrm{mc}} - G_\star = M\delta + (S(\delta) - \mathbb{E}[S(\delta)])$, so

$$\operatorname{tr}\operatorname{Cov}(G_{\mathrm{mc}} \mid B) = \mathbb{E}\|M\delta\|^2 + 2\,\mathbb{E}\left[(M\delta)^\top\left(S(\delta) - \mathbb{E}[S(\delta)]\right)\right] + \mathbb{E}\,\|S(\delta) - \mathbb{E}[S(\delta)]\|^2.$$

The first term equals $\operatorname{tr}\left(M\operatorname{Cov}(\delta)M^\top\right) = \tau^2 v$ with $v = \operatorname{tr}\left(M\Sigma_0 M^\top\right)$. In the cross term, $\mathbb{E}[(M\delta)^\top\mathbb{E}[S(\delta)]] = 0$, and

$$\mathbb{E}\left[(M\delta)^\top S(\delta)\right] = \tfrac{1}{2}\,\mathbb{E}\left[(M\delta)^\top Q(\delta)\right] + \mathbb{E}\left[(M\delta)^\top R(\delta)\right].$$

The first expectation is a third moment of a zero-mean Gaussian and vanishes. The second is at most $\|M\|\,\mathbb{E}\left[\|\delta\|\,\|R(\delta)\|\right] \le c\,\tau^4$ by Eqs. (23) and (22). The last term is at most $\mathbb{E}\|S(\delta)\|^2 \le c\,\tau^4$ by Eq. (24). Hence, for $\tau \le 1$,

$$\left|\operatorname{tr}\operatorname{Cov}(G_{\mathrm{mc}} \mid B) - \tau^2 v\right| \le K_v\tau^4.$$

Step 3: crossover. Let $v > 0$ and $\tau^\star = \min\left\{1, (v/(K_b + K_v))^{1/2}\right\}$. For $\tau < \tau^\star$,

$$\|b\|^2 \le K_b\tau^4 < \tau^2 v - K_v\tau^4 \le \operatorname{tr}\operatorname{Cov}(G_{\mathrm{mc}} \mid B),$$

so by Eq. (10) VRR has strictly lower conditional gradient error. □

Remark: dependence on batch size. Suppose the batch loss is a mean over N transitions with independent latent actions, $h_\theta = \frac{1}{N}\sum_{n=1}^{N} h_{\theta,n}(L_n)$. Transitions from the same trajectory share latent actions through overlapping windows, so N here counts independent trajectories in B. Then $G_{\mathrm{mc}} = \frac{1}{N}\sum_n g_n(L_n)$ is an average of independent terms, and $\operatorname{tr}\operatorname{Cov}(G_{\mathrm{mc}} \mid B) = \frac{1}{N^2}\sum_n \operatorname{tr}\operatorname{Cov}(g_n(L_n))$, so v and K_v are $O(1/N)$. The bias $b = \frac{1}{N}\sum_n b_n$ is an average of per-transition biases and does not shrink with N unless they cancel. Hence $K_b = O(1)$ and $\tau^\star \propto N^{-1/2}$: larger labeled batches narrow the regime in which VRR wins on gradient error alone, which is why the mechanism in Section A, not the proposition, carries the argument early in training.

C Results Table

We provide the full numerical results for all experiments here, including means and standard deviations over seeds, as tables complementing the plots in Section 5.

D Compute

All numbers in Table 3 are measured on a single NVIDIA A40 (46 GB) with 4 CPUs and 48 GB of RAM. JEVA requires compute comparable to other world-model planners (PLDM, LeWM), while DINO-WM is roughly 700× more expensive per plan due to its DINOv2 encoder and patch-level predictor.

E Hyperparameters

We provide hyperparameters for training JEVA in Table 4.

