LeWorldModel: Stable End-to-End Joint-Embedding Predictive Architecture from Pixels

Lucas Maes[*1] **Quentin Le Lidec**[*2] **Damien Scieur**[1,3] **Yann LeCun**[2] **Randall Balestriero**[4]

[1]Mila & Université de Montréal [2]New York University [3]Samsung SAIL [4]Brown University

Website Code

Abstract

Joint Embedding Predictive Architectures (JEPAs) offer a compelling framework for learning world models in compact latent spaces, yet existing methods remain fragile, relying on complex multi-term losses, exponential moving averages, pre-trained encoders, or auxiliary supervision to avoid representation collapse. In this work, we introduce LeWorldModel (LeWM), a JEPA that trains stably end-to-end from raw pixels using only two loss terms: a next-embedding prediction loss and a regularizer enforcing Gaussian-distributed latent embeddings. This reduces tunable loss hyperparameters from six to one compared to the only existing end-to-end alternative. With 15M parameters trainable on a single GPU in a few hours, LeWM plans up to $48\times$ faster than foundation-model-based world models while remaining competitive across diverse 2D and 3D control tasks. Beyond control, we show that LeWM's latent space encodes meaningful physical structure through probing of physical quantities. Surprise evaluation confirms that the model reliably detects physically implausible events. Code available here.

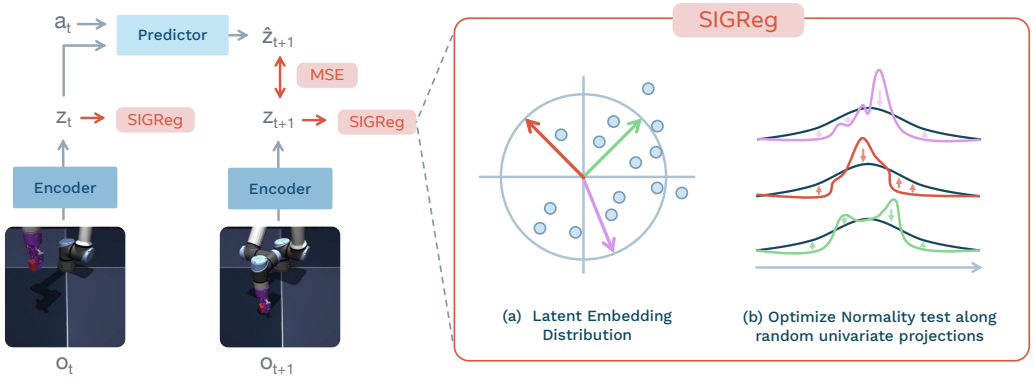

Figure 1: **LeWorldModel Training Pipeline.** Given frame observations $\boldsymbol{o}_{1:T}$ and actions $\boldsymbol{a}_{1:T}$, the encoder maps frames into low-dimensional latent representations $\boldsymbol{z}_{1:T}$. The predictor models the environment dynamics by autoregressively predicting the next latent state $\boldsymbol{z}_{t+1}$ from the current latent state $\boldsymbol{z}_t$ and action $\boldsymbol{a}_t$. The encoder and predictor are jointly optimized using a mean-squared error (MSE) prediction loss. LeWM does not rely on any training heuristics, such as stop-gradient, exponential moving averages, or pre-trained representations. To prevent trivial collapse, the SIGReg regularization term enforces Gaussian-distributed latent embeddings, promoting feature diversity. For tractability, latent embeddings are projected onto multiple random directions, and a normality test is applied to each one-dimensional projection. Aggregating these statistics encourages the full embedding distribution to match an isotropic Gaussian.

* Equal contribution. Correspondence to `lucas.maes@mila.quebec`

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

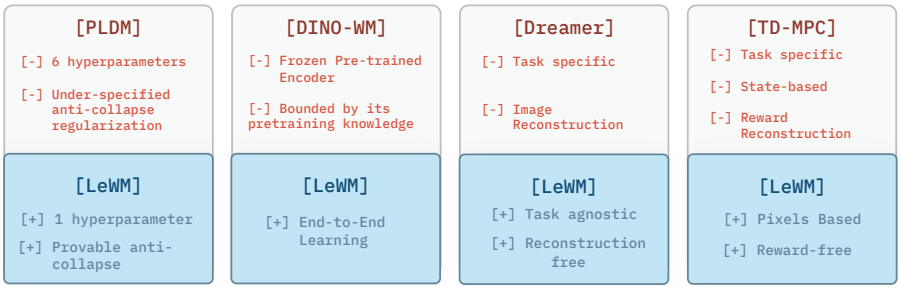

Figure 2: **Characteristics of latent world model approaches.** Methods are grouped by training paradigm. *End-to-end* methods (PLDM) learn the encoder and predictor jointly from pixels without pre-trained representations or heuristics like stop-gradient or EMAs, but require many hyperparameters and lack collapse guarantees. *Foundation-based* methods (DINO-WM) avoid collapse by freezing a pre-trained vision encoder, forgoing end-to-end learning. *Task-specific* methods (Dreamer, TD-MPC) require reward signals or privileged state access. LeWM combines the strengths of all three: end-to-end, task-agnostic, pixel-based, reconstruction- and reward-free, with a single hyperparameter with provable anti-collapse guarantees.

1 Introduction

A central goal of artificial intelligence is to develop agents that acquire skills across diverse tasks and environments using a single, unified learning paradigm—one that operates directly from sensory inputs of its surroundings–without hand-engineered state representations or domain-specific calibration. Vision is ideally suited for this aim: cameras are inexpensive and scalable, and learning from pixels enables fully end-to-end training from raw sensory input to action [1]. World Models (WMs) are a powerful family of methods [2] that learn to predict the consequences of actions in the environment. When successful, WMs allows agents to plan and to improve themselves solely form their model of the world, i.e., in imagination space. This is particularly valuable in the offline setting, where agents must learn from fixed datasets without environment interaction—leveraging the model to generate synthetic experience and evaluate counterfactual action sequences [3, 4].

A recent popular approach for learning world models is the Joint Embedding Predictive Architecture (JEPA) [5]. Instead of attempting to model every aspect of the environment, JEPA focuses on capturing the most relevant features needed to predict future states. Concretely, JEPA learns to encode observations into a compact, low-dimensional latent space and models temporal dynamics by predicting the latent representation of future observations.

However, despite their conceptual simplicity, existing JEPA methods are highly prone to collapse. In this failure mode, the model maps all inputs to nearly identical representations to trivially satisfy the temporal prediction objective leading to unusable representations. Preventing collapse is therefore one of the central challenges in training JEPA models. Many influential works have proposed methods to address this issue. Yet, these approaches typically rely on heuristic regularization, multi-objective loss functions, external sources of information, or architectural simplifications such as pre-trained encoders. In practice, these strategies often introduce additional instability or significantly increase training complexity (see App. C).

To overcome these limitations, we propose LeWorldModel (LeWM), a simple and principled recipe for learning a stable JEPA end-to-end from raw pixels without heuristics (cf. Fig 1). We evaluate LeWM across a diverse set of manipulation, navigation, and locomotion tasks in both 2D and 3D environments. In addition, we probe its intuitive physical understanding through targeted probing and surprise-quantification evaluations in latent space. Overall, our key findings and contributions are:

- We propose an end-to-end JEPA method for learning a latent world model from raw pixels on a single GPU. The method relies on a simple and stable two-term objective that remains robust across architectures and hyperparameter choices, while enabling efficient logarithmic-time hyperparameter search.
- Our experiment demonstrates that LeWM achieves competitive control performance across diverse 2D and 3D tasks with only a compact 15M-parameter model, surpassing existing end-to-end JEPA-based approaches while remaining competitive with foundation-model-based world models at substantially lower cost, enabling planning up to $48\times$ faster.

