ONLYACT: Extracting Action Representations from Foundation Models for World Models

Runjia Li **Minghao Chen** **Junyu Xie** **Philip Torr** **Andrea Vedaldi** **Tomas Jakab**
University of Oxford
runjiali-rl.github.io/onlyact/

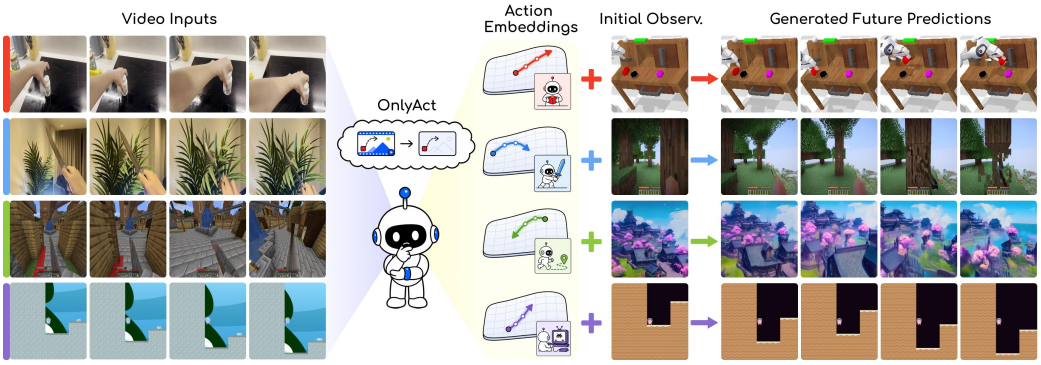

Figure 1: **OnlyAct: an unlabeled action representation learning framework** builds on a pretrained foundation model to extract *expressive* and *transferable* action representations, keeping only the action while discarding the scene in which it occurs. Actions from diverse domains are mapped into a single shared action space, inheriting broad spatio-temporal priors from large-scale video pretraining. These action embeddings can be derived from video frames alone and used to generate videos with the same actions from different initial images across diverse domains. Real actions can also be adaptively mapped into the same action space for real-world control.

Abstract

Video world models promise general-purpose interactive simulators, but their training is limited by action supervision: action labels are scarce, fragmented across incompatible specifications, and do not scale with the unlabeled video pretraining corpus. A common workaround jointly trains an action encoder to compress consecutive frames into a latent action and a decoder to reconstruct the next frame from the previous frame and that latent, forcing the latent to capture only new information. This has two limitations: learning the action space from scratch can limit generalization to unseen domains, and the bottleneck trades expressiveness for separation, with tighter settings losing nuance and looser ones risking scene leakage under domain shifts. We argue for a different starting point: rather than learning an action representation from scratch, we start from a pretrained foundation model (V-JEPA 2), whose embeddings of a frame pair already carry substantial action information after large-scale, diverse video pretraining. This provides a more generalizable foundation for action representation. The remaining challenge is to extract the action while discarding the scene. To this end, we propose a label-free contrastive adaptation framework that pulls together frame pairs sharing the same motion under a different scene and pushes apart pairs from the same scene with different motion, reshaping the geometry of the pretrained space without imposing a reconstruction bottleneck. We introduce a unified evaluation protocol that probes action representations along three axes—action expressiveness,

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

scene invariance, and action-conditioned video generation quality—across robot manipulation, camera navigation, game, and driving benchmarks, most of which are unseen during adaptation. Compared with prior annotation-free latent-action methods, our representations are substantially more expressive while matching their scene invariance, and they yield world models that transfer demonstrated actions to new scenes more faithfully.

1 Introduction

Over the past few years, video generative models [10, 21, 34, 58] have reached unprecedented levels of realism. They are trained on vast quantities of video data and acquire broad priors over scene structure and dynamics, which has motivated their use well beyond video generation itself. A growing line of work explores how they can be turned into video world models by conditioning generation on action signals [11, 46, 20, 19]. Video world models have the potential to serve as general-purpose interactive simulators, with applications ranging from robotics to gaming.

Realizing this potential, however, hinges on action supervision. While unlabeled video is abundant, only a small fraction is paired with the actions that produced it, and the labels that do exist are fragmented across incompatible specifications, such as joint angles in robotics or controller inputs in games. Direct action conditioning therefore does not scale with the unlabeled video pretraining corpus and does not transfer well across environments. To overcome this, latent-action models [43, 11] extract latent actions from unlabeled video in a self-supervised manner and use them to condition the video generator during pretraining. Real actions can then be mapped onto this latent space afterward with a small amount of labeled data [11, 19].

Latent-action models are typically trained from scratch as inverse-dynamics models: given a pair of consecutive frames, an action encoder infers a latent intended to capture the action, while a decoder reconstructs the next frame from the previous frame together with this latent. A bottleneck on the latent forces it to encode only what the decoder cannot recover from the previous frame alone, ideally isolating the action signal from scene content. This comes with two limitations: first, training the action space from scratch can prevent it from generalizing to unseen domains; second, tightening the bottleneck improves separation between action and scene at the cost of expressiveness, while loosening it preserves nuance at the cost of scene leakage, which can also hurt generalization when scene, embodiment, or dynamics shift.

In this work, we propose OnlyAct, an unlabeled action representation learning framework that takes a different starting point. Rather than learning the action representation from scratch, we start from a pretrained foundation model, V-JEPA 2 [4], and extract actions from it with contrastive learning. Pretrained on large-scale, diverse video, such a model encodes both what is in a scene and how it changes over time, so the action-relevant signal is already present in its embeddings, alongside scene content. We confirm this empirically: when a pair of consecutive frames is encoded as a two-frame clip, even the frozen V-JEPA 2 embedding carries more action information than latent-action models trained on the same data, as measured by a linear probe (Table 1). However, it also carries more scene information (Table 2), and its probing performance is unstable across domains. The main challenge is therefore not to learn an action representation from scratch, but to disentangle the action-relevant signal from scene content.

We do this by adapting the pretrained foundation model into an action encoder on unlabeled video with a contrastive objective and a lightweight low-rank adapter. Given two frames from a video, we form a positive example that keeps the motion but changes the scene: when the camera is static, we use optical flow to cut out the moving region and paste it, at a shifted location, onto the background of a different scene. Hard negatives are drawn from other frame pairs of the same video, where action provides the dominant axis of variation. Together, these signals shift the encoder embedding space toward action. Crucially, this is achieved without a reconstruction bottleneck: the objective reshapes the geometry of the pretrained space rather than compressing the action into a small code from which the next frame must be decoded.

The adapted encoder supports two main uses: transfer from demonstrations, and direct control with real actions. In the first, the encoder extracts a latent action from a pair of demonstration frames; this latent then conditions a video world model to reproduce the same action in a different scene. In

the second, a small action-labeled dataset can be used to map an environment's real actions onto the latent space, after which the video world model can be controlled directly through them.

Both uses require an action representation that captures rich and discriminative action dynamics, isolates the action signal from scene content so the same action transfers across scenes, and drives faithful action-conditioned video generation across different domains. We construct a unified evaluation protocol around these axes and benchmark against prior annotation-free latent-action methods and the frozen pretrained encoder on thirteen robot, camera-navigation, and game datasets, ten of which are unseen during adaptation, as well as on world-model training and fine-tuning across robot, navigation, game, and driving domains. Our representation is the most expressive on almost all datasets, matches the scene invariance of bottlenecked latent-action models, and yields world models with markedly better action transfer and controllability.

Our main contributions are as follows:

1. We introduce a new perspective on unlabeled action representation learning: rather than learning actions from scratch, we extract them from the representation space of a pretrained foundation model.
2. We propose a label-free contrastive adaptation framework, built on flow-guided scene-swapped positives and same-video hard negatives, that disentangles action from scene without a reconstruction bottleneck and improves both action expressiveness and scene invariance over the frozen encoder.
3. We introduce a unified evaluation protocol that systematically probes action expressiveness, scene invariance, and action-conditioned video generation quality, and show across robot, navigation, game, and driving benchmarks that our representations improve over prior annotation-free latent-action baselines.

Code and models will be released on our project page: runjiali-rl.github.io/onlyact/.

2 Related Work

Video World Models. Action-conditioned video world models have become a core approach for modeling environment dynamics in domains such as gaming and robotics. Existing methods typically obtain action information in one of two ways. The first assumes explicit action or intervention supervision, such as controller inputs, robot states, camera trajectories, or action prompts [3, 63, 1, 56, 67, 23, 53, 27, 46, 20, 40, 36, 54, 61, 47]. This provides a precise control interface, but the supervision is tied to particular environments, embodiments, or data-collection pipelines, and therefore does not naturally scale to uncurated in-the-wild video. The second introduces latent-action interfaces learned from video [11, 19, 18, 64, 49, 12], reducing the need for explicit labels but requiring a separate action-learning process. Current interactive video world models therefore face a trade-off between precise but environment-specific action supervision and scalable but separately learned latent actions. Our work is complementary: we use pretrained foundation-model representations as an effective starting point and ask whether they already contain transferable transition structure that can be extracted for world model pretraining.

Latent Action Models from Video. A growing line of work learns latent actions from unlabeled video by jointly training an action encoder on frame pairs together with a future-frame decoder, with a bottleneck on the latent. Introduced by earlier work [16, 43, 42] and scaled by recent methods [11, 51, 64, 19, 18, 49, 12], these latents serve as proxy actions for controllable video generation or policy learning. The key limitation is generalization: because these encoders are trained from scratch with reconstruction objectives, the learned action space is shaped by the scenes, embodiments, and dynamics present in the training data. As a result, latents can encode scene-specific cues such as appearance, object identity, or viewpoint rather than the action itself, making transfer difficult when scene or embodiment shifts [15, 66]. Prior methods reduce this leakage with low-dimensional bottlenecks, discrete codebooks, or other architectural constraints, but these constraints come at the cost of reducing the resolution of the action space. A separate line of work augments reconstruction-based latent-action models with contrastive objectives, but relies on additional supervision such as action labels or kinematic trajectories [15, 66]. We instead start from a pretrained foundation model [4] and use a contrastive objective to extract action as a signal already present in its embeddings, neither learning the action space from scratch nor requiring additional supervision.