	Policy	Plan.	5%	10%	15%	20%
PushT						
LeWM		✓	51.3 ± 6.4	58.3 ± 8.5	76.3 ± 5.0	83.3 ± 1.5
PLDM		✓	35.3 ± 5.0	59.0 ± 9.6	68.0 ± 1.7	69.0 ± 1.7
DINO-WM		✓	49.7 ± 3.8	58.7 ± 5.5	58.0 ± 3.6	53.0 ± 4.4
LeWM-BC	✓		81.0 ± 2.0	89.3 ± 3.1	88.7 ± 3.1	92.7 ± 4.0
GCBC	✓		45.3 ± 4.0	56.0 ± 3.5	62.3 ± 6.4	66.0 ± 2.6
GCIVL	✓		15.7 ± 2.5	26.7 ± 0.6	31.0 ± 6.0	29.0 ± 4.0
Ours		✓	75.0 ± 6.6	84.7 ± 2.1	82.0 ± 3.6	83.7 ± 4.9
Ours-BC	✓		**90.0 ± 2.6**	**93.0 ± 1.0**	**97.0 ± 2.6**	**95.3 ± 1.2**
Reacher						
LeWM		✓	70.0 ± 4.4	74.7 ± 6.5	78.7 ± 1.2	85.0 ± 2.6
PLDM		✓	71.7 ± 1.5	75.3 ± 4.0	75.0 ± 2.0	71.7 ± 4.2
DINO-WM		✓	75.3 ± 2.1	81.3 ± 0.6	77.3 ± 2.9	81.7 ± 1.2
LeWM-BC	✓		**79.7 ± 2.5**	**85.0 ± 3.6**	**89.3 ± 4.6**	**91.7 ± 2.1**
GCBC	✓		74.7 ± 2.3	78.7 ± 3.1	83.3 ± 5.5	89.7 ± 2.1
GCIVL	✓		55.0 ± 3.0	62.3 ± 4.6	59.3 ± 2.1	63.7 ± 2.3
Ours		✓	74.7 ± 4.2	81.7 ± 8.1	80.7 ± 2.3	83.7 ± 2.1
Ours-BC	✓		73.0 ± 2.0	78.7 ± 2.1	79.3 ± 0.6	77.3 ± 3.8
OGBench Cube-Single						
LeWM		✓	66.7 ± 10.0	65.3 ± 9.5	69.3 ± 5.0	67.3 ± 3.1
PLDM		✓	62.0 ± 12.1	60.7 ± 6.8	62.7 ± 9.3	61.3 ± 7.6
DINO-WM		✓	61.3 ± 5.8	61.7 ± 2.1	62.3 ± 4.6	61.3 ± 3.8
LeWM-BC	✓		99.3 ± 0.6	**100.0 ± 0.0**	**100.0 ± 0.0**	99.7 ± 0.6
GCBC	✓		81.0 ± 3.6	82.3 ± 5.7	86.0 ± 3.0	84.7 ± 0.6
GCIVL	✓		59.3 ± 3.5	61.3 ± 5.1	58.0 ± 5.6	64.0 ± 3.6
Ours		✓	78.0 ± 4.4	83.7 ± 3.1	83.3 ± 2.1	87.7 ± 2.9
Ours-BC	✓		**99.7 ± 0.6**	97.3 ± 2.3	99.7 ± 0.6	**100.0 ± 0.0**
OGBench Cube-Double						
LeWM		✓	35.3 ± 3.1	34.7 ± 5.1	36.0 ± 6.0	35.3 ± 6.0
PLDM		✓	33.7 ± 1.5	32.7 ± 2.1	33.7 ± 4.2	35.0 ± 3.5
DINO-WM		✓	32.0 ± 4.4	32.3 ± 2.9	31.3 ± 3.8	33.3 ± 1.5
LeWM-BC	✓		58.3 ± 3.5	57.7 ± 2.3	59.0 ± 2.6	59.7 ± 2.5
GCBC	✓		41.3 ± 2.1	45.3 ± 3.1	54.7 ± 1.2	63.3 ± 2.1
GCIVL	✓		44.3 ± 3.5	41.0 ± 1.0	49.3 ± 2.5	48.7 ± 4.0
Ours		✓	55.0 ± 3.6	57.0 ± 5.6	55.3 ± 5.1	56.3 ± 2.9
Ours-BC	✓		**68.7 ± 6.5**	**73.7 ± 3.8**	**80.0 ± 2.0**	**71.7 ± 7.0**
OGBench Cube-Triple						
LeWM		✓	32.0 ± 2.0	38.0 ± 4.4	36.3 ± 2.9	37.0 ± 5.2
PLDM		✓	36.3 ± 6.0	35.0 ± 5.0	34.7 ± 2.3	35.0 ± 2.0
DINO-WM		✓	35.0 ± 1.0	36.0 ± 1.0	35.3 ± 4.0	34.7 ± 4.2
LeWM-BC	✓		39.7 ± 6.4	47.3 ± 2.5	50.7 ± 4.6	56.0 ± 3.6
GCBC	✓		40.3 ± 2.5	47.7 ± 3.8	54.3 ± 4.9	57.0 ± 2.6
GCIVL	✓		38.3 ± 4.5	40.7 ± 7.0	44.7 ± 0.6	42.3 ± 1.5
Ours		✓	49.7 ± 4.2	54.0 ± 4.6	56.7 ± 4.2	58.3 ± 2.1
Ours-BC	✓		**58.0 ± 1.7**	**65.3 ± 4.7**	**74.0 ± 3.5**	**70.0 ± 5.0**