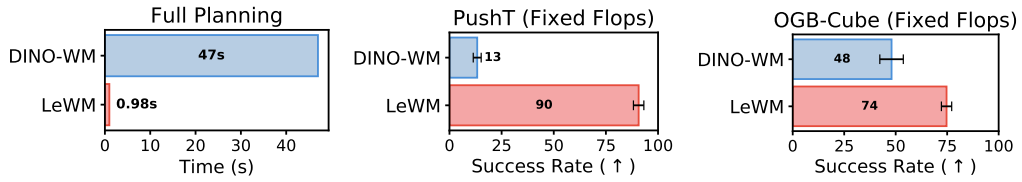

Figure 3: **Planning time and performance under fixed compute. Left:** Planning time comparison averaged over 50 runs. Encoding observations with $\sim 200\times$ fewer tokens than DINO-WM allows LeWM to achieve planning speeds comparable to PLDM while being up to $\sim 50\times$ faster than DINO-WM. **Center–Right:** Planning performance under the same computational budget (fixed FLOPs). LeWM significantly outperforms DINO-WM on Push-T (center) and OGBench-Cube (right). See App. D for planning setup details.

- We evaluate physical understanding in the latent space through probing of physical quantities and a violation-of-expectation test for detecting unphysical trajectories.

2 Related Work

World Models aim to learn predictive models of environment dynamics from data, enabling agents to reason about future states in imagination. A prominent class of WMs consists of *generative* approaches that explicitly model environment dynamics in pixel space. These action-conditioned generative models act as learned simulators by producing future observations conditioned on past states and actions. Generative world models have been successfully applied to simulate existing game-like environments. For example, IRIS [3], DIAMOND [6], Δ-IRIS [7], OASIS [8], and DreamerV4 [4] model environments such as Minecraft, Counter-Strike, and Crafter, improving policy sample efficiency in reinforcement learning. Other methods generate entirely new interactive simulators, e.g., Genie [9] and HunyuanWorld [10], while learned simulators have also been applied to robot policy evaluation [11]. Importantly, many generative WMs assume access to datasets containing reward signals, enabling joint modeling of dynamics and value-relevant information for downstream reinforcement learning. In contrast, we focus on the reward-free setting, corresponding to the setup considered in the JEPA line of work, which aims at learning generic, task-agnostic world models from observational data without relying on reward supervision.

JEPA is a framework for learning world models that predict the dynamic evolution of a system in a compact, low-dimensional latent space. Since their introduction by LeCun [5], JEPA methods have evolved considerably, differing mainly in their target tasks and in the strategies used to learn non-collapsing representations. One prominent line of work applies JEPA to self-supervised representation learning by predicting the latent embeddings of masked input patches. Examples include I-JEPA [12] for images, V-JEPA [13, 14] for videos, and Echo-JEPA and Brain-JEPA [15, 16] for medical data. These approaches typically employ an exponential moving average (EMA) of the target encoder together with stop-gradient (SG) updates to stabilize training and prevent representation collapse. However, the theoretical understanding of EMA and SG remains limited, as they do not in general correspond to the minimization of a well-defined objective [17]. A second line of work uses the JEPA recipe for action-conditioned latent world modeling. Some approaches rely on pretrained encoders to obtain representations [14, 18–20]. This avoids collapse but limits the expressivity of representation to the pretrained encoder used. In contrast, PLDM [21, 22] learns representations end-to-end using VICReg [23] with additional regularization terms, at the cost of known training instabilities and scalability limitations [24]. Several works further improve stability by incorporating auxiliary signals or architectural components, such as proprioceptive inputs or action decoders [18, 19]. In this work, we propose a stable method for training end-to-end JEPAs directly from raw pixels using a simple two-term loss: a predictive objective on future embeddings and a regularization objective that enforces Gaussian-distributed embeddings [25].

Planning with Latent Dynamics. World Models [26] pioneered learning policies directly from compact latent representations of high-dimensional observations. Some works leverage learned latent dynamics models to train policies using reinforcement learning [27–29, 4]. In these approaches, the generative world model acts as a simulator in which trajectories are rolled out in imagination, allowing policy optimization to occur largely in imagination in latent space. Once training is complete, the policy is executed directly, and the world model is no longer required at test time. More recent works

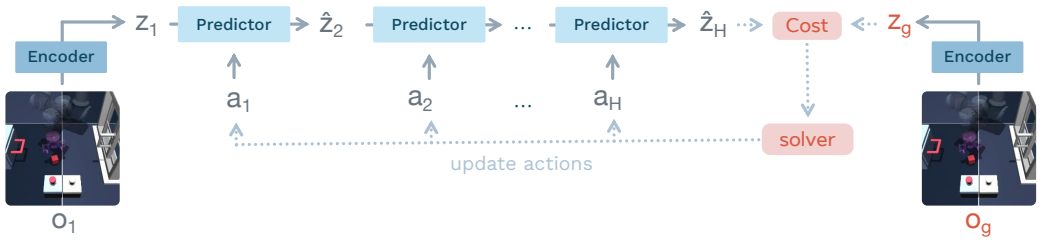

Figure 4: **Latent Planning with LeWorldModel .** Given an initial observation $\boldsymbol{o}_1$ and a goal $\boldsymbol{o}_g$, the world model learned in Fig. 2 performs planning in the LeWM latent space. The initial state embedding $\boldsymbol{z}_1$ and the goal embedding $\boldsymbol{z}_g$ are obtained from the encoder. The predictor then rolls out future latent states up to a horizon H. A latent cost between the final predicted state and the goal embedding guides a solver to optimize the action sequence. This prediction–optimization loop is repeated until convergence to a good plan candidate.

instead perform planning directly in the latent space at test time using Model Predictive Control (MPC) [30–33, 18, 22].

3 Method: LeWorldModel

In this section, we introduce LeWorldModel (LeWM). We first describe the streamlined training procedure used to learn the latent world model from offline data, including the dataset, model architecture, and training objective. We then explain how the learned model can be leveraged for decision making through latent planning using model predictive control (MPC).

3.1 Learning the Latent World Model

Offline Dataset. We consider a fully offline and reward-free setting. LeWorldModel is trained solely from unannotated trajectories of observations and actions, without access to reward signals or task specifications. This setup aligns with the JEPA line of work [18, 14], which aims to learn generic, task-agnostic world models from observational data. Our objective is not to optimize behavior for a specific task, but to learn representations that capture environment dynamics and can later be controlled or adapted to a diverse set of tasks.

The training data consists of trajectories of length T composed of raw pixel observations $\boldsymbol{o}_{1:T}$ and associated actions $\boldsymbol{a}_{1:T}$. Trajectories are collected offline from behavior policies with no optimality requirements; they may be pseudo-expert or exploratory, as long as they sufficiently cover the environment dynamics. Additional implementation details (batch size, resolution, and sub-trajectory construction) are provided in App. D.