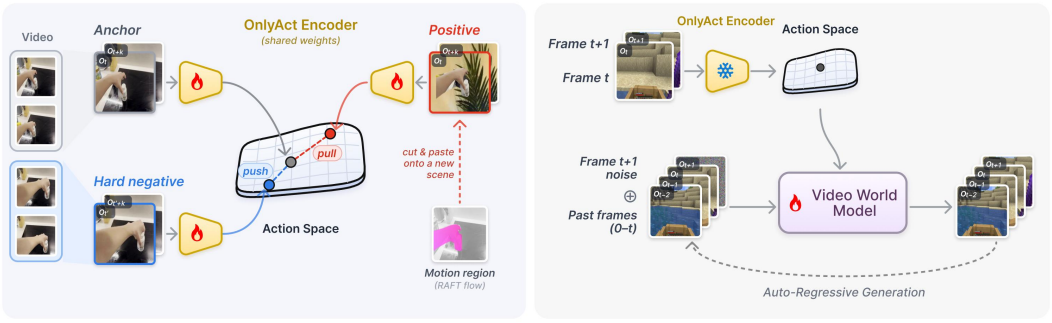

Figure 2: **Overview of the proposed learning pipeline. Left: Contrastive action adaptation.** An anchor frame pair (o_t, o_{t+k}) is encoded as a two-frame clip by the OnlyAct encoder ϕ_θ, a pretrained foundation model with low-rank adapters whose weights are shared across all branches. The positive keeps the action but changes the scene: the moving region of the anchor, found by optical flow, is cut out and pasted at a shifted location onto the background of a different scene (for moving cameras, a different photometric augmentation of the anchor is used instead). The hard negative is drawn from a non-overlapping window $(o_{t'}, o_{t'+k})$ of the same video, sharing the scene but corresponding to a different action. In the action space, the anchor is pulled towards its positive and pushed away from its hard negative. **Right: Action-conditioned video world model.** The frozen OnlyAct encoder ϕ_θ maps consecutive frame pairs into the learned action space, yielding an action stream $\alpha_{1:t}$ that conditions an autoregressive video world model G_ψ. Given past frames $o_{1:t}$ and the action stream, G_ψ predicts the next frame o_{t+1}, and is rolled out autoregressively to generate subsequent frames.

Pretrained Representations. Pretrained visual representations [44, 38, 60, 39, 30, 68, 32, 9] provide strong and transferable features for downstream perception, control, and world modeling. Large-scale foundation models such as V-JEPA 2 [4] extend this to predictive spatio-temporal representations that support video understanding, and V-JEPA 2-AC further post-trains an action-conditioned predictor in this representation space for planning with robot actions. These works use pretrained representations to encode observations or predict future states, with actions, when needed, supplied as labels. We instead use the pretrained foundation model itself as an *action* encoder: encoding a frame pair as a short clip exposes the motion between the frames, and we adapt the encoder without labels so that this motion signal, rather than scene content, dominates its embedding space.

3 Method

Given a corpus of videos without action labels, each consisting of frames $(o_1, \ldots, o_T)$, our goal is to learn a latent action α_t for each consecutive frame pair (o_t, o_{t+1}) and use these latent actions to pretrain an action-conditioned video world model G_ψ that predicts the next frame o_{t+1} from $(o_{1:t}, \alpha_{1:t})$. Our pipeline has two stages: we first adapt a pretrained foundation model ϕ_θ on unlabeled video so that $\alpha_t = \phi_\theta(o_t, o_{t+1})$ isolates the action signal from scene content, and then freeze ϕ_θ and use the resulting action stream $\alpha_{1:T-1}$ to train an auto-regressive video world model G_ψ. Following prior latent-action work [16, 11, 51, 64, 19], we define the action as the change between two frames that is caused by the agent or the camera, and treat what can be inferred from a single frame, such as appearance, layout, and object identity, as scene content. An overview of the training pipeline is provided in Fig. 2. Once trained, G_ψ supports two control modes: *action transfer* and *real-action control*. Sec. 3.1 introduces the foundation-model-based action encoder, Sec. 3.2 details its contrastive adaptation, and Sec. 3.3 describes the action-conditioned video world model.

3.1 Frame-Pair Embeddings from a Foundation Model

We instantiate the action encoder ϕ_θ with the encoder of V-JEPA 2 [4], a ViT-L foundation model pretrained on large-scale video. Because it is pretrained to model how video content evolves across space and time, the encoder represents motion, not only appearance, which makes it a natural starting point from which to extract action representations.

Pair encoding. Given a frame pair (o_t, o_{t+1}), we stack the two frames into a two-frame clip and pass it through the encoder only; the V-JEPA 2 predictor is not used. Since the encoder groups every two frames into one spatio-temporal tubelet, the clip yields a single temporal slice of 16×16 patch tokens, and each token jointly observes both frames from the first layer onward. Motion is therefore fused into the representation from the start, rather than obtained by differencing two independently encoded frames. We average-pool the output tokens, apply a fixed linear projection to d=640 dimensions, and ℓ_2-normalize,

$$\alpha_t \;=\; \phi_\theta(o_t, o_{t+1}) \;=\; \frac{P\,\bar{h}_t}{\|P\,\bar{h}_t\|_2}, \qquad \bar{h}_t \;=\; \frac{1}{N}\sum_{n=1}^{N} h_{t,n},$$

where $h_{t,n}$ is the n-th output token and $P \in \mathbb{R}^{640\times 1024}$ is a random projection that is fixed at initialization and never trained. A random linear projection approximately preserves the geometry of the pooled embedding, so P serves only to reduce dimensionality; all learning happens in the encoder.

The frozen embedding already encodes substantial action information (Table 1), but it is entangled with scene content: it leaks more scene information than latent-action models (Table 2), and its action information degrades sharply on some domains. A low-dimensional reconstruction bottleneck is the standard remedy for scene leakage, but it trades *scene invariance against action expressiveness* and would discard the capacity that makes the pretrained space attractive in the first place. We therefore adapt ϕ_θ with a contrastive objective that reshapes its geometry without compressing it (Sec. 3.2).

3.2 Contrastive Action Adaptation

Building on the encoder ϕ_θ of Sec. 3.1, we now reshape its embedding space to better isolate action from scene while retaining the capacity of the pretrained representation. We frame this adaptation around two properties that we want the action embedding space to satisfy: *action expressiveness* and *scene invariance*. Without action labels, neither can be enforced directly, so we approximate both through a contrastive objective on positive and hard-negative pairs that are constructed from raw video. To avoid overwriting the general-purpose features acquired during pretraining, we keep the pretrained weights θ frozen and attach low-rank adapters [26] to the linear layers of every transformer block, yielding an update $\Delta\theta$ and the adapted encoder $\phi_{\theta+\Delta\theta}$. With slight abuse of notation, we write ϕ_θ for this adapted encoder in the remainder of the paper, with the understanding that θ subsumes $\Delta\theta$.

Pair construction. For each training sample we draw an anchor pair, a positive pair, and a hard-negative pair. The *anchor* is a frame pair (o_t, o_{t+k}) from a video, with a small random offset k. The *positive* $g(o_t, o_{t+k})$ should carry the same action in a different scene. We construct it with *flow-guided scene swapping*: we estimate the dense optical flow between the two frames with an off-the-shelf estimator [55], and threshold its magnitude to obtain the moving core region, i.e., the pixels that carry the action. We cut out this region from both frames and paste it onto the same background frame taken from a different scene of the same dataset, translating it in both frames by a common random offset. The pixels in motion are thus preserved exactly, while the scene, background, and absolute location of the motion are all replaced. This construction is only valid when the camera is static. When the camera moves, as in egocentric, hand-held, or driving video, the background motion is itself part of the action and cannot be swapped; in this case the positive falls back to the anchor pair under a different photometric augmentation. Both anchor and positive receive a color jitter that is shared between their two frames, so that inter-frame changes are not altered. The *hard negative* $(o_{t'}, o_{t'+k})$ is drawn from a non-overlapping time window of the same video: it shares the scene with the anchor but likely corresponds to a different action, so it cannot be told apart from the anchor by appearance. We write α_t, $\alpha_t^+ = \phi_\theta\big(g(o_t, o_{t+k})\big)$, and $\alpha_t^- = \phi_\theta(o_{t'}, o_{t'+k})$ for the embeddings of the anchor, positive, and hard negative, respectively.

Contrastive objective. We organize these triplets into a candidate bank $\mathcal{Z}$ containing every anchor, positive, and hard negative in a batch of size N, and apply a symmetric InfoNCE objective [57] that pulls each anchor toward its matched positive and pushes it away from every other element of the

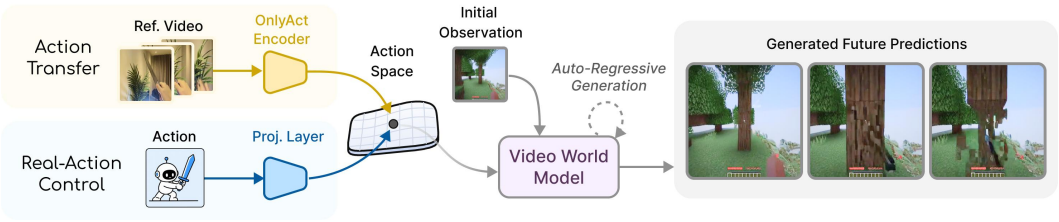

Figure 3: **Two control modes.** After pretraining the video world model on latent actions from the OnlyAct encoder, the learned action space supports two modes. First, *action transfer* extracts an action stream from a demonstration video and uses it to condition the world model from a new initial frame, reproducing the demonstrated action with no labels. Second, *real-action control* learns a lightweight interface from real actions onto the learned latent-action space on a small action-labeled set, so that real actions drive the world model directly.

bank in both directions,

$$\mathcal{L}_{\text{con}} = -\frac{1}{2N}\sum_{i=1}^{N}\left[\log\frac{\exp\left(\alpha_i^\top \alpha_i^+/\tau\right)}{\sum_{z\in\mathcal{Z}\setminus\{\alpha_i\}}\exp\left(\alpha_i^\top z/\tau\right)} + \log\frac{\exp\left(\alpha_i^{+\top}\alpha_i/\tau\right)}{\sum_{z\in\mathcal{Z}\setminus\{\alpha_i^+\}}\exp\left(\alpha_i^{+\top} z/\tau\right)}\right],$$

where τ is a temperature. The same-video hard negatives in $\mathcal{Z}$ make the objective non-trivial: in contrast, random in-batch negatives from unrelated scenes would be easily separated by appearance alone, allowing the encoder to satisfy the loss without disentangling action from scene. The scene-swapped positives, in turn, prevent the encoder from separating the anchor from its hard negative by memorizing scene-specific cues, since the positive shares none of the anchor's background.

