BRANCH-JEPA: ACTION-ALIGNED PATCH ROUTING FOR LATENT WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Predicting future embeddings offers a compact route to learning visual world models, but a well-dispersed latent space need not expose the state changes that distinguish actions. We present Branch-JEPA, a representation-level extension of an end-to-end joint-embedding world model. Two independent spatial routers aggregate shared Patch features into State and Act residuals, which augment rather than replace a global CLS pathway. A contrastive action-alignment objective supervises changes in the fused Act coordinates, while one shared Predictor models the complete representation. The design preserves latent prediction, global anti-collapse regularization and goal-conditioned planning without a pretrained visual backbone or a denoising frontend. Across PushT, TwoRoom, Reacher and Cube, task-averaged planning success increases from 83.25% to 88.25% on clean observations and from 35.50% to 59.75% under Gaussian corruption. Direct action-regression controls show that action supervision and Patch access account for much of the gain. Paired intervals resolve smaller improvements from separate routing and Act-only alignment under Gaussian noise; their clean intervals include zero. Frozen readouts, interventions and success-aligned candidate ranking connect these behavioral results to relative specialization within the shared representation.

1 INTRODUCTION

A visual world model should distinguish futures that demand different decisions, such as whether a displacement creates contact or meets an obstacle. These distinctions need not require reconstructing every pixel. Joint-embedding predictive architectures instead predict representations, as in I-JEPA for images and V-JEPA for videos (Assran et al., 2023; Bardes et al., 2024). The resulting question is which information the prediction space should retain.

A representation may preserve appearance while compressing action-relevant changes; an action-focused representation may discard the geometry that determines their consequences. A useful latent state must expose both configuration and action-associated transitions. We distinguish *numerical non-collapse*, concerning feature variation, from *dynamical accessibility*, concerning whether a practical decoder and planner can use that variation.

LeWorldModel provides a concise starting point: learn an Encoder and action-conditioned Predictor jointly from pixels using embedding prediction and distributional regularization (Maes et al., 2026). Retaining this framework is attractive because it avoids a separate pretrained visual pipeline. Yet constraining the embedding distribution does not assign particular coordinates to state content or action-associated change. This is a question of inductive bias, not evidence that the original method lacks physical representations. In particular, low readout under a distribution shift should not automatically be called complete information loss.

We propose **Branch-JEPA**, which adds organization to the representation without replacing its global path or splitting its dynamics model. Two soft routers pool shared visual Patch features into State and Act residuals. Their concatenation is added to the original CLS projection, preserving a direct path for global scene information. Changes in the final Act coordinates are aligned with recorded actions through a contrastive objective. State coordinates remain jointly trained for prediction, rather than being forced to stay constant as the agent moves. A shared Predictor continues to forecast the entire embedding, so interactions between the two partitions remain expressible.

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.

We target relative specialization, not a hard division between controllable and uncontrollable objects. Visual corruption stress-tests this organization without entering training as a labeled nuisance. The contributions are:

- A diagnostic distinction between numerical spread and accessible transition information.
- CLS-preserving Patch routing and action-aligned latent changes, compatible with a shared Predictor and the original planning interface.
- A four-task evaluation connecting paired run uncertainty and direct action-regression controls to source-resolved readouts and success-aligned ranking diagnostics.

2 Related Work

Predictive representations and latent planning. I-JEPA predicts representations of image regions from context, while V-JEPA uses feature prediction to learn from video (Assran et al., 2023; Bardes et al., 2024). These objectives establish a representation-prediction paradigm, but they are not by themselves complete action-conditioned planners. DINO-WM learns dynamics over pretrained spatial features and optimizes action sequences toward observational goals (Zhou et al., 2025). Planning with reward-free offline latent dynamics provides another route to goal-directed behavior without reward annotations in the training data (Sobal et al., 2025). Branch-JEPA focuses on the visual representation within an end-to-end framework; its spatial residual does not require a frozen foundation-model feature space.

Preventing collapse versus organizing information. VICReg explicitly regulates feature variance and covariance in self-supervised learning (Bardes et al., 2022). LeJEPA introduces SIGReg, which encourages an isotropic Gaussian embedding distribution (Balestriero & LeCun, 2025); LeWorldModel combines related regularization with action-conditioned embedding prediction (Maes et al., 2026). Sub-JEPA instead applies Gaussian constraints in random subspaces to relax the global distributional restriction (Zhao et al., 2026). These approaches address optimization and distributional issues. Our question is complementary: after numerical degeneracy is controlled, which information is readily accessible to the predictor and planner? We retain global regularization and test organization separately, rather than treating additional branch structure as a replacement for anti-collapse control.

Action-guided visual representations. Inverse-dynamics supervision has been used to learn features associated with agent interaction (Pathak et al., 2017). Delta-JEPA directly decodes executed actions from differences between observed endpoint embeddings, including multi-step action blocks, making displacement-level action supervision especially relevant to this setting (Zhang et al., 2026). Accordingly, action supervision of a latent difference is not claimed as new here. The proposed distinction is its combination with a CLS-preserving dual aggregation path and supervision of one fused partition while predicting both jointly. We therefore include matched latent-difference regression controls and a five-query Transformer decoder patterned after Delta-JEPA. These retain our backbone, SIGReg and planner; they test its supervision scheme in this framework rather than reproduce the published system. Contrastive predictive coding motivates the matching formulation (van den Oord et al., 2018); our filtered negative set is used as an alignment surrogate, not advertised as an exact standard mutual-information bound.

Adaptive spatial aggregation. Vision Transformers provide spatial Patch tokens alongside global summaries (Dosovitskiy et al., 2021). TokenLearner illustrates adaptive token aggregation, whereas DynamicViT emphasizes token sparsification for efficient inference (Ryoo et al., 2021; Rao et al., 2021). Our routers do not prune tokens or promise a lower backbone cost. They construct two soft summaries from the same token set, with gradients shaped by predictive learning and action alignment. The relevant comparison is therefore learned aggregation versus uniform pooling under matched training, not token count alone.

3 MOTIVATION: FROM NUMERICAL SPREAD TO DYNAMICS INFORMATION

Does preventing numerical collapse also make a representation useful for distinguishing transitions? We compare prediction-only and prediction-plus-SIGReg Encoders with the same backbone, data and training budget. Clean feature covariance gives normalized effective rank, while feature standard deviation checks absolute variation. A linear probe fitted on clean training-split features decodes recorded actions from observed endpoint differences. We freeze it for clean and Gaussian-corrupted PushT test endpoints ($\sigma = 0.05$), pairing physical transitions across conditions (Appendix C).

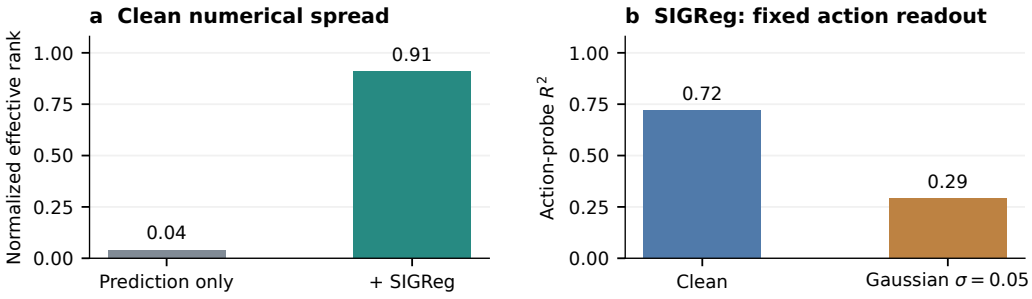

Figure 1: Numerical spread and transferable action information. (a) Clean numerical spread increases with SIGReg. (b) Action readout under Gaussian noise decreases with the decoder fixed. Rank is measured on clean features only. Points reuse Table 9; no uncertainty is implied.

SIGReg increases normalized rank from 0.04 to 0.91 and mean feature standard deviation from 0.002 to 0.98 (Figure 1a), supporting its role in preventing numerical collapse. Yet the regularized Encoder's action-probe R^2 falls from 0.72 to 0.29 under noise (Figure 1b). The actions remain unchanged while their accessibility to the clean-fitted decoder declines. This readout-transfer gap does not prove complete information loss: coordinate changes, nonlinear coding and lost information could each contribute. Numerical spread and transferable transition information therefore require separate tests.

This motivates three design requirements: preserve the global representation and anti-collapse objective, expose action-associated change, and couple that change with state context during prediction. Independent Patch pooling and branch-selective action alignment implement these choices. The diagnostic motivates testing this organization; it does not establish the architecture's necessity. Section 5 compares routing and alignment controls, and Section 6 connects readouts with ranking and planning.

