NESTED LATENT REPRESENTATIONS FOR SIZE-FLEXIBLE END-TO-END WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent world models for planning typically assume a fixed representation size. While recent work, such as SubJEPA, considered operating in subspaces of smaller dimensionality, they require additional task-specific tuning of the subspace granularity. In this work, we ask whether world models can learn representations that remains useful across multiple post-hoc embedding sizes. As an answer, we introduce ElasticJEPA which jointly trains a world model at multiple nested granularities. ElasticJEPA combines subspace regularization with prediction losses applied to the same prefixes used at inference. The novel prediction loss objective induces a coordinate-supervision profile that is naturally monotonically-ordered. ElasticJEPA also ensures that privileging prefixes are retained under truncation. With such a construction, a single trained model can be evaluated at multiple representation sizes with no need for retraining. Importantly, inference-time evaluation can also include widths not supervised during training. Across four environments, ElasticJEPA is competitive with prior methods at full width, retains most of its performance under substantial truncation, and degrades less than baselines truncated to the same size, most clearly at the smallest widths.

1 INTRODUCTION

World models (Ha & Schmidhuber, 2018; Hafner et al., 2019; 2020) planning in latent space requires the representations be sufficiently informative to support predictions and well-structured to avoid trivial collapse of the latent representations. To this end, recent approaches have proposed representation regularizers Maes et al. (2026); Balestriero & LeCun (2025) to reduce reliance on heuristics, such as stop-gradients, EMA targets, and multi-term objectives, used in self-supervised world models. Building on this further, recent work (Zhao et al., 2026) proposed subspace-based regularization, which is often desirable when the task dynamics lie on lower-dimensional manifolds. These approaches, however, still commit to a fixed representation size during training. While (Zhao et al., 2026) do show that subspace regularization subspaces can improve performance, the approach requires a careful specification of a granularity parameter K that must be tuned carefully for each task, and the performance is reported to be sensitive to the value of K (Zhao et al., 2026).

Matryoshka Representation Learning (MRL) (Kusupati et al., 2022) provides a natural starting point for addressing this limitation. MRL trains a single embedding whose nested prefixes are each independently useful by applying the task loss at every prefix length. In principle, this would allow one model to represent all candidate values of K. However, combining MRL with Sub-JEPA is not straightforward. Sub-JEPA's regularizer is permutation symmetric, so all subspaces are treated interchangeably. Which is precisely what makes Sub-JEPA less restrictive than LeWM. Matryoshka-style nesting, in contrast, relies on an ordered representation in which earlier dimensions have a distinguished role. As a result, applying the same symmetric regularizer at multiple prefix lengths does not by itself induce the hierarchy that Matryoshka learning requires. Some additional mechanism is needed to break the symmetry and make the prefixes meaningfully distinct.

We introduce **ElasticJEPA** to address this limitation by combining nested regularization with a nested prediction loss applied across prefix lengths, Figure 1. For each candidate K, the same coordinate prefix that would be used at inference time is trained to be predictive on its own. This supervision is applied across all candidate values of K, which forces the model to concentrate on useful information in the earlier dimensions of the embedding. This constrain cannot be provided

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.

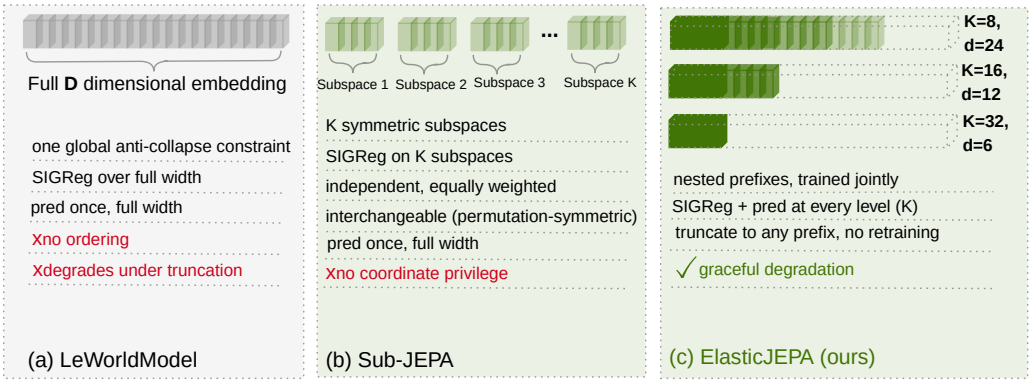


Figure 1: (a) LeWM enforces one global anti-collapse constraint on the full D-dim embedding. (b) SubJEPA relaxes this to K independent, symmetric subspaces that are interchangeable and have no ordering. Our ElasticJEPA instead nests K prefixes and re-applies both the regularizer and the task loss L_{pred} redundantly at every nesting depth. Shading in (c) visualizes the resulting coordinate privilege. The first block is trained at every K, the last only at the finest. Absence of this asymmetry from both baselines is what makes post-hoc truncation (bottom) viable without retraining.

alone by nested regularization. This allows a single trained model to be truncated to several smaller embedding sizes without retraining or a separate grid search over K. Across four planning tasks, embeddings an eighth of the full size perform close to the full embedding, and at the smallest widths ElasticJEPA degrades less than LeWM or Sub-JEPA truncated in the same way.

Concretely, we make the following contributions.

- **We introduce ElasticJEPA**, a size-flexible end-to-end JEPA world model. ElasticJEPA learns *nested* representation prefixes using nested subspace regularization and prediction losses. This enables a single model to operate at multiple embedding granularities without retraining.
- **We identify why nested regularization alone is insufficient**. We show analytically and empirically that prediction supervision at each nesting depth induces coordinate privilege needed for useful truncation.
- **We demonstrate post-hoc size flexibility** across four continuous-control environments. Our demonstration also includes successful evaluation at embedding widths not explicitly supervised during training.
- **We analyze what information is retained under truncation** using linear and nonlinear representation probes. Our analysis shows that some information can remain present even when it becomes difficult to access with a linear planning metric.

