SEPARATING ACTION FROM PREDICTION IN WORLD ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World action models (WAMs) learn latent variables from observations and use them both to predict how the scene evolves and to generate actions. Prediction has to account for the whole scene, whereas the actions depend on only part of it. However, WAMs fitted to demonstrations provide no guarantee of generating actions only from the latent variables on which the demonstrated actions depend, termed action-relevant latent variables, because a model may imitate equally well while entangling them with nuisance variables. We therefore separate what the actions may depend on from what prediction needs, and impose a sparsity constraint only on the estimated structure between latent states and actions. We prove that, under this constraint, the latent variables used to generate actions are an invertible function of the action-relevant ones alone, and the true structure is identified up to a permutation of the latent variables. These guarantees require neither sufficient temporal dependence nor assumptions on how unobserved regimes evolve, and hold even when the action-relevant latent variables are statistically dependent on the rest. We approximate the constraint in a pretrained WAM by adding a sparse routing path to its action head, while the world-modeling objective stays unchanged. Synthetic experiments support the theory, and on simulated and real robots, the routing path improves aggregate closed-loop success.

1 INTRODUCTION

When a robot stacks one cube on another, its motion is determined by where the two cubes are and whether its gripper holds the top one. Replacing the tablecloth or moving the clutter behind the table changes many pixels in the camera images but leaves the motion unchanged. World action models (WAMs) such as Fast-WAM (Yuan et al., 2026) and Cosmos 3 (NVIDIA, 2026) infer latent variables from these images and are trained to predict future frames and to generate actions from the same variables. Since prediction has to account for the whole scene, these latent variables also describe the tablecloth and the clutter. We refer to the latent variables on which the demonstrated actions depend as action-relevant latent variables, and to the others as nuisance variables.

However, fitting the demonstrations well does not guarantee that a WAM generates its actions from the action-relevant latent variables. For each step of a demonstration, one may write $o_t = f(s_t)$ and $a_t = \pi(s_t, \eta_t)$, where s_t is the latent state, f an unknown invertible function, π an unknown action mechanism, and η_t exogenous noise. Separating action from prediction amounts to recovering, up to certain indeterminacies, the action-relevant latent variables and which of them each action dimension depends on. The difficulty is that, without further assumptions, latent variables cannot be identified from their observations (Hyvärinen & Pajunen, 1999; Locatello et al., 2019). Observing the actions as well does not remove this ambiguity, because two models can induce the same distribution of (o_t, a_t) while their latent variables differ by an arbitrary invertible transformation. This transformation can mix the action-relevant latent variables with nuisance ones, so a flexible action head may generate actions from variables that also encode the tablecloth, which does not cause the motion, and still fit the demonstrations equally well.

To make latent variables identifiable, existing works in nonlinear independent component analysis (ICA) introduce auxiliary variables, such as domain or time indices, or exploit temporal dependence (Hyvärinen & Morioka, 2016; 2017; Hyvärinen et al., 2019; Khemakhem et al., 2020). For latent causal processes, identifiability has been established under sufficient temporal dependence (Yao

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

et al., 2022) and, when the nonstationarity is unknown, under additional assumptions on how the unobserved regimes evolve (Song et al., 2023). Besides, Lippe et al. (2022) rely on interventions with known targets. In demonstrations, however, the temporal dependence need not be sufficient, the unobserved regimes the robot passes through (e.g., skills or contact phases) need not follow a Markov chain, and controlled interventions are rarely available.

Another line of work constrains the generating process, through the function class or the sparsity of its dependency structure (Kivva et al., 2022; Zheng et al., 2022; Lachapelle et al., 2022). Most of these results, like those above, aim to recover all latent variables, so their conditions have to hold even for the nuisance variables a robot can ignore. Results on partial identifiability target only a block of latent variables, which is set by side information that comes with the data, such as the content shared across views (von Kügelgen et al., 2021; Yao et al., 2024), the part invariant across domains (Kong et al., 2022), or the part relevant to given tasks (Zheng et al., 2026). For a robot learning from demonstrations, the block that matters is the set of latent variables that the demonstrated actions depend on. No view, domain, or task label specifies this set, so the actions themselves have to reveal which latent variables belong to it. Therefore, one may wonder:

From demonstrations alone, can we separate what actions depend on from what prediction needs?

In this work, we answer this question with a constraint on the action side that leaves prediction free to use every latent variable. Since each action is generated from only part of the latent state, constraining the number of dependencies between latent variables and action dimensions can separate that part from the rest. Specifically, we show that, without such a constraint and under nondegeneracy conditions, fitting the demonstrations only ensures that each action dimension of the estimated mechanism depends on at least as many latent variables as the true one (Prop. 1). We then constrain the estimated mechanism to have no more of these dependencies than the true one, summed over action dimensions. Under the same conditions, the lower bound becomes an equality and the dependency structure is identified up to a permutation of the latent variables (Thm. 1).

Moreover, the estimated latent variables used to generate actions are computed from observations only and, under the same constraint, are an invertible function of the action-relevant latent variables alone (Thm. 2). The argument treats each step on its own, so it needs neither sufficient temporal dependence nor any assumption on how unobserved regimes evolve. It also allows the action-relevant latent variables to be statistically dependent on the nuisance ones. Inspired by the theory, we add to Cosmos 3 a sparse routing path from visual and task tokens to the action head and apply a token-level surrogate of the constraint to this path (Sec. 4). The joint video and action loss of Cosmos 3 is unchanged, so prediction still accounts for the whole scene. On synthetic processes with known latent variables, the latent variables selected with a sparsity penalty recover the action-relevant block, and the fitted action head relies on them. On RoboCasa365, LIBERO, and a real Franka R3 robot, the routing path improves aggregate closed-loop success over Cosmos 3 without it, and in most visualized frames its routing mass falls on the object being moved and its target (Sec. 5).

2 Problem setup

Data. Demonstration data are sequences $\{o_t, a_t\}_{t=1}^{T}$ with $o_t \in \mathbb{R}^{d_o}$ and $a_t \in \mathbb{R}^{d_a}$. Relevance is defined by the executed action at each step, and no task, skill, or segment labels are assumed.

Observational world action process. We define latent states $\{s_t\}_{t=1}^{T}$, $s_t \in \mathbb{R}^n$, and hidden nonparametric maps

$$o_t = f(s_t), \tag{1a}$$
$$a_t = \pi(s_t, \eta_t), \tag{1b}$$

where $f : \mathbb{R}^n \to \mathbb{R}^{d_o}$ is a diffeomorphism onto its image ($d_o \gg n$ is allowed), π is an unknown action mechanism, and η_t is exogenous noise. The same mechanism applies to an H-step chunk $a_{t:t+H-1}$ when the policy emits a short horizon at once (Section 4); identifiability is stated per action coordinate. We do not require temporal dependence: consecutive steps may be disconnected and each (o_t, a_t) may be i.i.d. Within a step, action-relevant latent variables are those on which π functionally depends. Since f is invertible, o_t depends on every latent variable, whereas π depends only on the action-relevant ones. A fitted action mechanism may *over-cover* the support of J_π, that

is, depend on more latent variables than π, and we ask when the latent variables used to generate actions are separated from the nuisance ones (notation below).

Mechanism supports and the compact state. For a differentiable structural map m of the state, let $J_m(s) = \partial m/\partial s$, and let $\mathcal{I}(J_m) = \{(i,j) : \Pr[J_m(s)_{ij} \neq 0] > 0\}$ be its functional support, with $\|\mathcal{I}(J_m)\|_0$ its cardinality. For π, the Jacobian is taken with respect to s_t at fixed η_t. We write $I_i := \{j : (i,j) \in \mathcal{I}(J_\pi)\}$ for the coordinates that drive action dimension i, set $I_{\mathrm{A}} = \bigcup_i I_i$, and write $s_{t,I_{\mathrm{A}}} = (s_{t,j})_{j \in I_{\mathrm{A}}}$ for the action-relevant subvector. Relevance is functional (a perturbation of coordinate j can change a) and is not a statement about correlation. A learner returns an encoder $\hat{f}^{-1}$ with $\hat{s}_t = \hat{f}^{-1}(o_t)$, an action mechanism $\hat{\pi}$, estimated index sets $\hat{I}_i := \{j : (i,j) \in \mathcal{I}(J_{\hat{\pi}})\}$ and $\hat{I}_{\mathrm{A}} = \bigcup_i \hat{I}_i$, and the *compact action-relevant state* $C_t := \hat{s}_{t,\hat{I}_{\mathrm{A}}}$. At deployment, C_t is computed from observations only; training may use a_t to fit $\hat{\pi}$, but actions are not inputs to $\hat{f}^{-1}$.

Identification goal. Write $\mathcal{L}_{f,\pi}$ for the law of (o,a) induced by equation 1a–equation 1b. Two processes (f,π) and $(\hat{f},\hat{\pi})$ are *observationally equivalent* if

$$\mathcal{L}_{f,\pi} \;=\; \mathcal{L}_{\hat{f},\hat{\pi}}. \tag{2}$$

Under observational equivalence on (o,a), the estimated state $\hat{s}_t$ determines every latent variable, as prediction requires, so we only ask about the action side. We first ask whether the estimated action mechanism matches the true state-to-action dependency pattern. We then ask whether the compact state C_t used to generate actions is separated from the nuisance variables, in the sense that it carries the action-relevant block $s_{t,I_{\mathrm{A}}}$ up to a block-wise invertible transformation. Section 3 states sufficient conditions for both questions. Section 5.1 is the direct empirical check with known I_{A}, and Appendix C records the synthetic generator, estimators, and metrics in full.

Estimated process. An estimated process is any pair $(\hat{f},\hat{\pi})$ that satisfies equation 1a–equation 1b with $\hat{f} : \mathbb{R}^n \to \mathbb{R}^{d_o}$ a diffeomorphism onto its image. With $\hat{s}_t = \hat{f}^{-1}(o_t)$ and $\phi := \hat{f}^{-1} \circ f$, observational equivalence allows ϕ to be an arbitrary diffeomorphism on $\mathbb{R}^n$ compatible with equation 2. By itself, observational equivalence does not identify the full latent state s_t or a unique encoder $\hat{f}^{-1}$. Section 3 asks what *is* identified (the compact state C_t and the functional support of J_π) under additional inductive bias.