4 BRANCH-JEPA

4.1 SETTING AND OVERVIEW

Let $\mathcal{D}$ contain offline trajectories of observations o_t and recorded action blocks $a_t \in \mathbb{R}^{d_a}$. The model learns an Encoder F_θ and an action-conditioned Predictor P_ϕ from these trajectories. Physical states and task success are evaluation signals only. At test time, a current image and a goal image define a latent planning problem. "Reward-free" here refers to training without reward labels; planning still uses an explicit goal-distance objective.

Figure 2 shows the computational division. The State Encoder produces a complete 192D embedding with a 128D State partition and a 64D Act partition. The Action Encoder maps the task's action block to 192D. A shared causal Predictor takes these embeddings and predicts the complete next latent. Two contrastive heads are used only for representation learning and are absent from planning.

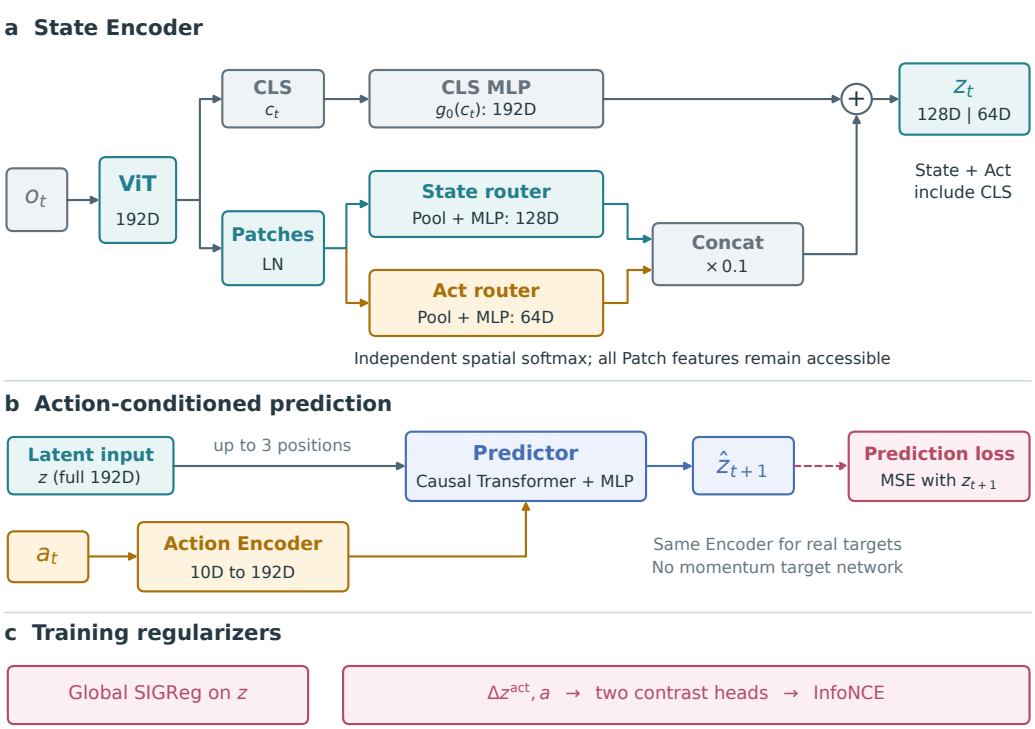

Figure 2: **Branch-JEPA computation.** Global CLS features are augmented with two independently pooled Patch residuals. Action alignment operates on differences of the final fused Act coordinates. One shared Predictor forecasts both partitions. The displayed 10D action input is the PushT instantiation; other tasks use their native action-block dimension.

4.2 CLS-Preserving Dual Patch Aggregation

A randomly initialized ViT maps 224×224 observations into a global CLS feature c_t and $N = 256$ Patch features of dimension 192. Let $h_{t,i}$ be the non-affine layer-normalized feature of Patch i. For branch $b \in \{\text{state}, \text{act}\}$, a two-layer score network s_b defines

$$w^b_{t,i} = \frac{\exp(s_b(h_{t,i})/\tau_r)}{\sum_{j=1}^{N} \exp(s_b(h_{t,j})/\tau_r)}, \qquad p^b_t = \sum_{i=1}^{N} w^b_{t,i} h_{t,i}, \qquad r^b_t = g_b(p^b_t). \tag{1}$$

We use $\tau_r = 1$. Each router normalizes over space independently, so $w^{\text{state}}_{t,i} + w^{\text{act}}_{t,i}$ need not equal one. A Patch may contribute to both branches. This preserves access to shared contact or geometric information and avoids a hard assignment bottleneck. Zero-initialized final score layers start the routers at uniform pooling.

The branch heads produce residuals of dimensions 128 and 64. They augment the original global projection g_0:

$$z_t = g_0(c_t) + \alpha[r^{\text{state}}_t; r^{\text{act}}_t], \qquad z_t = [z^{\text{state}}_t; z^{\text{act}}_t], \qquad \alpha = 0.1. \tag{2}$$

The final partitions therefore contain both CLS and Patch information. The State branch is not defined as static background, and the Act branch is not a segmentation of controllable pixels. They are feature partitions with different direct supervision. The global pathway remains available even when a router has not yet learned useful spatial preferences.

A small α limits the direct scale of the residual, but neither protects the jointly trained CLS path nor guarantees noise-invariant pooling. The residual bound in Appendix D makes this distinction explicit; the uniform-pooling control tests the benefit of learned spatial weights.

4.3 Aligning Actions with Observed Representation Changes

For each sampled transition, the fused Act displacement is $\delta_m = z_{t+1}^{\text{act}} - z_t^{\text{act}}$. Projection heads u and v map the displacement and recorded action to unit-normalized 64D vectors q_m and k_m. To avoid declaring nearly identical continuous actions to be negatives, define

$$\mathcal{N}_m = \left\{ n \neq m : \frac{\|a_m - a_n\|_2}{\sqrt{d_a}} \geq \eta \right\}, \qquad \eta = 0.05. \tag{3}$$

Actions are standardized with fitting-data statistics. With temperature $\tau_a = 0.2$ and $\mathcal{U} = \{m : |\mathcal{N}_m| > 0\}$, the alignment loss is

$$\mathcal{L}_{\text{act}} = -\frac{1}{|\mathcal{U}|} \sum_{m \in \mathcal{U}} \log \frac{\exp(q_m^\top k_m / \tau_a)}{\exp(q_m^\top k_m / \tau_a) + \sum_{n \in \mathcal{N}_m} \exp(q_m^\top k_n / \tau_a)}. \tag{4}$$

We sample at most 256 local transitions per worker and use zero loss if no valid negatives remain. This is an InfoNCE-style matching objective (van den Oord et al., 2018); action-dependent negative filtering changes the sampling assumptions of the usual mutual-information argument.

Crucially, δ_m is computed from two encoded observations, not from a Predictor-generated next state. Thus, the objective directly shapes the Encoder's Act coordinates. Its gradients reach the shared backbone, CLS projector and Act residual path, but do not directly supervise the planning Action Encoder or Predictor. State learns through prediction and global regularization. This asymmetric pressure encourages relative action specialization without forcing independence or forbidding State from encoding action-relevant context.

4.4 Joint Prediction and Training

Let E_ξ be the Action Encoder. For a context of at most $H = 3$ positions, the shared Predictor estimates the next embeddings from observed inputs:

$$\hat{z}_{b,1:H} = P_\phi(z_{b,0:H-1}, E_\xi(a_{b,0:H-1})), \qquad \mathcal{L}_{\text{pred}} = \frac{1}{BHD} \sum_{b=1}^{B} \sum_{j=0}^{H-1} \|\hat{z}_{b,j+1} - z_{b,j+1}\|_2^2, \tag{5}$$

where $D = 192$. Targets use the same trainable Encoder without a momentum teacher or stop-gradient. Global SIGReg regularizes the complete fused latent, following the underlying end-to-end framework (Balestriero & LeCun, 2025; Maes et al., 2026). The objective is

$$\mathcal{L} = \mathcal{L}_{\text{pred}} + 0.09\mathcal{L}_{\text{SIGReg}} + 0.005\mathcal{L}_{\text{act}}. \tag{6}$$

There is no separate branch-invariance penalty, Patch-separation loss, denoising frontend or corruption augmentation. In particular, State is free to change when object geometry, agent location or contact conditions change.