2 Related Work

World models and latent planning. Learning the predictive models of environment dynamics for planning has a long history. Schmidhuber (1990) used recurrent neural networks as learned models for reinforcement learning and planning, and Ha & Schmidhuber (2018) later showed that control can be learned through compact latent world models. This idea was developed further by PlaNet (Hafner et al., 2019), Dreamer (Hafner et al., 2020), and TD-MPC2 (Hansen et al., 2024), which learn latent dynamics that support prediction or control without modeling every detail of the observation. More recent reward free approaches move closer to our setting. PLDM (Sobal et al., 2025) learns latent dynamics from offline trajectories for test time planning, while DINO-WM (Zhou et al., 2025) predicts the evolution of pretrained visual features. These methods show that useful planning models can operate entirely in representation space, but the dimensionality of that space is fixed once the model is trained.

JEPA world models and collapse prevention. JEPAs learn representations by predicting in latent space instead of reconstructing observations (Assran et al., 2023; Bardes et al., 2024; Assran et al., 2025). A central challenge in training such models end to end is preventing the representation from collapsing. Earlier self supervised methods such as Barlow Twins (Zbontar et al., 2021) and VICReg (Bardes et al., 2022) address collapse by directly constraining the statistics of the learned embeddings. LeJEPA (Balestriero & LeCun, 2025) develops this idea through SIGReg, which encourages the embedding distribution to match an isotropic Gaussian. LeWM (Maes et al., 2026) brings this objective to action conditioned world models and enables stable training directly from pixels. Sub-JEPA (Zhao et al., 2026) relaxes the full space constraint by regularizing random subspaces, giving the representation more freedom while preserving the anti collapse effect. The subspace dimension K, however, remains a training choice and must be selected before the model is deployed.

Ordered and elastic representations. The idea of making a single representation useful at several dimensionalities predates modern world models. Nested Dropout (Rippel et al., 2014) learns an explicit ordering over latent dimensions by repeatedly training on nested subsets of the representation. Matryoshka Representation Learning (Kusupati et al., 2022) builds on the same principle by directly supervising multiple coordinate prefixes of a shared embedding. Related work on universally slimmable networks (Yu & Huang, 2019) and once for all networks (Cai et al., 2020) similarly trains a single model that can operate at different capacities, although the flexibility is expressed through network width instead of the size of a shared embedding. What these approaches have in common is that training gives different parts of the representation distinct roles.

ElasticJEPA brings this idea into end to end JEPA world models. This requires resolving a tension that does not arise in the settings above. Sub-JEPA deliberately treats its regularized subspaces interchangeably, while a nested representation needs particular coordinate prefixes to be useful on their own. ElasticJEPA introduces this structure through prediction losses at multiple nesting depths while retaining subspace regularization for stable training. A single trained world model can therefore support several embedding sizes without choosing one fixed K in advance.

3 ELASTICJEPA

In this section, we explain our method in detail, answering questions like why naïve nesting does not work, how we construct the projections, what is the loss function for our method, and finally designing control experiments to identify the performance effects of different loss setups.

3.1 WHY NAÏVE NESTING IS INSUFFICIENT

Sub-JEPA's regularizer is invariant to how the K subspaces are labeled. Which one is called "first", or which coordinates constitute the "j-th subspace" after permutation, does not change $\mathrm{SIGReg}(\{P^{(j)}z\}_{j=1}^{K})$. What matters is that the marginal of each subspace should be Gaussian, independently of the other subspaces. But the nesting property is quite the opposite. The same first m coordinates should always be privileged over the rest. This ensures that even after the truncation to that length, the information loss is minimal. Training the symmetric regularizer at several values of K does not ensure this. In every K, the subspaces are still interchangeable in terms of loss. As a result, the regularizer term in the loss will not give any signal about which subspace to keep at inference, and any post-hoc choice, such as always keeping "subspace 0," is arbitrary. The regularizer never trained the model to treat it as special. Section 3.3 addresses this gap by adding an asymmetric task loss, applied at the same truncation depths as the regularizer. Section 5.2 tests this argument empirically rather than keeping it purely analytical.

3.2 CONSTRUCTION

Let $\mathcal{K} = \{1, 2, 4, 8, 16, 32\}$ be the granularities we train jointly, all divisors of max_k $= 32$. We build one global orthogonal matrix $Q \in \mathbb{R}^{D\times D}$ ($Q^\top Q = I$) and reuse it for every granularity. To build it, we fill a $D \times D$ matrix with random numbers and take the Q factor of its QR decomposition, producing a genuinely random orthogonal matrix, with no direction favored over any other. Q is fixed once, at initialization, and never updated again, through training or at inference, so it is not a learned parameter.

We cut Q's rows into max_k $= 32$ blocks of $d_{\min} = D/\text{max_k} = 6$ rows each. For a granularity $K \in \mathcal{K}$, the K projections $P_K^{(1)}, \ldots, P_K^{(K)} \in \mathbb{R}^{D/K \times D}$ are formed by grouping $g = \text{max_k}/K$ adjacent blocks together, always the same rows of the same fixed Q, never redrawn. Because every level reuses rows from the same matrix, $P_{2K}^{(1)}$ is always exactly the row-prefix of $P_K^{(1)}$: truncating to any granularity's first block truncates to a literal prefix of every coarser granularity's first block too, the nesting property this construction is built to give. Since the retained view at any width w is just the first w rows of this, one, fixed matrix, the construction is well-defined at *any* width from 1 to D, not only the six granularities we explicitly train on (Section 5.1 makes direct use of this).