Inference. After adaptation, a latent action α_t is obtained by a single forward pass of ϕ_θ on a frame pair. No labels, prompts, or trajectory-level optimization are required at inference. The adapted encoder ϕ_θ thus serves as a drop-in action encoder on arbitrary video, yielding the action stream $\alpha_{1:t}$ that conditions the video world model G_ψ (Sec. 3.3).

3.3 Action-Conditioned Video World Model

We obtain the action-conditioned video world model G_ψ by adopting the autoregressive world model architecture of AdaWorld [19] and replacing its latent actions by our action stream. Given a trajectory $(o_1, \ldots, o_T)$, we apply the frozen action encoder of Sec. 3.1 to consecutive frame pairs to form the *action stream* $\alpha_{1:T-1}$ and use it as continuous action conditioning, with G_ψ predicting each next frame o_{t+1} from $(o_{1:t}, \alpha_{1:t})$. Before entering G_ψ, our action embeddings are whitened by a fixed, parameter-free covariance equalization computed on the training data, while the baselines' latent actions enter unchanged; whitening a baseline in the same way leads to the same conclusions (Appendix A.2). Otherwise, every action representation we compare uses exactly the same backbone, recipe, and data.

Action injection and training objective. Following AdaWorld, the action representation conditions the denoiser through two pathways: it is concatenated with the CLIP image embedding to form the cross-attention context, and concatenated with the context-length embedding into the vector condition, which a label-embedding MLP maps and adds to the timestep embedding. One action vector is given per predicted transition, with no learned projection in between, and the action slots of the pretrained weights are zero-initialized. G_ψ predicts the next frame from up to six context frames, with the context length sampled uniformly from 1 to 6 during training, and is trained with the v-prediction objective under EDM noise scaling and loss weighting [31]. Each condition, including the action, is dropped with probability 0.1 to enable classifier-free guidance [24].

Action transfer and real-action control. As illustrated in Fig. 3, the trained pipeline supports two control modes that share the same autoregressive rollout of the video world model G_ψ but differ in how the conditioning action stream $\alpha_{1:T-1}$ is obtained. For *action transfer*, we extract $\alpha^{\text{demo}}_{1:T-1}$ by running the frozen action encoder ϕ_θ over a demonstration video and feed it to G_ψ together with an initial frame o_1 from a different scene. The model then reproduces the demonstrated action in the

Table 1: **Linear probing of action information.** A linear layer predicts the ground-truth action from each frozen representation. We report test R^2 on held-out scenes (LIBERO, RLBench: held-out demonstrations of training scenes; Procgen: macro-F1 over 15 discrete actions), averaged over 3 probe seeds (std ≤ 0.03). MetaWorld actions are clipped to $[-1, 1]$. °Videos seen by our encoder during training, without action labels; all other datasets are entirely unseen. †Retrained on the same 168K training pairs as our encoder. V-JEPA 2 (frozen): our backbone, embedded identically but without contrastive adaptation. Avg.: mean R^2 over the 12 continuous-action datasets. Δ: ours minus the best other method. **Bold**: best; underline: second best.

	Robot manipulation									*Camera navigation*			*Game (F1)*	
Representation	LIBERO°	DROID°	RLBench	RoboCasa	DROID-IPRL	MetaWorld	MUTEX	Robosuite	RoboDesk	SpatialVID°	ARKitScenes	RE10K	Procgen	Avg.
Random	−0.032	−0.112	−0.021	−0.100	−0.094	−0.062	−0.156	−0.093	−0.072	−0.092	−0.090	−0.074	0.073	−0.083
AdaWorld† [19]	0.199	0.011	0.136	−0.154	0.009	−0.045	0.191	0.121	0.170	0.108	0.006	0.046	0.063	0.067
DreamDojo† [18]	0.324	0.071	0.246	0.034	0.162	0.156	0.425	0.319	0.268	0.266	0.087	0.173	0.092	0.211
V-JEPA 2 (frozen) [4]	**0.718**	−0.131	0.556	−0.368	0.186	0.144	0.527	0.448	0.520	0.301	**0.665**	0.598	0.145	0.347
OnlyAct (ours)	0.634	**0.203**	**0.566**	**0.322**	**0.279**	**0.277**	**0.566**	**0.695**	**0.581**	**0.560**	0.588	**0.609**	**0.200**	**0.490**
Δ	−0.08	+0.13	+0.01	+0.29	+0.09	+0.12	+0.04	+0.25	+0.06	+0.26	−0.08	+0.01	+0.05	+0.14

new scene without any action labels. For *real-action control*, we fine-tune on a small action-labeled set. Following [18], a small learned interface maps real actions into the latent-action space and replaces ϕ_θ at the input of G_ψ: one vector per discrete action, initialized at the mean latent action of its labeled clips, or, for continuous actions, a two-layer MLP first fitted to regress the latent actions of the labeled clips. The interface and the denoiser of G_ψ are then fine-tuned end-to-end (Appendix B.5). Because the latent-action space has already been shaped by large-scale unlabeled pretraining, only a modest amount of labeled data is required, and at inference time the mapped real actions serve as the conditioning stream.

4 Experiments

We evaluate an action representation for action-conditioned video world models along three axes: (i) **action expressiveness** (Sec. 4.2) — how well the embedding distinguishes different actions; (ii) **scene invariance** (Sec. 4.3) — whether the embedding avoids encoding scene or future-state content that a video generator could exploit as a shortcut; and (iii) **action-conditioned video generation quality** (Sec. 4.4) — how well the embedding serves as a conditioning signal for video generation, measured by fidelity, transferability, and controllability. On both seen and unseen benchmarks, OnlyAct is substantially more expressive than prior annotation-free latent-action baselines, matches their scene invariance, and yields world models with better action transfer and controllability. Implementation details are provided in Appendix B.1.

4.1 Datasets and Baselines

We evaluate OnlyAct on datasets both seen and unseen during encoder adaptation. The action encoder is adapted on 168K unlabeled frame-pair samples from nine video sources spanning robot manipulation, human and egocentric video, camera navigation, and games (Appendix B.2). Among our evaluation datasets, only LIBERO [37], DROID [33], and SpatialVID [59] contribute videos to this set (marked ° in the tables), and all other datasets are entirely unseen by the encoder. For action probing and scene leakage, we use thirteen datasets with action annotations: nine robot-manipulation datasets (LIBERO, DROID, RLBench [28], RoboCasa [45], DROID-IPRL [33], MetaWorld [65], MUTEX [52], Robosuite [70], and RoboDesk [29]), three camera-navigation datasets (SpatialVID, ARKitScenes [6], and RealEstate10K [69]), and Procgen [14] with 15 discrete actions. For action-conditioned video generation, we evaluate on LIBERO, RLBench, SpatialVID, and nuScenes [13], and further fine-tune with real actions on DMLab [7] and Minecraft from the TECO datasets [62], Habitat [50] in HM3D scenes [48], and nuScenes.

We compare against two families of baselines. The first is latent-action models, namely DreamDojo [18] and AdaWorld [19]. Since their publicly released weights are not trained on our data, we retrain both on the same 168K training pairs as our encoder for a fair comparison (marked †). The second is the frozen V-JEPA 2 encoder [4], embedded in the same way as OnlyAct (Sec. 3.1) but without contrastive adaptation, which isolates the effect of our adaptation from that of the pretrained backbone.

Table 2: **Scene leakage.** A decoder with the same architecture for every representation (trained with L1 + 0.5 × LPIPS, 10 epochs, 128×128) reconstructs the post-action frame from the representation and is evaluated on scenes held out from its training; values are averaged over 4 decoder seeds. Per-dataset columns: held-out LPIPS↑ (higher = less scene information recoverable). Last three columns: LPIPS↑, SSIM↓ and L1↑ averaged over the 13 datasets. A representation that carries no information at all also leaks nothing, so read this table together with Table 1. Markers and baselines as in Table 1. Δ: ours minus the best other method (green: ours leaks less). **Bold**: least leakage; underline: second least.

	Robot manipulation									*Camera navigation*			*Game*	*Average*		
Representation	LIBERO°	DROID°	RLBench	RoboCasa	DROID-IPRL	MetaWorld	MUTEX	Robosuite	RoboDesk	SpatialVID°	ARKitScenes	RE10K	Procgen	LPIPS↑	SSIM↓	L1↑
AdaWorld† [19]	0.288	0.314	0.273	0.363	0.320	0.136	0.211	0.307	**0.127**	0.366	0.405	0.369	0.325	0.293	0.374	0.256
DreamDojo† [18]	0.306	0.315	**0.283**	0.375	0.317	**0.137**	**0.212**	0.308	0.119	0.379	0.405	0.385	0.347	0.299	**0.369**	**0.260**
V-JEPA 2 (frozen) [4]	0.257	0.289	0.216	**0.480**	0.247	0.066	0.130	0.282	0.069	0.402	0.384	**0.442**	0.311	0.275	0.478	0.200
OnlyAct (ours)	**0.314**	**0.328**	0.269	0.411	**0.322**	0.105	0.177	**0.317**	0.100	**0.407**	**0.411**	0.436	**0.352**	**0.304**	0.391	0.255
Δ	+0.008	+0.014	−0.014	−0.069	+0.002	−0.032	−0.035	+0.009	−0.027	+0.005	+0.005	−0.006	+0.005	+0.005	+0.021	−0.004

4.2 Action Expressiveness

To evaluate whether the representation captures rich and discriminative action dynamics, we conduct action probing experiments. Specifically, we train a linear probe to predict ground-truth actions from each representation, and report the results in Table 1 (details in Appendix B.3).