4.5 Planning without Additional Inference Heads

At inference, the frozen Encoder maps current and goal images to z_t and z_g. For a candidate sequence $A = (a_t, \ldots, a_{t+K-1})$, the Predictor rolls forward autoregressively. CEM minimizes

$$J(A) = \|\hat{z}_{t+K}(A) - z_g\|_2^2. \tag{7}$$

Both partitions enter the same terminal distance without learned planning weights. Each replanning call begins with the current observed latent; subsequent context positions come from the imagined rollout. The inherited PushT configuration uses five action blocks, 300 candidates, 30 CEM iterations and 30 elites. Each block contains five environment steps, and all five blocks are executed before replanning: the replan interval is therefore **25 environment steps**, with success checked after every step. The 50-step evaluation budget permits at most two planning calls. Keeping this interface fixed isolates representation design from changes to search capacity. No contrastive head is executed by CEM.

Task	Clean		Gaussian		Background		Brightness/contrast	
	Ref.	Ours	Ref.	Ours	Ref.	Ours	Ref.	Ours
PushT	88	**94**	24	**74**	59	**82**	68	**86**
TwoRoom	85	**90**	30	**49**	64	**80**	70	**83**
Reacher	90	**93**	45	**59**	74	**86**	78	**88**
Cube	70	**76**	43	**57**	55	**65**	59	**67**

Table 1: Four-task mean success (%) over five runs. Table 6 reports SDs and paired intervals; Appendix A defines cases and corruptions.

5 EXPERIMENTS

We evaluate planning performance, the contributions of routing and action supervision, and the information exposed by the learned representation. Reference rows report our LeWorldModel control under the same protocol, rather than scores quoted from its authors.

5.1 BENCHMARK DESIGN

PushT, TwoRoom, Reacher and Cube share task-specific offline data, preprocessing, latent dimension, training-step budget and planning solver across methods. The core comparison fixes the Action Encoder and Predictor architecture. Checkpoints are selected using clean validation success. Appendix A specifies task predicates, episode-disjoint splits and the hyperparameter search.

Gaussian noise perturbs RGB pixels in $[0, 1]$ as $x' = \mathrm{clip}(x + \epsilon, 0, 1)$, $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$, before Encoder normalization. Both current and goal images are corrupted: σ is 0.05 for PushT, 0.30 for TwoRoom and Reacher, and 0.20 for Cube. Background replacement and brightness–contrast shifts are separate conditions. No corruption enters world-model training.

Planning summaries use five runs with seeds 11, 22, 33, 44 and 55, each evaluated on 500 common test cases with one clean evaluation or five corruption repeats per case. We average repeats within runs before computing the mean and sample standard deviation (SD) across runs. Paired differences use two-sided 95% Student-t intervals with four degrees of freedom; corruption repeats do not count as independent training runs. Seed values and paired intervals appear in Appendix B.

5.2 PLANNING ACROSS TASKS AND OBSERVATION CONDITIONS

Branch-JEPA improves clean success by 3–6 percentage points (Table 1). Gaussian improvements reach 50 points on PushT and remain positive on the other tasks. Background improvements are 23, 16, 12 and 10 points, respectively. These results concern the specified corruptions; they do not establish robustness to arbitrary backgrounds or lighting.

5.3 SEPARATING CAPACITY, ROUTING AND ALIGNMENT

Table 2 compares a wider global control, a single Patch space with full-latent alignment, shared routing with two residual heads, and dual routing with full-latent alignment. These respectively test additional capacity, the partitioned design, independent spatial selection and alignment location. The last comparison fixes the representation and Predictor but widens the auxiliary head's input. Exact changes are specified in Appendix E.

Direct action-regression controls. For training-standardized action blocks $\tilde{a}$, we replace contrastive alignment with

$$\mathcal{L}_{\mathrm{reg}} = \frac{1}{Bd_a} \sum_{m=1}^{B} \|D_\psi(\Delta z_m) - \tilde{a}_m\|_2^2, \tag{8}$$

where Δz_m is an observed endpoint difference spanning five environment steps. Two-layer MLPs decode the full 192D displacement for global and single-Patch controls, or the 64D Act displacement for dual routing. A stronger single-Patch control uses Delta-JEPA's five-query, three-layer Transformer decoder (Zhang et al., 2026). These adapt its decoding objective, rather than reproduce

Retrained variant	Clean (%)	Gaussian (%)	Action selectivity
Reference	88.0 ± 1.6	24.0 ± 2.2	–
Parameter-matched single space	89.0 ± 1.6	29.0 ± 2.2	–
CLS dual + alignment, no Patch	90.0 ± 1.6	45.0 ± 2.2	0.17
Uniform pooling + alignment	91.0 ± 1.6	57.0 ± 2.2	0.19
Learned routing, no alignment	90.0 ± 1.6	61.0 ± 2.2	0.04
Single Patch space + full alignment	91.0 ± 1.6	65.0 ± 2.2	–
Shared router, two heads + alignment	92.0 ± 0.7	69.0 ± 1.4	0.23
Dual routers + full alignment	92.0 ± 1.4	70.0 ± 1.0	0.08
Global Delta regression	90.0 ± 1.6	60.0 ± 2.9	–
Single Patch Delta regression	91.0 ± 1.6	65.0 ± 3.1	–
Single Patch + Delta Transformer	92.0 ± 1.6	71.0 ± 3.2	–
Dual routers + Act regression	93.0 ± 1.6	72.0 ± 2.7	–
Branch-JEPA	94.0 ± 1.6	74.0 ± 2.7	0.25

Table 2: PushT controls: success as mean $\pm$ sample SD over five runs. Selectivity is the descriptive Act-minus-State action-probe R^2, undefined for an unsplit latent. Every variant uses the same data and update budget; paired intervals appear in Table 8.

its complete system. All controls retain the backbone, data, Predictor, SIGReg, optimizer-step budget and six-configuration validation search. Equal updates do not imply equal FLOPs or wall-clock cost.

The single-Patch contrastive control reaches 91/65% clean/Gaussian success. Global and single-Patch MLP regression reach 90/60% and 91/65%; the Transformer decoder reaches 92/71%. Dual-router Act regression reaches 93/72%, close to Branch-JEPA's 94/74%. Action supervision and Patch access therefore account for substantial gains. The paired Branch-JEPA-minus-Act-regression intervals include zero in both conditions, so these results do not establish contrastive-loss superiority.

Uncertainty in organization effects. Separate versus shared routers gives $+2$ pp clean, 95% interval $[-0.48, 4.48]$, and $+5$ pp Gaussian, $[1.60, 8.40]$. Act-only versus full-latent alignment gives $+2$ pp clean, $[-1.17, 5.17]$, and $+4$ pp Gaussian, $[0.38, 7.62]$. The clean intervals include zero. Every shared-router Gaussian difference is positive, while one full-alignment comparison reverses sign. These intervals are not multiplicity-adjusted discovery tests. Uniform pooling and no alignment reach 91/57% and 90/61%; Appendix E reports the complete routing-by-alignment control.

5.4 Corruption Curves and Multi-Step Prediction

Figure 3a evaluates seven Gaussian strengths with each selected checkpoint fixed. Area under the success curve, normalized by the evaluated noise range, is 67.60% for the reference, 78.60% for uniform pooling and 87.25% for Branch-JEPA. This is average success over a noise interval, not ROC-AUC.

At horizon five, normalized autoregressive error is 0.25 for Branch-JEPA versus 0.39 for the reference (Figure 3b). Different learned latent spaces still prevent interpreting these errors as a common physical accuracy measure. Section 6 complements this diagnostic with candidate ranking against the complete terminal success predicate.

6 Representation and Functional Analysis

6.1 Relative Specialization without a Static Branch

Frozen PushT probes use the same episode-disjoint split to decode recorded action blocks, physical transitions and instantaneous state (Table 3). Transition targets include pusher and object displacement and wrapped angle change; state targets include positions and sine/cosine of orientation. Noisy evaluation freezes clean-fitted preprocessing and decoder parameters (Appendix C).