Drawing one global Q and slicing it, rather than drawing each of the max_k blocks separately, is what makes every $P_K^{(j)}$ exactly row-orthonormal and mutually orthogonal to every other block at every level. We initially tried the per-block alternative: each block is valid on its own, but nothing coordinates the separate draws, so blocks end up slightly overlapping instead of perfectly perpendicular. The per-block construction has a condition number of 600–1300$\times$ across random seeds, where a true isometry (a transform that preserves distances exactly) should measure exactly 1.0; the global-Q construction used throughout this paper measures 1.0000008, with cross-block dot products of order 10^{-7}. Beyond the condition number itself, the two constructions give different downstream results: under the per-block construction, PushT and Cube's truncated-inference success rate fell substantially more steeply with K, because a non-isometric basis distorts the distances the CEM planner minimizes when selecting actions, making its cost function a less reliable guide to which actions reach the goal, independent of whether the representation itself is any less informative at that width.

3.3 Nested Regularization and Prediction Objective

Let z_{t+1} denote the target encoder's embedding of the true next observation and $\hat{z}_{t+1}$ the predictor's embedding of the predicted next observation, both in $\mathbb{R}^D$. We train two nested terms jointly with the standard prediction loss:

$$\mathcal{L}_{\text{reg}} = \frac{1}{|\mathcal{K}|} \sum_{K \in \mathcal{K}} w_K \cdot \text{SIGReg}\big(\{P_K^{(j)} z\}_{j=1}^{K}\big), \tag{1}$$

$$\mathcal{L}_{\text{trunc-pred}} = \frac{1}{|\mathcal{K}|} \sum_{K \in \mathcal{K}} w_K \cdot \big\| P_K^{(1)} (\hat{z}_{t+1} - z_{t+1}) \big\|_2^2, \tag{2}$$

with weights w_K normalized so $\frac{1}{|\mathcal{K}|} \sum_K w_K = 1$.

Nested prediction induces ordered coordinate supervision. The asymmetry introduced by $\mathcal{L}_{\text{trunc-pred}}$ can be characterized exactly. Let $e = Q(\hat{z}_{t+1} - z_{t+1}) = (e_1, \ldots, e_{K_{\max}})$, where each $e_b \in \mathbb{R}^{d_{\min}}$ is one minimal block in the fixed orthogonal basis Q, with $d_{\min} = D/K_{\max}$.

Proposition 1 (Ordered supervision from nested prediction)**.** *Let every $K \in \mathcal{K}$ divide $K_{\max}$, and assume $1 \in \mathcal{K}$ and $w_K > 0$ for all $K \in \mathcal{K}$. Then*

$$\mathcal{L}_{\text{trunc-pred}} = \sum_{b=1}^{K_{\max}} \alpha_b \, \|e_b\|_2^2, \qquad \alpha_b = \frac{1}{|\mathcal{K}|} \sum_{\substack{K \in \mathcal{K} \\ b \le K_{\max}/K}} w_K,$$

and the block weights satisfy

$$\alpha_1 \;\ge\; \alpha_2 \;\ge\; \cdots \;\ge\; \alpha_{K_{\max}} \;>\; 0.$$

Thus nested prediction is equivalent to a block-weighted prediction loss in which supervision is non-increasing along the representation: every level that retains a block also retains all earlier blocks, so earlier blocks are supervised by at least as many truncation levels as later ones.

This result makes precise the coordinate privilege of ElasticJEPA. Unlike SubJEPA regularizer, nested prediction explicitly distinguishes the prefix retained at inference time. The proof and the induced exposure profile for our choice $\mathcal{K} = \{1, 2, 4, 8, 16, 32\}$ are given in Appendix A.

The full training objective combines the nested prediction objection and the nested regularization with a single scalar weight λ in a similar manner as LeWM and Sub-JEPA

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{trunc-pred}} + \lambda\,\mathcal{L}_{\text{reg}}, \qquad \lambda = 0.09. \tag{3}$$

The $K{=}1$ term of $\mathcal{L}_{\text{trunc-pred}}$ uses the full $D \times D$ orthogonal transform, so it is numerically identical to the plain full-embedding prediction loss: the standard loss is not replaced, it is the $K{=}1$ case of $\mathcal{L}_{\text{trunc-pred}}$, with the smaller-prefix terms added alongside it. $\mathcal{L}_{\text{trunc-pred}}$ supplies the asymmetry that Section 3.1 argues $\mathcal{L}_{\text{reg}}$ cannot. It directly supervises the exact truncated view used at inference time (Section 4), giving the model a real reason to make that specific prefix predictively useful. Nesting the regularizer alone, no matter how finely, cannot do this.

We consider two weighting schemes for w_K: **uniform** ($w_K \equiv 1$) and **sqrt-K** ($w_K \propto \sqrt{K}$, normalized to mean 1), which upweights finer granularities in both Equations 1 and 2. The motivation is that small prefixes are the ones most at risk of losing predictive signal under truncation and might need more supervision; Section C tests whether this is true.

3.4 A FACTORIAL VIEW OF THE OBJECTIVE

The construction above changes two things relative to LeWM: the regularizer is applied at multiple nested granularities, and the prediction loss is applied to nested prefixes rather than only to the full embedding. These are logically separable, and Section 5.2 tests them separately. Write $\mathcal{L}_{FP} = \|\hat{z}_{t+1} - z_{t+1}\|_2^2$ for the plain full-width prediction loss and $\mathcal{R}_F = \text{SIGReg}(z)$ for the plain full-space regularizer (together, exactly LeWM's objective), and $\mathcal{L}_{NP}$, $\mathcal{R}_N$ for their nested counterparts (Equations 2 and 1). Four combinations are then possible:

$\mathcal{R}_F + \mathcal{L}_{FP}$	LeWM baseline
$\mathcal{R}_N + \mathcal{L}_{FP}$	nested regularizer, ordinary full-width prediction
$\mathcal{R}_F + \mathcal{L}_{NP}$	full-space regularizer, nested prediction
$\mathcal{R}_N + \mathcal{L}_{NP}$	ElasticJEPA, ours