As shown in Table 1, OnlyAct captures more expressive and discriminative action dynamics than the baselines on 11 of the 13 benchmarks, raising the average R^2 from 0.211 for the strongest latent-action model to 0.490. The frozen V-JEPA 2 encoder already outperforms the latent-action models on average, confirming that the pretrained foundation model provides a strong starting point for action representation. However, its action information is unreliable across domains, and on DROID and RoboCasa its R^2 even falls below that of the latent-action models. Contrastive adaptation removes these failure cases and improves the average R^2 from 0.347 to 0.490, with large gains also on datasets unseen during adaptation (e.g., +0.69 on RoboCasa and +0.25 on Robosuite). We attribute this improved transferability to the broad spatio-temporal priors of the pretrained foundation model, which the adaptation preserves rather than overwrites.

High action prediction accuracy alone, however, does not guarantee a representation suitable for action-conditioned video generation, as we analyze next.

4.3 Scene Invariance

Since action embeddings are inferred from consecutive frames, they may inadvertently encode future-state information that a video model can exploit as a shortcut, directly reconstructing future content rather than learning genuine action-conditioned dynamics and thereby hurting both controllability and transferability.

To quantify how much scene information is encoded in the action embeddings, we train a decoder to reconstruct the future frame from the action embedding alone and evaluate it on held-out scenes. A more scene-invariant embedding should yield worse reconstruction. The results in Table 2 show that the frozen V-JEPA 2 embedding leaks the most scene information (average LPIPS 0.275). With contrastive adaptation, however, OnlyAct matches the scene invariance of latent-action models, attaining the highest average LPIPS (0.304) and comparable SSIM and L1, while retaining a substantially more expressive action representation. The adaptation therefore improves expressiveness and scene invariance together, without relying on a bottleneck. The right side of Fig. 4 further shows that OnlyAct embeddings form meaningful clusters for the same actions across different datasets. The decoder structure is described in Appendix B.4.

4.4 Action-Conditioned Video Generation Quality

The ultimate goal of the learned action representation is to provide a general conditioning signal for action-conditioned video generation. We hypothesize that embeddings that are both expressive and scene-invariant enable a video model to learn action-conditioned dynamics more effectively, generalize across domains, and transfer to downstream tasks after lightweight fine-tuning with real actions.

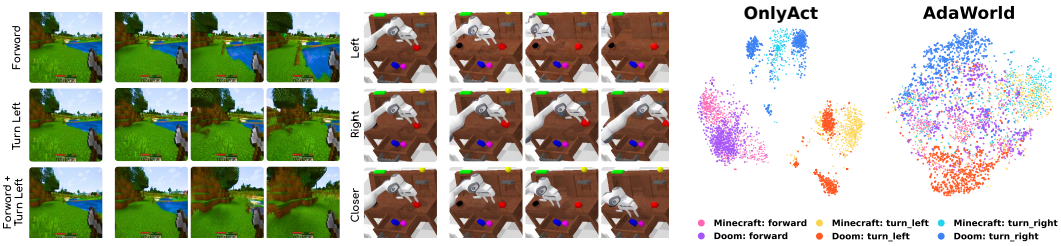

Figure 4: **Real-action control**. **Left:** OnlyAct learns to generate videos that faithfully respond to different actions, **Right:** comparison of action representations between OnlyAct and AdaWorld using t-SNE.



Figure 5: **Action transfer.** Latent actions extracted from the demonstration frames (top) condition the video world model generations (bottom). OnlyAct faithfully reproduces the demonstrated action in the new target scene, e.g. in the cross-source example it turns right and then performs the attack action. Best seen as a video in the supplementary material.

To evaluate this hypothesis, we train the same action-conditioned autoregressive world model (Sec. 3.3) with each action representation as input, using the same backbone, recipe, and data for all of them. We assess generation quality along three sub-axes: *fidelity*, *transferability*, and *controllability*. Fidelity is measured by the PSNR of generations driven by the actions inferred from the clip itself (Self PSNR). Transferability is measured by driving a different scene with the actions of a source clip and computing the cosine between the optical-flow direction [17] of the generated video and that of the source (Transfer Dir), where 0 means unrelated and 1 means the same direction. Since a representation that leaks scene content can reach a high Self PSNR by reconstructing the future frame but cannot carry its action to a new scene, Transfer Dir is the more direct test of an action representation.

The results in Table 3 show that OnlyAct achieves the best Transfer Dir on all five evaluation settings, improving over DreamDojo by $+0.03$ to $+0.13$, including on RLBench and nuScenes, which are unseen by our encoder, while AdaWorld's latent actions stay close to or below the no-action reference. Fidelity is comparable across methods: OnlyAct has the highest Self PSNR on LIBERO and RLBench and is within 0.31 dB of the best method on SpatialVID and nuScenes. In addition, Fig. 5 and further video results in the supplementary material show that the action-transfer control mode (Sec. 3.3) faithfully reproduces a demonstrated action in a new target scene, including under cross-source transfer and from in-the-wild demonstrations.

Controllability is evaluated by fine-tuning each pretrained world model on a small action-labeled set of a target domain (real-action control, Sec. 3.3; details in Appendix B.5). We report Follow, the accuracy of an inverse-dynamics classifier at recognizing the commanded action in the generated frame (on nuScenes, the Spearman correlation between commanded and generated forward motion), and Control, the PSNR gap between generations under the commanded action and under other actions. As shown in Table 4, OnlyAct yields the most controllable world model on all four domains, with the largest gain on nuScenes (Follow 0.928 vs. 0.722); on Habitat, the Control gain is within noise. OnlyAct also reaches the highest fidelity early in fine-tuning on DMLab and Habitat, while after 800 updates fidelity is on par across representations. This indicates that the action space shaped by unlabeled adaptation transfers effectively to real-action fine-tuning, as further illustrated qualitatively on the left of Fig. 4.

Table 3: **Action-conditioned world model.** The same world model (AdaWorld backbone, same recipe and data) is trained with each action representation. *Self*: PSNR↑ (dB) when driven by the clip's own actions. *Transfer*: a source clip's actions drive a different scene; Dir↑ is the cosine between the optical-flow direction of the generated video and of the source (0: unrelated; 1: same). °Seen by our encoder without action labels. †Retrained on our data. Δ: ours minus the best baseline. **Bold**: best; underline: second best (no-action row excluded).

	Robot manipulation				*Camera navigation*		*Driving*			
	LIBERO°		RLBench		SpatialVID°		nuScenes dev.		nuScenes conf.	
Action representation	Self PSNR↑	Transfer Dir↑	Self PSNR↑	Transfer Dir↑	Self PSNR↑	Transfer Dir↑	Self PSNR↑	Transfer Dir↑	Self PSNR↑	Transfer Dir↑
No action	18.41	0.186	18.51	−0.054	15.42	−0.188	18.60	−0.092	19.10	0.398
AdaWorld† [19]	18.90	0.162	18.32	−0.109	15.52	−0.274	18.44	−0.142	19.46	0.432
DreamDojo† [18]	19.95	0.803	18.08	0.572	**16.40**	0.480	**18.57**	0.574	**19.67**	0.571
OnlyAct (ours)	**20.32**	**0.917**	**18.63**	**0.686**	16.10	**0.608**	18.56	**0.659**	19.38	**0.603**
Δ	+0.37	+0.114	+0.31	+0.113	−0.31	+0.128	−0.01	+0.085	−0.29	+0.032

Table 4: **Downstream world-model fine-tuning: controllability and fidelity.** Each pretrained world model from Table 3 is fine-tuned on a small labeled set of a target domain (all unseen by our encoder, which was trained on VPT Minecraft recordings, not these TECO videos). **(a) Controllability** after 800 updates. Follow↑: accuracy of an inverse-dynamics classifier at recognizing the commanded action in the generated frame (chance 0.333; *Real frames*: the same classifier on real frames); on nuScenes, Spearman correlation between commanded and generated forward motion (chance 0). Control↑: PSNR under the commanded action minus under other actions (dB; 0 = action ignored). **(b) Fidelity**: PSNR↑ (dB) after 0, 200 and 800 updates (0: interface initialized from the labeled clips, no gradient update). Markers as in Table 3. Δ: ours minus the best baseline; grey when the 95% paired bootstrap interval contains 0. **Bold**: best; underline: second best.

	DMLab		Minecraft		Habitat		nuScenes	
(a) Controllability after 800 updates								
Action representation	Follow↑	Control↑	Follow↑	Control↑	Follow↑	Control↑	Follow↑	Control↑
Real frames	0.997	–	0.900	–	0.787	–	–	–
AdaWorld† [19]	0.508	4.04	0.647	6.88	0.527	3.04	0.628	0.04
DreamDojo† [18]	0.863	7.99	0.787	7.59	0.749	7.33	0.722	0.15
OnlyAct (ours)	**0.901**	**8.95**	**0.821**	**7.81**	**0.767**	**7.35**	**0.928**	**0.64**
Δ	+0.038	+0.96	+0.034	+0.22	+0.018	+0.02	+0.207	+0.49

	DMLab			Minecraft			Habitat			nuScenes		
(b) Fidelity, PSNR↑ (dB) after 0 / 200 / 800 updates												
Action representation	0	200	800	0	200	800	0	200	800	0	200	800
AdaWorld† [19]	18.77	21.35	22.34	**23.07**	23.36	23.46	18.44	18.95	20.99	**24.59**	**24.85**	**24.77**
DreamDojo† [18]	19.14	22.56	23.08	22.80	23.48	23.49	19.24	23.00	**23.78**	24.53	24.83	**24.77**
OnlyAct (ours)	**19.28**	**22.82**	**23.11**	22.57	**23.59**	**23.54**	**20.56**	**23.27**	23.74	**24.59**	24.83	24.76
Δ	+0.14	+0.26	+0.03	−0.50	+0.11	+0.05	+1.31	+0.27	−0.04	0.00	−0.02	0.00

5 Conclusion

We presented OnlyAct, a scalable approach to action representation learning for video world models. Previous latent-action models typically learn action spaces from scratch using unlabeled videos, often relying on restrictive bottlenecks to disentangle actions from scene content. In contrast, OnlyAct adapts a pretrained foundation model into an action encoder and uses a label-free contrastive objective, with flow-guided scene-swapped positives and same-video hard negatives, to steer its embeddings towards action-centric representations, without imposing such a bottleneck. We show that the resulting action representations are substantially more expressive than those learned by prior annotation-free latent-action methods while matching their scene invariance, and that they enable world models with better action transfer and controllability on new domains.