Model Architecture. LeWM is built upon two components: an encoder and a predictor. The encoder maps a given frame observation $\boldsymbol{o}_t$ into a compact, low-dimensional latent representation $\boldsymbol{z}_t$. The predictor models the environment dynamics in latent space by predicting the embedding of the next frame observation $\hat{\boldsymbol{z}}_{t+1}$ given the latent embedding $\boldsymbol{z}_t$ and an action $\boldsymbol{a}_t$.

$$\begin{aligned} \text{Encoder:} \quad & \boldsymbol{z}_t = \text{enc}_\theta(\boldsymbol{o}_t) \\ \text{Predictor:} \quad & \hat{\boldsymbol{z}}_{t+1} = \text{pred}_\phi(\boldsymbol{z}_t, \boldsymbol{a}_t) \end{aligned} \tag{LeWM}$$

The encoder is implemented as a Vision Transformer (ViT) [34]. Unless otherwise specified, we use the tiny configuration ($\sim$5M parameters) with a patch size of 14, 12 layers, 3 attention heads, and hidden dimensions of 192. The observation embedding $\boldsymbol{z}_t$ is constructed from the `[CLS]` token embedding of the last layer, followed by a projection step. The projection step maps the `[CLS]` token embedding into a new representation space using a 1-layer MLP with Batch Normalization [35]. This step is necessary because the final ViT layer applies a Layer Normalization [36], which prevents our anti-collapse objective from being optimized effectively.

The predictor is a transformer with 6 layers, 16 attention heads, and 10% dropout ($\sim$10M parameters). Actions are incorporated into the predictor through Adaptive Layer Normalization (AdaLN) [37] applied at each layer. The AdaLN parameters are initialized to zero to stabilize training and ensure that action conditioning impacts the predictor training progressively. The predictor takes as input a

history of N frame representations and predicts the next frame representation auto-regressively with temporal causal masking to avoid looking at future embeddings. The predictor is also followed by a projector network with the same implementation as the one used for the encoder. All components of our world model are learned jointly using the loss described in the following paragraph.

Training Objective. Our objective is to learn latent representations useful for predicting the future, i.e., modeling the environment dynamics. LeWorldModel training objective is the sum of two terms: a prediction loss and a regularization loss. The prediction loss $\mathcal{L}_{\text{pred}}$ (teacher-forcing) computes the error between the predicted embedding of consecutive time-steps:

$$\mathcal{L}_{\text{pred}} \triangleq \|\hat{\boldsymbol{z}}_{t+1} - \boldsymbol{z}_{t+1}\|_2^2, \qquad \hat{\boldsymbol{z}}_{t+1} = \text{pred}_\phi(\boldsymbol{z}_t, \boldsymbol{a}_t). \tag{1}$$

Through the prediction loss, the encoder is incentivized to learn a predictable representation for the predictor.

However, if alone, the loss in Eq. 1 leads to representation collapse, yielding a trivial solution in which the encoder maps all inputs to a constant representation. To prevent this behavior, we introduce an anti-collapse regularization term that promotes feature diversity in the embedding space. Specifically, we adopt the Sketched-Isotropic-Gaussian Regularizer (SIGReg) [25] due to its simplicity, scalability, and stability. SIGReg encourages the latent embeddings to match an isotropic Gaussian target distribution.

Let $\boldsymbol{Z} \in \mathbb{R}^{N \times B \times d}$ denote the tensor of latent embeddings collected over the history length N, the batch size B, and where d denotes the embedding dimension. Assessing normality directly in high-dimensional spaces is challenging, as most classical normality tests are designed for univariate data and do not scale reliably with dimensionality. SIGReg circumvents this limitation by projecting embeddings onto M random unit-norm directions $\boldsymbol{u}^{(m)} \in \mathbb{S}^{d-1}$ and optimizing the univariate Epps–Pulley [38] test statistic $T(\cdot)$ along the resulting one-dimensional projections $\boldsymbol{h}^{(m)} = \boldsymbol{Z}\boldsymbol{u}^{(m)}$, as illustrated in Fig.1. By the Cramér–Wold theorem [39], matching all one-dimensional marginals is equivalent to matching the full joint distribution.

$$\text{SIGReg}(\boldsymbol{Z}) \triangleq \frac{1}{M} \sum_{m=1}^{M} T(\boldsymbol{h}^{(m)}). \tag{2}$$

Additional details on SIGReg and the definition of the Epps–Pulley statistical test are provided in appendix A.

The complete LeWM training objective is defined as:

$$\mathcal{L}_{\text{LeWM}} \triangleq \mathcal{L}_{\text{pred}} + \lambda\, \text{SIGReg}(\boldsymbol{Z}). \tag{3}$$

The method introduces only two training hyperparameters: the number of random projections M used in SIGReg and the regularization weight λ. Unless otherwise specified, we use $M = 1024$ projections and $\lambda = 0.1$. In practice, we observe that the number of projections has negligible impact on downstream performance (see Sec. 4 and App. G), making λ the only effective hyperparameter to tune. This greatly simplifies hyperparameter selection, as λ can be efficiently optimized using a simple bisection search with logarithmic complexity. We do not employ stop-gradient, exponential moving averages, or additional stabilization heuristics. Gradients are propagated through all components of the loss, and all parameters are optimized jointly in an end-to-end manner, resulting in a streamlined and easy-to-implement training procedure. The training logic is summarized in Alg. 3.

3.2 Latent Planning

At inference time, we perform trajectory optimization in our world model latent space, as illustrated in Fig.4. Given an initial observation $\boldsymbol{o}_1$, we initialize a candidate action sequence randomly and iteratively rollout predicted latent states up to a planning horizon H. The model predicts latent transitions according to

$$\hat{\boldsymbol{z}}_{t+1} = \text{pred}_\phi(\hat{\boldsymbol{z}}_t, \boldsymbol{a}_t), \quad \hat{\boldsymbol{z}}_1 = \text{enc}_\theta(\boldsymbol{o}_1),$$

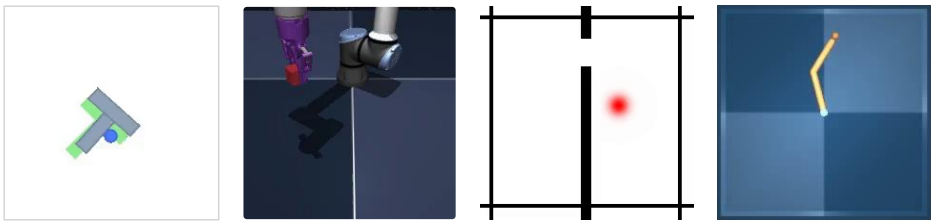

Figure 5: Environments used for evaluation. **Left:** Push-T, a 2D manipulation task where the agent must push a block toward a target configuration, commonly used as a robotics benchmark. **Center (1):** OGBench-Cube, a visually richer 3D manipulation environment where a robotic arm interacts with a cube to reach a target position. **Center (2):** Two-Room, a simple 2D navigation environment where an agent moves between rooms to reach target positions. **Right:** Reacher, a task where a 2-joint arm needs to reach a target configuration in a 2D plane. All environments have a continuous action space. More details on environment and datasets are available in appendix E.

Planning is performed by optimizing the action sequence to minimize a terminal latent goal-matching objective,

$$\mathcal{C}(\hat{\boldsymbol{z}}_H) = \|\hat{\boldsymbol{z}}_H - \boldsymbol{z}_g\|_2^2, \quad \boldsymbol{z}_g = \text{enc}_\theta(\boldsymbol{o}_g), \tag{4}$$

where $\hat{\boldsymbol{z}}_H$ is the predicted latent state at the end of the rollout and $\boldsymbol{z}_g$ is the latent embedding of the goal observation $\boldsymbol{o}_g$. The world model parameters remain fixed during planning. This procedure corresponds to a finite-horizon optimal control problem,

$$\boldsymbol{a}^*_{1:H} = \arg\min_{\boldsymbol{a}_{1:H}} \mathcal{C}(\hat{\boldsymbol{z}}_H), \tag{5}$$

which we solve using the Cross-Entropy Method (CEM) [40], a sampling method that iteratively selects the best plan and updates the parameters of the sampling distribution with the statistics of the best plans. The planning horizon H trades off long-term lookahead against increased computational cost and model bias. In particular, auto-regressive rollouts accumulate prediction errors as the horizon grows, which can deteriorate the quality of the optimized action sequence. To mitigate this effect, we adopt a Model Predictive Control (MPC) strategy: only the first K planned actions are executed before replanning from the updated observation. We provide more details on the planning strategy in appendix D.