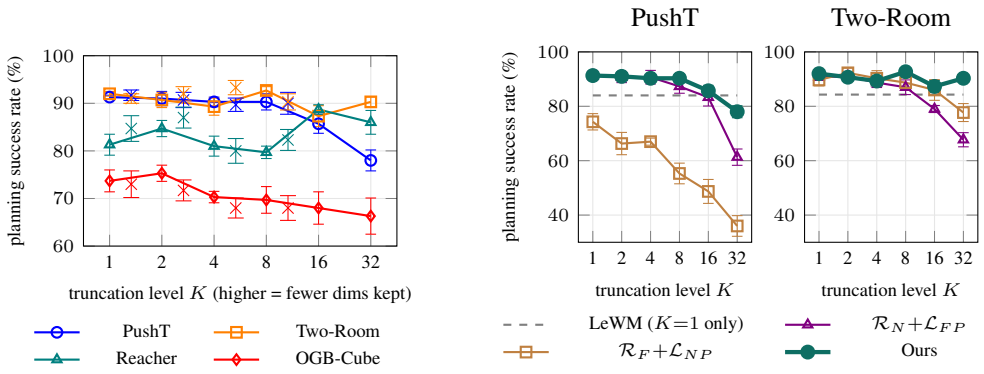
Only the middle two require new training runs. $\mathcal{R}_N{+}\mathcal{L}_{FP}$ tests whether nesting the regularizer alone is sufficient for truncatability. Section 3.1 argues analytically that it should not be, and this makes that argument directly falsifiable. $\mathcal{R}_F{+}\mathcal{L}_{NP}$ tests the opposite question: whether nesting the prediction loss alone already accounts for most of the benefit, with no multi-subspace regularization at all. If so, a simpler method than ours would suffice. We implement all four conditions from the same code, not four separate modules. The regularizer and the prediction loss each read their own K-level set from Equations 1-2: our full method sets both to the full six-level set $\mathcal{K}$; the middle two conditions come from setting only one of the two to that full set and leaving the other at just 1. Everything else (basis Q, architecture, optimizer, data) stays identical across all four conditions.

4 EXPERIMENTAL SETUP

Environments. We follow the experimental protocol of (Maes et al., 2026) and (Zhao et al., 2026) and use four continuous control benchmarks: **TwoRooms**, a 2-D low-dimensional manifold task where an agent moves from one room to another at a target position; **PushT**, a complex 2-D task where an agent pushes the object to a specific configuration; **OGBench-Cube**, a visually rich complex 3-D task where a robotic arm grabs and places an object to a specified position; and **Reacher**, a 2-D task where a robotic arm moves from one specified configuration to another.

Baselines. We compare ElasticJEPA against state-of-the-art baselines on 2-D and 3-D control tasks, specifically, **LeWorldModel** (Maes et al., 2026), which uses JEPA-style training with a prediction loss and a SIGReg regularizer for high-dimensional embeddings, and **SubJEPA** (Zhao et al., 2026), which uses subspace regularization for higher-dimensional embeddings, specifically helpful in intrinsically lower-dimensional manifold tasks.

Inputs and architecture. All four environments provide pixel observations (224×224 RGB images), encoded by a ViT-style patch encoder (patch size 14, "tiny" scale) shared by all three methods we compare. None of the reported differences come from a different encoder architecture or input modality; the $D{=}192$ embedding dimension is this encoder's output width (and $D{=}384$ for the capacity-scaling experiments of Section 5.6).



(a) Overall truncation sweep **(b) Factorial objective ablation**

Figure 2: **(a)** Truncated-inference planning success rate vs. truncation level K—one model per environment, evaluated at every K with no retraining. Circles/squares/triangles/diamonds connected by lines are the explicitly trained granularities; the four $\times$-marked points per environment are widths never supervised during training (144/72/36/18 dims), plotted at their equivalent $K_{\text{eff}} = D/\text{width}$ position. **(b)** Factorial objective ablation: truncated-inference success rate vs. K, isolating regularizer nesting from prediction-loss nesting. LeWM is shown as a dashed reference at its single $K = 1$ value; $\mathcal{R}_N + \mathcal{L}_{FP}$ and $\mathcal{R}_F + \mathcal{L}_{NP}$ are the two factorial conditions, with Ours as the full method.

Evaluation protocol. All success rates are planning success rate (%) over n=50 episodes per seed, using CEM planning (population 300, 30 iterations, elite set 30, horizon 5 steps), identical across all methods and all experiments in this paper. Unless stated otherwise, every number is a mean over 6 seeds, matching the 6-seed protocol Sub-JEPA's own paper documents using for its own reported numbers; error bars are standard error over these 6 seeds.

Truncated-inference evaluation. At inference, truncation applies to exactly one thing: the planning *cost function* used inside CEM. Predicted and goal embeddings are projected through $P_K^{(1)}$ before their distance is compared, the same projection $\mathcal{L}_{\text{trunc-pred}}$ used during training. The dynamics predictor itself is never truncated. It always consumes and produces the full D-dimensional embedding, no matter which K the planning cost is evaluated at.

5 Results

5.1 Post-hoc truncation across trained and untrained widths

Figure 2(a) highlights the major contribution of our idea: for each environment, a single ElasticJEPA model is trained once, then evaluated at all six granularities with no retraining. The results depict that 2 of the environments, Reacher and Two-Room, degrade gracefully, i.e., their $K = 1$ and $K = 32$ performance all lie within the same statistical range, whereas PushT shows a clear drop as the prefix shrinks, and Cube a smaller one. Section 5.5 examines why these two environments differ from other two.

Another major highlight of the Figure 2(a) is that we evaluate our method on the untrained embedding prefixes (144, 72, 36, and 18 dimensions) using the same checkpoints and no further training. Every one of the sixteen (environment, untrained-width) combinations falls within the noise of its two trained neighbors, including on PushT and Cube, where the trained sweep itself declines. The representation degrades continuously across the whole range, rather than behaving like six separate, memorized operating points.

5.2 Factorial Ablation: What makes the representation truncatable?

Section 3.4 separates ElasticJEPA's objective into two independent factors: whether the regularizer is nested, and whether the prediction loss is nested. Figure 2(b) reports the two new conditions this produces, trained on PushT and Two-Room under otherwise identical settings, alongside the two already-reported endpoints (LeWM and our full method).