Acknowledgments and Disclosure of Funding

This work was supported by the Clarendon Scholarship at the University of Oxford. The authors declare no competing interests.

References

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

[2] AgiBot-World-Contributors, Qingwen Bu, Jisong Cai, Li Chen, et al. AgiBot World Colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

[3] Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. *Advances in Neural Information Processing Systems*, 37:58757–58791, 2024.

[4] 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.

[5] Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.

[6] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, et al. ARKitScenes: A diverse real-world dataset for 3d indoor scene understanding using mobile RGB-D data. In *NeurIPS Datasets and Benchmarks Track*, 2021.

[7] Charles Beattie, Joel Z. Leibo, Denis Teplyashin, et al. DeepMind Lab. *arXiv preprint arXiv:1612.03801*, 2016.

[8] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of artificial intelligence research*, 47:253–279, 2013.

[9] Gabrijel Boduljak, Yushi Lan, Christian Rupprecht, and Andrea Vedaldi. Vfmf: World modeling by forecasting vision foundation model features. *arXiv preprint arXiv:2512.11225*, 2025.

[10] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Wing Yin Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators, 2024.

[11] Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Maria Elisabeth Bechtle, Feryal Behbahani, Stephanie C.Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando De Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *International Conference on Machine Learning (ICML)*, 2024.

[12] Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

[13] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[14] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588*, 2019.

[15] Weisheng Dai, Kai Lan, Jianyi Zhou, Bo Zhao, Xiu Su, Junwen Tong, Weili Guan, and Shuo Yang. Conla: Contrastive latent action learning from human videos for robotic manipulation. *arXiv preprint arXiv:2602.00557*, 2026.

[16] Ashley D. Edwards, Himanshu Sahni, Yannick Schroecker, and Charles L. Isbell. Imitating latent policies from observation. In *International Conference on Machine Learning (ICML)*, 2019.

[17] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian Conference on Image Analysis (SCIA)*, volume 2749 of *Lecture Notes in Computer Science*, pages 363–370. Springer, 2003.

[18] Shenyuan Gao, William Liang, Kaiyuan Zheng, Ayaan Malik, Seonghyeon Ye, Sihyun Yu, Wei-Cheng Tseng, Yuzhu Dong, Kaichun Mo, Chen-Hsuan Lin, Qianli Ma, Seungjun Nah, Loic Magne, Jiannan Xiang, Yuqi Xie, Ruijie Zheng, Dantong Niu, You Liang Tan, K.R. Zentner, George Kurian, Suneel Indupuru, Pooya Jannaty, Jinwei Gu, Jun Zhang, Jitendra Malik, Pieter Abbeel, Ming-Yu Liu, Yuke Zhu, Joel Jang, and Linxi "Jim" Fan. Dreamdojo: A generalist robot world model from large-scale human videos. *arXiv preprint arXiv:2602.06949*, 2026.

[19] Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. Adaworld: Learning adaptable world models with latent actions. In *International Conference on Machine Learning (ICML)*, 2025.

[20] Google DeepMind. Genie 3: A new frontier for world models. `https://deepmind.google/blog/genie-3-a-new-frontier-for-world-models/`, 2025. Blog post.

[21] Google DeepMind. Veo 3. `https://deepmind.google/models/veo/`, 2025.

[22] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, et al. The "something something" video database for learning and evaluating visual common sense. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[23] Xianglong He, Chunli Peng, Zexiang Liu, Boyang Wang, Yifan Zhang, Qi Cui, Fei Kang, Biao Jiang, Mengyin An, Yangyang Ren, Baixin Xu, Hao-Xiang Guo, Kaixiong Gong, Cyrus Wu, Wei Li, Xuchen Song, Yang Liu, Eric Li, and Yahui Zhou. Matrix-game 2.0: An open-source, real-time, and streaming interactive world model. *arXiv preprint arXiv:2508.13009*, 2025.

[24] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[25] Ryan Hoque, Peide Huang, David J. Yoon, Mouli Sivapurapu, and Jian Zhang. EgoDex: Learning dexterous manipulation from large-scale egocentric video. *arXiv preprint arXiv:2505.11709*, 2025.

[26] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[27] Tianyu Huang, Wangguandong Zheng, Tengfei Wang, Yuhao Liu, Zhenwei Wang, Junta Wu, Jie Jiang, Hui Li, Rynson WH Lau, Wangmeng Zuo, and Chunchao Guo. Voyager: Long-range and world-consistent video diffusion for explorable 3d scene generation. *arXiv preprint arXiv:2506.04225*, 2025.

[28] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 2020.

[29] Harini Kannan, Danijar Hafner, Chelsea Finn, and Dumitru Erhan. RoboDesk: A multi-task reinforcement learning benchmark. `https://github.com/google-research/robodesk`, 2021.

[30] Siddharth Karamcheti, Suraj Nair, Annie S Chen, Thomas Kollar, Chelsea Finn, Dorsa Sadigh, and Percy Liang. Language-driven representation learning for robotics. *arXiv preprint arXiv:2302.12766*, 2023.

[31] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[32] Efstathios Karypidis, Ioannis Kakogeorgiou, Spyros Gidaris, and Nikos Komodakis. DINO-foresight: Looking into the future with DINO. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[33] Alexander Khazatsky, Karl Pertsch, Suraj Nair, et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems (RSS)*, 2024.

[34] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

[35] Mingxin Li, Yanzhao Zhang, Dingkun Long, Keqin Chen, Sibo Song, Shuai Bai, Zhibo Yang, Pengjun Xie, An Yang, Dayiheng Liu, et al. Qwen3-vl-embedding and qwen3-vl-reranker: A unified framework for state-of-the-art multimodal retrieval and ranking. *arXiv preprint arXiv:2601.04720*, 2026.

[36] Runjia Li, Philip Torr, Andrea Vedaldi, and Tomas Jakab. Vmem: Consistent interactive video scene generation with surfel-indexed view memory. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 25690–25699, October 2025.

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

[38] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

[39] Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Jason Ma, Claire Chen, Sneha Silwal, Aryan Jain, Vincent-Pierre Berges, Tingfan Wu, Jay Vakil, et al. Where are we in the search for an artificial visual cortex for embodied intelligence? *Advances in Neural Information Processing Systems*, 36:655–677, 2023.

[40] Xiaofeng Mao, Shaoheng Lin, Zhen Li, Chuanhao Li, Wenshuo Peng, Tong He, Jiangmiao Pang, Mingmin Chi, Yu Qiao, and Kaipeng Zhang. Yume: An interactive world generation model. *arXiv preprint arXiv:2507.17744*, 2025.

[41] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7327–7334, 2022.

[42] Willi Menapace, Stéphane Lathuilière, Aliaksandr Siarohin, Christian Theobalt, Sergey Tulyakov, Vladislav Golyanik, and Elisa Ricci. Playable environments: Video manipulation in space and time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[43] Willi Menapace, Stéphane Lathuilière, Sergey Tulyakov, Aliaksandr Siarohin, and Elisa Ricci. Playable video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[44] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. In *Conference on Robot Learning*, pages 892–909. PMLR, 2023.

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

[46] Jack Parker-Holder, Philip Ball, Jake Bruce, Vibhavari Dasagi, Kristian Holsheimer, Christos Kaplanis, Alexandre Moufarek, Guy Scully, Jeremy Shar, Jimmy Shi, et al. Genie 2: A large-scale foundation world model. 2, 2024.

[47] Alexander Pondaven, Ziyi Wu, Igor Gilitschenski, Philip Torr, Sergey Tulyakov, Fabio Pizzati, and Aliaksandr Siarohin. Actionparty: Multi-subject action binding in generative video games. *arXiv preprint arXiv:2604.02330*, 2026.

[48] Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X. Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3D dataset (HM3D): 1000 large-scale 3D environments for embodied AI. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2021.

[49] Nedko Savov, Naser Kazemi, Mohammad Mahdi, Danda Pani Paudel, Xi Wang, and Luc Van Gool. Exploration-driven generative interactive environments. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 27597–27607, June 2025.

[50] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, et al. Habitat: A platform for embodied AI research. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[51] Dominik Schmidt and Minqi Jiang. Learning to act without actions. In *International Conference on Learning Representations (ICLR)*, 2024.

[52] Rutav Shah, Roberto Martín-Martín, and Yuke Zhu. MUTEX: Learning unified policies from multimodal task specifications. In *Conference on Robot Learning (CoRL)*, 2023.

[53] Skywork AI Matrix-Game Team. Matrix-game 3.0: Real-time and streaming interactive world model with long-horizon memory. Technical report, 2026.

[54] Robbyant Team, Zelin Gao, Qiuyu Wang, Yanhong Zeng, Jiapeng Zhu, Ka Leong Cheng, Yixuan Li, Hanlin Wang, Yinghao Xu, Shuailei Ma, Yihang Chen, Jie Liu, Yansong Cheng, Yao Yao, Jiayi Zhu, Yihao Meng, Kecheng Zheng, Qingyan Bai, Jingye Chen, Zehong Shen, Yue Yu, Xing Zhu, Yujun Shen, and Hao Ouyang. Advancing open-source world models. *arXiv preprint arXiv:2601.20540*, 2026.

[55] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.

[56] Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines, 2024.

[57] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2018.

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

[59] Jiahao Wang et al. SpatialVID: A large-scale video dataset with spatial annotations. *arXiv preprint arXiv:2509.09676*, 2025.

[60] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

[61] Zeqi Xiao, Yushi Lan, Yifan Zhou, Wenqi Ouyang, Shuai Yang, Yanhong Zeng, and Xingang Pan. Worldmem: Long-term consistent world simulation with memory, 2025.

[62] Wilson Yan, Danijar Hafner, Stephen James, and Pieter Abbeel. Temporally consistent transformers for video generation. In *International Conference on Machine Learning (ICML)*, 2023.

[63] Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 2023.

[64] Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos, 2024.

[65] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019.

[66] Chubin Zhang, Jianan Wang, Zifeng Gao, Yue Su, Tianru Dai, Cai Zhou, Jiwen Lu, and Yansong Tang. Clap: Contrastive latent action pretraining for learning vision-language-action models from human videos. *arXiv preprint arXiv:2601.04061*, 2025.