4 Latent Planning Performance

4.1 Planning evaluation setup

Environments. We evaluate LeWM on a diverse set of tasks, including navigation, motion planning and manipulation, in both two- and three-dimensional environments, all illustrated in Fig. 5. We provide more details on dataset generation and environments in App. E.

Baselines. We compare the performance of LeWM against several baselines: DINO-WM [18] and PLDM [22], two state-of-the-art JEPA-based methods; a goal-conditioned behavioral cloning policy (GCBC); and two goal-conditioned offline reinforcement learning algorithms, GCIVL and GCIQL [41]. Among these baselines, PLDM is the closest to our setup, as it also learns a world model end-to-end directly from pixel observations. However, it relies on a seven-term training objective derived from the VICReg [23] criterion, which introduces training instability and increases the complexity of hyperparameter tuning. DINO-WM, in contrast, models dynamics using DINOv2 [42] as feature encoder to mitigate representation collapse, but its original formulation additionally incorporates other modalities, such as proprioceptive inputs; for a fair comparison, unless specified otherwise, we exclude proprioceptive information from DINO-WM. Additional implementation details for the baselines (App. C) and evaluation settings (App. F.1) are provided in the appendix. For each method, we keep the hyperparameters fixed across all environments.

4.2 Towards Efficient Planning with WMs

We report planning performance in Fig. 6. LeWM outperforms PLDM on the more challenging planning tasks, achieving an 18% higher success rate on PushT, while remaining competitive with

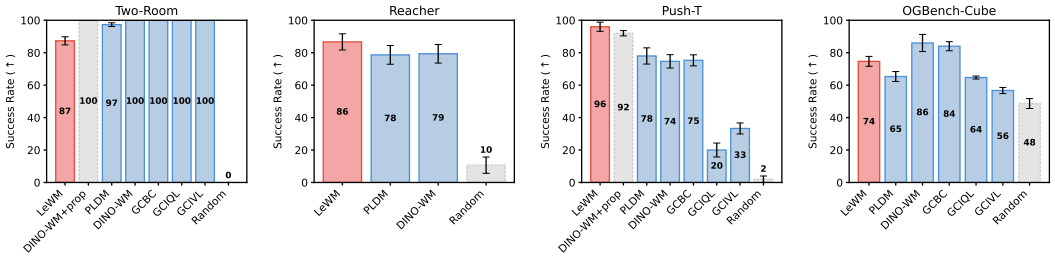

Figure 6: **Planning performance across environments.** Results are shown for *Two-Room* (left), *Reacher* (center 1), *PushT* (center-2) and *OGBench-Cube* (right). LeWM consistently outperforms PLDM and DINO-WM on Push-T and Reacher. On OGBench-Cube, DINO-WM slightly outperforms LeWM, possibly due to the higher visual complexity and the 3D nature of the environment, which makes encoder training more challenging. In the simpler Two-Room environment, PLDM and DINO-WM outperform LeWM, which may be explained by the SIGReg regularization encouraging a Gaussian distribution in a high-dimensional latent space, while the intrinsic dimensionality of the environment is much lower.

DINO-WM. Notably, on PushT, LeWM (pixels-only) surpasses DINO-WM even when DINO-WM has access to additional proprioceptive information, demonstrating LeWM's ability to capture underlying task-relevant quantities. Interestingly, LeWM performs worse on the simplest environment, Two-Room. A possible explanation is that the low diversity and low intrinsic dimensionality of this dataset make it difficult for the encoder to match the isotropic Gaussian prior enforced by SIGReg in a high-dimensional latent space, which may lead to a less structured latent representation. This highlights a potential limitation of the SIGReg regularization in very low-complexity environments.

Moreover, when comparing planning speedups (Fig. 3), LeWM achieves a $48\times$ faster planning time, with the full planning completing in under one second while preserving competitive performance across tasks. This planning time remains consistent across environments for a fixed planning setup, narrowing the gap toward real-time control.

4.3 Towards Stable Training of World Models

Ablations. We perform ablations on several design choices of LeWM. First, we analyze the sensitivity of SIGReg to its internal parameters, namely the number of random projections and the number of integration knots. The performance is largely unaffected by these quantities, indicating that they do not require careful tuning. As a result, the regularization weight λ remains the only effective hyperparameter. Since only a single hyperparameter needs to be tuned, grid search can be performed efficiently using a simple bisection strategy ($\mathcal{O}(\log n)$), whereas PLDM requires search in polynomial time ($\mathcal{O}(n^6)$). We also study the effect of the embedding dimensionality. While the representation dimension must be sufficiently large for the method to perform well, performance quickly saturates beyond a certain threshold, suggesting that the approach is robust to the precise choice of encoder capacity. Additionally, we examine the impact of the encoder architecture by replacing the default ViT encoder with a ResNet-18 backbone (Tab. 8). LeWM achieves competitive performance with both architectures, indicating that it is largely agnostic to the choice of vision encoder. Details on all ablations are available in App. G.

Training Curves. We report the training loss curves on PushT for LeWM in Fig. 18 and PLDM in Fig. 19. The two-term objective of LeWM exhibits smooth and monotonic convergence: the prediction loss decreases steadily while the SIGReg regularization term drops sharply in the early phase of training before plateauing, indicating that the latent distribution quickly approaches the isotropic Gaussian target. In contrast, PLDM's seven-term objective displays noisy and non-monotonic behavior across several of its loss components. These observations highlight a key advantage of LeWM: by reducing the training objective to only two well-behaved terms, the training becomes significantly more stable, removing the need to balance competing gradients from multiple regularizers.

5 Quantifying Physical Understanding in LeWM

In this section, we evaluate the quality of the dynamics captured by LeWM's latent space, either by learning to extract physical quantities from latent embeddings or by measuring the world model's ability to detect changes in physics.

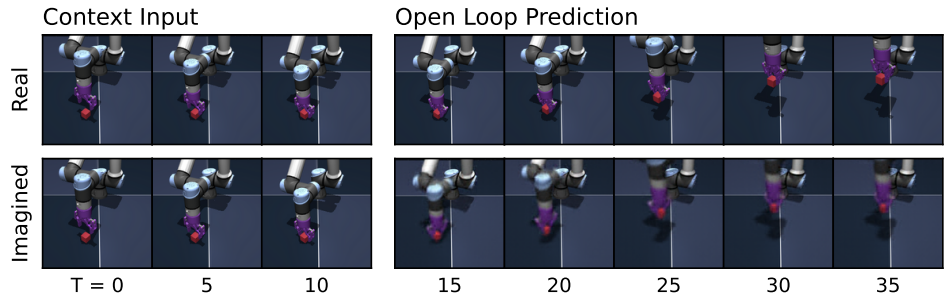

Figure 7: **Predictor rollout on OGBench-Cube**. We visualize decoded latent plans from LeWM given context and action sequences. Each rollout encodes three image observations as context, then autoregressively generates future latents conditioned on the actions in an open-loop fashion. Latents are decoded via a decoder trained *a posteriori*. The imagined rollout suggests that latent representations capture the global scene structure while finer details like end-effector angle are not fully preserved. Additional rollouts for Push-T and OGBench-Cube are provided in Fig. 9.