Table 1: Truncated-inference success rate across matched widths for all three methods, all four environments. LeWM and Sub-JEPA use a fixed slice of their already-trained embeddings, no retraining (mean ± SE, 2 seeds); ElasticJEPA is the existing nested-basis result (6 seeds). 192d values are the full-embedding numbers from Table 2.

Env.	Method	192d	96d	24d	6d
PushT	LeWM	84.0 ± 2.7	87.0 ± 9.0	76.0 ± 8.0	49.0 ± 7.0
	Sub-JEPA	$\mathbf{92.0 \pm 2.1}$	$\mathbf{91.0 \pm 7.0}$	82.0 ± 6.0	57.0 ± 3.0
	ElasticJEPA	91.3 ± 0.4	$\mathbf{91.0 \pm 1.5}$	$\mathbf{90.3 \pm 1.7}$	$\mathbf{78.0 \pm 2.2}$
Two-Room	LeWM	84.3 ± 1.7	81.0 ± 7.0	68.0 ± 6.0	53.0 ± 7.0
	Sub-JEPA	$\mathbf{95.0 \pm 1.1}$	$\mathbf{92.0 \pm 4.0}$	$\mathbf{94.0 \pm 2.0}$	79.0 ± 9.0
	ElasticJEPA	92.0 ± 0.5	90.7 ± 1.6	92.7 ± 0.8	$\mathbf{90.3 \pm 1.0}$
Reacher	LeWM	$\mathbf{85.3 \pm 2.2}$	81.0 ± 1.0	76.0 ± 0.0	74.0 ± 6.0
	Sub-JEPA	85.0 ± 1.7	80.0 ± 0.0	76.0 ± 2.0	80.0 ± 6.0
	ElasticJEPA	81.3 ± 2.2	$\mathbf{84.7 \pm 1.7}$	$\mathbf{79.7 \pm 1.3}$	$\mathbf{86.0 \pm 2.5}$
Cube	LeWM	67.0 ± 2.4	65.0 ± 7.0	63.0 ± 5.0	54.0 ± 6.0
	Sub-JEPA	$\mathbf{76.7 \pm 3.0}$	71.0 ± 9.0	$\mathbf{71.0 \pm 7.0}$	63.0 ± 3.0
	ElasticJEPA	73.7 ± 2.3	$\mathbf{75.3 \pm 1.7}$	69.7 ± 2.8	$\mathbf{66.3 \pm 3.8}$

Nesting the regularizer alone ($\mathcal{R}_N$+$\mathcal{L}_{FP}$) fails in the same way on both environments. It tracks close to our full method through moderate truncation, then collapses sharply at the single finest level, K=32 (PushT: 83.3 → 61.3; Two-Room: 79.0 → 67.7). This empirically supports Section 3.1: nesting the regularizer alone cannot manufacture coordinate privilege on its own, and the failure is concentrated exactly where there is no privileged prefix left to fall back on.

Nesting the prediction loss alone ($\mathcal{R}_F$+$\mathcal{L}_{NP}$) does not replicate across environments. On PushT it performs poorly even at K=1 (74.3, below LeWM's 84.0) and degrades steeply under truncation. On Two-Room it is competitive with the other ablated condition throughout, and even exceeds it at K=32. We do not have a confirmed explanation for this difference, and did not extend the ablation to Reacher and Cube. On PushT, neither component alone matches the full method; on Two-Room, nested prediction alone comes close. Whether both components are needed may therefore depend on the task as well.

5.3 Truncated Baseline Comparison

We tested whether LeWM and Sub-JEPA degrade just as gracefully under the same kind of truncation. Both were sliced with a fixed random projection of their already-trained embeddings, with no retraining. Table 1 shows that they do not. At the smallest size, ElasticJEPA is ahead of both baselines on every environment. The margin is large on PushT and Two-Room, where both baselines have already dropped substantially, and within noise on Reacher and Cube. At wider truncations, the picture is more mixed, and a baseline occasionally matches or slightly exceeds ElasticJEPA; the benefit of the nested construction shows mainly as the embedding gets small. The truncated baselines are averaged over two seeds, so their error bars are less reliable than ours.

5.4 Success Rate at Full Scale

To check whether the additional functionality that ElasticJEPA provides does not come at the cost of full-scale embedding performance, Table 2 presents the results for ElasticJEPA($D = 1$) directly comparable with LeWM and SubJEPA. We reproduced the results for LeWM and SubJEPA based on their checkpoints and the same controlled setup. Their own numbers from the paper may vary a little, and a comparison for the same is presented in Table 7- 8 (Appendix D), alongwith a discussion of why that happens and why our reproduction is within the noise margin of what each paper reports.

The table highlights that our method is sufficiently competitive with the existing baseline methods, not necessarily dominant. But being dominant at fixed embedding sizes is not the main contribution of our method as well; instead, it is to provide additional post-hoc size flexibility at competitive full-scale performance, which the baselines do not provide without substantial degradation (Section 5.3).

Table 2: Full-embedding performance comparison across environments. Values are success rate (%), reported as mean ± standard error over 6 seeds.

Environment	LeWM	Sub-JEPA (hand-tuned K)	ElasticJEPA (ours)
PushT	84.0 ± 2.7	92.0 ± 2.1	91.3 ± 0.4
Two-Room	84.3 ± 1.7	95.0 ± 1.1	92.0 ± 0.5
Reacher	85.3 ± 2.2	85.0 ± 1.7	81.3 ± 2.2
Cube	67.0 ± 2.4	76.7 ± 3.0	73.7 ± 2.3

Table 3: Linear-probe versus MLP-probe held-out R^2 at the largest (K=1) and smallest (K=32) representation sizes.