Regularity and coupling. Throughout, f, $\hat{f}$, $\pi(\cdot,\eta)$, and $\hat{\pi}(\cdot,\eta)$ are continuously differentiable, f and $\hat{f}$ have the same image (so ϕ is a diffeomorphism of $\mathbb{R}^n$), and s_t has a density that is positive on $\mathbb{R}^n$ (hence so does $\hat{s}_t = \phi(s_t)$). The results below use a pathwise coupling of the two processes, in which they share the noise and share the latent state up to ϕ, that is, $o_t = f(s_t) = \hat{f}(\hat{s}_t)$ and $a_t = \pi(s_t,\eta_t) = \hat{\pi}(\hat{s}_t,\eta_t)$ with $\hat{s}_t = \phi(s_t)$. This coupling implies equation 2.

3 Theory

We now study, within each time step, when fitting (o,a) separates the latent coordinates that generate the action from the nuisance ones that only prediction needs. In equation 1, only the block $s_{t,I_{\mathrm{A}}}$ need enter π. Remaining coordinates may still be required for $o_t = f(s_t)$, so a latent state sufficient for observations need not isolate the coordinates that cause the action. All guarantees below concern one application of equation 1, or equivalently one pair (o,a) drawn from $\mathcal{L}_{f,\pi}$ in equation 2. We keep the subscript t when referring to demonstration tuples (o_t,a_t), but the argument does not use cross-time structure (Appendix A fixes a step and drops t). The within-step conclusion applies at every t, and if the steps are identically distributed, the law in equation 2 also does not depend on t.

3.1 Identifiability without sparsity

Without a support-sparsity constraint on $\hat{\pi}$, observational equivalence on (o,a) may still hold while $\hat{\pi}$ depends on more latent coordinates than π. The action side then need not be separated from the nuisance coordinates, and fitting guarantees only the one-sided bound in Proposition 1.

Proposition 1. *Consider two observationally equivalent processes $o_t = f(s_t) = \hat{f}(\hat{s}_t)$, $a_t = \pi(s_t,\eta_t) = \hat{\pi}(\hat{s}_t,\eta_t)$ of the form equation 1 that satisfy the regularity conditions of Section 2.*

Assume that, for each action dimension $i \in [d_a]$, there exist $|I_i|$ states $s^{(1)}, \dots, s^{(|I_i|)}$ at which the rows $(J_\pi(s^{(l)}))_{i,\cdot}$ are linearly independent, and for $\phi = \hat{f}^{-1} \circ f$, $P = J_{\phi^{-1}}$, and a matrix M with $\mathcal{I}(M) = \mathcal{I}(P)$, $\mathcal{I}\big((J_\pi(s^{(l)})M)_{i,\cdot}\big) \subseteq \mathcal{I}\big((J_{\hat{\pi}}(\hat{s}^{(l)}))_{i,\cdot}\big)$ with $\hat{s}^{(l)} = \phi(s^{(l)})$. Then, for every action dimension $i \in [d_a]$,

$$\|\mathcal{I}\big((J_{\hat{\pi}})_{i,\cdot}\big)\|_0 \;\geq\; \|\mathcal{I}\big((J_\pi)_{i,\cdot}\big)\|_0.$$

Proof sketch. The span condition ensures that reparameterization between s and $\hat{s}$ cannot eliminate a true functional dependency on I_i. The inclusion condition links $J_{\hat{\pi}}$ to J_π through ϕ and rules out accidental cancellations, so, up to a permutation ρ of the latent coordinates that is common to all action dimensions, each row of $J_{\hat{\pi}}$ must activate at least the coordinates required by the corresponding row of J_π. Counting these coordinates gives the bound. Details are in Appendix A.

Discussion on assumptions. The span and inclusion conditions in These conditions and the argument follow those of Zheng et al. (2026) but we adopt them for actions. Proposition 1 ensure that Jacobian rows carry enough variation to represent their supports under reparameterization. They rule out degenerate training distributions and accidental derivative cancellations. Under a continuous state density, the span condition is mild. A fuller discussion is in Appendix B.1.

Implication. Proposition 1 makes over-coverage precise. Without a bias toward a *minimal* action pathway, observational fitting need not shrink $\hat{\pi}$ to π. It need only learn a mechanism that is no smaller in support than the true one. Neither matching $\mathcal{I}(J_\pi)$ nor invertible recovery of s_{t,I_A} in C_t is guaranteed.

3.2 Identifiability under support sparsity

The negative result points to the design choice used in our method: constrain *which* latent coordinates $\hat{\pi}$ may depend on. Support sparsity requires that the estimated mechanism activate no more coordinates, counted over action dimensions, than the true one,

$$\|\mathcal{I}(J_{\hat{\pi}})\|_0 \;\leq\; \|\mathcal{I}(J_\pi)\|_0. \tag{3}$$

Combined with Proposition 1, this inequality bounds the size of the estimated support from both sides and identifies the state-to-action dependency pattern up to relabeling. The mechanism-level guarantee is as follows.

Theorem 1. *Consider two observationally equivalent processes $o_t = f(s_t) = \hat{f}(\hat{s}_t)$, $a_t = \pi(s_t, \eta_t) = \hat{\pi}(\hat{s}_t, \eta_t)$ of the form equation 1, and assume the regularity conditions of Section 2, the conditions in Proposition 1, and equation 3. Then there is a permutation ρ of $[n]$ such that*

$$\mathcal{I}(J_{\hat{\pi}}) \;=\; \mathcal{I}(J_\pi)\, P_\rho,$$

and, for every action dimension $i \in [d_a]$, $\mathcal{I}\big((J_{\hat{\pi}})_{i,\cdot}\big) = \rho(I_i)$. Here P_ρ is the permutation matrix with $(P_\rho)_{j,k} = \mathbf{1}\{\rho(j) = k\}$, and supports are identified with $0/1$ matrices.

Proof sketch. By the proof of Proposition 1, reparameterization through $\phi = \hat{f}^{-1} \circ f$ gives a common permutation ρ with $\rho(I_i) \subseteq \mathcal{I}((J_{\hat{\pi}})_{i,\cdot})$ for every i. Sparsity equation 3 forbids extra activations, so the inclusions become equalities and assemble into $\mathcal{I}(J_{\hat{\pi}}) = \mathcal{I}(J_\pi)P_\rho$ (Appendix A).

Implication. Each action dimension i of the fitted mechanism depends on exactly the estimated coordinates $\rho(I_i)$, which correspond to its true parents I_i, and the bipartite pattern from latent coordinates to actions is matched, whereas without sparsity it can be enlarged. Sparsity turns the lower bound in Proposition 1 into an exact answer to *which* coordinates each action dimension functionally depends on. Theorem 1 concerns the Jacobian of $\hat{\pi}$, that is, the bipartite pattern from latent coordinates to executed actions. However, the robot deploys the compact state $C_t = \hat{s}_{t,\hat{I}_\mathrm{A}}$, which is computed from observations without access to a_t. Even with matched supports, one could imagine a lossy projection that drops information within the action-relevant block or leaks nuisance coordinates into C_t. The next result shows that this does not happen under the coupling of Section 2 once supports align. Specifically, the reparameterization $\phi = \hat{f}^{-1} \circ f$ cannot mix the nuisance coordinates into the estimated coordinates $\rho(I_\mathrm{A})$, so C_t is an invertible function of s_{t,I_A}.

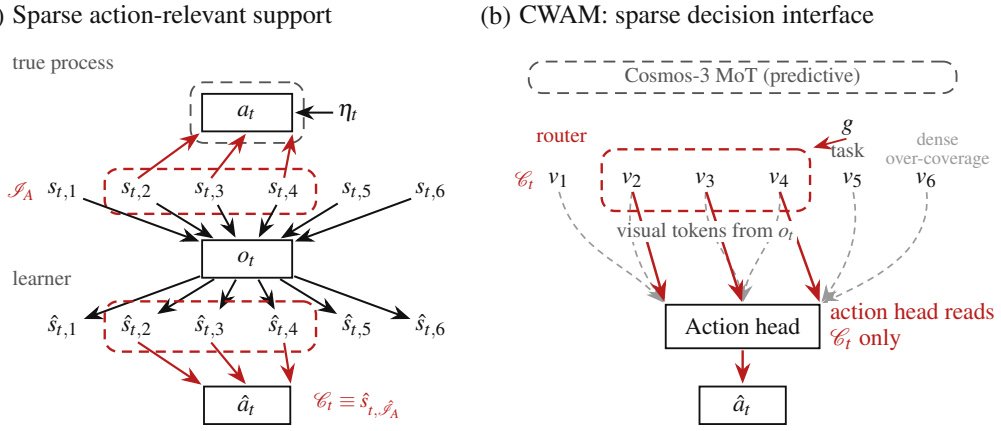


Figure 1: The action depends only on a block of the latent state, and CWAM routes an estimate of this block to the action head through a sparse path. (a) In the world action process equation 1 this block is s_{t,I_A}, and the learner $(\hat{f}, \hat{\pi})$ estimates it from observations as the compact state C_t. (b) CWAM keeps the joint rectified-flow training of Cosmos 3 and adds a sparse router over clean context tokens, whose top-k routed tokens form C_t. Gray dashed arrows mark the dense path to the action head, which the default policy keeps.

Theorem 2. *Under the hypotheses of Theorem 1, including the regularity conditions of Section 2, with the same permutation ρ, there exists an invertible map h such that*

$$C_t \;=\; \hat{s}_{t,\rho(I_A)} \;=\; h\big(s_{t,I_A}\big).$$

Proof sketch. For $\phi = \hat{f}^{-1} \circ f$ and $P = J_{\phi^{-1}}$, the aligned row supports $\mathcal{I}((J_{\hat{\pi}})_{i,\cdot}) = \rho(I_i)$ from Theorem 1 imply that P is block lower-triangular, with a zero block in rows I_A and columns $\rho([n] \setminus I_A)$. The invertibility of P then yields $\hat{s}_{t,\rho(I_A)} = h(s_{t,I_A})$ with invertible h. By Theorem 1, $\hat{I}_i = \rho(I_i)$ for every i, so $\hat{I}_A = \rho(I_A)$ and $C_t = \hat{s}_{t,\rho(I_A)}$. Details are in Appendix A.