5.1 Physical Structure of the Latent Space

Probing physical quantities. As a first measure of physical understanding, we evaluate which physical quantities are recoverable from LeWM's latent representations. We train both linear and non-linear probes to predict physical quantities of interest from a given embedding. Results on the Push-T environment are reported in Tab. 1. Our method consistently outperforms PLDM while remaining competitive with representations produced by large pretrained models such as DINOv2. We provide probing results on other environments in App. F.2.

Table 1: **Physical latent probing results on Push-T.** LeWM consistently outperforms PLDM while remaining competitive with DINO-WM. The strong probing performance of DINO-WM on certain properties may stem from its foundation-model pretraining: the DINOv2 encoder is trained on two orders of magnitude more data ($\sim$124M images) spanning a far more diverse distribution, which likely allows it to capture some physical properties in its embeddings by default.

		Linear		MLP	
Property	**Model**	MSE ↓	r ↑	MSE ↓	r ↑
Agent Location	DINO-WM	1.888 ± 0.500	**0.977**	$\mathbf{0.003 \pm 0.022}$	**0.999**
	PLDM	0.090 ± 0.311	0.955	0.014 ± 0.119	0.993
	LeWM	$\mathbf{0.052 \pm 0.149}$	0.974	0.004 ± 0.056	0.998
Block Location	DINO-WM	$\mathbf{0.006 \pm 0.007}$	**0.997**	0.002 ± 0.003	**0.999**
	PLDM	0.122 ± 0.341	0.938	0.011 ± 0.066	0.994
	LeWM	0.029 ± 0.073	0.986	$\mathbf{0.001 \pm 0.006}$	**0.999**
Block Angle	DINO-WM	$\mathbf{0.050 \pm 0.101}$	**0.979**	$\mathbf{0.009 \pm 0.052}$	**0.995**
	PLDM	0.446 ± 0.625	0.745	0.056 ± 0.184	0.972
	LeWM	0.187 ± 0.359	0.902	0.021 ± 0.139	0.990

Decoding Latent Space. To further assess the information captured in the latent representation, we report in Fig. 10 images produced by a decoder trained to reconstruct pixel observations from a single latent embedding (192 dim) during training. Although reconstruction is never used during training, the decoder is able to recover the visual scene from the learned representation, confirming that the low-dimensional and compact latent space retains sufficient information about the underlying physical state. Details on the decoder architecture are provided in App. D.

Visualizing Latent Space. We further visualize the structure of the latent space using t-SNE. Fig. 13 provides a qualitative visualization of the latent space in the *PushT* environment. The visualization suggests that the learned representation captures the spatial structure of the environment, preserving neighborhood relationships and relative positions in the latent space.

Temporal Latent Path Straightening. Inspired by the temporal straightening hypothesis from neuroscience [43] and recent work from [44, 45], we measure the cosine similarity between consecutive latent velocity vectors throughout training (Eq. 9). We find that LeWM's latent trajectories become

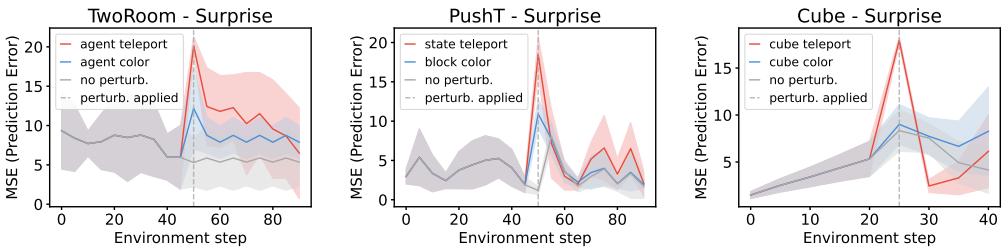

Figure 8: **Violation-of-expectation evaluation across three environments.** Each plot shows LeWM surprise along three trajectories: an unperturbed reference, a visually perturbed trajectory with abrupt object color change, and a physically perturbed trajectory where objects teleport to random positions. Teleportation violates physical continuity and produces a pronounced surprise spike, while the unperturbed trajectory stays at a low baseline. The increase is significant for teleportation across all environments (paired t-test, $p < 0.01$) but weaker and non-significant for color changes, indicating greater sensitivity to physical than visual perturbations. Environments, left to right: *TwoRoom*, *PushT*, *OGBench Cube*.

increasingly straight on PushT over training as a purely emergent phenomenon, without any explicit regularization encouraging this behavior, cf. Fig. 17. Remarkably, LeWM achieves higher temporal straightness than PLDM, despite PLDM employing a dedicated temporal smoothness regularization term. We detail our findings in App. H.

5.2 Violation-of-expectation Framework

Another approach to quantifying physical understanding is the ability to detect violations of the learned world model. Inspired by the violation-of-expectation (VoE) paradigm used in developmental psychology and recently adopted in machine learning [46–48], this framework evaluates whether a model assigns higher surprise to events that contradict learned physical regularities.

Following prior work, we quantify surprise by measuring the discrepancy between the model's predicted future observations and the actual observed future. We evaluate this framework across three environments: *TwoRoom*, *PushT*, and *OGBench Cube*. For each environment, we introduce two types of perturbations. The first is a visual perturbation, where the color of an object changes abruptly during the trajectory. The second is a physical perturbation, where one or more objects are teleported to a random location, violating the expected physical continuity of the scene. Fig. 8 shows that LeWM consistently assigns higher surprise to frames containing physical violations compared to their unperturbed counterparts. We provide more details on VoE in App. F.3.

6 Conclusion

We introduced LeWorldModel (LeWM), a stable end-to-end method for learning latent world models. LeWM is a Joint-Embedding Predictive Architecture in which an encoder maps image observations to a latent space and a predictor models temporal dynamics by forecasting future embeddings conditioned on actions. Across continuous control environments with raw pixel inputs, LeWM matches or exceeds prior end-to-end approaches in task performance while requiring no large-scale pre-training, a single anti-collapse coefficient instead of six, and under a second per plan. Training stability stems from explicitly encouraging latent embeddings toward an isotropic Gaussian distribution to prevent collapse, offering a scalable and principled alternative to existing work.

Limitations & Future Work. Several limitations point to future directions. Planning remains restricted to short horizons, motivating hierarchical world modeling for long-horizon reasoning. Our method also relies on offline datasets with sufficient coverage; in particular, low data diversity weakens SIGReg in simple, low-dimensional environments where matching a high-dimensional Gaussian prior is harder. Pre-training on large, diverse video datasets could provide stronger priors and reduce domain-specific data needs. Finally, dependence on action labels could be alleviated by inverse dynamics modeling.

References

[1] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[2] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3), 2018.

[3] Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vhFu1Acb0xb.

[4] Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models, 2025. URL https://arxiv.org/abs/2509.24527.

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

[6] Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=NadTwTODgC.

[7] Vincent Micheli, Eloi Alonso, and François Fleuret. Efficient world models with context-aware tokenization. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=BiWIERWBFX.

[8] Decart, Julian Quevedo, Quinn McIntyre, Spruce Campbell, Xinlei Chen, and Robert Wachen. Oasis: A universe in a transformer. 2024. URL https://oasis-model.github.io/.

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

[10] Team HunyuanWorld. Hunyuanworld 1.0: Generating immersive, explorable, and interactive 3d worlds from words or pixels. *arXiv preprint*, 2025.

[11] Julian Quevedo, Ansh Kumar Sharma, Yixiang Sun, Varad Suryavanshi, Percy Liang, and Sherry Yang. Worldgym: World model as an environment for policy evaluation, 2025. URL https://arxiv.org/abs/2506.00613.

[12] 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*, pages 15619–15629, 2023.