Environment	Target	K=1 (linear → MLP)	K=32 (linear → MLP)
PushT	**block angle**	$0.763 \rightarrow 0.877$	$\mathbf{0.056 \rightarrow 0.543}$
Two-Room	agent position	$0.997 \rightarrow 0.995$	$0.436 \rightarrow 0.959$
Reacher	finger position	$0.999 \rightarrow 0.908$	$0.282 \rightarrow 0.995$
Cube	block position	$0.994 \rightarrow 0.994$	$0.489 \rightarrow 0.947$
Cube	**block yaw**	$-0.046 \rightarrow -0.069$	$\mathbf{-0.002 \rightarrow 0.001}$

5.5 Representation-level probing

Section 5.1 shows the performance of PushT and Cube degrades significantly under truncation, while that of Two-Room and Reacher does not. So does truncation destroy information which the small representation needs, or is the information still present but is inaccessible to the planning cost function, which compares embeddings by linear distance? We test this by freezing a truncated embedding and training a small readout model to predict a known quantity from it, first a linear probe, then a small MLP. If the linear probe fails but the MLP succeeds, the information has survived truncation, it is simply arranged in a way a straight-line readout cannot extract, exactly the arrangement the planning cost function assumes.

PushT's block angle at K=32 conveys that a linear probe explains almost none of its variance, but a nonlinear probe recovers over half from the exact same embedding. Much of this information has not been destroyed by truncation, it has just become non-linearly arranged, and the planning cost function, which measures distances linearly, cannot use it. Some information is genuinely lost too: the nonlinear probe itself declines from K=1 to K=32, consistent with the assumption that truncation simply reduces capacity. But the linear probe's much sharper decline conveys that most of the information is lost, which does not appear to be true.

Cube's block yaw is a check against this pattern being an artifact of probing itself: both probes stay near zero at every K (full sweep in Appendix B). Yaw is unlikely to be visually resolvable from this observation type at all, so finding nothing there is expected. That block angle's score moves sharply with K while block yaw's does not shows the pattern is specific to what should be recoverable, not generic probe noise. Table 5 in B reports the full linear-probe sweep across all six granularities and every probed target, not just the two endpoints.

5.6 Robustness under capacity scaling

To test the robustness of different methods, we retrained all three methods at $D = 384$ and compared the results to the existing $D = 192$ numbers, on all four environments. Table 4 shows that both the environment and the method affect performance under this change. Of the two methods that partition embeddings into subspaces, our method is generally more robust to dimensionality change than Sub-JEPA: where Sub-JEPA suffers one severe collapse, our worst case is far milder. The claim this supports is narrow: among methods that must pick a subspace granularity, picking all of them jointly is safer than committing to one.

Two-Room benefits most from increasing the dimensionality, for all three methods. At 384d, our method has the highest success rate on three environments, PushT, Reacher, and Cube. On the fourth, Two-Room, LeWM nominally leads the other two. The scaling rule itself is one choice

Table 4: Performance under $2\times$ embedding-dimension scaling. Values are success rate (%), mean $\pm$ standard error over 6 seeds. Δ is 384d minus 192d, in percentage points.

Env.	Method	192d	384d	Δ
PushT	LeWM	84.0 ± 2.7	83.3 ± 2.2	-0.7pp
	Sub-JEPA-native (K doubled)	92.0 ± 2.1	69.7 ± 3.7	-22.3pp
	ElasticJEPA (ours)	91.3 ± 0.4	84.7 ± 2.9	-6.6pp
Two-Room	LeWM	84.3 ± 1.7	96.7 ± 0.4	$+12.4$pp
	Sub-JEPA-native	95.0 ± 1.1	96.3 ± 1.5	$+1.3$pp
	ElasticJEPA (ours)	92.0 ± 0.5	96.3 ± 1.1	$+4.3$pp
Reacher	LeWM	85.3 ± 2.2	83.0 ± 1.8	-2.3pp
	Sub-JEPA-native	85.0 ± 1.7	80.3 ± 3.2	-4.7pp
	ElasticJEPA (ours)	81.3 ± 2.2	84.3 ± 1.9	$+3.0$pp
Cube	LeWM	67.0 ± 2.4	72.7 ± 2.6	$+5.7$pp
	Sub-JEPA-native	76.7 ± 3.0	75.0 ± 1.7	-1.7pp
	ElasticJEPA (ours)	73.7 ± 2.3	76.3 ± 2.9	$+2.6$pp

among others. We doubled the number of subspaces and held each one's size fixed; holding the count fixed and doubling each subspace's size instead remains untested. Sub-JEPA's own paper treats its subspace count as a sensitive, hand-tuned hyperparameter, so PushT's collapse may be about changing that count specifically, not about capacity scaling in general.

6 Discussion and Limitations

Scope of the ablations. The factorial and weighting ablations cover PushT and Two-Room only; extending them to Reacher and Cube was outside the time available here. That gap matters more than it might otherwise: capacity scaling started with a clean two-environment pattern too, and it did not survive extension to all four. A third or fourth environment could easily complicate the factorial and weighting results the same way, not just confirm them. The truncated baselines in Section 5.3 are also averaged over only two seeds, and we plan to extend them to the full six-seed protocol.

Training cost. ElasticJEPA adds no trainable parameters, since the basis Q is a fixed buffer, so all three methods have the same model size. The extra loss terms do make training slower, by up to about a third in our runs, which is much less than their number would suggest because the encoder and predictor dominate the per-step cost.

Why truncation still matters without a speedup. Wall-clock planning cost does not change with K in this implementation as the predictor always runs at full size. So the value of truncation is not a faster planner but a smaller representation: cheaper to store, cheaper to send to a remote planner, and cheaper for any downstream module, a probe or a retrieval index, that does scale with embedding width even when the predictor does not. A genuine speedup would need the predictor itself to be prefix-closed, which we do not attempt here.