Table 2: Success rate (%) against the fraction of labeled episodes. **Plan.**: actions chosen by CEM search in a learned world model. **Policy**: actions from a learned goal-conditioned policy network, trained by behavior cloning (GCBC, LeWM-BC, JEVA-BC) or extracted from a learned value function (GCIVL). Baselines train on $\mathcal{D}_L$ only; JEVA additionally uses $\mathcal{D}_U$. Mean ± std over 3 seeds, 100 episodes each.

Method	Params (M)	FLOPs / solve (G) ↓	Plan time (s) ↓
PLDM	18.0	8 ,299	1.76
LeWM	17.3	8,299	1.75
DINO-WM	42.3	5,759,563	561.9
JEVA	18.6	8,299	1.86

Table 3: Computational cost and planning time on Push-T.

F Physical Latent Probing

Table 5 details how the choice of action bottleneck shapes the learned representations. For agent location, JEVA achieves a linear MSE of 0.0074, an improvement over LeWM's 0.0579, and an MLP r of 0.9986. This strong linear readability corresponds to JEVA's co-training objective: because the forward predictor consumes latent actions l_t inferred by the IDM rather than ground-truth actions, the encoder is pressured to capture transition-determining variables, like paddle motion, directly in z. Meanwhile, task-critical block-state properties are still recovered to high fidelity by JEVA nonlinearly, achieving MLP r values of 0.9922 for location and 0.9250 for angle. However, its higher

Shared hyperparameter	Value
λ_{sig} (SIGReg, state / latent action)	0.09 / 0.09
λ_{dec} (decoder MSE)	1.0
β (KL on l_t)	0.01
Context length (history H)	3
Prediction horizon	1
Learning rate	5×10^{-5}
Batch size (unlabeled / labeled)	256 / 128

Task-specific	PushT	Reacher	C-S	C-D	C-T
Latent action dim $\dim(l_t)$	8	8	16	16	16
Env action dim (d_a)	2	2	5	5	5

Table 4: JEVA hyperparameters. Top: shared across all tasks. Bottom: task-specific.

linear MSE on these properties compared to LeWM indicates this block information is encoded less linearly. LeWM, which receives ground-truth actions directly, achieves both lower linear and MLP MSE on these block states. Ultimately, JEVA's representational allocation—highly linear for action-determining variables and nonlinear for task-state variables—is empirically sufficient: JEVA's planning success rates exceed LeWM's despite the linear probing gap on rotation.