[67] Yifan Zhang, Chunli Peng, Boyang Wang, Puyi Wang, Qingcheng Zhu, Fei Kang, Biao Jiang, Zedong Gao, Eric Li, Yang Liu, and Yahui Zhou. Matrix-game: Interactive world foundation model. *arXiv preprint arXiv:2506.18701*, 2025.

[68] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning, 2024.

[69] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

[70] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, et al. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

Appendix

Qualitative samples are shown in the local HTML in the supplementary materials.

A Additional Experiments **15**

A.1 Action-Transfer Videos 15

A.2 Whitening Control 15

A.3 Multimodal Embedding Backbone 15

B Experiment Details **17**

B.1 Implementation Details 17

B.2 Contrastive Learning Details 18

B.3 Action Expressiveness Details 19

B.4 Scene Invariance Details 19

B.5 Real-Action Fine-Tuning Details 19

C Licenses of Existing Assets **20**

D Limitations **20**

E Broader Impact **21**

A Additional Experiments

A.1 Action-Transfer Videos

Extensive qualitative videos for the action-transfer control mode (Sec. 3.3) are provided in the supplementary material, covering same-source action transfer, cross-source action transfer, and in-the-wild demonstrations. Fig. 6 shows a few such examples; please see the supplementary HTML for the full set.

A.2 Whitening Control

Only the OnlyAct embeddings are whitened before entering the world model (Appendix B.1), while the baselines in Table 3 use their latent actions unchanged. To check that the whitening is not what drives the comparison, we train a second DreamDojo† world model whose latent actions pass through the same whitening rule, with statistics computed on its own embeddings of the same training clips, and keep everything else identical (backbone, recipe, data, seed, and sampler). Table 5 evaluates it on the same clips and source–target pairs as Table 3. Whitening changes DreamDojo's results only slightly (by at most 0.23 dB in Self PSNR and 0.023 in Transfer Dir). OnlyAct remains ahead of the whitened DreamDojo in Self PSNR on LIBERO and RLBench and in Transfer Dir on LIBERO, is level with it in Self PSNR on SpatialVID, and keeps a positive but not statistically significant lead in Transfer Dir on SpatialVID. The whitening therefore does not explain the advantage of OnlyAct.

A.3 Multimodal Embedding Backbone

The submitted version of this work extracted actions from a multimodal embedding model, Qwen3-VL-Embedding-8B [35], instead of V-JEPA 2. To compare the two backbones under identical conditions, we apply the OnlyAct recipe unchanged to Qwen3-VL-Embedding-8B: the same 168K training pairs, augmentation, positives and hard negatives, loss, optimizer, schedule and 10,000 steps (Appendix B.2). Only the backbone, its input format and its LoRA targets differ. A frame pair is resized and center-cropped to 256×256 exactly as for V-JEPA 2 and given to the model as two

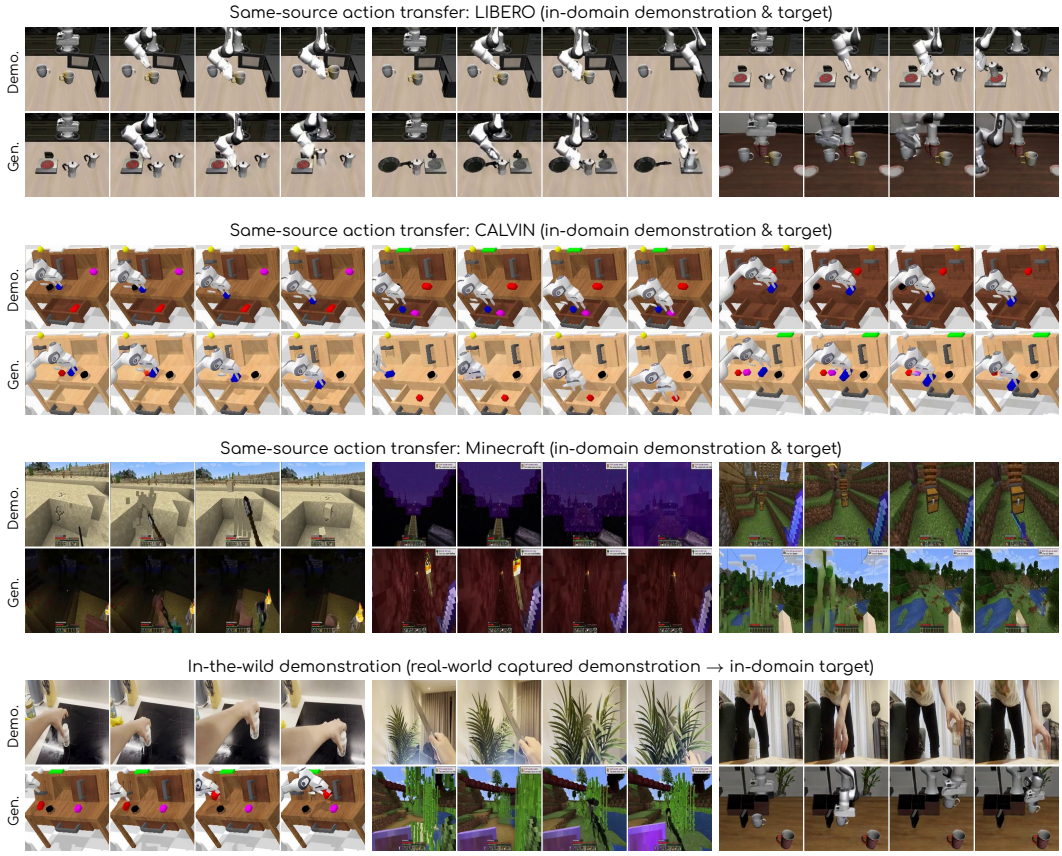

Figure 6: **Additional action-transfer results.** Latent actions extracted from the demonstration frames (top) condition the video world model generations (bottom). OnlyAct reproduces the demonstrated actions across various settings. Best seen as videos in the supplementary material which also includes more examples.

images after the instruction *"Describe what action is happening in this scene as an embedding."*; the embedding is the model's own last-token hidden state (4096-d), mapped to 640 dimensions by a fixed random linear layer and ℓ_2-normalized, as for OnlyAct. LoRA adapters with the same rank 16, α=32 and dropout 0.05 as OnlyAct are attached to the query, key, value and MLP projections of the language model (41.6M trainable parameters; with gradient checkpointing, training fits on one RTX 4090). Action expressiveness and scene leakage are evaluated with exactly the protocols, clips, splits and seeds of Tables 1 and 2.

Results. The contrastive task itself is learned: the anchor-to-positive retrieval accuracy among the five in-batch candidates (chance 0.20) rises from 0.72 over the first 500 steps to 0.96 over the last 2,000 (0.98 for OnlyAct over the same steps). The resulting representation, however, carries much less action information than OnlyAct (Table 6, left): its average probe R^2 is 0.142 against 0.490, it is lower on all 13 datasets, and it is also below the frozen V-JEPA 2 embedding (0.347). It lets slightly less of the scene be reconstructed than OnlyAct (Table 6, right; average held-out LPIPS 0.315 vs. 0.304), as expected for a representation that carries less information overall, which is why scene leakage has to be read together with action expressiveness: OnlyAct gains 0.35 in average R^2 for 0.011 in average LPIPS. These results motivated the switch to V-JEPA 2 in the final version; the results reported for the multimodal backbone in the submitted version are superseded by this re-evaluation.

Table 5: **Whitening control.** DreamDojo† world models trained with their native latent actions (as in Table 3) and with the same whitening rule as OnlyAct (statistics computed on its own embeddings), everything else identical, evaluated on the same clips and source–target pairs as Table 3 (LIBERO and RLBench: 40; SpatialVID: 20). RLBench transfer and nuScenes were not evaluated for the whitened model (–). Δ: ours minus the whitened DreamDojo, tested with a paired t-test over clips (Self PSNR) and a two-sided Wilcoxon signed-rank test over source–target pairs (Transfer Dir); grey: not significant at the 0.05 level. Single training seed.

	LIBERO°		RLBench		SpatialVID°	
Action representation	Self PSNR↑	Transfer Dir↑	Self PSNR↑	Transfer Dir↑	Self PSNR↑	Transfer Dir↑
DreamDojo†, native input (Table 3)	19.95	0.803	18.08	0.572	16.40	0.480
DreamDojo†, same whitening as ours	19.93	0.826	18.00	–	16.17	0.487
OnlyAct (ours)	20.32	0.917	18.63	0.686	16.10	0.608
Δ vs. whitened DreamDojo	+0.40	+0.091	+0.63	–	−0.07	+0.121
test statistic	t=2.42	$p<10^{-5}$	t=3.46	–	t=−0.31	p=0.31

Table 6: **The OnlyAct recipe on a multimodal embedding backbone.** Left: linear probing of action information (test R^2; Procgen: macro-F1; mean over 3 probe seeds) with the protocol, clips, splits and seeds of Table 1. Right: scene leakage (held-out LPIPS↑, higher = less scene information recoverable; mean over 4 decoder seeds) with the decoders and held-out scenes of Table 2. V-JEPA 2: the frozen backbone of OnlyAct; OnlyAct: main-text results; Qwen recipe: the same recipe (168K training pairs, augmentation, loss, optimizer, 10,000 steps) on Qwen3-VL-Embedding-8B (Appendix A.3). Δ: Qwen recipe minus OnlyAct; green: better for the Qwen recipe (higher R^2, less leakage). Avg.: probe over the 12 continuous-action datasets, leakage over all 13. °Videos seen during contrastive training (the same for both backbones).