7 Conclusion

In this paper we proposed **ElasticJEPA**, a size-flexible end-to-end JEPA world model that learn nested representations using nested subspace regularization and prediction loss. Nesting the regularizer alone does not order the representation, because SIGReg treats all subspaces alike; the ordering comes from prediction losses applied to the same prefixes used at inference. This makes it useful even with post-hoc truncation. We analytically and empirically show that prediction supervision at each nesting depth induces coordinate privilege needed for useful truncation. A single trained model stays close to its full-width performance under substantial truncation, works at widths it was never trained on, and degrades less than LeWM or Sub-JEPA truncated to the same size. The smallest prefixes still lose performance on PushT and Cube, and our probes suggest that part of this loss comes from the Euclidean planning cost rather than from missing information. A planning cost suited to small prefixes and a prefix-closed predictor, which would also make planning faster, are the natural next steps.

AI USE STATEMENT

As part of the disclosure of AI use in this work, we used AI assistance for our work in these ways: (1) literature search to identify relevant literature, (2) taking assistance for refining the paper text of this research paper.

REPRODUCIBILITY STATEMENT

Currently we have not released the source code of our research work publicly.

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 (CVPR)*, pp. 15619–15629, June 2023.

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

Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics, 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, Mido Assran, and Nicolas Ballas. V-JEPA: Latent video prediction for visual representation learning, 2024. URL https://openreview.net/forum?id=WFYbBOEOtv.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HylxE1HKwS.

David Ha and Jürgen Schmidhuber. World models. 2018. doi: 10.5281/ZENODO.1207631. URL https://zenodo.org/record/1207631.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels, 2019. URL https://arxiv.org/abs/1811.04551.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination, 2020. URL https://arxiv.org/abs/1912.01603.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Oxh5CstDJU.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30233–30249. Curran Associates, Inc., 2022. doi: 10.52202/068431-2192. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c32319f4868da7613d78af9993100e42-Paper-Conference.pdf.

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

Oren Rippel, Michael Gelbart, and Ryan Adams. Learning ordered representations with nested dropout. In Eric P. Xing and Tony Jebara (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1746–1754, Beijing, China, 22–24 Jun 2014. PMLR. URL https://proceedings.mlr.press/v32/rippel14.html.

J. Schmidhuber. An on-line algorithm for dynamic reinforcement learning and planning in reactive environments. In *1990 IJCNN International Joint Conference on Neural Networks*, pp. 253–258 vol.2, 1990. doi: 10.1109/IJCNN.1990.137723.

Vlad 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, 2025. URL https://arxiv.org/abs/2502.14819.

Jiahui Yu and Thomas S. Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction, 2021. URL https://arxiv.org/abs/2103.03230.

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, 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 *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=D5RNACOZEI.

A Analysis of nested prediction

A.1 Proof of the ordered-supervision proposition

The rows of the fixed orthogonal matrix Q are divided into $K_{\max}$ adjacent blocks of size

$$d_{\min} = \frac{D}{K_{\max}}.$$

For a granularity K, the retained projection $P_K^{(1)}$ consists of the first D/K rows of Q, equivalently the first $K_{\max}/K$ minimal blocks. Therefore,

$$\left\| P_K^{(1)} \left(\hat{z}_{t+1} - z_{t+1} \right) \right\|_2^2 = \sum_{b=1}^{K_{\max}/K} \|e_b\|_2^2.$$

Substituting into the nested prediction objective gives

$$\mathcal{L}_{\text{trunc-pred}} = \frac{1}{|\mathcal{K}|} \sum_{K \in \mathcal{K}} w_K \sum_{b=1}^{K_{\max}/K} \|e_b\|_2^2 \tag{4}$$

$$= \sum_{b=1}^{K_{\max}} \left(\frac{1}{|\mathcal{K}|} \sum_{\substack{K \in \mathcal{K} \\ b \le K_{\max}/K}} w_K \right) \|e_b\|_2^2 \tag{5}$$

$$= \sum_{b=1}^{K_{\max}} \alpha_b \|e_b\|_2^2. \tag{6}$$

Now consider two blocks $b < b'$. Every retained prefix containing b' also contains b, so the set of truncation levels contributing to $\alpha_{b'}$ is a subset of those contributing to α_b. Since all w_K are positive, $\alpha_b \geq \alpha_{b'}$. Finally, because $1 \in \mathcal{K}$, the full-width term contains every block, hence $\alpha_b > 0$ for all b. □

A.2 Exposure profile for the trained granularities

For $\mathcal{K} = \{1, 2, 4, 8, 16, 32\}$, $K_{\max} = 32$ and uniform weighting $w_K \equiv 1$, the 32 minimal blocks receive, up to the common factor $1/|\mathcal{K}| = 1/6$, the relative exposure profile

$$6,\ 5,\ \underbrace{4, 4}_{\text{2 blocks}},\ \underbrace{3, \ldots, 3}_{\text{4 blocks}},\ \underbrace{2, \ldots, 2}_{\text{8 blocks}},\ \underbrace{1, \ldots, 1}_{\text{16 blocks}}.$$

Hence the first block participates in all six prediction losses, whereas each of the final sixteen blocks participates only in the full-width ($K = 1$) loss. This is the optimization-level counterpart of the coordinate-privilege shading in Fig. 1.

A.3 Interpretation of non-uniform level weighting

Changing the level weights w_K changes the induced coordinate profile $\{\alpha_b\}$. This is different from a mere rescaling of the overall objective. More specifically, increasing the relative weight of larger K places more emphasis on the blocks that survive aggressive truncation. Therefore, the $\sqrt{K}$ weighting ablation can equivalently be interpreted as testing a steeper coordinate-privilege schedule. However, we note that this does not by itself predict whether the modified schedule improves planning performance, which is generally environment-dependent.

A.4 Intermediate prefixes

For any fixed prediction-error vector e and prefix widths $m_1 \leq m \leq m_2$,

$$\|e_{1:m_1}\|_2^2 \leq \|e_{1:m}\|_2^2 \leq \|e_{1:m_2}\|_2^2.$$

Thus the projected squared prediction error is well-defined and monotone in retained width even at intermediate prefixes not explicitly used during training. We caution that this should not be interpreted as a guarantee of monotone or interpolating planning success, which depends on the learned representation, dynamics model, and planning objective.