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

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

[15] Zijian Dong, Li Ruilin, Yilei Wu, Thuan Tinh Nguyen, Joanna Su Xian Chong, Fang Ji, Nathanael Ren Jie Tong, Christopher Li Hsian Chen, and Juan Helen Zhou. Brain-JEPA: Brain dynamics foundation model with gradient positioning and spatiotemporal masking. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=gtU2eLSAmO.

[16] Alif Munim, Adibvafa Fallahpour, Teodora Szasz, Ahmadreza Attarpour, River Jiang, Brana Sooriyakanthan, Maala Sooriyakanthan, Heather Whitney, Jeremy Slivnick, Barry Rubin, Wendy Tsang, and Bo Wang. Echojepa: A latent predictive foundation model for echocardiography, 2026. URL https://arxiv.org/abs/2602.02603.

[17] Jean Ponce, Basile Terver, Martial Hebert, and Michael Arbel. Dual perspectives on non-contrastive self-supervised learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=f5MC1G6XhB.

[18] 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 (ICML 2025)*, 2025.

[19] Raktim Gautam Goswami, Prashanth Krishnamurthy, Yann LeCun, and Farshad Khorrami. Osvi-wm: One-shot visual imitation for unseen tasks using world-model-guided trajectory generation, 2025. URL https://arxiv.org/abs/2505.20425.

[20] Heejeong Nam, Quentin Le Lidec, Lucas Maes, Yann LeCun, and Randall Balestriero. Causal-jepa: Learning world models through object-level latent interventions, 2026. URL https://arxiv.org/abs/2602.11389.

[21] Vlad Sobal, Jyothir S V, Siddhartha Jalagam, Nicolas Carion, Kyunghyun Cho, and Yann LeCun. Joint embedding predictive architectures focus on slow features, 2022. URL https://arxiv.org/abs/2211.10831.

[22] Vlad Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Stress-testing offline reward-free reinforcement learning: A case for planning with latent dynamics models. In *7th Robot Learning Workshop: Towards Robots with Human-Level Abilities*, 2025. URL https://openreview.net/forum?id=jON7H6A9UU.

[23] 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.

[24] Randall Balestriero and Yann LeCun. Contrastive and non-contrastive self-supervised learning recover global and local spectral embedding methods. *Advances in Neural Information Processing Systems*, 35:26671–26685, 2022.

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

[26] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

[27] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

[28] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[29] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[30] J Testud, J Richalet, A Rault, and J Papon. Model predictive heuristic control: Applications to industial processes. *Automatica*, 14(5):413–428, 1978.

[31] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning (ICML)*, 2022.

[32] 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.

[33] Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models, 2025. URL https://arxiv.org/abs/2412.03572.

[34] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[35] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. URL https://arxiv.org/abs/1502.03167.

[36] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[37] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[38] Thomas W Epps and Lawrence B Pulley. A test for normality based on the empirical characteristic function. *Biometrika*, 70(3):723–726, 1983.

[39] Harald Cramér and Herman Wold. Some theorems on distribution functions. *Journal of the London Mathematical Society*, 1(4):290–294, 1936.

[40] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2004.

[41] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[42] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=a68SUt6zFt. Featured Certification.

[43] Olivier J Hénaff, Robbe LT Goris, and Eero P Simoncelli. Perceptual straightening of natural videos. *Nature neuroscience*, 22(6):984–991, 2019.

[44] Christian Internò, Robert Geirhos, Markus Olhofer, Sunny Liu, Barbara Hammer, and David Klindt. AI-generated video detection via perceptual straightening. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=LsmUgStXby.

[45] Ying Wang, Oumayma Bounou, Gaoyue Zhou, Randall Balestriero, Tim GJ Rudner, Yann LeCun, and Mengye Ren. Temporal straightening for latent planning. *arXiv preprint arXiv:2603.12231*, 2026.

[46] Francesco Margoni, Luca Surian, and Renée Baillargeon. The violation-of-expectation paradigm: A conceptual overview. *Psychological Review*, 131(3):716, 2024.

[47] Quentin Garrido, Nicolas Ballas, Mahmoud Assran, Adrien Bardes, Laurent Najman, Michael Rabbat, Emmanuel Dupoux, and Yann LeCun. Intuitive physics understanding emerges from self-supervised pretraining on natural videos. *arXiv preprint arXiv:2502.11831*, 2025.

[48] Florian Bordes, Quentin Garrido, Justine T Kao, Adina Williams, Michael Rabbat, and Emmanuel Dupoux. Intphys 2: Benchmarking intuitive physics understanding in complex synthetic environments, 2025. URL https://arxiv.org/abs/2506.09849.

[49] Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=M992mjgKzI.

[50] Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*, 2019.

[51] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

A SIGReg

SIGReg proposes to match the distribution of embeddings towards the isotropic Gaussian target distribution. Achieving that match in high-dimension is gracefully done by combining two statistical components (i) Cramer-Wold theorem, and (ii) the univariate Epps-Pulley test-statistic. In short, SIGReg first produces M unit-norm directions $\boldsymbol{u}^{(m)}$ and projects the embeddings $\boldsymbol{Z}$ onto them as

$$\boldsymbol{h}^{(m)} \triangleq \boldsymbol{Z}\boldsymbol{u}^{(m)}, \boldsymbol{u}^{(m)} \in \mathbb{S}^{D-1}, \tag{6}$$

where the directions are sampled uniformly on the hypersphere. Then, SIGReg performs univariate distribution matching as

$$\mathrm{SIGReg}(\boldsymbol{Z}) \triangleq \frac{1}{M}\sum_{m=1}^{M} T^{(m)}, \tag{SIGReg}$$

with T the univariate Epps-Pulley test-statistic

$$T^{(m)} = \int_{-\infty}^{\infty} w(t)\left|\phi_N(t;\boldsymbol{h}^{(m)}) - \phi_0(t)\right|^2 dt, \tag{EP}$$

where the empirical characteristic function (ECF) is defined as $\phi_N(t;\boldsymbol{h}) = \frac{1}{N}\sum_{n=1}^{N} e^{it\boldsymbol{h}_n}$, w is a weighting function, e.g., $w(t) = e^{-\frac{t^2}{2\lambda^2}}$. Lastly, because the target is an isotropic Gaussian in $\mathbb{R}^D$, the univariate projection through $\boldsymbol{u}^{(m)}$ makes the univariate target distribution ϕ_0 the standard Gaussian $N(0,1)$. By Cramér–Wold, matching all 1D marginals implies matching the joint distribution, i.e., in the asymptotic limit over M we have the following weak convergence result

$$\mathrm{SIGReg}(\boldsymbol{Z}) \to 0 \iff \mathbb{P}_{\boldsymbol{Z}} \to N(0, \boldsymbol{I}). \tag{Cramer-Wold}$$

Practically, the integral in equation EP employs a quadrature scheme, e.g., trapezoid with T nodes uniformly distributed in $[0.2, 4]$.

B Cross-Entropy Method

The *Cross-Entropy Method* (CEM) [40] is a sampling-based (zero-order) optimization algorithm. Intuitively, CEM is an iterative sampling procedure that progressively refines a plan, defined as a sequence of actions, at each iteration.

At every iteration, the algorithm samples a pool of candidate plans from a distribution, typically a Gaussian (with initial parameters $\mu = \mathbf{0}$ and $\sigma = \mathbf{I}$). Next, each candidate plan is evaluated using the world model, and a cost is associated with it. The algorithm then selects the top k plans with the lowest cost, referred to as *elites*. These elites are used to compute statistics that update the parameters of the sampling distribution for the next iteration. Through this iterative process, the method explores the action space while gradually concentrating the sampling distribution around regions associated with lower costs. The final action plan is obtained from the mean of the sampling distribution at the last iteration.