	Linear probe R^2 ↑				*Scene leakage*: held-out LPIPS↑			
Dataset	V-JEPA 2	OnlyAct	Qwen recipe	Δ	V-JEPA 2	OnlyAct	Qwen recipe	Δ
LIBERO°	0.718	0.634	0.443	−0.19	0.257	0.314	0.323	+0.009
DROID°	−0.131	0.203	−0.085	−0.29	0.289	0.328	0.343	+0.015
RLBench	0.556	0.566	0.426	−0.14	0.216	0.269	0.266	−0.003
RoboCasa	−0.368	0.322	0.034	−0.29	0.480	0.411	0.433	+0.022
DROID-IPRL	0.186	0.279	−0.035	−0.31	0.247	0.322	0.337	+0.015
MetaWorld	0.144	0.277	−0.049	−0.33	0.066	0.105	0.125	+0.020
MUTEX	0.527	0.566	0.229	−0.34	0.130	0.177	0.197	+0.020
Robosuite	0.448	0.695	0.175	−0.52	0.282	0.317	0.331	+0.014
RoboDesk	0.520	0.581	0.301	−0.28	0.069	0.100	0.101	+0.001
SpatialVID°	0.301	0.560	0.263	−0.30	0.402	0.407	0.418	+0.011
ARKitScenes	0.665	0.588	0.104	−0.48	0.384	0.411	0.424	+0.014
RE10K	0.598	0.609	−0.102	−0.71	0.442	0.436	0.418	−0.018
Procgen (F1)	0.145	0.200	0.126	−0.07	0.311	0.352	0.376	+0.025
Avg.	0.347	0.490	0.142	−0.35	0.275	0.304	0.315	+0.011
Avg. SSIM↓					0.478	0.391	0.379	−0.012
Avg. L1↑					0.200	0.255	0.266	+0.010

B Experiment Details

B.1 Implementation Details

Action encoder. We use the encoder of V-JEPA 2 ViT-L (`vjepa2-vitl-fpc64-256`; 24 layers, hidden size 1024, 16×16 patches, tubelet size 2) [4]. Frames are resized to a short side of 292, center-cropped to 256×256, and normalized with ImageNet statistics using the official V-JEPA 2 preprocessing. A frame pair is stacked into a two-frame clip, which produces a single temporal slice of 16×16=256 tokens; the V-JEPA 2 predictor is not used. The output tokens are average-pooled to a 1024-dimensional vector, projected to 640 dimensions by a linear map that is randomly initialized (seed 0) and kept fixed, and ℓ_2-normalized.

Low-rank adaptation. We attach LoRA adapters [26] with rank 16, α=32, and dropout 0.05 to the query, key, value, and output projections of the attention and to both MLP layers in all 24 encoder blocks, as well as to the patch-embedding convolution. The pretrained weights are frozen and kept in bf16. This amounts to 7.1M trainable parameters, about 2.3% of the 300M encoder parameters.

World model. We use the AdaWorld world model [19] as the backbone for all action representations. We attach LoRA adapters with rank 128 and α=128 to the query, key, value, and output projections of the denoiser attention, and train the label-embedding MLP in full, which amounts to 61.5M trainable parameters for OnlyAct and 55.7M for the 32-dimensional latent-action baselines. We use AdamW with a learning rate of 5×10^{-5}, 1k linear warm-up steps followed by a constant schedule, a batch of 4 clips of 7 frames, 20k training steps, bf16 precision, gradient clipping at 0.3, no EMA, and seed 32. All world models are trained on the same 19,946 clips (25 frames, 256×256) from 9 datasets. Before entering the world model, the OnlyAct embeddings are whitened by a fixed, parameter-free covariance equalization. Each dimension is first standardized; the covariance $C = Q\,\mathrm{diag}(e)\,Q^\top$ of the standardized embeddings z is then equalized as $y = z\,Q\,\mathrm{diag}(g)\,Q^\top$ with gains $g_i = s/\sqrt{e_i + r}$, where the ridge $r \geq 0$ is the smallest value for which no gain exceeds 2, and s keeps the total variance equal to the dimensionality, so that no direction is removed. All statistics are computed once on the 478,704 frame pairs of the world-model training clips. The latent actions of AdaWorld and DreamDojo enter the world model unchanged; Appendix A.2 reports a DreamDojo world model trained with the same whitening. Sampling uses the Euler EDM sampler with 25 steps, a guidance scale of 1.1, and 6 context frames, rolled out autoregressively for 19 frames. Each world model takes about 5 GPU-hours on a single RTX 4090.

Compute. All experiments are run on single 24 GB GPUs (NVIDIA RTX 4090 or RTX A5500). Encoder adaptation takes about 1.5 GPU-hours on an RTX 4090.

B.2 Contrastive Learning Details

Training data. No action labels are used in this stage. We pre-sample 168K training samples from nine video sources with the following sampling weights: VPT Minecraft [5] 0.25, CALVIN [41] 0.15, SpatialVID [59] 0.15, AgiBot World [2] 0.10, EgoDex [25] 0.10, Something-Something v2 [22] 0.10, and LIBERO [37], DROID [33], and Atari [8] 0.05 each.

Anchor. The anchor is a frame pair (o_t, o_{t+k}) from one video, with k sampled uniformly from $\{1, 2, 3\}$. Both frames receive the same color jitter (brightness, contrast, and saturation ±0.3, hue ±0.05).

Positive: flow-guided scene swapping. We first decide whether the camera is static by the mean optical-flow magnitude of the background: if it is at most 0.5 pixels, the camera is treated as static. In this case, we compute the moving core region as the pixels whose flow magnitude exceeds $\max(1\,\mathrm{px},\ 0.35 \times q_{99})$, where q_{99} is the 99th percentile of the flow magnitude, without dilation or feathering. We cut out the original pixels of this region from both frames and paste them onto the same background frame, taken from a different scene of the same dataset, translated in both frames by a common random integer offset of at least 16 pixels. The moving pixels are therefore preserved exactly, while the scene and the absolute location of the motion are replaced. A color jitter shared between the two frames is applied afterwards. When the camera moves (e.g., hand-held, egocentric, or driving video), the background motion is itself part of the action and cannot be replaced; the positive then falls back to the anchor pair under a different color-jitter draw. During training, about 43% of samples use the scene-swapping path (mean over the training log): it dominates on static-camera datasets (LIBERO 99%, DROID 88%, Atari 75%, AgiBot World 62%), whereas hand-held and ego-motion datasets mostly fall back. Optical flow is precomputed with RAFT-large [55] (12 iterations, 256×256 input).

Hard negative. The hard negative is a frame pair from a different, non-overlapping time window of the same video as the anchor. It shares the scene with the anchor but generally corresponds to a different action.

Objective and optimization. We use the symmetric InfoNCE loss of Sec. 3.2 with temperature τ=0.07, over a candidate bank formed by all anchors, positives, and hard negatives in the batch. We

train on a single RTX 4090 with 2 samples per step, so that each query is contrasted against one positive and four negatives: its own same-video hard negative and the anchor, positive, and hard negative of the other sample; we accumulate gradients over 8 steps for an effective batch size of 16 (gradient accumulation does not enlarge the negative set, and no memory queue is used). We use AdamW with learning rate 10^{-4}, 500 warm-up steps followed by cosine decay to zero, no weight decay, and seed 0, for 10,000 updates (about 160K samples, roughly one pass over the training set). At the end of training, the positive is retrieved for every query (chance: 0.2), and the mean cosine similarity is 0.88 to the positive and 0.215 to the hard negative, compared with about 0.94 for both at initialization.

B.3 Action Expressiveness Details

For each embedding family, we train an independent linear probe $W \in \mathbb{R}^{D\times K}$ (zero-initialized weight and bias) to map the per-pair embedding $z_t \in \mathbb{R}^D$ (for OnlyAct, $z_t = \alpha_t$ from Sec. 3.1) to the ground-truth action a_t, where D is the embedding dimensionality and K is the action dimensionality for continuous datasets or the number of classes for the discrete datasets. Inputs are z-scored using mean and standard deviation estimated from a $20{,}000$-sample subset of the training split; for continuous datasets, the targets are z-scored with the same procedure, and for the discrete datasets, we apply softmax cross-entropy directly. We train for 50 epochs with AdamW (learning rate 10^{-2}, weight decay 10^{-3}) and batch size $4{,}096$, without early stopping, and report the last epoch averaged over 3 probe seeds on a fixed test split. The probe is deliberately kept linear so that performance differences reflect representation quality rather than probe capacity.

B.4 Scene Invariance Details

To measure how much pixel-level appearance can be reconstructed from the per-pair embedding $z_t \in \mathbb{R}^D$, we train a small convolutional decoder $g_\phi : \mathbb{R}^D \to \mathbb{R}^{3\times 128\times 128}$ to recover the second frame o_{t+1} of the pair. The decoder consists of a single linear projection from z_t to a $512\times4\times4$ feature map, followed by five upsampling blocks ($4\to8\to16\to32\to64\to128$); each block applies a 4×4 stride-2 ConvTranspose2d, GroupNorm, GELU, a 3×3 Conv2d, GroupNorm, and GELU. A final 3×3 convolution with $\tanh$ produces the RGB output. We train with a combined L1 + LPIPS loss with $\lambda_{\text{LPIPS}}=0.5$, using AdamW (learning rate 10^{-4}, weight decay 10^{-4}) with cosine annealing to zero and batch size 64, for 10 epochs per (embedding, dataset) pair. We report the last-epoch metrics averaged over 4 seeds, each with its own random split that holds out 20% of the scenes for evaluation.

B.5 Real-Action Fine-Tuning Details

Domains and data. DMLab and Minecraft use the released datasets of TECO [62] (DMLab actions: look left, look right, forward; Minecraft actions: forward with jump, camera left $20°$, camera right $20°$). Habitat uses frames rendered with Habitat [50] in the HM3D example scenes [48] (move forward, turn left, turn right), with fine-tuning clips from one scene and test cases from two other scenes. nuScenes uses the front camera, with the ego motion (forward and lateral displacement in meters) as a continuous action. Each discrete-action domain has 300 labeled fine-tuning clips (100 per action) and 300 test cases; nuScenes has 100 fine-tuning clips from 4 driving logs and 60 test cases from 12 other driving logs.

Action interface. For the discrete-action domains, the interface is an action book with one row per action. Each row is initialized at the mean of the representation's latent actions, after the world model's input transform, over the 100 labeled clips of that action. The *0 updates* column of Table 4 uses this initialization without any gradient update, so it already uses the labeled clips. For nuScenes, the interface is a two-layer MLP (linear, SiLU, linear) from the two-dimensional ego motion to the latent-action space, fitted before fine-tuning by 3,000 SGD steps (learning rate 0.01, batch 10, mean-squared error) to regress the transformed latent actions of the labeled clips.

Two recipes. DMLab and Minecraft use a *standardized* interface: the book stores u, and the condition is $\mu + \sigma \odot u$, where μ and σ are the per-dimension statistics of each representation's own world-model training features. The book is trained with a learning rate of 2.42×10^{-3}, so that every representation receives the same step size relative to the scale of its features. Habitat and nuScenes

use the *native* interface: the book rows or the MLP output the condition directly and are trained with a learning rate of 5×10^{-5}.