Alignment raises Act action readout from 0.61 to 0.81 while State changes from 0.57 to 0.56; State configuration readout remains similar (0.88 versus 0.87; Table 14). Thus increased selectivity mainly

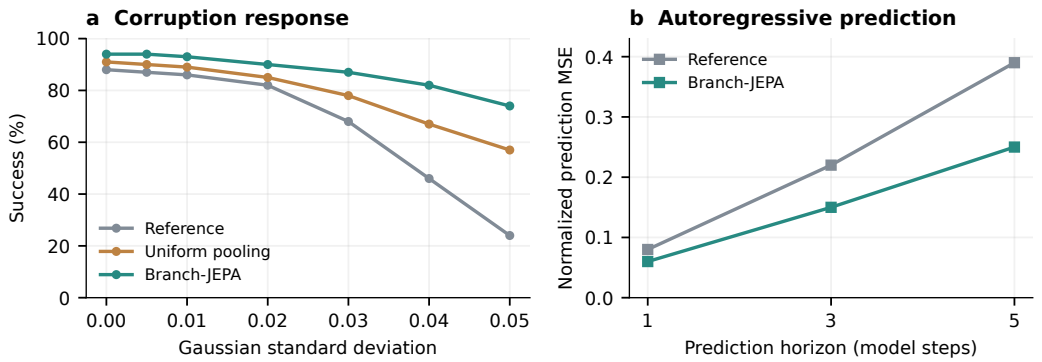

Figure 3: (a) PushT success versus Gaussian strength, averaged over five runs. (b) Autoregressive prediction error, normalized by each Encoder's target latent variation. Prediction errors are descriptive point values without run-level uncertainty.

Diagnostic	Reference	State	Act	Full
Action R^2, clean	0.72	0.56	0.81	0.83
Action R^2, noisy	0.29	0.41	0.69	0.71
State R^2, clean	0.92	0.87	0.80	0.96
State R^2, noisy	0.77	0.81	0.72	0.91
Transition R^2, clean	0.61	0.58	0.71	0.85
Transition R^2, noisy	0.28	0.43	0.57	0.72
Normalized effective rank	0.91	0.90	0.89	0.88

Table 3: Frozen PushT readouts and numerical spread. Reference and Full use the complete reference and Branch-JEPA latents. R^2 uses test-mean centering. Probe values are descriptive point estimates.

reflects better Act readout. State better decodes orientation (0.78 versus 0.58), while Act better decodes pusher position (0.98 versus 0.93), supporting relative roles rather than static-background semantics. Equal-64D state readouts are 0.91 for Full, 0.83 for State and 0.80 for Act. This controls probe input dimension without establishing coordinate-level nonredundancy. Under noise, Act action readout is 0.69 and Full state readout is 0.91. The action-selectivity gap grows from 0.25 to 0.28 despite decreasing absolute action readouts.

6.2 Where Is the Readable Information?

Equal-64D probes compare CLS, scaled Patch residuals and fused partitions within the complete-model checkpoint (Appendix Figure 4). Act residuals yield action readout 0.75 versus State's 0.38; State residuals yield state readout 0.79 versus Act's 0.60. Fusion gives 0.81 for Act action readout and 0.83 for State state readout. Joint learning and probe standardization prevent assigning these differences to causal contributions of routing (Appendix E.3).

6.3 Connecting Readout, Ranking and Planning

Appendix Table 16 compares full-latent action readouts, object-only candidate ranking and planning. Action probes use complete 192D observed differences with identical fitting rules; ranking and planning likewise retain both partitions. Checkpoints use clean validation selection, with common test splits and paired planning cases.

Full-latent alignment gives higher action readout (0.84 versus 0.83), while Act-only alignment gives higher object-only ranking (0.66 versus 0.62) and clean/noisy success (94/74% versus 92/70%). Branch selectivity therefore need not imply more total readable action information. These comparisons do not establish causal mediation; the clean planning interval includes zero, and noisy ranking is not evaluated.

Variant	Object-only τ_b	Success-aligned τ_b
Reference	0.333 ± 0.013	0.430 ± 0.013
Dual routers, no alignment	0.390 ± 0.013	0.507 ± 0.023
Shared router, Act alignment	0.423 ± 0.010	0.555 ± 0.014
Dual routers, full alignment	0.426 ± 0.006	0.565 ± 0.003
Dual routers, Act regression	0.443 ± 0.014	0.588 ± 0.016
Branch-JEPA	0.454 ± 0.010	0.601 ± 0.012

Table 4: Common-candidate ranking, mean $\pm$ sample SD over five runs. Both costs use the same endpoints and predictions; success-aligned cost includes pusher position, object position and orientation.

Ranking with the complete success predicate. Object-only cost omits pusher position. For $q = (x_p, y_p, x_o, y_o)$, define

$$d_{\text{succ}}(s, g) = \max \left\{ \frac{\|q - q_g\|_2}{20}, \frac{|\operatorname{wrap}(\theta_o - \theta_g)|}{\pi/9} \right\}. \tag{9}$$

Its strict sublevel set $d_{\text{succ}} < 1$ matches terminal success. A separate common-candidate evaluation computes both physical costs on 100 states with 64 candidate endpoints each, shared across methods. Within each state, Kendall's τ_b compares predicted latent costs with physical costs; scores are averaged across states and then five runs (Table 4). These shared candidate sets differ from the object-only diagnostic in Table 16.

This diagnostic ranks terminal outcomes at 25 environment steps. Main success can occur at any step within 50 steps, with replanning at step 25. Intermediate success, trajectory feasibility and the second planning call therefore limit what terminal ranking explains about planning gains.

6.4 Does Planning Depend on Both Patch Residuals?

Removing the State or Act Patch residual while preserving CLS reduces clean/noisy success from 94/74% to 88/63% or 90/66%, respectively (Table 17). Norm- and coordinate-matched random perturbations give 93/71% for State and 93/72% for Act. Larger targeted drops suggest dependence on residual content beyond perturbation magnitude. However, current and goal codes change while the Predictor remains frozen; mismatch remains possible, and retraining could compensate. These PushT diagnostics do not establish identical branch semantics across all four tasks.

7 Discussion and Conclusion

Branch-JEPA augments a global visual pathway with independent Patch summaries and aligns observed changes with actions, while retaining a shared dynamics model. State remains dynamic; Act receives direct supervision to expose action-associated changes.

The results connect planning behavior with routing and supervision controls, source-resolved readouts and success-aligned ranking. Patch access and action supervision account for much of the reference gap. Smaller clean gains from independent routing and Act-only alignment have paired intervals that include zero; corresponding Gaussian intervals exclude zero. The contrastive-versus-regression comparison does not establish superiority of the loss family.

Limitations remain: offline associations do not identify causal effects, residual removal can shift the model's input distribution despite matched-norm controls, and soft routing does not guarantee corruption invariance. Terminal ranking differs from anytime success, and representation diagnostics focus on PushT. These boundaries motivate testing representation organization through distinct behavioral, readout and intervention measurements.

AI Use Statement

Generative AI tools assisted with manuscript organization, language editing, reference checking, and preparation of LaTeX sources and figure-generation code. They also assisted with refining

experimental reporting and the presentation of result interpretations. The authors take responsibility for the scientific claims, reported results and final content.

Reproducibility Statement

Section 4 specifies the architecture, objectives and planner. Appendix A gives the training, evaluation and selection protocols; the remaining appendices define uncertainty estimates, readouts, controls and ranking diagnostics. The accompanying LaTeX source embeds the table values and vector figures needed to compile the complete manuscript.

References

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2023. URL https://arxiv.org/abs/2301.08243.

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

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xm6YD62D1Ub.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024. URL https://arxiv.org/abs/2404.08471.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

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

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2778–2787. PMLR, 2017. URL https://proceedings.mlr.press/v70/pathak17a.html.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems*, volume 34, pp. 13937–13949. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/747d3443e319a22747fbb873e8b2f9f2-Abstract.html.

Michael S. Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. TokenLearner: Adaptive space-time tokenization for videos. In *Advances in Neural Information Processing Systems*, volume 34, pp. 12786–12797. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/6a30e32e56fce5cf381895dfe6ca7b6f-Abstract.html.

Uladzislau Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 43905–43941. Curran Associates, Inc., 2025. doi: 10.52202/085713-1465.

URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3e7cf447f21cd11c846463affefce665-Abstract-Conference.html.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. URL https://arxiv.org/abs/1807.03748.

Zhenghao Zhang, Yuanxiang Wang, Zhenyu Guan, Yujia Yang, Bingkang Shi, Tianyu Zong, Hongzhu Yi, Guoqing Chao, Xingchen Chen, Tiankun Yang, Chenxi Bao, Tao Yu, Jingjing Zhou, and Jungang Xu. Delta-JEPA: Learning action-sensitive world models via latent difference decoding. *arXiv preprint arXiv:2606.31232*, 2026. URL https://arxiv.org/abs/2606.31232.

Kai Zhao, Dongliang Nie, Yuchen Lin, Zhehan Luo, Yixiao Gu, Deng-Ping Fan, and Dan Zeng. Sub-JEPA: Subspace gaussian regularization for stable end-to-end world models. *arXiv preprint arXiv:2605.09241*, 2026. URL https://arxiv.org/abs/2605.09241.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhou25t.html.