Implication. The compact state C_t is an invertible re-encoding of the true action-relevant block s_{t,I_A}. This is stronger than being correlated with pose or gripper-like factors. The conclusion is functional (C_t depends only on s_{t,I_A} and determines it), so it holds whatever the statistical dependence between s_{t,I_A} and the nuisance coordinates.

4 Estimation Method

Estimation principle. Under the pathwise coupling and regularity conditions of Section 2, the nondegeneracy conditions of Proposition 1, and support sparsity equation 3, Section 3 identifies the support of J_π up to a permutation and the action-relevant latent variables in C_t up to a block-wise invertible transformation. We refer to a model fitted to demonstrations with a sparsity penalty on its action mechanism as CWAM (causal WAM). Synthetic experiments impose coordinate sparsity on a linear head directly (Section 5.1). The deployed action interface extends equation 1b to an H-step chunk:

$$\hat{a}_{t:t+H-1} \;=\; \hat{\pi}\big(o_t,\, \epsilon_t;\, C_t\big), \qquad C_t \;=\; \mathcal{S}\big(\hat{f}^{-1}(o_t)\big), \tag{4}$$

where ϵ_t is exogenous noise for generation (an initial draw for rectified flow, in parallel with η_t in equation 1b), $\mathcal{S}$ selects a support-sparse subset of the estimated state, and H is the chunk horizon. Figure 1 shows the split between world modeling and sparse routing to the action head.

Backbone. Cosmos 3 Nano (NVIDIA, 2026) co-trains video prediction and action generation from shared context. CWAM keeps that backbone and post-trains it on demonstrations (o, a) with the standard joint rectified-flow recipe (Lipman et al., 2023; Liu et al., 2023b). A mixture-of-transformers (MoT) generator (Liang et al., 2024) denoises future video latents and the executed action chunk, conditioned on clean history and language. The leading frame in each window is

observed, and later video tokens are noised during training. CWAM instantiates the estimation principle above as

$$\min_{\hat{f},\hat{\pi}} \; \mathcal{D}\big(\mathcal{L}_{f,\pi},\, \mathcal{L}_{\hat{f},\hat{\pi}}\big) \;+\; \mathcal{R}_{\text{sparse}}(\hat{\pi}), \tag{5}$$

where $\mathcal{D}$ penalizes mismatch between the true and estimated laws of (o, a) from equation 1a–equation 1b, and $\mathcal{R}_{\text{sparse}}$ biases $\hat{\pi}$ toward a minimal action pathway as in equation 3. In Cosmos 3, $\mathcal{D}$ is the joint post-training loss $\mathcal{L}_{\text{WAM}}$ (the rectified-flow video and action co-training described above), and $\mathcal{R}_{\text{sparse}} = \mathcal{L}_{\text{sparse}}$ in equation 7. CWAM leaves $\mathcal{L}_{\text{WAM}}$ unchanged, adds the sparse router, and applies $\mathcal{L}_{\text{sparse}}$ to the router readout only.

Action-relevant router. After the generator tower, each action hidden state h_i routes over clean context indices $j \in \mathcal{J}_t$ (task tokens and *clean* vision, excluding future noised tokens):

$$w_{ij} \;=\; \text{softmax}_{j\in\mathcal{J}_t}\Big(\frac{q_i^\top k_j}{\sqrt{d_r}} + b_{ij}(\text{relations})\Big), \qquad \tilde{h}_i \;=\; \sum_{j\in\mathcal{J}_t} w_{ij}\, v_j, \tag{6}$$

with relation-aware biases b_{ij} on position, modality, and chunk progress ($d_r = 512$, 8 heads). The routed signal enters through a learned gate $g \in (0, 1)$, initialized small, as $h_i \leftarrow h_i + g\,\tilde{h}_i$. Training adds a token-level surrogate for equation 3 on this pathway,

$$\mathcal{L}_{\text{sparse}} \;=\; \lambda\, \mathbb{E}_i\Big[1 - \sum_{j\in T_k(i)} w_{ij}\Big], \tag{7}$$

where $T_k(i) \subseteq \mathcal{J}_t$ is the index set of the k largest weights w_{ij}, so the sum is the attention mass on the top-k context tokens ($\lambda = 0.01$, ramped in after 10^3 steps). Post-training minimizes

$$\mathcal{L}_{\text{total}} \;=\; \mathcal{L}_{\text{WAM}} \;+\; \mathcal{L}_{\text{sparse}}, \tag{8}$$

with $\mathcal{L}_{\text{WAM}}$ identical to that of the matched Cosmos 3 baseline on the same demonstration windows (rectified-flow targets for noised future video tokens and the action chunk).

Compact state. The interface in equation 4 idealizes $C_t = \mathcal{S}(\hat{f}^{-1}(o_t))$ on encoder coordinates; on Cosmos 3 the same role is played by clean context tokens:

$$C_t \;=\; \Big\{ j \in \mathcal{J}_t \;\Big|\; \exists\, i :\; j \in T_k(i) \Big\}, \tag{9}$$

i.e. clean tokens in the top-k routed mass of some action query. The default policy retains the dense MoT visual to action path, and equation 7 regularizes the router branch of $\hat{\pi}$.

5 Experiments

We evaluate CWAM in two settings. On synthetic world action processes with known latent variables, the compact state C_t used by the sparse action head recovers the action-relevant block, which is consistent with Theorem 2 (Section 5.1). On RoboCasa365, LIBERO, and a real Franka robot, the sparse routing path improves aggregate closed-loop success over the matched Cosmos 3 baseline (Section 5.2), and on the four RoboCasa365 tasks where CWAM is strongest, the routed tokens alone keep most of that success (Section 5.3). Appendix C gives the synthetic protocol in full.

5.1 The compact state recovers the action-relevant block

Theorem 2 predicts that C_t is an invertible function of the action-relevant block s_{t,I_A} alone. Comparing C_t with this prediction needs the true latent variables and I_A, which a simulator provides.

Setup. The process follows equation 1 with $n = 20$ latent variables, $|I_A| = 4$ of which drive the action, and observes each latent variable through its own noisy coordinates. A hidden regime, never given to the estimator, scales the action mechanism and a transition that couples the nuisance variables to the action-relevant ones. Each estimator pairs an observation-only MLP encoder with a linear action head, whose weight matrix is its Jacobian, so CWAM with sparsity uses a group penalty on the head columns, a column-level surrogate for equation 3. The compact state C_t keeps the $|I_A|$

Table 1: The coordinates selected by CWAM with sparsity decode the action-relevant latent variables as well as the dense readout, and the nuisance variables only about as well as the true block. Zeroing them breaks the fitted head. Held-out linear probes give the relevant R^2 and Δ_{nuis}, the nuisance R^2 of C_t minus that of s_{t,I_A}. Selected/full (Complement/full) is the held-out action MSE of the fitted head, not refitted, after zeroing the coordinates outside (inside) C_t, relative to the full $\hat{s}_t$. Mean $\pm$ population std over five seeds, with ratios computed per seed.

Method	$\lvert C_t \rvert$	Relevant $R^2 \uparrow$	$\Delta_{\text{nuis}} \downarrow$	Selected/full	Complement/full
Dense (20D readout)	20	0.9961 ± 0.0015	0.646 ± 0.115	n/a	n/a
Random matched (4D)	4	0.5493 ± 0.2964	0.392 ± 0.144	48.067 ± 31.855	6.36 ± 6.42
CWAM without sparsity	4	0.9876 ± 0.0164	-0.008 ± 0.006	3.456 ± 3.180	50.58 ± 28.81
Standard MLP VAE	4	0.9440 ± 0.0448	0.316 ± 0.186	3.259 ± 2.211	51.13 ± 28.54
CWAM with sparsity	4	0.9957 ± 0.0014	0.039 ± 0.093	1.704 ± 1.398	54.65 ± 25.78

coordinates with the largest mean absolute head weight. The baselines are a standard MLP VAE (Zheng et al., 2026) fitted without head sparsity and three readouts of one fit without the penalty, namely the full latent state (dense), $|I_A|$ random coordinates (random matched), and the selected coordinates (CWAM without sparsity). Held-out linear probes give the R^2 from C_t to the action-relevant latent variables. For the nuisance variables we report $\Delta_{\text{nuis}} = R^2(C_t \to \text{nuis}) - R^2(s_{t,I_A} \to \text{nuis})$, the excess of C_t over the oracle reference, i.e., over how well the true block predicts them.

Block recovery. A block that depends on s_{t,I_A} alone can predict the nuisance variables only through their dependence on s_{t,I_A}, so under Theorem 2 the nuisance excess of C_t should be near zero. For the coordinates selected by CWAM with sparsity it is indeed close to zero, and they decode the action-relevant latent variables about as well as the dense readout (Table 1). In contrast, the dense readout and the coordinates selected from the VAE decode the nuisance variables well beyond the true block, and four random coordinates recover the action-relevant ones poorly.

Head ablation. A probe measures what C_t contains, while the fitted head may still draw on other coordinates. Setting C_t to zero raises the action error by a similarly large factor for every head that selected a block, including the VAE, while setting the complement to zero costs much less (Table 1). Random matched shows the opposite pattern, and keeping only its coordinates breaks the head, so the head depends on which coordinates are kept more than on how many.

Hidden environments. The theorems assume nothing about how unobserved regimes evolve, so hidden variation in the nuisance dynamics should leave the conclusion of Theorem 2 unchanged. A second hidden index e_t, resampled at every step and never given to the encoder, rescales only the nuisance variables in the transition, and we increase its number of values E with the action mechanism fixed. As E grows, neither readout loses accuracy on the action-relevant latent variables, and only the dense readout keeps a large nuisance excess (Appendix D, Table 4).

5.2 The sparse routing path improves closed-loop success

For a pretrained WAM, the true latent variables are unknown, so on Cosmos 3 we ask whether the sparse routing path improves closed-loop success. We compare with Cosmos 3 which is post-trained identically with matched setups but without the router and $\mathcal{L}_{\text{sparse}}$ (Section 4). CWAM succeeds more often on RoboCasa365 (Nasiriany et al., 2026), LIBERO, and a real Franka robot that stacks cubes (Figure 2).