		Linear		MLP	
Property	**Model**	MSE ↓	r ↑	MSE ↓	r ↑
Agent Location	LeWM	0.0579	0.9709	0.0078	0.9962
	JEVA	**0.0074**	**0.9963**	**0.0028**	**0.9986**
Block Location	LeWM	**0.0272**	**0.9869**	**0.0026**	**0.9988**
	JEVA	0.0468	0.9777	0.0165	0.9922
Block Angle	LeWM	1.7794	**0.8993**	**0.0631**	**0.9683**
	JEVA	**1.6329**	0.8081	0.1460	0.9250

Table 5: Physical latent probing results on Push-T, 5% data.

G Open-loop latent rollouts

At each rollout step, the predictor consumes a sliding three-frame window of latent embeddings z alongside a matched latent-action window $l_{1:3}$ to autoregressively predict the next embedding, which is appended to the window. As illustrated in figure 4, by $T{=}35$ the model rolls out entirely on its own past predictions. Despite this extended horizon, the imagined frames preserve task-relevant kinematics . i.e. the paddle and T-block displacement in Push-T, gripper-cube interaction in the manipulation task. Fine textures degrade, but the model successfully captures the underlying dynamics.

H Dataset Descriptions

- **PushT**: a continuous manipulation task where a pusher must push a T block to a target position and orientation. Actions are 2-D pusher displacements. We use 20,000 expert demonstrations following Zhou et al. (2025).
- **Reacher**: a 2-link planar arm must reach a target joint configuration (Tassa et al., 2018). Actions are joint-position deltas. The dataset contains 2,000 trajectories from an SAC policy that drives the arm to sampled joint targets.
- **OGBench Cube-Single/Double/Triple** Park et al. (2025): a simulated manipulation task where a robot arm must move one, two or three cubes to their goal positions; the number

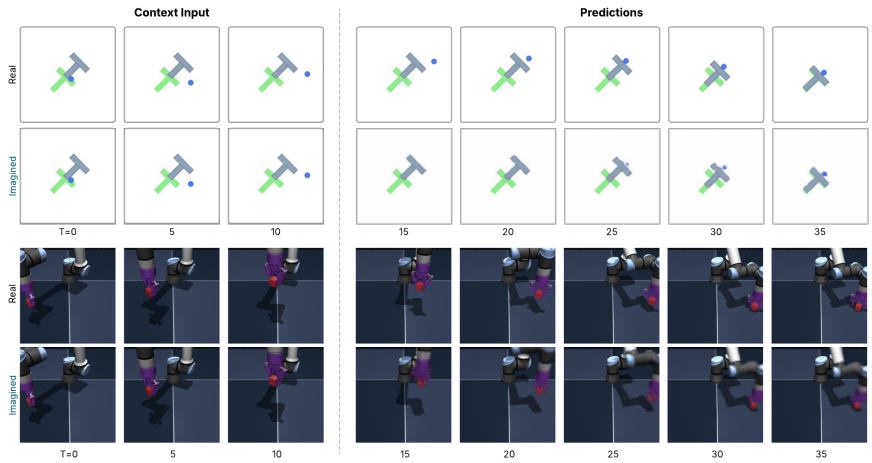

Figure 4: **Open-loop latent rollouts for PushT and OGBench-Cube.** For each domain we show the *real* trajectory (rows 1, 3) and the model's *imagined* rollout decoded to pixels (rows 2, 4). The encoder receives three real context frames ($T{=}0, 5, 10$) and the predictor autoregressively rolls forward for five macro-steps ($T{=}15, 20, 25, 30, 35$), with no real observations past $T{=}10$. A frozen pixel decoder is used *only for visualization*; it is not part of the dynamics model or the planning loop.

of cubes sets the compositional difficulty. Actions are continuous end-effector commands. Each dataset contains $10{,}000$ trajectories collected by OGBench's oracle expert.