A Benchmark and Experimental Protocol

This appendix specifies the data splits, training and model-selection procedure, evaluation conditions and diagnostic definitions used for the reported experiments. Numerical summaries are reported in the tables below.

Dataset sizes and episode-disjoint splits. Each of PushT, TwoRoom, Reacher and Cube has 10,000 episodes, each containing 101 observations and 100 actions. The episode-disjoint split assigns 8,000 episodes to training, 1,000 to validation and 1,000 to testing. Each task has 800,000 training, 100,000 validation and 100,000 test transition positions. No frames, actions, goals or probe targets from validation or test enter Encoder fitting, normalization, contrastive negatives or PCA fitting. The complete episode, rather than a frame or a window, is the splitting unit.

A training clip starts at environment step $t \in \{0, 5, \ldots, 85\}$ and contains observations at $t, t+5, t+10, t+15$, together with the three intervening five-step action blocks. This gives 144,000 clips per task. Each epoch uses a seeded permutation of those clips, followed by the first 384 clips of the same permutation to fill 188 optimizer steps at effective batch size 768. Every variant receives the same permutation for a given training seed. No corruption is applied during training.

Training, search and checkpoint selection. The search allocates six hyperparameter candidates per task/variant, each fitted for exactly 100 epochs with search seed 7. For variants with an auxiliary loss the grid is learning rate $\{5\times10^{-5}, 10^{-4}\}$ crossed with auxiliary weight $\{0.0005, 0.005, 0.05\}$. For a variant without an auxiliary loss, the six candidates cross the same learning rates with weight decay $\{0.01, 0.05, 0.1\}$. All other settings in Table 5 are fixed; there is no architecture, noise or test-set search. The main results cover Reference and Branch-JEPA on four tasks; the architecture and supervision controls are PushT-only. The ridge-probe grid specified below is a separate decoder fit and does not select a world model. The reported configuration uses learning rate 10^{-4}, weight decay 0.05 and auxiliary weight 0.005 when applicable.

We evaluate checkpoints at epochs $10, 20, \ldots, 100$ on the same 250 clean validation cases, using the success fraction averaged equally over cases. The checkpoint with maximum validation success is selected; ties select the earlier epoch. Hyperparameters are selected by that score, breaking remaining ties by the ascending trial ID. After selection, we independently fit five 100-epoch reporting runs with seeds 11, 22, 33, 44 and 55, selecting each run's checkpoint by the same clean-validation rule. All 100 epochs are completed even if an earlier checkpoint is selected. The auxiliary heads are absent from inference. No noisy validation, confirmation test, ranking or probe score participates in model selection.

Evaluation sizes and pairing. Per task, the first 250 validation episodes define 250 validation cases; the first 500 test episodes define 500 confirmation cases. A case starts at step 0 with the same episode's step-25 image as its goal, and has a maximum 50-environment-step rollout. Each reporting seed has one clean rollout per case and five corruption replicates per case for each non-clean condition. Thus a clean cell has 500 outcomes per seed and a corrupted cell has $500 \times 5 = 2,500$; five reporting seeds give 2,500 and 12,500 outcomes, respectively. We average repeats within each case, cases within each seed, and then the five seed estimates with equal weight. Additional corruption repeats do not increase the number of independently trained models.

All methods share case IDs, seed IDs, repeat IDs, initial conditions, goals and exogenous-randomness keys. CEM keys are paired separately from image noise. A deterministic key comprises task, reporting seed, case ID, corruption repeat, stream name, camera ID and environment-step index; it excludes method ID, worker ID and batch order. A BLAKE2b-derived 64-bit integer (8-byte digest, big-endian) seeds a PCG64 stream. Current and goal noise use distinct stream names, with the goal draw held fixed throughout the rollout. Re-reading a frame reuses its draw; early termination never advances another case's stream. Once trajectories diverge, equal keys give equal noise tensors, not equal current observations.

Corruption conditions. Gaussian noise is added to RGB pixels in $[0, 1]$ before Encoder normalization, then clipped to $[0, 1]$. Both current and goal images are corrupted; σ is task-specific (Table 5). Every nonzero point in the PushT noise curve uses the same five-repeat rule, paired by case and repeat. Background replacement uses a renderer mask excluding the agent, manipulated objects, goals and structural obstacles. A color sampled uniformly from $[0.2, 0.8]^3$ is fixed for the episode and shared between current and goal images; each of the five repeats has a separate color.

For brightness–contrast, compute $x' = \mathrm{clip}(\beta x, 0, 1)$, then $x'' = \mathrm{clip}(\mu + \gamma(x' - \mu), 0, 1)$, with μ the per-image channel mean of x'. The ordered menu is the eight lexicographic pairs in $\{0.8, 1.0, 1.2\}^2 \setminus \{(1, 1)\}$. For each case and seed, a fixed hash chooses a start index; the five repeats use the next five menu entries cyclically. The same factors apply to current and goal images. This is a five-repeat mixture, not an equal-weight average over all eight combinations. Corruption parameters are paired across methods.

Task success predicates. The PushT state uses simulator coordinates $q = (x_p, y_p, x_o, y_o)$ in a 512×512 workspace; angle differences are wrapped to $[-\pi, \pi)$. Define

$$\rho_{\mathrm{pos}}(s, g) = \frac{\|q - q_g\|_2}{20}, \qquad \rho_{\mathrm{ang}}(s, g) = \frac{|\,\mathrm{wrap}(\theta_o - \theta_g)|}{\pi/9}, \qquad d_{\mathrm{succ}}(s, g) = \max(\rho_{\mathrm{pos}}, \rho_{\mathrm{ang}}). \tag{10}$$

PushT succeeds precisely when $d_{\mathrm{succ}} < 1$ at any visited state after an executed environment step, up to step 50. This includes pusher position and object position and orientation; it is not area coverage. TwoRoom succeeds when the agent is within Euclidean distance 0.10 of its goal in normalized $[0, 1]^2$ coordinates. Reacher uses two wrapped joint-angle errors, each strictly below 0.10 radian (the `qpos_match` condition). Cube uses a single object and camera; success requires object center error below 0.05 m and quaternion geodesic orientation error below $\pi/12$. Each task uses an anytime-within-50-steps predicate with the task-specific definitions above. PushT actions are desired pusher positions in $[0, 512]^2$; TwoRoom and Reacher actions are 2D bounded controls in $[-1, 1]^2$; Cube actions are 7D bounded controls in $[-1, 1]^7$. Each action coordinate is standardized using training episodes only, with a standard-deviation floor of 10^{-6}.

CEM initializes a diagonal Gaussian at the midpoint of each native action bound, with standard deviation half its width; clips samples to the bounds; and replaces mean and variance by the elite estimates at every iteration (no smoothing, variance floor 10^{-6} in normalized bound coordinates). It executes the final best candidate's 25 actions, checking success after each action and terminating immediately on success. If needed, it replans once at step 25 for the remaining 25 steps. Thus "closed-loop" here uses *25-step replanning*, not per-step replanning.

B Replicates and Uncertainty Estimates

Every success summary is the mean of five reporting runs. For seed values $x_1, \ldots, x_5$ in percent, the sample standard deviation is $s_x = \sqrt{\sum_i (x_i - \bar{x})^2/4}$. For a paired comparison, form $d_i = x_i^{\mathrm{Branch}} -$

Component	Experimental setting
Visual input / backbone	224×224 RGB; randomly initialized ViT-Tiny, 12 blocks, 3 attention heads, MLP ratio 4; 14-pixel Patches; 16×16 tokens, width 192; no backbone dropout
CLS / prediction MLP	$192 \to 2048 \to 192$; BatchNorm and GELU
Independent routers	Each $192 \to 64 \to 1$; GELU; spatial softmax, temperature 1; zero-initialized final score layer
State / Act residual heads	$192 \to 128 \to 128$ / $192 \to 128 \to 64$; LayerNorm and GELU; residual scale 0.1
Planning Action Encoder	1×1 Conv $d_a \to d_a$; MLP $d_a \to 768 \to 192$ with SiLU
Causal Predictor	6 layers; 16 attention heads of dimension 64; feed-forward width 2048; dropout 0.1; latent width 192
Contrast heads	Change: $64 \to 64 \to 64$; action: $d_a \to 64 \to 64$; LayerNorm, GELU, output unit normalization; temperature 0.2, negative-distance cutoff 0.05
MLP action-regression controls	Two linear layers $d_z \to 192 \to d_a$, $d_z = 192$ for global/single-Patch, 64 for Act; GELU; coordinate-mean MSE on training-standardized action blocks
Delta-Transformer control	5 learned queries; difference-conditioned AdaLN; 3 noncausal layers, 8 heads $\times$ 64D, feed-forward width 512; one native action/query; coordinate-mean MSE
Training clips / local statistics	4 observations, 3 real transitions; at most 256 contrast pairs per worker; no cross-worker negative pool
SIGReg / loss weights	1024 projections, 17 frequency knots; prediction / SIGReg / auxiliary: $1/0.09/0.005$
Batch / schedule	$128 \times 3 \times 2 = 768$ clips; 188 optimizer steps/epoch, 100 epochs; 5 reporting seeds; validation at epochs $10, 20, \ldots, 100$
Optimizer / precision	AdamW $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$; peak learning rate 10^{-4}, weight decay 0.05; 5-epoch linear warmup, cosine decay to zero; bfloat16; gradient-norm cap 1.0
Action block / goal / budget	5 environment steps/block; goal offset 25 steps; confirmation horizon 50 steps
CEM / execution	5-block horizon; 300 candidates, 30 iterations, 30 elites; execute all 5 blocks = 25 environment steps before replanning; at most two planning calls in a 50-step rollout