On RoboCasa365, CWAM achieves a higher mean success rate than the baseline in every foundational skill family (Figure 2(a)). The gain is small for button pressing and largest for drawer opening and closing, rack sliding, insertion, and pick-and-place tasks. Averaged across tasks, CWAM's success rate is nearly twice that of the baseline (Table 5). On LIBERO (Liu et al., 2023a), both policies achieve high success rates, with CWAM outperforming the baseline across all suites and showing the largest gain on LIBERO-Goal (Figure 2(b)). In real-robot experiments, CWAM succeeds in a majority of trials, whereas the baseline succeeds in fewer than half (Figure 2(c)).

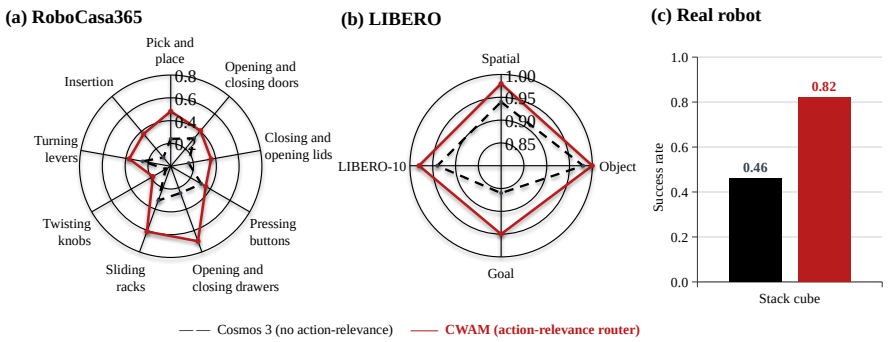


Figure 2: CWAM improves closed-loop success over the matched Cosmos 3 baseline in every RoboCasa365 skill family, every LIBERO suite, and on the real robot. (a) RoboCasa365, mean success per skill family over 17 tasks (tasks in Table 5). (b) LIBERO, success per suite over 40 tasks. (c) Franka stack cube, 50 trials per policy. Panels (a) and (b) use 100 episodes per task. The baseline (Cosmos 3 in the legend) is post-trained on the same data and recipe, without the router and $\mathcal{L}_{\text{sparse}}$.

Table 2: On the four RoboCasa365 tasks where CWAM succeeds most, keeping only the routed tokens costs far less success than keeping as many random tokens. The CWAM checkpoint is fixed at 50k steps and run on the episodes of the main evaluation. The action queries see the full visual context (CWAM), only the routed tokens that form C_t (C-only), or as many visual tokens drawn uniformly at random from the clean context (random matched). Neither intervention refits the policy.

Task	CWAM	C-only	Random matched
Pick place sink to counter	74%	63%	28%
Pick place toaster to counter	72%	61%	21%
Open drawer	70%	59%	33%
Open stand mixer head	69%	58%	16%
Average (4 tasks)	71.3%	60.3%	24.5%

Where does the router look? In sampled frames of the real-robot sequence in Figure 3 (right), routing weights are concentrated on the two cubes involved in the stacking task described in the introduction. The RoboCasa365 rollout (left) shows a similar pattern, with most of the weight assigned to the manipulated object and its target. In some frames, however, the weights are more dispersed, extending to the robot arm and the wall, as in the fourth frame of the real-robot sequence.

5.3 Routed tokens alone keep most of CWAM's success on its best tasks

Routing visualizations do not establish whether the routed tokens alone are sufficient for action prediction, since the default policy retains a dense visual-to-action pathway (Section 4). They also do not show whether a different set of tokens of the same size would support comparable performance. To test these questions, we evaluate a fixed CWAM checkpoint under two visual-context interventions on the four RoboCasa365 tasks with the highest success rates. *C-only* restricts the visual context available to the action queries to the routed tokens defining C_t in equation 9. *Random matched* instead provides the action queries with an equal number of randomly selected visual tokens.

As shown in Table 2, *C-only* retains most of the default policy's performance, whereas *Random matched* results in a substantially larger drop in success rate, with the same ordering across all four tasks. These results suggest that the routed tokens retain much of the visual information needed to perform these tasks. Consistent with the synthetic-head results, the gap between the two conditions shows that token selection affects performance even when the number of retained tokens is fixed.

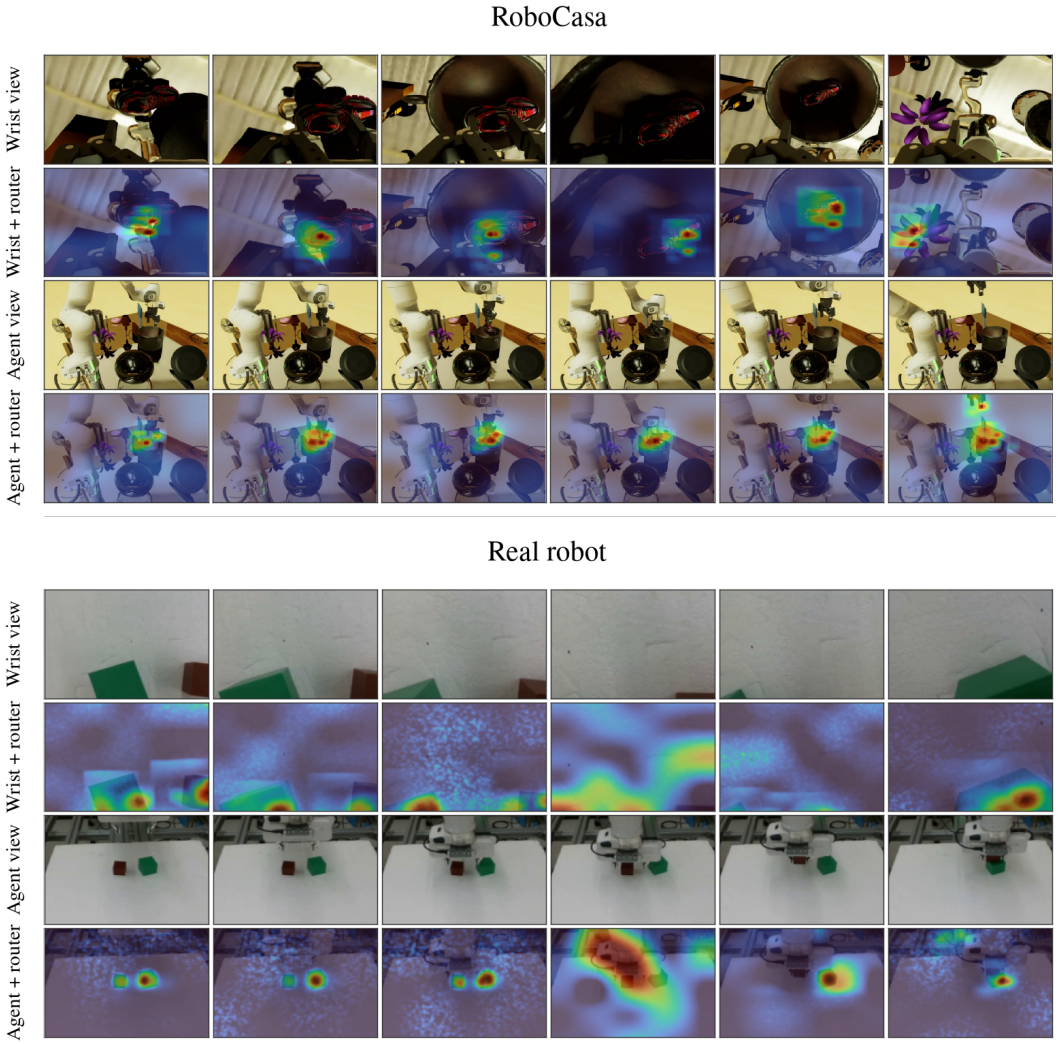


Figure 3: In most frames, the routing mass falls on the object being moved and its target. (Left) A RoboCasa365 rollout. (Right) The real robot stacking the red cube on the green cube. Rows show the wrist and agent (third-person) views, each followed by the same view with the routing mass overlaid. Warmer colors mark more of the top-k routing mass over clean visual tokens.

6 Conclusion

The latent variables of a world action model serve both prediction and action, but only the action side needs to be restricted to the action-relevant ones. We formalize this restriction and study when demonstrations alone guarantee it. Specifically, we prove that, under nondegeneracy conditions and without a sparsity constraint, each action dimension of any estimated mechanism that fits the demonstrations depends on at least as many latent variables as the true one. A sparsity constraint that caps the total number of these dependencies at the true number turns this bound into an equality and identifies the structure between latent states and actions up to a permutation of the latent variables. The latent variables used to generate actions are then an invertible function of the action-relevant ones alone, while the full estimated state still describes the whole scene for prediction. Neither identification result uses temporal structure or supervision. Synthetic experiments agree with the theory, and our method helps task completion in both simulation and on a real robot. The main limitation is the lack of component-wise differentiability that pints down individual latents, which may not be needed for separating actions but can be helpful for precise interventions.

AI STATEMENT

In this paper, we used generative AI tools to assist coding, language editing, and figure plotting. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Aapo Hyvärinen and Hiroshi Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ICA. In *Advances in Neural Information Processing Systems*, volume 29, pp. 3765–3773, 2016.

Aapo Hyvärinen and Hiroshi Morioka. Nonlinear ICA of temporally dependent stationary sources. In *Artificial Intelligence and Statistics*, pp. 460–469. PMLR, 2017.

Aapo Hyvärinen and Petteri Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(3):429–439, 1999.

Aapo Hyvärinen, Hiroaki Sasaki, and Richard Turner. Nonlinear ICA using auxiliary variables and generalized contrastive learning. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 859–868. PMLR, 2019.

Ilyes Khemakhem, Diederik P. Kingma, Roberto P. Monti, and Aapo Hyvarinen. Variational autoencoders and nonlinear ICA: A unifying framework. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 2207–2217, 2020. URL https://proceedings.mlr.press/v108/khemakhem20a.html.

Bohdan Kivva, Goutham Rajendran, Pradeep Ravikumar, and Bryon Aragam. Identifiability of deep generative models without auxiliary information. In *Advances in Neural Information Processing Systems*, volume 35, pp. 15687–15701, 2022.

Lingjing Kong, Shaoan Xie, Weiran Yao, Yujia Zheng, Guangyi Chen, Petar Stojanov, Victor Akinwande, and Kun Zhang. Partial disentanglement for domain adaptation. In *International Conference on Machine Learning*, 2022.