B Linear Probe Sweep across all granularities

Table 5 depicts full linear-probe sweep across all six granularities and every probed target, not just the two endpoints. Every target declines gradually as K grows, except three: Two-Room's target position, Reacher's target position, and Cube's block yaw all stay near zero throughout. These are deliberate negative controls. Two-Room's and Reacher's targets are randomized per episode, so nothing in the embedding should predict them; block yaw is unlikely to be visible in this observation type at all. Everything else declines with K; these three do not. That contrast confirms block angle's earlier drop is a real, specific pattern, not generic noise.

C Non-uniform level weighting

We ran non-uniform weighting ablations on PushT and TwoRooms and found different informative answers. Giving the small representations more training weight ($w_K \propto \sqrt{K}$) made things worse on PushT, at every size, including the full one, which had nothing to gain from the change in the first place. That fits what probing already suggested: PushT's problem is not a lack of training pressure, it is that the information is arranged in a way the linear readout cannot use. Pushing harder does not fix that.

The same change helped in Two-Room, and helped most at the smallest sizes. An explanation for that could be that Two-Room's gap is a genuine shortage of capacity, not a readout problem, so giving the small sizes more attention during training helps. So whether weighting helps or not depends on the problem at hand, and is not a by default recommended mechanism. We did not test it on Reacher or Cube.

Table 5: Linear-probe held-out R^2 across ElasticJEPA representation sizes, all probed targets.

Environment	Target	$K=1$	$K=2$	$K=4$	$K=8$	$K=16$	$K=32$
PushT	agent position	0.947	0.903	0.801	0.681	0.560	0.475
PushT	block position	0.981	0.963	0.861	0.691	0.617	0.423
PushT	**block angle**	0.763	0.645	0.540	0.249	0.144	**0.056**
Two-Room	agent position	0.997	0.990	0.839	0.632	0.549	0.436
Two-Room	target position	0.042	0.064	0.067	0.056	0.058	0.017
Reacher	finger position	0.999	0.996	0.972	0.719	0.410	0.282
Reacher	target position	−0.071	−0.031	−0.011	−0.008	−0.001	0.000
Reacher	joint angles (qpos)	0.450	0.459	0.435	0.354	0.321	0.267
Cube	block position	0.994	0.984	0.890	0.771	0.680	0.489
Cube	**block yaw**	−0.046	−0.028	−0.020	−0.009	−0.004	**−0.002**
Cube	effector position	0.977	0.954	0.859	0.722	0.641	0.487

Table 6: Effect of $\sqrt{K}$ weighting vs. uniform weighting on truncated-inference success rate. Values are the change in percentage points, $\sqrt{K}$ minus uniform.

	K=1	K=2	K=4	K=8	K=16	K=32
PushT	−2.3	−3.3	−2.0	−5.3	−2.0	−5.0
Two-Room	+1.3	+1.3	+3.7	+1.0	+7.7	+5.4

D SINGLE-SEED TO MULTI-SEED BASELINE CORRECTION

Early in this project, LeWM and Sub-JEPA baselines were evaluated with a single seed (`seed=42`). This produced real, sometimes large discrepancies against each paper's own reported numbers – e.g. a single-seed PushT LeWM read 96.0% against the paper's reported 84.67 ± 6.53. We traced this to Sub-JEPA's own README documenting its reported numbers as an average over the same 6 seeds used throughout this paper, which our single-seed runs did not match by construction. Re-running every baseline on those 6 seeds (Table 2) brought every mean within approximately one standard error of the corresponding paper's reported number – Two-Room LeWM matched to two decimal places (84.3 vs. 84.33).

Table 7: ElasticJEPA's best operating point (any K) against LeWM and Sub-JEPA's own *published* numbers. "At size" is relative to the full embedding. The LeWM/Sub-JEPA columns reproduce the values reported in their respective papers; per the reconstruction below, these are very likely standard *deviation* over 6 seeds, not standard error. The ElasticJEPA column is our own standard *error* over 6 seeds – the two statistics are not directly comparable without accounting for this (roughly a $\sqrt{6} \approx 2.45\times$ factor).

Env.	LeWM (paper)	Sub-JEPA (paper)	ElasticJEPA (ours)	At size
PushT	84.67 ± 6.53	89.00 ± 5.33	$\mathbf{91.3 \pm 0.4}$ (K=1)	full
Two-Room	84.33 ± 4.23	95.00 ± 2.76	92.7 ± 0.8 (K=8, 24d)	8× smaller
Reacher	82.67 ± 4.42	84.00 ± 4.00	$\mathbf{88.7 \pm 1.0}$ (K=16, 12d)	16× smaller
Cube	67.33 ± 5.01	76.33 ± 5.99	75.3 ± 1.7 (K=2, 96d)	2× smaller

This is also the source of the apparent variance mismatch visible in Table 7: dividing each paper's reported "±" by $\sqrt{6}$ (converting an assumed standard deviation to an implied standard error, for a fair comparison to our own numbers, which are standard error throughout) brings 7 of 8 baseline rows within about 1 point of the standard error we compute directly from our own raw 6 seeds for the equivalent reproduction (Table 8 converts this in-table) – consistent with the published statistic being standard deviation rather than standard error, though we have not confirmed this from the original paper's stated methodology and flag it as an assumption, not a verified fact.

Table 8: Table 7 converted to consistent units: each paper's reported "±" divided by $\sqrt{6}$, i.e. the implied standard error *if* the original statistic is standard deviation over 6 seeds as we believe (unconfirmed – see discussion above and below). All three columns are now standard error and directly comparable; means are unchanged from Table 7.

Env.	**LeWM (implied SE)**	**Sub-JEPA (implied SE)**	**ElasticJEPA (SE)**
PushT	84.67 ± 2.67	89.00 ± 2.18	91.3 ± 0.4
Two-Room	84.33 ± 1.73	95.00 ± 1.13	92.7 ± 0.8
Reacher	82.67 ± 1.80	84.00 ± 1.63	88.7 ± 1.0
Cube	67.33 ± 2.05	76.33 ± 2.45	75.3 ± 1.7