Task	Per-step action dim.	Block dim. d_a	σ
PushT	2	10	0.05
TwoRoom	2	10	0.30
Reacher	2	10	0.30
Cube	7	35	0.20

Table 5: Experimental settings. The common backbone, input data, schedule, optimizer family, checkpoint rule and six-trial search budget apply to every trained variant. d_a concatenates five native actions. Full-latent alignment changes its displacement-head input to 192; the regression control replaces contrastive alignment with action MSE.

x_i^{control} using the same seed, cases and repeats; report $\bar{d} \pm t_{4,0.975} s_d/\sqrt{5}$ with $t_{4,0.975} = 2.776445$. The unit is percentage points. These unadjusted intervals describe variation across the five paired runs. No evaluation repeat is treated as an additional training seed, and no multiple-comparison claim is made.

Task	Condition	Reference	Branch-JEPA	Paired Δ [95% interval]
PushT	Clean	88.0 ± 1.6	94.0 ± 1.6	$+6.0$ $[4.48, 7.52]$
PushT	Gaussian	24.0 ± 2.2	74.0 ± 2.7	$+50.0$ $[45.79, 54.21]$
PushT	Background	59.0 ± 2.2	82.0 ± 2.2	$+23.0$ $[19.96, 26.04]$
PushT	Bright./contrast	68.0 ± 2.2	86.0 ± 2.2	$+18.0$ $[14.38, 21.62]$
TwoRoom	Clean	85.0 ± 1.6	90.0 ± 1.6	$+5.0$ $[2.37, 7.63]$
TwoRoom	Gaussian	30.0 ± 2.2	49.0 ± 2.2	$+19.0$ $[15.96, 22.04]$
TwoRoom	Background	64.0 ± 2.2	80.0 ± 2.2	$+16.0$ $[15.12, 16.88]$
TwoRoom	Bright./contrast	70.0 ± 2.2	83.0 ± 2.2	$+13.0$ $[11.24, 14.76]$
Reacher	Clean	90.0 ± 1.6	93.0 ± 1.6	$+3.0$ $[-0.62, 6.62]$
Reacher	Gaussian	45.0 ± 2.2	59.0 ± 2.2	$+14.0$ $[11.09, 16.91]$
Reacher	Background	74.0 ± 2.2	86.0 ± 2.2	$+12.0$ $[7.35, 16.65]$
Reacher	Bright./contrast	78.0 ± 2.2	88.0 ± 2.2	$+10.0$ $[5.11, 14.89]$
Cube	Clean	70.0 ± 1.6	76.0 ± 1.6	$+6.0$ $[2.49, 9.51]$
Cube	Gaussian	43.0 ± 2.2	57.0 ± 2.2	$+14.0$ $[11.37, 16.63]$
Cube	Background	55.0 ± 2.2	65.0 ± 2.2	$+10.0$ $[6.83, 13.17]$
Cube	Bright./contrast	59.0 ± 2.2	67.0 ± 2.2	$+8.0$ $[4.38, 11.62]$

Table 6: Main success results as mean $\pm$ sample SD over five seeds. Differences and intervals use seed-paired scores. Counts are 500 cases/seed for clean and 500 cases $\times$ 5 repeats/seed for every corrupted condition.

Method	Condition	11	22	33	44	55	Mean $\pm$ SD
Branch-JEPA	Clean	92.0	95.0	93.0	96.0	94.0	94.0 ± 1.6
Branch-JEPA	Gaussian	70.0	74.0	73.0	77.0	76.0	74.0 ± 2.7
Shared router	Clean	92.0	91.0	93.0	92.0	92.0	92.0 ± 0.7
Shared router	Gaussian	68.0	68.0	70.0	68.0	71.0	69.0 ± 1.4
Full-latent alignment	Clean	94.0	93.0	91.0	91.0	91.0	92.0 ± 1.4
Full-latent alignment	Gaussian	71.0	69.0	69.0	71.0	70.0	70.0 ± 1.0
Global Δ regression	Clean	89.0	92.0	90.0	88.0	91.0	90.0 ± 1.6
Global Δ regression	Gaussian	59.0	61.0	56.0	64.0	60.0	60.0 ± 2.9
Single Patch Δ regression	Clean	90.0	93.0	91.0	89.0	92.0	91.0 ± 1.6
Single Patch Δ regression	Gaussian	63.0	65.0	62.0	70.0	65.0	65.0 ± 3.1
Single Patch + Δ Transformer	Clean	93.0	90.0	92.0	94.0	91.0	92.0 ± 1.6
Single Patch + Δ Transformer	Gaussian	69.0	73.0	67.0	75.0	71.0	71.0 ± 3.2
Dual routers + Act regression	Clean	94.0	91.0	92.0	95.0	93.0	93.0 ± 1.6
Dual routers + Act regression	Gaussian	73.0	70.0	69.0	76.0	72.0	72.0 ± 2.7

Table 7: PushT reporting runs underlying the architectural and action-supervision comparisons. All entries are success percentages.

C Readouts, Ranks and Dynamics Diagnostics

Diagnostic model	Norm. rank	Mean feature SD	Action R^2	Noisy action R^2
Prediction only	0.04	0.002	-0.01	-0.02
Prediction + SIGReg	0.91	0.980	0.72	0.29

Table 9: Complete values for the pre-Method diagnostic. Rank and feature SD use clean observations. Gaussian strength is 0.05; both action readouts use the same clean-fitted probe.

The PushT probe evaluation uses 8,000 fitting windows (one from step 0 of each training episode), 1,000 validation windows (one from step 0 of each validation episode), and 5,000 test windows

Comparison	Condition	Mean Δ (pp)	Paired 95% interval (pp)
Ours − Shared router	Clean	$+2.0$	$[-0.48, 4.48]$
Ours − Shared router	Gaussian	$+5.0$	$[1.60, 8.40]$
Ours − Full-latent alignment	Clean	$+2.0$	$[-1.17, 5.17]$
Ours − Full-latent alignment	Gaussian	$+4.0$	$[0.38, 7.62]$
Ours − Global Δ regression	Clean	$+4.0$	$[1.22, 6.78]$
Ours − Global Δ regression	Gaussian	$+14.0$	$[10.96, 17.04]$
Ours − Single Patch Δ regression	Clean	$+3.0$	$[0.22, 5.78]$
Ours − Single Patch Δ regression	Gaussian	$+9.0$	$[6.52, 11.48]$
Ours − Single Patch + Δ Transformer	Clean	$+2.0$	$[-0.78, 4.78]$
Ours − Single Patch + Δ Transformer	Gaussian	$+3.0$	$[0.09, 5.91]$
Ours − Dual routers + Act regression	Clean	$+1.0$	$[-1.63, 3.63]$
Ours − Dual routers + Act regression	Gaussian	$+2.0$	$[-1.83, 5.83]$

Table 8: Paired differences between Branch-JEPA and each control. Intervals use five seed differences and a t_4 multiplier. An interval spanning zero does not resolve the sign of the corresponding mean gain at this confidence level.

(starts $0, 5, \ldots, 45$ in each of the first 500 test episodes). A window has a five-step action target and the corresponding endpoint observations. Validation and test source episodes are reserved before world-model Encoder fitting. Feature and target coordinates are standardized using fitting data only. Ridge coefficients are selected from $\{0.1, 1, 10, 100\}$ by validation error. We evaluate R^2 using the test mean of each target coordinate and average coordinates uniformly. The fitted decoder and preprocessing are reused for clean-to-noisy testing. These linear readouts characterize accessible information under a fixed decoder family; shuffled-target and nonlinear-probe controls are not reported.