Sébastien Lachapelle, Pau Rodríguez López, Yash Sharma, Katie E Everett, Rémi Le Priol, Alexandre Lacoste, and Simon Lacoste-Julien. Disentanglement via mechanism sparsity regularization: A new principle for nonlinear ICA. In *Conference on Causal Learning and Reasoning*, pp. 428–484. PMLR, 2022.

Weixin Liang, Lili Yu, Liang Luo, Srinivasan Iyer, Ning Dong, Chunting Zhou, et al. Mixture-of-transformers: A sparse and scalable architecture for multi-modal foundation models. *arXiv preprint arXiv:2411.04996*, 2024. URL https://arxiv.org/abs/2411.04996.

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

Phillip Lippe, Sara Magliacane, Sindy Löwe, Yuki M Asano, Taco Cohen, and Stratis Gavves. CITRIS: Causal identifiability from temporal intervened sequences. In *International Conference on Machine Learning*, pp. 13557–13603. PMLR, 2022.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2023a.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023b.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning*, pp. 4114–4124. PMLR, 2019.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. RoboCasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2603.04356.

NVIDIA. Cosmos 3: Omnimodal world models for physical AI. *arXiv preprint arXiv:2606.02800*, 2026.

Xiangchen Song, Weiran Yao, Yewen Fan, Xinshuai Dong, Guangyi Chen, Juan Carlos Niebles, Eric Xing, and Kun Zhang. Temporally disentangled representation learning under unknown nonstationarity. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2310.18615.

Julius von Kügelgen, Yash Sharma, Luigi Gresele, Wieland Brendel, Bernhard Schölkopf, Michel Besserve, and Francesco Locatello. Self-supervised learning with data augmentations provably isolates content from style. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Dingling Yao, Danru Xu, Sébastien Lachapelle, Sara Magliacane, Perouz Taslakian, Georg Martius, Julius von Kügelgen, and Francesco Locatello. Multi-view causal representation learning with partial observability. In *International Conference on Learning Representations*, 2024.

Weiran Yao, Guangyi Chen, and Kun Zhang. Temporally disentangled representation learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2210.13647.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

Yujia Zheng, Ignavier Ng, and Kun Zhang. On the identifiability of nonlinear ICA: Sparsity and beyond. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Yujia Zheng, Fan Feng, Yuke Li, Shaoan Xie, Kevin Murphy, and Kun Zhang. From generalist to specialist representation, 2026. URL https://arxiv.org/abs/2605.12733.

Symbol	Meaning
$o \in \mathbb{R}^{d_o}$	Observation
$s \in \mathbb{R}^n$	Latent state
$a \in \mathbb{R}^{d_a}$	Action
f	Observation map $s \mapsto o$, diffeomorphism onto its image
$\hat{f}$	Learner observation map $\hat{s} \mapsto o$, diffeomorphism onto its image; its inverse $\hat{f}^{-1}$ is the encoder
π	Action mechanism $(s, \eta) \mapsto a$ in equation 1b
$\hat{\pi}$	Estimated action mechanism
η	Exogenous action noise (held fixed in J_π, $J_{\hat{\pi}}$)
$J_m(s)$	Jacobian of a vector-valued map m with respect to s
$\mathcal{I}(A)$	Index set of functionally nonzero entries of matrix or vector A
$\|\mathcal{I}(A)\|_0$	Cardinality of $\mathcal{I}(A)$ (support size, ℓ_0 norm)
$I_i \subseteq [n]$	Latent indices that drive action dimension $i \in [d_a]$
I_{A}	$\bigcup_{i=1}^{d_a} I_i$; action-relevant coordinate set
$s_{I_{\mathrm{A}}}$	Subvector of s on coordinates I_{A}
$\hat{I}_i$, $\hat{I}_{\mathrm{A}}$, C	Learner's index sets $\hat{I}_i = \{j : (i,j) \in \mathcal{I}(J_{\hat{\pi}})\}$ and $\hat{I}_{\mathrm{A}} = \bigcup_i \hat{I}_i$; compact state $C = \hat{s}_{\hat{I}_{\mathrm{A}}}$ (denoted C_t in Section 2)
Observational equivalence	Same law of (o, a); the results use the pathwise coupling $\hat{s} = \phi(s)$ with shared η, which implies it (Section 2)
ϕ	Reparameterization $\phi := \hat{f}^{-1} \circ f$, $\hat{s} = \phi(s)$
$P(\hat{s})$	$J_{\phi^{-1}}(\hat{s})$; chain rule links $J_{\hat{\pi}}$ and J_π
ρ	Permutation of $[n]$ aligning true and estimated coordinates
P_ρ	Permutation matrix, $(P_\rho)_{j,k} = \mathbf{1}\{\rho(j) = k\}$

Table 3: Summary of notation for the identifiability proofs.

A PROOFS

Notation. Table 3 summarizes the symbols used in the proofs below. The main text (Sections 2 and 3) indexes samples by time t as in equation 1a–equation 1b; the proofs fix one step, drop the index, and write $o = f(s)$, $a = \pi(s, \eta)$. Identifiability does not use cross-time structure, so this is equivalent to applying the argument at each t separately.

A.1 PROOF OF PROPOSITION 1

Proposition 1. Consider two observationally equivalent models $o = f(s) = \hat{f}(\hat{s})$, $a = \pi(s, \eta) = \hat{\pi}(\hat{s}, \eta)$ as in Table 3. Assume that, for each action dimension $i \in [d_a]$, there exist $|I_i|$ states $s^{(1)}, \ldots, s^{(|I_i|)}$ at which the rows $(J_\pi(s^{(l)}))_{i,\cdot}$ are linearly independent, and for $\phi = \hat{f}^{-1} \circ f$, $P = J_{\phi^{-1}}$, and a matrix M with $\mathcal{I}(M) = \mathcal{I}(P)$, $\mathcal{I}\big((J_\pi(s^{(l)})M)_{i,\cdot}\big) \subseteq \mathcal{I}\big((J_{\hat{\pi}}(\hat{s}^{(l)}))_{i,\cdot}\big)$ with $\hat{s}^{(l)} = \phi(s^{(l)})$, and that the regularity conditions of Section 2 hold. Then, for every action dimension $i \in [d_a]$,

$$\|\mathcal{I}\big((J_{\hat{\pi}})_{i,\cdot}\big)\|_0 \;\geq\; \|\mathcal{I}\big((J_\pi)_{i,\cdot}\big)\|_0. \tag{10}$$

Proof. Step 1 (reparameterization). The pathwise coupling in the hypothesis gives, for all s,

$$f(s) = \hat{f}(\phi(s)), \qquad \phi := \hat{f}^{-1} \circ f, \qquad \hat{s} = \phi(s), \tag{11}$$

and ϕ is a diffeomorphism of $\mathbb{R}^n$ by the regularity conditions of Section 2. For fixed η, the same coupling gives the following identity almost surely, hence for all s by continuity and full support,

$$\pi(s, \eta) = \hat{\pi}(\phi(s), \eta) \qquad \text{for all } s. \tag{12}$$

Differentiating in s and applying the chain rule,

$$\frac{\partial \pi}{\partial s}(s, \eta) = \frac{\partial \hat{\pi}}{\partial \hat{s}}(\phi(s), \eta)\, \frac{\partial \phi}{\partial s}(s) = J_{\hat{\pi}}(\phi(s))\, J_\phi(s), \tag{13}$$

and $J_\phi(s) = \big(J_{\phi^{-1}}(\phi(s))\big)^{-1}$, so with $P(\hat{s}) := J_{\phi^{-1}}(\hat{s})$,

$$J_{\hat{\pi}}(\phi(s)) = J_\pi(s)\, P(\phi(s)). \tag{14}$$

Step 2 (nondegenerate rows). Fix $i \in [d_a]$. Let $s^{(1)}, \dots, s^{(|I_i|)}$ be as in the proposition, so the rows $(J_\pi(s^{(l)}))_{i,\cdot}$, $l = 1, \dots, |I_i|$, are linearly independent. Write $\hat{s}^{(l)} = \phi(s^{(l)})$ and let M be the matrix in the hypothesis, with $\mathcal{I}(M) = \mathcal{I}(P)$. The assumed inclusion is, for each l,

$$\mathcal{I}\big((J_\pi(s^{(l)})M)_{i,\cdot}\big) \subseteq \mathcal{I}\big((J_{\hat{\pi}}(\hat{s}^{(l)}))_{i,\cdot}\big). \tag{15}$$

Write $S_i := \mathcal{I}\big((J_{\hat{\pi}})_{i,\cdot}\big)$ for the functional row support (Section 2) and $S^{(l)} := \mathcal{I}\big((J_{\hat{\pi}}(\hat{s}^{(l)}))_{i,\cdot}\big)$. If a row vector $v \in \mathbb{R}^n$ satisfies $\mathcal{I}(v) \subseteq T$, then

$$v = \sum_{r \in T} v_r e_r \in \operatorname{span}\{e_r : r \in T\}. \tag{16}$$

By equation 15, $\mathcal{I}\big((J_\pi(s^{(l)})M)_{i,\cdot}\big) \subseteq S^{(l)}$. Each $S^{(l)}$ is contained in S_i: an entry of $J_{\hat{\pi}}$ that is nonzero at $\hat{s}^{(l)}$ is nonzero on a neighborhood of $\hat{s}^{(l)}$ by continuity, and this neighborhood has positive probability because $\hat{s} = \phi(s)$ has a positive density on $\mathbb{R}^n$ (Appendix B.1). Applying the display above with $v = (J_\pi(s^{(l)})M)_{i,\cdot}$ gives

$$(J_\pi(s^{(l)})M)_{i,\cdot} \in \operatorname{span}\{e_r : r \in S_i\}, \qquad l = 1, \dots, |I_i|. \tag{17}$$

Step 3 (from row span to every row of M on I_i). Fix $j \in I_i$. By definition of I_i, coordinate $k \notin I_i$ implies $(J_\pi(s))_{i,k} = 0$ almost surely, hence everywhere by continuity and full support, and in particular $(J_\pi(s^{(l)}))_{i,k} = 0$ for every l. Thus each row $v^{(l)} := (J_\pi(s^{(l)}))_{i,\cdot}$ lies in $V := \operatorname{span}\{e_k : k \in I_i\}$. The $|I_i|$ rows $v^{(1)}, \dots, v^{(|I_i|)}$ are linearly independent and contained in the $|I_i|$-dimensional space V, so they form a basis of V. In particular $e_j^\top \in V$, and there exist $\beta_1, \dots, \beta_{|I_i|}$ with

$$e_j^\top = \sum_{l=1}^{|I_i|} \beta_l \, v^{(l)} = \sum_{l=1}^{|I_i|} \beta_l \, (J_\pi(s^{(l)}))_{i,\cdot}. \tag{18}$$