However, in non-convex settings, there is no guarantee that the solution to which CEM converges is a global optimum. Furthermore, CEM suffers from the curse of dimensionality and becomes increasingly difficult to apply when the action space is large.

In our experiments, we use a CEM solver with 300 sampled action sequences per iteration and perform 30 optimization steps. At each step, the top 30 candidates are selected as elites to update the sampling distribution. We provide the algorithm pseudo-code in Alg. 2.

Algorithm 2 Cross-Entropy Method (CEM) for Action Sequence Optimization

Require: World model f, planning horizon H, number of samples N, number of elites K, number of iterations T

Initialize sampling distribution parameters $\mu_0 = \mathbf{0}$, $\Sigma_0 = I$
for $t = 1$ to T **do**
  Sample N candidate action sequences $\{a^{(i)}_{1:H}\}_{i=1}^{N} \sim \mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$
  for $i = 1$ to N **do**
    Roll out $a^{(i)}_{1:H}$ in the world model f
    Compute cost $J^{(i)}$
  end for
  Select the K sequences with lowest cost (elites)
  Update distribution parameters using elite set:
  $\mu_t \leftarrow \frac{1}{K} \sum_{i \in \mathcal{E}} a^{(i)}_{1:H}$
  $\Sigma_t \leftarrow \text{Var}_{i \in \mathcal{E}} \left(a^{(i)}_{1:H} \right)$
end for
return best action sequence found or first action of μ_T

C Baselines

C.1 DINO-WM

DINO world model (DINO-WM) focused on learning a predictor by leveraging DINOv2 frozen pre-trained representation to avoid collapse. Because not trained end-to-end, the loss simply is to minimize the predicted next-embedding with the ground trught next-state embedding produced by DINOv2.

$$\mathcal{L}_{\text{DINO-WM}} = \frac{1}{BT} \sum_{i}^{B} \sum_{t}^{T} \| \hat{\boldsymbol{z}}^{(i)}_{t+1} - \boldsymbol{z}^{(i)}_{t+1} \|_2^2 \tag{7}$$

We use the same setup as the original paper [18] (architecture, hyper-paremeters, etc..)

C.2 PLDM

PLDM [22] proposed a method for learning an end-to-end joint-embedding predictive architecture (JEPA). To avoid collapse, their approach takes inspiration from the variance-invariance-covariance regularization (VICReg, [23]) with extra terms to take into account the temporality of the next state prediction. The PLDM objective is the following:

$$\mathcal{L}_{\text{PLDM}} = \mathcal{L}_{\text{pred}} + \alpha \mathcal{L}_{\text{var}} + \beta \mathcal{L}_{\text{cov}} + \gamma \mathcal{L}_{\text{time-sim}} + \zeta \mathcal{L}_{\text{time-var}} + \nu \mathcal{L}_{\text{time-cov}} + \mu \mathcal{L}_{\text{IDM}} \tag{8}$$

where,

$$\mathcal{L}_{\text{pred}} = \frac{1}{BT} \sum_{i}^{B} \sum_{t}^{T} \| \hat{\boldsymbol{z}}^{(i)}_{t+1} - \boldsymbol{z}^{(i)}_{t+1} \|_2^2$$

$$\mathcal{L}_{\text{var}} = \frac{1}{TD} \sum_{t}^{T} \sum_{d}^{D} \max \left(0, 1 - \sqrt{\text{Var}(\boldsymbol{z}^{(:)}_{t,d})} + \epsilon \right)$$

$$\mathcal{L}_{\text{cov}} = \frac{1}{T} \sum_{t}^{T} \frac{1}{D} \sum_{i \neq j}^{D} \left[\text{Cov}(\boldsymbol{Z}_t) \right]_{ij}$$

$$\mathcal{L}_{\text{time-sim}} = \frac{1}{BT}\sum_{i}^{B}\sum_{t}^{T}\|\boldsymbol{z}_t^{(i)} - \boldsymbol{z}_{t+1}^{(i)}\|_2^2$$

$$\mathcal{L}_{\text{time-var}} = \frac{1}{BD}\sum_{i}^{B}\sum_{d}^{D}\max\left(0, 1 - \sqrt{\text{Var}(\boldsymbol{z}_{:,d}^{(i)})} + \epsilon\right)$$

$$\mathcal{L}_{\text{time-cov}} = \frac{1}{B}\sum_{b}^{B}\frac{1}{D}\sum_{i\neq j}^{D}[\text{Cov}(\boldsymbol{Z})]_{ij}$$

$$\mathcal{L}_{\text{IDM}} = \frac{1}{BT}\sum_{i}^{B}\sum_{t}^{T}\|\hat{\boldsymbol{a}}_t^{(i)} - \boldsymbol{a}_t^{(i)}\|_2^2$$

with $\boldsymbol{z}_t^{(i)} \in \mathbb{R}^D$ correspond to step $t \in [T]$ of trajectory $i \in [B]$ and T is trajectory length and B the batch size, and $\boldsymbol{Z}_t \in \mathbb{R}^{B\times D}$ denote the matrix whose i-th row is $\boldsymbol{z}_t^{(i)}$, i.e.,

$$\boldsymbol{Z}_t = \begin{bmatrix} (\boldsymbol{z}_t^{(1)})^\top \\ \vdots \\ (\boldsymbol{z}_t^{(B)})^\top \end{bmatrix},$$

Let $\bar{\boldsymbol{Z}}_t$ be the row-centered version of $\boldsymbol{Z}_t$:

$$\bar{\boldsymbol{Z}}_t = \boldsymbol{Z}_t - \frac{1}{B}\mathbf{1}\mathbf{1}^\top \boldsymbol{Z}_t.$$

Then, for each time step t and feature dimension d, the variance across the batch is

$$\text{Var}(\boldsymbol{z}_{t,d}^{(:)}) = \frac{1}{B-1}\sum_{i=1}^{B}\left(z_{t,d}^{(i)} - \frac{1}{B}\sum_{i'=1}^{B} z_{t,d}^{(i')}\right)^2,$$

and the covariance matrix across feature dimensions is

$$\text{Cov}(\boldsymbol{Z}_t) = \frac{1}{B-1}\bar{\boldsymbol{Z}}_t^\top \bar{\boldsymbol{Z}}_t \in \mathbb{R}^{D\times D}.$$

Similarly, for the temporal regularization, let $\boldsymbol{Z}^{(i)} \in \mathbb{R}^{T\times D}$ denote the matrix whose t-th row is $\boldsymbol{z}_t^{(i)}$, and let $\bar{\boldsymbol{Z}}^{(i)}$ be its row-centered version:

$$\bar{\boldsymbol{Z}}^{(i)} = \boldsymbol{Z}^{(i)} - \frac{1}{T}\mathbf{1}\mathbf{1}^\top \boldsymbol{Z}^{(i)}.$$

Then the variance across time is

$$\text{Var}(\boldsymbol{z}_{:,d}^{(i)}) = \frac{1}{T-1}\sum_{t=1}^{T}\left(z_{t,d}^{(i)} - \frac{1}{T}\sum_{t'=1}^{T} z_{t',d}^{(i)}\right)^2,$$

and the temporal covariance matrix is

$$\text{Cov}(\boldsymbol{Z}^{(i)}) = \frac{1}{T-1}(\bar{\boldsymbol{Z}}^{(i)})^\top \bar{\boldsymbol{Z}}^{(i)} \in \mathbb{R}^{D\times D}.$$

$\hat{\boldsymbol{z}}_t^{(i)} \in \mathbb{R}^d$ is the predicted embedding at step t for traj i using the predictor. $\boldsymbol{a}_t^{(i)} \in \mathbb{R}^A$ is the action associated to step t and $\hat{\boldsymbol{a}}_t^{(i)} \in \mathbb{R}^A$ is the predicted action for the inverse dynamic model (IDM) $\text{idm}(\boldsymbol{z}_t, \boldsymbol{z}_{t+1})$.