For PushT, the action target is a 10D block of five consecutive 2D actions. The physical-transition target contains the two pusher displacements, two object displacements and wrapped object-angle change over the same interval. The state target is six-dimensional: pusher and object positions plus sine and cosine of object angle. Readouts of a real two-frame difference describe observed transitions; they are not future-state predictions.

Target	Reference (64D)	State (64D)	Act (64D)	Full (64D)
Action	0.68	0.53	0.81	0.79
State	0.86	0.83	0.80	0.91
Transition	0.57	0.54	0.71	0.79

Table 10: Equal-dimension clean readout controls. PCA is fitted separately on each model or partition's fitting features; all decoders use 64 inputs. Act is already 64D.

For centered feature covariance with eigenvalues λ_i, define $p_i = \lambda_i / \sum_j \lambda_j$ and normalized effective rank $\exp(-\sum_i p_i \log p_i)/d$. Pair it with feature standard deviation, because a scale-normalized rank alone cannot detect all features shrinking toward a constant. Rank and feature SD use the 5,000 clean probe-test snapshots, with the same sample IDs for every representation.

For horizon h, normalize autoregressive prediction error as

$$E_h = \frac{\mathbb{E}\|\hat{z}_{t+h} - F(o_{t+h})\|_2^2}{\mathbb{E}\|F(o_{t+h}) - \bar{z}_{\text{fit}}\|_2^2 + \epsilon}. \tag{11}$$

Autoregressive evaluation uses only the initial observed context and recorded future actions, without replacing intermediate predictions with observed latents. Different Encoders still define different target spaces, so lower E_h is not a universal cross-model physical-accuracy guarantee. We use the same 5,000 test-window starts for horizons $h = 1, \ldots, 5$ model steps (5 to 25 environment steps), with recorded actions and clean observations. The target mean uses fitting episodes only. Prediction-error and probe values are descriptive point estimates; the five-seed success-rate uncertainty calculations do not apply to these diagnostics.

Success-aligned candidate ranking. The ranking evaluation uses the step-0 state in each of the first 100 PushT test episodes, with its step-25 goal. We draw 64 bounded five-block action sequences per state from one state-keyed uniform proposal stream, share them across methods, and reset the simulator to the same physical state before every candidate. This gives 100 ranking sets and 6,400 terminal outcomes per reporting seed, with no samples entering training or selection. For each candidate, the Predictor rolls out the five action blocks and scores the final latent against the encoded goal using the planning cost. The simulator endpoint supplies the physical state used for evaluation.

The primary physical cost is d_{succ} in Eq. 10, including the pusher–object position vector and wrapped orientation. For each state, we compare the 64 predicted latent costs with the 64 physical costs using Kendall's τ_b. Both are ordered from smaller to larger; agreeing order gives positive τ_b. Costs are rounded to 10^{-8} for tie identification, with the usual tie correction. Undefined coefficients are rejected; all 3,000 model/seed/state groups are valid and the degenerate-group count is zero. States are averaged equally within each seed; we report the five-seed mean and sample SD.

The 0.47/0.66 diagnostic in the linked analysis concerns an *object-only* cost,

$$d_{\text{obj}}(s, g) = \frac{\|p_{\text{obj}} - p_{g,\text{obj}}\|_2^2}{L^2} + 0.25[1 - \cos(\theta_{\text{obj}} - \theta_g)], \qquad L = 512\sqrt{2}. \tag{12}$$

It omits pusher position and cannot be identified with the main success predicate. Table 4 evaluates both costs under shared candidate sets, separating object-only ordering from success-aligned ordering. This five-seed evaluation is separate from the descriptive 0.47/0.66 point estimates in the linked analysis; the two evaluations are not interchangeable. The two physical costs need not give the same ranking. Even success-aligned ranking is a diagnostic of a single 25-environment-step candidate endpoint; main success is an anytime event within a 50-step rollout with up to two planning calls. Ranking therefore does not fully explain success-rate differences or demonstrate a causal mechanism. Gaussian ranking is not reported, so clean ordering cannot establish the source of noisy success.

D Residual Scale and Corruption-Area Analysis

For a perturbed observation o', write $r = [r^{\text{state}}; r^{\text{act}}]$. The triangle inequality gives

$$\|z(o') - z(o)\|_2 \le \|g_0(c(o')) - g_0(c(o))\|_2 + \alpha\|r(o') - r(o)\|_2. \tag{13}$$

This bounds the direct contribution of residual variation for a fixed model, not the global path's sensitivity or the effect of end-to-end training. It is not a robustness theorem for the complete Encoder.

The residual-scale comparison in Table 11 illustrates why a small residual can be useful without implying that larger residuals are always better. Each row uses matched retraining. At $\alpha = 0$, Patch features make no direct residual contribution, although action alignment still shapes the CLS coordinates. The 0.1 row agrees with the main model.

Residual scale α	Clean success (%)	Gaussian success (%)
0	90	45
0.05	93	68
0.1	94	74
0.2	92	71
0.5	86	58

Table 11: PushT residual-scale sensitivity under matched retraining.

For noise levels $\sigma_0 < \cdots < \sigma_m$, normalized corruption area is

$$\text{AUC}_{\text{noise}} = \frac{1}{\sigma_m - \sigma_0} \sum_{j=0}^{m-1} (\sigma_{j+1} - \sigma_j) \frac{\text{SR}(\sigma_j) + \text{SR}(\sigma_{j+1})}{2}. \tag{14}$$

The units remain percentage success when SR is expressed in percent. Unlike ROC-AUC, this area summarizes performance over an explicitly chosen corruption range; changing the range changes the statistic.

Model	Normalized noise AUC (%)
Reference	67.60
Uniform pooling	78.60
Branch-JEPA	87.25

Table 12: Normalized area computed from the mean success rates in Figure 3a over $\sigma \in [0, 0.05]$. No separate uncertainty estimate is reported for these integrated summaries.

E Controls for Representation Organization

E.1 What Each Retrained Variant Changes

All controls retain the 192D latent, global CLS path, shared Predictor, Action Encoder, planning objective and input preprocessing. For the capacity-matched single-space controls, used projection-head widths target a total trainable parameter count within 1% of the complete model; unused dummy parameters do not count. Shared-router and factorial controls instead fix head widths to isolate the specified change. These comparisons do not establish equal total training cost; parameter counts and runtimes are not reported.

Single Patch space with full-latent alignment. One spatial softmax pools the same Patch features, followed by one 192D residual head. The original CLS projection is preserved. The contrastive displacement head receives the complete 192D difference and projects to 64D; positive actions, negative filtering, temperature and loss weight are the same as the complete model. This controls access to Patch information and an action-matching auxiliary loss, but jointly changes partitioning and where that loss acts. It tests the proposed package, not branch count in isolation.

Adapted action-regression controls. The global-CLS and single-Patch regression variants decode the full 192D observed endpoint difference using two linear layers $192 \to 192 \to d_a$ with GELU. The dual-router Act-regression variant keeps the complete dual-branch Encoder but uses $64 \to 192 \to d_a$ on the Act difference. All regress the same training-standardized five-action block with coordinate-mean MSE (Eq. 8); no action-derived statistics are fitted on validation or test. Auxiliary weight has the same $\{0.0005, 0.005, 0.05\}$ grid as contrastive alignment.

The stronger single-Patch control adopts Delta-JEPA's action-decoding structure: five learned queries, one per native action in a block, with AdaLN conditioning on the observed full-latent displacement; three noncausal Transformer layers, eight attention heads of dimension 64 and feed-forward width 512; and a per-query native-action output projection. Its outputs are concatenated and scored by the same standardized MSE. The global backbone, Predictor, prediction loss and SIGReg remain those of this framework, including for the Transformer control. Delta-JEPA's original complete system is therefore not reproduced; the comparison isolates an adapted decoder and supervision scheme (Zhang et al., 2026). All four controls have the same data, 100-epoch schedule, optimizer steps and six-trial search budget as Branch-JEPA. Auxiliary parameter counts and training FLOPs differ and have not been measured; equal update budgets do not imply equal total computation. All regression/contrastive decoders are removed for planning.

Shared router with two heads. A single spatial score network supplies identical weights to the two residual heads. Their output dimensions remain 128 and 64; action alignment still uses only the fused Act difference. This comparison changes independent spatial selection while retaining the output partitions and loss location. It is therefore more specific than comparing the full model with a wider global representation.