Right-multiplying by M and using equation 17,

$$M_{j,\cdot} = e_j^\top M = \sum_{l=1}^{|I_i|} \beta_l \, (J_\pi(s^{(l)})M)_{i,\cdot} \in \operatorname{span}\{e_r : r \in S_i\}, \tag{19}$$

because each term lies in $\operatorname{span}\{e_r : r \in S_i\}$ and that set is a linear subspace. Record

$$M_{j,\cdot} \in \operatorname{span}\{e_r : r \in S_i\}. \tag{20}$$

Step 4 (a permutation common to all action dimensions). Fix a point $\hat{s}^* \in \mathbb{R}^n$, the same for every i. The matrix $P(\hat{s}^*)$ is invertible, so $\det P(\hat{s}^*) \neq 0$. By the Leibniz formula,

$$\det P(\hat{s}^*) = \sum_{\sigma \in \mathfrak{S}_n} \operatorname{sgn}(\sigma) \prod_{k=1}^{n} P_{k,\sigma(k)}(\hat{s}^*), \tag{21}$$

where $\mathfrak{S}_n$ is the set of permutations of $[n]$. Some summand is nonzero; fix one such $\rho \in \mathfrak{S}_n$, chosen once for all i, so that

$$P_{j,\rho(j)}(\hat{s}^*) \neq 0 \qquad \text{for all } j \in [n]. \tag{22}$$

By continuity of P, these entries are nonzero on a neighborhood of $\hat{s}^*$, and this neighborhood has positive probability because $\hat{s} = \phi(s)$ has a positive density on $\mathbb{R}^n$. Hence $(j, \rho(j)) \in \mathcal{I}(P) = \mathcal{I}(M)$, that is,

$$M_{j,\rho(j)} \neq 0 \qquad \text{for all } j \in [n]. \tag{23}$$

Step 5 (conclusion). Fix $j \in I_i$. Vectors in $\operatorname{span}\{e_r : r \in S_i\}$ vanish off S_i, so equation 20 gives $M_{j,k} = 0$ for $k \notin S_i$; in particular, $M_{j,\rho(j)} = 0$ if $\rho(j) \notin S_i$. Step 4 gives $M_{j,\rho(j)} \neq 0$, hence

$$\rho(j) \in S_i = \mathcal{I}\big((J_{\hat{\pi}})_{i,\cdot}\big). \tag{24}$$

This holds for every $j \in I_i$, so $\rho(I_i) \subseteq S_i$ and

$$\|\mathcal{I}((J_{\hat{\pi}})_{i,\cdot})\|_0 = |S_i| \geq |\rho(I_i)| = |I_i| = \|\mathcal{I}((J_\pi)_{i,\cdot})\|_0, \tag{25}$$

where $|\rho(I_i)| = |I_i|$ because ρ is a bijection of $[n]$. Step 4 does not use i, so the same ρ serves every action dimension i. □

A.2 Proof of Theorem 1

Theorem 1. Consider two observationally equivalent models as in Proposition 1 and assume support sparsity equation 3. Then there is a permutation ρ of $[n]$ such that $\mathcal{I}(J_{\hat{\pi}}) = \mathcal{I}(J_\pi)P_\rho$ and $\mathcal{I}((J_{\hat{\pi}})_{i,\cdot}) = \rho(I_i)$ for every $i \in [d_a]$.

Proof. Step 1 (one-sided row supports). Apply Steps 1 to 5 of the proof of Proposition 1 to each $i \in [d_a]$, with the same $\hat{s}^*$ and the same ρ in Step 4. This gives one permutation ρ of $[n]$, common to all i, such that

$$\rho(I_i) \subseteq \mathcal{I}\big((J_{\hat{\pi}})_{i,\cdot}\big), \tag{26}$$

hence

$$\|\mathcal{I}((J_{\hat{\pi}})_{i,\cdot})\|_0 \;\geq\; |\rho(I_i)| = |I_i| = \|\mathcal{I}((J_\pi)_{i,\cdot})\|_0. \tag{27}$$

Step 2 (global one-sided bound). Each index pair $(i, j) \in \mathcal{I}(J_\pi)$ satisfies $j \in I_i$, so

$$\|\mathcal{I}(J_\pi)\|_0 = \sum_{i=1}^{d_a} \|\mathcal{I}((J_\pi)_{i,\cdot})\|_0 = \sum_{i=1}^{d_a} |I_i|. \tag{28}$$

Similarly,

$$\|\mathcal{I}(J_{\hat{\pi}})\|_0 = \sum_{i=1}^{d_a} \|\mathcal{I}((J_{\hat{\pi}})_{i,\cdot})\|_0. \tag{29}$$

Summing equation 27 over i gives

$$\|\mathcal{I}(J_{\hat{\pi}})\|_0 \;\geq\; \|\mathcal{I}(J_\pi)\|_0. \tag{30}$$

Step 3 (sparsity forces equalities). Support sparsity equation 3 gives

$$\|\mathcal{I}(J_{\hat{\pi}})\|_0 \leq \|\mathcal{I}(J_\pi)\|_0. \tag{31}$$

Together with equation 30,

$$\|\mathcal{I}(J_{\hat{\pi}})\|_0 = \|\mathcal{I}(J_\pi)\|_0. \tag{32}$$

If some row inequality in equation 27 were strict, summing would make equation 30 strict, a contradiction. Therefore equation 27 holds with equality for every i, and with $\rho(I_i) \subseteq \mathcal{I}((J_{\hat{\pi}})_{i,\cdot})$ this implies

$$\mathcal{I}\big((J_{\hat{\pi}})_{i,\cdot}\big) = \rho(I_i) \qquad \text{for each } i \in [d_a]. \tag{33}$$

Step 4 (support-level equivalence). For every $i \in [d_a]$ and $j \in [n]$, the definition of I_i, the bijectivity of ρ, and equation 33 give

$$(i, j) \in \mathcal{I}(J_\pi) \iff j \in I_i \iff \rho(j) \in \rho(I_i) = \mathcal{I}\big((J_{\hat{\pi}})_{i,\cdot}\big) \iff (i, \rho(j)) \in \mathcal{I}(J_{\hat{\pi}}). \tag{34}$$

Step 5 (matrix support). By definition of P_ρ, $(i, k) \in \mathcal{I}(J_\pi)P_\rho$ if and only if $(i, \rho^{-1}(k)) \in \mathcal{I}(J_\pi)$. For $k = \rho(j)$ this is equivalent to $(i, j) \in \mathcal{I}(J_\pi)$, i.e. to $(i, \rho(j)) \in \mathcal{I}(J_{\hat{\pi}})$ by equation 34. Hence

$$\mathcal{I}(J_{\hat{\pi}}) = \mathcal{I}(J_\pi)P_\rho. \tag{35}$$

□

A.3 Proof of Theorem 2

Theorem 2. Under the hypotheses of Theorem 1, with the same permutation ρ, there exists an invertible map h such that $C = \hat{s}_{\rho(I_A)} = h(s_{I_A})$ (equivalently $C_t = \hat{s}_{t,\rho(I_A)} = h(s_{t,I_A})$ in Section 2).

Proof. Assume equation 33 and fix the permutation ρ from Theorem 1.

Step 1 (a zero block of P). Take any $i \in [d_a]$, $j \in I_i \subseteq I_A$, and $r \in [n] \setminus I_A$. Suppose $P_{j,\rho(r)} \neq 0$ for some s. By equation 33, $\mathcal{I}((J_{\hat{\pi}})_{i,\cdot}) = \rho(I_i)$, so $\rho(r) \in \rho(I_i)$ implies

$$r = \rho^{-1}(\rho(r)) \in I_i, \tag{36}$$

using $\rho(I_i) = \{\rho(j) : j \in I_i\}$. This contradicts $r \notin I_{\mathrm{A}}$. Hence $P_{j,\rho(r)} = 0$ for all s and all such (j, r). Because $P = J_{\phi^{-1}}$ is continuous, these entries vanish identically on $\mathbb{R}^n$:

$$P_{j,\rho(r)} = 0 \qquad \text{for all } j \in I_{\mathrm{A}},\ r \in [n] \setminus I_{\mathrm{A}}. \tag{37}$$

The same argument with I_{A} in place of I_i gives equation 37 for every $j \in I_{\mathrm{A}}$.

Step 2 (block triangular form). Order rows of P as $(I_{\mathrm{A}}, [n]\setminus I_{\mathrm{A}})$ and columns as $(\rho(I_{\mathrm{A}}), \rho([n]\setminus I_{\mathrm{A}}))$. Then equation 37 is exactly $P_{j,\rho(r)} = 0$ for $j \in I_{\mathrm{A}}$ and $r \in [n] \setminus I_{\mathrm{A}}$, i.e. the upper-right block of P vanishes:

$$P = \begin{pmatrix} P_{11} & 0 \\ P_{21} & P_{22} \end{pmatrix}, \qquad P_{11} \in \mathbb{R}^{|I_{\mathrm{A}}|\times|\rho(I_{\mathrm{A}})|}, \quad P_{22} \in \mathbb{R}^{(n-|I_{\mathrm{A}}|)\times(n-|\rho(I_{\mathrm{A}})|)}. \tag{38}$$

Because P is invertible,

$$\det P = \det P_{11} \det P_{22} \neq 0, \tag{39}$$

so P_{11} and P_{22} are invertible. For an invertible block-triangular matrix,

$$P^{-1} = \begin{pmatrix} P_{11}^{-1} & 0 \\ -P_{22}^{-1} P_{21} P_{11}^{-1} & P_{22}^{-1} \end{pmatrix}, \tag{40}$$

which is checked by direct multiplication. Since $P^{-1} = J_\phi(s)$, the upper-right block of P^{-1} is zero, so

$$\frac{\partial \hat{s}_{\rho(j)}}{\partial s_r} = 0 \qquad \text{for } j \in I_{\mathrm{A}},\ r \in [n] \setminus I_{\mathrm{A}}; \tag{41}$$

equation 37 gives

$$\frac{\partial s_j}{\partial \hat{s}_{\rho(r)}} = 0 \qquad \text{for the same index pairs.} \tag{42}$$