Optimization. In both recipes, the whole denoiser (1.56B parameters) is fine-tuned with a learning rate of 5×10^{-6} for 800 AdamW updates (weight decay 0.01) with a global batch of 32 clips, 100 linear warm-up updates, gradient clipping at 0.3, an EMA of the denoiser weights (decay 0.9999), bf16 mixed precision, and seed 32. The 200- and 800-update checkpoints come from the same run, and each cell of Table 4 is a single fine-tuning run.

Evaluation. Each test case has six real context frames, and the next frame is generated under every candidate action with the Euler EDM sampler (5 steps, guidance scale 1.05), using the same noise for all actions and all representations. On the discrete-action domains, Follow uses an inverse-dynamics classifier trained only on real frames, which receives the last real context frame and the generated frame. On nuScenes, Follow is the per-case Spearman correlation between five commanded forward displacements and the forward displacement that a regressor trained only on real frames reads from the last real context frame and the generated frame. Control is the PSNR under the commanded action minus the mean PSNR under the two other actions (on nuScenes, under four actions drawn from the training clips). Δ is tested with a paired bootstrap over test episodes (driving logs on nuScenes; 20,000 resamples), which covers the sampling of test cases but not the variance between training runs.

C Licenses of Existing Assets

Table 7 lists the license or terms of use of every dataset, model, and code base that we use, as stated by its source. Several datasets are restricted to non-commercial use (AgiBot World, EgoDex, SpatialVID, ARKitScenes, nuScenes, HM3D, and RLBench); we use them for non-commercial research only. We do not redistribute any data: our release contains only clip lists, splits, and preparation scripts that read each dataset from its original source.

D Limitations

OnlyAct demonstrates that action representations can be extracted from a pretrained foundation model and adapted for action-conditioned video world models without explicit action labels. However, several limitations remain. First, the method inherits both the strengths and weaknesses of the underlying foundation model. Although large-scale video pretraining provides broad spatio-temporal priors that improve transfer, the resulting action space may still reflect biases in the pretraining distribution, and its performance may degrade for domains with visual dynamics, embodiments, or interaction patterns that are poorly represented in that data. Second, OnlyAct relies on positive pairs that preserve the underlying action while changing the scene. Flow-guided scene swapping is only applicable to static cameras; for moving cameras we fall back to photometric augmentation, which provides a weaker scene-invariance signal. Optical-flow errors, thin or occluded moving parts, and interactions whose effect extends beyond the moving region may also make the swapped positive imperfect. Third, the contrastive stage uses a small batch with few negatives per query and saturates before the end of training, so larger batches, negative queues, or harder negatives may further improve the representation. Fourth, while contrastive adaptation substantially reduces scene leakage, it does not provide a formal guarantee that the learned representation is fully disentangled from scene content or future-state information.

In addition, real-action control still requires a small amount of action-labeled data to align environment-specific actions with the learned latent-action space. While this requirement is substantially weaker than training fully supervised action-conditioned models, it means that OnlyAct is not entirely label-free for direct control. Finally, our experiments cover robot manipulation, camera navigation, game, and driving data, but they remain offline evaluations. Further work is needed to evaluate robustness in long-horizon closed-loop interaction, real-world robotics, safety-critical settings, and domains involving complex multi-agent behavior.

Table 7: **Licenses of existing assets**, as stated by each source.

Asset	Used for	License or terms
VPT Minecraft [5]	training	MIT (repository; no separate license for the recordings)
CALVIN [41]	training	MIT
AgiBot World [2]	training	CC BY-NC-SA 4.0
EgoDex [25]	training	CC BY-NC-ND
Something-Something v2 [22]	training	Qualcomm license agreement
Atari [8]	training	none stated
LIBERO [37]	training, evaluation	CC BY 4.0
DROID [33]	training, evaluation	CC BY 4.0
SpatialVID [59]	training, evaluation	CC BY-NC-SA 4.0
RLBench [28] (PerAct demonstrations)	evaluation	Imperial College London non-commercial research license; none stated for the pre-generated demonstrations
DROID-IPRL [33]	evaluation	CC BY 4.0
RoboCasa [45]	evaluation	CC BY 4.0
MetaWorld [65] (LeRobot release)	evaluation	Apache-2.0
MUTEX [52]	evaluation	Apache-2.0
Robosuite [70], RoboDesk [29] (VP^2 data)	evaluation	MIT
ARKitScenes [6]	evaluation	Apple ARKitScenes license (non-commercial terms)
RealEstate10K [69]	evaluation	camera files CC BY 4.0; frames from YouTube, no license stated
Procgen [14]	evaluation	MIT
nuScenes [13]	evaluation, fine-tuning	CC BY-NC-SA 4.0 and the nuScenes Terms of Use
DMLab [7], Minecraft (TECO data [62])	fine-tuning	none stated
HM3D example scenes [48]	fine-tuning	Matterport academic, non-commercial license
V-JEPA 2 ViT-L [4]	encoder backbone	MIT
AdaWorld [19]	world model, baseline	Apache-2.0 (weights fine-tuned from Stable Video Diffusion; Stability AI terms apply)
DreamDojo [18]	baseline	Apache-2.0 (code)
RAFT [55] (torchvision)	positive construction	BSD-3-Clause

E Broader Impact

This work aims to reduce the dependence of video world models on expensive and fragmented action annotations. By extracting action representations from unlabeled video, OnlyAct could make it easier to train controllable simulators across domains where explicit action labels are difficult to obtain, including robotics, embodied AI, game environments, and interactive content creation. In robotics, such representations may support more scalable learning from observational video and lower the annotation burden for adapting world models to new embodiments. In gaming and simulation, they may enable more reusable action interfaces across visually diverse environments.

At the same time, more capable action-conditioned video world models raise several risks. If deployed without safeguards, they could be used to generate misleading interactive simulations or synthetic videos that appear physically plausible but do not faithfully reflect real-world dynamics. In robotics or embodied decision-making, inaccurate predictions could lead to unsafe behavior if treated as reliable simulators without appropriate validation. Because OnlyAct builds on a foundation model pretrained on large-scale web video, it may also inherit biases from that data, potentially leading to uneven performance across environments, objects, cultures, or human activities. Moreover, models that learn action structure from unlabeled video may amplify privacy concerns if trained on videos containing people or sensitive environments.

We therefore view OnlyAct primarily as a research step toward scalable action representation learning rather than a ready-to-deploy simulator for safety-critical decision-making. Responsible use should include careful dataset curation, evaluation for domain-specific failure modes, transparency about synthetic generation, and validation before deployment in real-world control systems. Releasing code, models, and evaluation protocols can help the community better audit the learned representations, reproduce results, and study both the benefits and risks of annotation-free action-conditioned world modeling.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Sec. 3 state the three claims of the paper—extracting action representations from a pretrained foundation model (V-JEPA 2), a label-free contrastive adaptation that disentangles action from scene without a reconstruction bottleneck, and a unified evaluation across robot, navigation, game, and driving benchmarks—which are matched by the method in Sec. 3 and the experiments in Sec. 4.3 and the surrounding subsections.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in Appendix D, including dependence on the pretrained foundation model's priors, the static-camera requirement of flow-guided scene swapping, the small contrastive batch, imperfect action disentanglement, limited action supervision requirements, and the need for broader real-world evaluation.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper presents an empirical method and evaluation; it contains no formal theorems or proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sec. 3 fully specifies the action encoder, contrastive objective, and action-conditioned video world model, and Appendix B.1 and B.2 report the encoder (V-JEPA 2 ViT-L), the LoRA configuration, the positive/negative construction, the contrastive training data and hyperparameters, the world-model backbone (AdaWorld), and the evaluation protocols used for all reported numbers.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code, the trained checkpoints, and the evaluation protocol of every table are released at `https://github.com/runjiali-rl/onlyact` (linked from the project page on the first page), with the exact software environments and one script per table. All datasets and the underlying pretrained models (V-JEPA 2, AdaWorld, DreamDojo) are publicly available; we do not redistribute any frames, and the repository provides the clip lists, splits, and preparation scripts of each dataset.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Sec. 4 describes the datasets, splits (held-out scenes; held-out demonstrations on LIBERO and RLBench), baselines, and metrics, and Appendix B.1–B.4 report the encoder and world-model hyperparameters, the contrastive training data and optimization, and the probe and decoder protocols.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Probing and scene-leakage results are averaged over 3 and 4 seeds, respectively, with the probe standard deviation reported in Table 1; in Table 4, differences whose 95% paired bootstrap interval contains zero are marked. World-model training runs (Table 3) and fine-tuning runs (Table 4) are single runs due to compute constraints, so the bootstrap intervals of Table 4 cover the sampling of test cases but not the variance between training runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix B.1 reports that all experiments run on single 24 GB GPUs (RTX 4090 / RTX A5500): encoder adaptation takes about 1.5 GPU-hours and each world model about 5 GPU-hours.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: The work uses publicly released datasets and pretrained models within their licensed terms, involves no human subjects, and does not pose obvious risks beyond those of the underlying video generation backbone.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The broader impacts are discussed in Appendix E, including the opportunities enabled by scalable action representation learning from unlabeled video, as well as the associated challenges of reliability, bias, misuse, and responsible deployment.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The released models are an action encoder and an action-conditioned video world model trained on public robotics, game, and human video data; their misuse risk is comparable to that of the underlying AdaWorld world model, and they are not general-purpose photorealistic generators.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets and pretrained models (V-JEPA 2, AdaWorld, DreamDojo) used in the paper are cited to their original publications, and Appendix C lists the license or terms of use of every asset as stated by its source. Datasets restricted to non-commercial use are used for non-commercial research only, and no data is redistributed.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new assets—the contrastively adapted action encoder and the action-conditioned video world model—are documented in Sec. 3 (architecture, training objective) and Appendix B.1 (hyperparameters); the code, weights, and documentation (environments, training and evaluation commands, checksums, and licenses) are released at `https://github.com/runjiali-rl/onlyact`, with the code under the Apache License 2.0.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects, so no IRB review was required.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were used only for assistance with writing, editing, and formatting, and did not influence the core methodology, scientific rigor, or originality of the research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.