Routing-by-alignment control. Keep both residual heads and vary only learned versus uniform weights and the alignment-loss coefficient (0 or 0.005). Table 13 supplies the uniform/no-alignment cell. Routing's noisy benefit is 10 pp without alignment and 17 pp with it. The difference-in-differences is 7 pp, a descriptive interaction of the reported scores, not a statistically established synergy. Accuracy bounds and optimization can also create nonadditive effects on this scale.

Spatial pooling	Action alignment	Clean (%)	Gaussian (%)
Uniform	Off	89	51
Uniform	On	91	57
Learned	Off	90	61
Learned	On	94	74

Table 13: Complete two-factor retraining control. All rows retain the two residual heads; changing either factor need not have the same effect at both levels of the other factor.

Full-latent alignment with the same dual-router Encoder. Retain both independent routers, the 128/64 residual heads, CLS fusion, Action Encoder, Predictor and planning objective. Replace the input to the displacement contrast head from $\Delta z^{\mathrm{act}} \in \mathbb{R}^{64}$ to $\Delta z \in \mathbb{R}^{192}$. Hidden and output widths, action projection, normalization, temperature, negative selection and loss weight remain the same. If the first displacement-head layer has width h, the full-latent control has $128h$ additional auxiliary weights. Thus the representation and inference architecture are fixed, but the training-head parameter counts are not identical. The comparison tests alignment location under a common head family; it does not establish a capacity-independent theorem. All compared checkpoints use clean validation selection and the same prespecified training budget, not corruption-conditioned selection.

E.2 Absolute Readouts and Specific State Variables

Table 14 reports both partitions' absolute scores, not only their difference. In the clean condition, Act improves by 0.20 while State decreases by 0.01; under noise, both increase, by 0.24 and 0.02. Thus, the selectivity effect is not produced solely by impoverishing the other partition. The small state-readout changes are descriptive; without repeated fits and uncertainty, they do not establish statistical equivalence.

	Action, clean		Action, noisy		State, clean	
Training variant	State	Act	State	Act	State	Act
No alignment	0.57	0.61	0.39	0.45	0.88	0.79
Shared router	0.54	0.77	0.40	0.65	0.86	0.79
Full-latent alignment	0.67	0.75	0.52	0.62	0.84	0.82
Act-only (full)	0.56	0.81	0.41	0.69	0.87	0.80

Table 14: Absolute readouts for the linked training variants. All entries are probe R^2 at native partition dimension. The Act-only row matches the main readout table; full-latent alignment retains both independent routers, whereas the shared-router row changes spatial selection.

State target	State branch	Act branch	Full latent
Pusher position	0.93	0.98	0.99
Object position	0.90	0.84	0.97
Orientation (sin/cos)	0.78	0.58	0.92

Table 15: Clean state readouts resolved by target group. Each group contains two target coordinates, so the mean of the three group scores equals the six-coordinate state score in the main table.

Position and orientation readouts concern PushT configuration. They are not evidence that a branch represents navigation terrain, affordances or semantic objects in another task. Those interpretations require their own labels and controls. Full-latent improvements over either branch also require the equal-dimension check in Table 10.

E.3 Source and Checkpoint-Matched Diagnostics

Information sources. For the complete-model checkpoint, extract the branch coordinates of $g_0(c_t)$, the scaled residual αr_t^b, and the fused z_t^b. Do not re-encode under a different interven-

tion or retrain the world model. For action targets, form endpoint differences before fitting PCA; for physical-state targets, use snapshots. Fit PCA on probe-fitting features only, reducing each 128D State source to 64D. Act sources already have 64 coordinates. Standardize the resulting inputs using fitting data and fit separate ridge probes with the same validation grid and targets. The fused rows reproduce Table 10's partition entries. These readouts localize accessible information; correlated sources and scale normalization preclude interpreting score differences as additive information shares or amplitude-weighted contributions to planning.

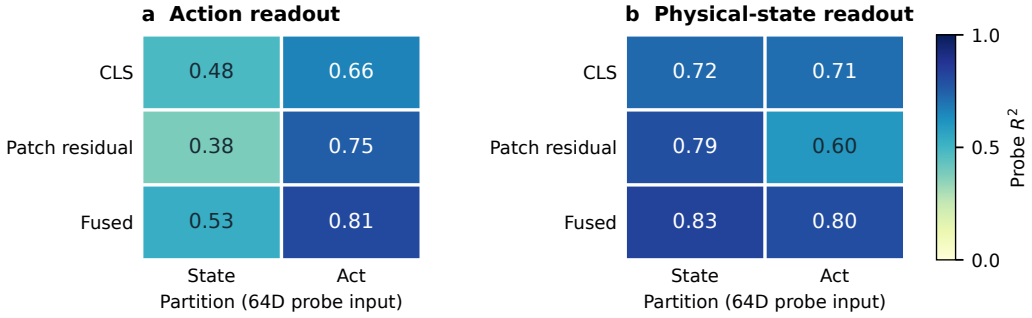

Figure 4: Clean PushT readouts from CLS, Patch residual and fused features. Each cell uses an independently fitted linear probe with 64 inputs. Fused scores match the equal-64D controls. Probe scores locate readable information; they are not additive causal contributions.

Linking readout to behavior. Checkpoints are selected by the common clean-validation rule and frozen during diagnostic evaluation. We use the same held-out episode split and complete 192D observed displacement for every full action probe, with no PCA or branch selection; standardization and ridge decoders are fitted using training episodes and the common validation grid. Native State/Act readouts in Table 14 ask where action information is accessible, whereas a complete-latent readout asks how much is accessible in total. The linked table reports descriptive probe and object-only ranking point estimates; only its success columns summarize five reporting seeds. The full-model full readout matches Table 3; other full readouts are fitted separately, rather than obtained by adding partition scores. Five-seed candidate ranking is reported separately in Table 4.

The evaluations share physical initial states, goals and proposal keys, as specified in Appendix C; clean planning pairs initial conditions, and noisy planning pairs corruption draws rather than diverging observations (Appendix A). Probes use observed transitions, candidate ranking branches from reset simulator states, and planning advances a trajectory. These evaluations use distinct samples. A given model's checkpoint is shared across these diagnostics, whereas probe parameters are fitted separately for each variant. Both full-latent readout and candidate ranking are clean; Gaussian success is a separate stress-test endpoint. The aggregate patterns do not establish within-checkpoint cross-metric correlations or causal mediation of planning performance.

Variant	Full R^2	Object τ_b	Clean SR (%)	Noisy SR (%)
No alignment	0.73	0.56	90	61
Shared router	0.82	0.61	92	69
Full-latent alignment	0.84	0.62	92	70
Act-only (full)	0.83	0.66	94	74

Table 16: PushT readouts, ranking and planning. Readout and object-only ranking are clean descriptive point estimates; noisy success uses Gaussian $\sigma = 0.05$. SR reuses five-run means from Table 2. Cross-metric run correlations are not estimated.

E.4 Interventions That Preserve the CLS Path

Let U_b embed branch b's residual into its designated coordinates of the full latent. For every real current or goal image, define

$$F_{-b}(o) = F(o) - \alpha U_b r^b(o). \tag{15}$$

Neither the CLS projection nor the other residual is replaced. Goal images use the same intervention type as current images. The Predictor and its predicted coordinates remain unchanged during imagined rollouts. This is an *observation-code intervention*, not a new dynamics model consistently trained in the intervened space; mismatch remains a possible contributor.

For a norm-matched control, use

$$F_{\mathrm{rand},b}(o) = F(o) + \alpha \|r^b(o)\|_2 U_b u_b, \qquad \|u_b\|_2 = 1, \tag{16}$$

where u_b is sampled uniformly on the unit sphere in the same branch coordinates. Draw and pair directions by case and observation identity; keep the goal perturbation fixed within an episode. The change magnitude matches residual deletion for that image. This controls norm and coordinate support, not semantic direction or manifold membership.

Encoder intervention	Clean (%)	Gaussian (%)
Full	94	74
State residual off	88	63
Act residual off	90	66
Both residuals off	84	53
Uniform routing	88	62
State random control	93	71
Act random control	93	72

Table 17: CLS-preserving interventions and matched-norm controls. These interventions test trained-model dependence, not the benefit of retraining each module or the impossibility of compensating for its removal.

F Numerical Summaries and Reproducibility

The source contains the table values and vector figures used throughout this manuscript. The appendix reports per-run planning percentages, aggregate means and sample SDs, paired intervals, and candidate-ranking statistics. Probe and prediction-error point estimates remain separate from the five-run planning and ranking summaries. The source compiles without external data or figure files; raw trajectories, training logs and checkpoint files are not part of this manuscript package.

The Reference and adapted comparison methods are evaluated within the protocol described here. Their numerical results are not attributed to the experiments of the cited publications.