Step 3 (block-wise functional dependence). Because ϕ and ϕ^{-1} are diffeomorphisms on $\mathbb{R}^n$, if $\partial \hat{s}_{\rho(j)}/\partial s_r \equiv 0$ for $j \in I_{\mathrm{A}}$ and $r \in [n] \setminus I_{\mathrm{A}}$, then $\hat{s}_{\rho(j)}$ does not depend on s_r as a function on $\mathbb{R}^n$; likewise $\partial s_j/\partial \hat{s}_{\rho(r)} \equiv 0$ implies s_j does not depend on $\hat{s}_{\rho(r)}$. Thus there exist continuously differentiable maps h and $\tilde{h}$ such that

$$\hat{s}_{\rho(I_{\mathrm{A}})} = h(s_{I_{\mathrm{A}}}), \qquad s_{I_{\mathrm{A}}} = \tilde{h}(\hat{s}_{\rho(I_{\mathrm{A}})}). \tag{43}$$

On the image of h, $\tilde{h} \circ h$ is the identity on $s_{I_{\mathrm{A}}}$ and $h \circ \tilde{h}$ is the identity on $\hat{s}_{\rho(I_{\mathrm{A}})}$, so h is invertible with inverse $\tilde{h}$. The Jacobian of h is the upper-left block of P^{-1}, hence

$$J_h = P_{11}^{-1}. \tag{44}$$

Step 4 (compact state). By equation 33, $\hat{I}_i = \mathcal{I}((J_{\hat{\pi}})_{i,\cdot}) = \rho(I_i)$ for every i, so $\hat{I}_{\mathrm{A}} = \bigcup_i \rho(I_i) = \rho(I_{\mathrm{A}})$ because ρ is a bijection. Step 3 therefore gives

$$C = \hat{s}_{\hat{I}_{\mathrm{A}}} = \hat{s}_{\rho(I_{\mathrm{A}})} = h(s_{I_{\mathrm{A}}}), \tag{45}$$

which is Theorem 2 with time subscript t restored ($C_t = \hat{s}_{t,\rho(I_{\mathrm{A}})}$). □

B Supplementary discussions

B.1 Detailed discussion on main conditions

The assumptions are those in Proposition 1. They require that each action dimension i has enough variation in $(J_\pi(s))_{i,\cdot}$ across $|I_i|$ states to span the true parents I_i, and that reparameterization through ϕ cannot hide a true dependency in $\hat{\pi}$ by cancellation at every such state. Both parts are mild in the usual smooth, nonparametric regime.

Span. The span condition prevents pathological cases where all demonstrations lie in a region so small that the rows $(J_\pi(s^{(l)}))_{i,\cdot}$ never span I_i, even though π_i functionally depends on those coordinates globally. Identifiability is asymptotic, so the assumption does not require finite datasets to

cover the states $s^{(l)}$ uniformly. If the latent state has a continuous density and π is smooth, then $(J_\pi(s))_{i,\cdot}$ is a continuous random vector in general position at random s, so $|I_i|$ independent draws suffice with probability one.

Support inclusion. The identity equation 14 implies that rows of $J_\pi(s)M$ with $\mathcal{I}(M) = \mathcal{I}(P)$ typically lie inside the corresponding rows of $J_{\hat{\pi}}(\hat{s})$. The inclusion assumed in Proposition 1 must hold at the $|I_i|$ states used for the span argument; it excludes measure-zero coincidences where partial derivatives cancel on all of those states simultaneously.

Functional row support. In the proof of Proposition 1, $S^{(l)} := \mathcal{I}\big((J_{\hat{\pi}}(\hat{s}^{(l)}))_{i,\cdot}\big)$ is compared with the functional row support $S_i = \mathcal{I}\big((J_{\hat{\pi}})_{i,\cdot}\big)$ from Section 2. If $(J_{\hat{\pi}}(\hat{s}^{(l)}))_{i,k} \neq 0$, then, since $\hat{\pi}$ is continuously differentiable, $J_{\hat{\pi}}(\hat{s})_{i,k} \neq 0$ on a neighborhood of $\hat{s}^{(l)}$, so $k \in S_i$ because $\hat{s} = \phi(s)$ has a positive density on $\mathbb{R}^n$ (Section 2).

C Evaluation protocol

Estimands. Block recovery is assessed with a ridge probe from C_t to the true latent variables of the synthetic process, fitted and reported on disjoint episodes, and reported as held-out R^2 for the action-relevant and for the nuisance variables. The same probe from the true s_{t,I_A} to the nuisance variables gives the oracle reference. On the robot benchmarks, routing mass is visualized in Figure 3, and on four RoboCasa365 tasks, closed-loop success with only the routed tokens is measured in Table 2.

Synthetic generator. The simulator implements equation 1a–equation 1b and adds a latent transition and a hidden regime that scales the action mechanism and the transition, to stress nuisance coupling; those extras are not part of the identification assumptions in Section 2. Latent $s_t \in \mathbb{R}^{20}$ with $I_\mathrm{A} = \{1, \dots, 4\}$. A hidden regime $c_t \in \{0, 1, 2\}$ follows a Markov chain with Dirichlet-random transition matrix and initial law. Action $a_t = M_{c_t}\, s_{t,I_\mathrm{A}} + 0.05\,\eta_t$, where $M_c = (1 + 0.15(c-1))M$ with Gaussian $M \in \mathbb{R}^{8\times 4}$; transition $s_{t+1} = \tanh\big((1 + 0.15\,c_t)(W_s s_t + W_a a_t)\big) + 0.05\,\epsilon_{t+1}$ with a dense Gaussian W_s (diagonal 0.85) and W_a feeding the action back into I_A only, so nuisance factors are coupled to relevant ones through W_s; observation $o_t = \tanh(W_\mathrm{mix} s_t) + 0.05\,\xi_t$ with block-diagonal Gaussian $W_\mathrm{mix} \in \mathbb{R}^{100\times 20}$ (five observation coordinates per latent factor, no cross-block entries). The environment suite draws $e_t \in [E]$ i.i.d. at every step and multiplies the nuisance coordinates of s_{t+1} by a per-environment factor in $[0.5, 1.5]$; the action family is unchanged across E. The regime thus modulates behavior and transition without changing the four-factor action support. It is not supplied to the estimator, and the fitted single-frame head learns a marginalized predictor rather than an explicitly recovered conditional policy. Simulator streams are seeded so that exogenous noise is paired across E.

Synthetic estimators and metrics. The encoder is an MLP $100 \to 256 \to 20$ with $\tanh$ followed by a non-affine LayerNorm, the action head is linear $20 \to 8$, and we use Adam with batch size 512. Development seeds $900, 901$ select learning rate from $\{3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}\}$ and epochs from $\{80, 160\}$, with 320 epochs tried only if no candidate meets the predeclared development targets. All six candidates meet the development-mean targets; minimum selected-state action MSE selects learning rate 10^{-3} and 160 epochs. This setting is shared by all baselines and evaluated once on seeds 200–204, held out from development tuning. The current action is a target, never an encoder input. Dense, random and λ=0 readouts share weights, and $E = 1$ is reused across tables: 70 entries come from 50 saved estimators, all reloaded for metric replay. CWAM penalty: mean over latent coordinates of the ℓ_2 norm of the corresponding head column, weight 0.01. VAE (Table 1): Gaussian posterior, MLP decoder, KL weight 10^{-3}, head sparsity weight 0; an optional VAE + ℓ_1 baseline (0.05 times the mean absolute head weight) is trained in the same matrix but omitted from the table. Support: the four coordinates with the largest mean absolute head weight; random matched: four uniformly random coordinates. Probes: ridge ($\alpha = 1$) fitted on 250 calibration episodes, R^2 averaged over target dimensions on 250 disjoint reporting episodes. Sel./full and Compl./full apply the fitted head to $\hat{s}_t$ with the complement (resp. C_t) set to zero on the reporting episodes.

Synthetic splits, readouts, and the oracle reference. Each fit uses 2,000 training episodes and 500 held-out episodes of length 128. The hidden regime c_t enters the action mechanism and the transition in every reported run. The dense readout applies the probes to all 20 coordinates of

Table 4: Block recovery by CWAM with sparsity does not degrade as hidden environments are added. At every E it decodes the action-relevant block as well as the dense readout, which alone decodes nuisance variables beyond the true block. Each hidden environment rescales the nuisance variables in the transition, with the action mechanism fixed across E, and is never an encoder input. Metrics as in Table 1, whose fits give the row $E = 1$. Mean $\pm$ population std over five seeds.

	Dense (20D readout)		CWAM with sparsity (4D)	
E	Relevant $R^2 \uparrow$	$\Delta_{\text{nuis}} \downarrow$	Relevant $R^2 \uparrow$	$\Delta_{\text{nuis}} \downarrow$
1	0.9961 ± 0.0015	0.646 ± 0.115	0.9957 ± 0.0014	0.039 ± 0.093
2	0.9961 ± 0.0016	0.640 ± 0.110	0.9961 ± 0.0016	-0.004 ± 0.003
3	0.9961 ± 0.0017	0.677 ± 0.109	0.9961 ± 0.0017	-0.004 ± 0.003
5	0.9961 ± 0.0017	0.683 ± 0.114	0.9961 ± 0.0017	-0.003 ± 0.002

$\hat{s}_t$. Random matched and CWAM without sparsity are taken from the same fit, trained with head sparsity weight $\lambda = 0$, and CWAM without sparsity keeps its four coordinates with the largest mean absolute head weight. CWAM with sparsity and the standard MLP VAE are separate fits, and C_t is selected from the head of the VAE in the same way. Every top-4 selection uses the true size $|I_{\text{A}}| = 4$. Because the transition couples the nuisance variables to the action-relevant ones, the true block already predicts the nuisance variables with $R^2 > 0$. The oracle reference $R^2(s_{t,I_{\text{A}}} \to \text{nuis})$ measures this coupling with the same linear probe and splits, and Δ_{nuis} subtracts it from the nuisance R^2 of C_t seed by seed before averaging. The reference is specific to this simulator and is computed from the true coordinates $s_{t,I_{\text{A}}}$. Since $C_t = \hat{s}_{t,\hat{I}_{\text{A}}}$ passes through a nonlinear encoder and need not align with $s_{t,I_{\text{A}}}$ coordinate by coordinate, a linear probe from C_t can land slightly above or below the reference even when block recovery is good, so Δ_{nuis} can be slightly negative. The environment suite uses $E \in \{1, 2, 3, 5\}$, and neither c_t nor e_t is an input to the encoder. Tables report the mean and the population standard deviation over seeds, with ratios computed per seed. Theorem 2 identifies the action-relevant latent variables up to a block-wise invertible transformation, so we measure block-level recovery with linear-probe R^2 and do not report the componentwise matched MCC.

D HIDDEN ENVIRONMENTS (SYNTHETIC)

The environment index $e_t \in [E]$ in Appendix C rescales nuisance transition coordinates while the action mechanism is fixed. Table 4 reports probes of the dense readout and of CWAM with sparsity as E grows (mean $\pm$ std over five seeds); metrics match Table 1.

E ROBOCASA365 PER-TASK SUCCESS

The RoboCasa365 panel of Figure 2 reports the mean success within each foundational skill family over the 17 evaluation tasks. Table 5 lists the per-task success rates behind those family means, together with the macro average over all tasks.

Robot evaluation. The matched Cosmos 3 baseline and CWAM are post-trained on the same demonstrations with the same recipe and optimization, and CWAM differs only by the action-relevance router and $\mathcal{L}_{\text{sparse}}$ (Section 4), so the baseline has no router. RoboCasa365 is evaluated on 17 tasks with 100 episodes per task, LIBERO on 40 tasks in four suites with 100 episodes per task, and the Franka stack-cube task with 50 trials per policy. For the context ablation in Table 2, the CWAM checkpoint is fixed, and only the visual tokens that the action queries attend to are changed at evaluation, with no refitting and with the episode keys of the main evaluation. C-only keeps the routed top-k tokens that define C_t in equation 9 and zeroes the other visual context to the action head. Random matched keeps the same number of visual tokens, drawn uniformly at random from the clean context. The four ablation tasks are those with the highest CWAM success in Table 5.

Table 5: CWAM raises success on 13 of the 17 RoboCasa365 tasks and the macro average from 0.24 to 0.43. Each cell is the success rate over 100 episodes. Tasks are grouped by skill family, and each point in Figure 2(a) averages the tasks of one family. Baseline and CWAM are as in Figure 2.

Foundational skill family	**Task**	**Baseline**	**CWAM**
Pick and place	PickPlaceCtrToCab	.31	.43
	PickPlaceCtrToStove	.22	.45
	PickPlaceDrawerToCtr	.15	.07
	PickPlaceSinkToCtr	.24	.74
	PickPlaceToasterToCtr	.28	.72
Opening / closing doors	CloseFridge	.14	.61
	CloseToasterOvenDoor	.51	.37
	OpenCabinet	.31	.24
Closing / opening lids	CloseBlenderLid	.02	.03
	OpenStandMixerHead	.30	.69
Pressing buttons	TurnOnMicrowave	.27	.26
	TurnOnElectricKettle	.35	.45
Opening / closing drawers	OpenDrawer	.25	.70
Sliding racks	SlideDishwasherRack	.32	.61
Twisting knobs	TurnOffStove	.05	.18
Turning levers	TurnOnSinkFaucet	.25	.37
Insertion	CoffeeSetupMug	.10	.37
Macro (unweighted mean over 17 tasks)		.24	.43

F COSMOS 3 POST-TRAINING SETUP

This appendix records the matched baseline and CWAM post-training recipe used for Section 5.2. The architectural difference is the action-relevance router and $\mathcal{L}_{\text{sparse}}$ (Appendix H); optimization, data, packing, and rectified-flow losses are otherwise identical (Appendix G).

F.1 TRAINING STAGES AND SERVED CHECKPOINT

We warm-start from the public `nvidia/Cosmos3-Nano` checkpoint converted to distributed checkpoint (DCP) format.

Stage 1 (initialization) trains experiment for 10,000 optimizer steps on RoboCasa365 with the router enabled and freshly initialized .

Stage 2 (continue) resumes from step 10,000 and runs to 50,000 steps with a flat LR multiplier ; scheduler state is not resumed. The matched baseline uses the same recipe with router disabled(λ=0).

Training runs on one 8×H200 node (FSDP shard degree 8), bf16, selective activation checkpointing on FMHA, and precomputed Wan 2.2 video latents on .

F.2 DEMONSTRATION DATA RECIPE (ROBOCASA365)

Corpus. Processed LeRobot-style RoboCasa365 demonstrations from the Cosmos 3 pipeline.

Windowing and modalities. Policy mode at 20 Hz: `chunk_length=16` frames per training window; the leading frame is observed and later video tokens are noised under the rectified-flow objective. Observations use `camera_mode=concat_view` (third-person and wrist, each 256×256, concatenated to 256×512 before the model canvas). Language follows `caption_mode=policy` with JSON-formatted prompts. Actions are `frame_wise_relative` rot6d (10D: position, rotation, gripper) with `quantile_rot` normalization and embodiment tag `robocasa365`. Classifier-free guidance dropout on conditioning is 0.1 on train and 0 on val.

Table 6: Shared Cosmos 3 RoboCasa365 post-training hyperparameters (baseline and CWAM).

Setting	Value
Optimizer	FusedAdamW (β_1=0.9, β_2=0.99, ϵ=10^{-8})
Base learning rate	5×10^{-5}
LR multipliers	5.0 on `action2llm`, `llm2action`, `action_modality_embed`
Weight decay	0.05
Gradient clipping	1.0 (force finite)
Precision	bf16
Parallelism	FSDP, 8 GPU shard / 1 replicate
Grad accumulation	2 (Stages 1 and 2)
Stage 1 schedule	linear warmup 500 steps, cycle 16,000, `max_iter=10,000`
Stage 2 schedule	flat multiplier 0.3871 (LR at step 10,000), cycle 90,000, `max_iter=50,000`
Checkpointing	save every 500 (Stage 1) / 10,000 (Stage 2); EMA rate 0.1
Rectified-flow loss scale	10.0 on generator (RoboCasa recipe override)
Action loss weight	10.0
CFG dropout (train)	0.1

F.3 Closed-loop task protocols

RoboCasa365 (17 tasks). We evaluate the *target* split on the fixed task list used in : `CloseBlenderLid`, `CloseFridge`, `CloseToasterOvenDoor`, `CoffeeSetupMug`, `OpenCabinet`, `OpenDrawer`, `OpenStandMixerHead`, `PickPlaceCounterToCabinet`, `PickPlaceCounterToStove`, `PickPlaceDrawerToCounter`, `PickPlaceSinkToCounter`, `PickPlaceToasterToCounter`, `SlideDishwasherRack`, `TurnOffStove`, `TurnOnElectricKettle`, `TurnOnMicrowave`, `TurnOnSinkFaucet`. Each task: 100 **episodes**, base seed 42 (paired keys across intervention conditions in Appendix E), 400 environment steps cap, 12-step open-loop action horizon (replan after each chunk).

LIBERO (40 tasks). Closed-loop evaluation uses the **LIBERO-post-trained** baseline and CWAM checkpoint (Appendix F.2). All LIBERO tasks are grouped into the four standard suites (40 tasks total); per-suite success rates appear in Figure 2(b). Protocol details follow the Cosmos 3 LIBERO closed-loop stack (20 Hz, quantile rotation actions, concatenated views).

F.4 Shared optimization hyperparameters

Table 6 lists settings shared by baseline and CWAM; router-only fields are deferred to Appendix H.

G Cosmos 3 base model and rectified-flow objective

Backbone. Cosmos 3-Nano: Qwen3-VL-8B-Instruct reasoner inside a mixture-of-transformers (MoT) generator with `joint_attn_implementation=two_way`, video generation enabled (`vision_gen=True`), and action generation enabled (`action_gen=True`). Diffusion expert loads pretrained weights; RoboCasa365 post-training *reinitializes* action adapters (`action2llm`, `llm2action`, `action_modality_embed`, `action_pos_embed`) from scratch while keeping the public video/reasoner stack.

Tokenization. Text: staged local Qwen3-VL processor (`QWEN3_VL_TOKENIZER_PATH`). Video: Wan 2.2 VAE latents (48 channels) with training-time exact encode durations $\{17, 61, 73\}$ frames when an online VAE is used; production runs use precomputed latents with metadata-only validation. Max packed tokens after packing: 74,000 for this recipe (override of the Nano default 45,056).

Objective $\mathcal{L}_{\text{WAM}}$. Joint rectified-flow training on noised future video tokens and the executed action chunk, conditioned on clean history and language. Training-time distributions: `waver` on video, `logitnormal` on image/action/sound branches; `loss_scale=10.0` on the generator path in the RoboCasa365 config; `action_loss_weight=10.0`. Inference defaults

(`rectified_flow_inference_config`): UniPC, 1000 train timesteps, shift 1. The baseline and CWAM share this loss; CWAM adds $\mathcal{L}_{\text{sparse}}$ only on the router readout (Section 4).

H ACTION-RELEVANCE ROUTER

Module. When `action_relevance_enabled=True` , a residual router sits after the generator tower: relation-aware attention over clean context indices $\mathcal{J}_t$ (task + clean vision; noised future vision excluded), with routing dimension d_r=512, 8 heads, top-k support k=32, and residual gate initialized to 0.01. Optimizer trains `action_relevance` with LR multiplier 5.0; router weights are skipped when loading the public Nano checkpoint.

Sparsity schedule. λ=0.01 on $\mathcal{L}_{\text{sparse}}$ (equation 7); λ=0 for the matched baseline. Sparsity loss activates at optimizer step 1,000 and anneals linearly over 4,000 steps (matches the main-text ramp). Gradients from $\mathcal{L}_{\text{sparse}}$ update the shared tower and router; the dense MoT visual→action path remains active at training and default eval time (Section 4).

Evaluation-only readouts. Closed-loop `selected` (C-only) and `random_matched` conditions (Section 5.3, Appendix E) intervene on the *fixed* 60k policy without refitting; they are not alternate training configs.