We select PLDM hyperparameters via a grid search over the loss coefficients. Since the overall objective includes six tunable weights (α, β, γ, ζ, ν, μ), an exhaustive search over all combinations is not tractable ($\mathcal{O}(n^6)$). Moreover, the original PLDM study reports coefficients that were extensively tuned per environment and dataset, which limits their transferability. We start from the set of hyperparameters from the config provided in their open-source codebase. We motivate this choice by mentioning that no mention of the time-var and time-cov regularization term are mentionned in the original paper. We then perform a grid search for each initial loss coefficient over 256 configurations on Push-T and keep the one performing the best on a held-out set. We report the best hyperparameters found in Table 2. We kept these coefficients fixed for all training.

Loss coefficient	Initial value
α	18.0
β	12
γ	0.2
ζ	0.7
ν	0.0
μ	0.0

Table 2: Best coefficient found from grid search.

C.3 GC-RL

To evaluate downstream control, we use goal-conditioned reinforcement learning (GC-RL) with offline training. In particular, we consider goal-conditioned variants of Implicit Q-Learning (IQL) and Implicit Value Learning (IVL). In both cases, observations and goals are encoded using DINOv2 patch embeddings, and policies are trained from offline datasets. Training proceeds in two phases: first learning a value function (and optionally a Q-function), followed by policy extraction via advantage-weighted regression.

GCIQL Implicit Q-Learning (IQL) [41] is an offline reinforcement learning algorithm that avoids querying out-of-distribution actions by learning a value function via expectile regression. In the goal-conditioned setting, the algorithm learns both a Q-function $Q_\psi(s_t, a_t, g)$ and a value function $V_\theta(s_t, g)$ conditioned on a goal g.

The Q-function is trained with Bellman regression, bootstrapping from a target value network $V_{\bar{\theta}}$:

$$\mathcal{L}_Q = \mathbb{E}_{(s_t,a_t,s_{t+1},g)\sim\mathcal{D}}\left[\left(Q_\psi(s_t,a_t,g) - (r(s_t,g) + \gamma m_t V_{\bar{\theta}}(s_{t+1},g))\right)^2\right],$$

where $m_t = 0$ if $s_t = g$ (terminal transition) and $m_t = 1$ otherwise.

The value network is trained using expectile regression against targets from the target Q-network $Q_{\bar{\psi}}$:

$$\mathcal{L}_V = \mathbb{E}_{(s_t,a_t,g)\sim\mathcal{D}}\left[L_\tau^2\left(Q_{\bar{\psi}}(s_t,a_t,g) - V_\theta(s_t,g)\right)\right],$$

where the expectile loss is defined as

$$L_\tau^2(u) = |\tau - \mathbb{1}(u<0)|u^2.$$

The total critic loss is given by

$$\mathcal{L}_{\text{critic}} = \mathcal{L}_Q + \mathcal{L}_V.$$

GCIVL Implicit Value Learning (IVL) [49] simplifies IQL by removing the Q-function and learning the value function directly through bootstrapped targets. The value network $V_\theta(s_t, g)$ is trained via expectile regression against a target network $V_{\bar{\theta}}$:

$$\mathcal{L}_V = \mathbb{E}_{(s_t,s_{t+1},g)\sim\mathcal{D}}\left[L_\tau^2\left(r(s_t,g) + \gamma V_{\bar{\theta}}(s_{t+1},g) - V_\theta(s_t,g)\right)\right].$$

As in IQL, L_τ^2 denotes the asymmetric expectile loss and γ is the discount factor.

Policy extraction. For both GCIQL and GCIVL, the policy $\pi_\theta(s_t, g)$ is trained via advantage-weighted regression (AWR). The policy objective is

$$\mathcal{L}_\pi = \mathbb{E}_{(s_t,a_t,g)\sim\mathcal{D}}\left[\exp\left(\beta A(s_t,a_t,g)\right)\|\pi_\theta(s_t,g) - a_t\|_2^2\right],$$

where the advantage is computed as

Algorithm 3. Pseudo-code for the training procedure of LeWorldModel. Pixel observations are encoded into latent embeddings, and a predictor estimates the dynamics by predicting the next-step embedding conditioned on actions. The model is optimized end-to-end using a next-embedding prediction loss together with a step-wise SIGReg regularization term to prevent representation collapse.

```
def LeWorldModel(obs,actions,lambd=0.1):
    """
    obs: (B, T, C, H, W) raw pixels sequence
    actions: (B, T, A) action sequence
    lambd: (float) SIGReg loss weight
    """

    emb = encoder(obs) # (B, T, D)
    next_emb = predictor(emb,actions) #(B, T, D)

    # — LeWorldModel training loss

    # next-embedding prediction loss
    pred_loss = F.mse_loss(emb[:, 1:] - next_emb[:, :-1])

    # step-wise sigreg (anti-collapse)
    sigreg_loss = mean(SIGReg(emb.transpose(0, 1))

    return pred_loss + lambd * sigreg_loss
```

$$A(s_t, a_t, g) = r(s_t, g) + \gamma V(s_{t+1}, g) - V(s_t, g),$$

and β is an inverse temperature parameter controlling the strength of advantage weighting.

C.4 GCBC

As a simple imitation learning baseline, we consider Goal-Conditioned Behavioral Cloning (GCBC) [50]. GCBC trains a goal-conditioned policy $\pi_\theta(s_t, g)$ to reproduce expert actions given the current observation s_t and a goal observation g. In our implementation, both observations and goals are encoded using DINOv2 patch embeddings before being provided to the policy network.

The policy is trained via supervised learning on an offline dataset $\mathcal{D}$ of state-action-goal tuples. Specifically, the objective minimizes the mean squared error between the predicted action and the action taken in the dataset:

$$\mathcal{L}_{\text{GCBC}} = \mathbb{E}_{(s_t, a_t, g) \sim \mathcal{D}} \left[\|\pi_\theta(s_t, g) - a_t\|_2^2 \right],$$

where s_t denotes the observation embedding, g the goal embedding, and a_t the corresponding expert action.

D Implementation details

We apply a frame-skip of 5, grouping consecutive actions between frames into a single action block. This choice enables computationally efficient longer-horizon predictions while maintaining informative temporal transitions. We use a batch size of 128 with sub-trajectories of size 4 corresponding to 4 frames and 4 blocks of 5 actions. Each frame is 224×224 pixels.

Encoder Architecture. The encoder is a Vision Transformer Tiny (ViT-Tiny) model from the Hugging Face library, using a patch size of 14.

Predictor Architecture. The predictor is implemented as a ViT-S backbone with learned positional embeddings and causal masking over the observation history. The history length is set to 3 for the *PushT* and *OGBench-Cube* environments, and to 1 for *TwoRoom*. During planning, the predictor is used autoregressively to generate rollouts of future latent states.

Decoder (Visualization Only). For visualization, we decode the `[CLS]` token embedding (192 dim) from the last encoder layer into an image using a lightweight transformer decoder. The `[CLS]` representation is first projected to a hidden dimension and used as the key and value in cross-attention. A fixed set of learnable query tokens, one for each patch of the target image, interacts with this global representation through several cross-attention layers with residual MLP blocks. For an image of size 224×224 with patch size 16, this corresponds to $P = (224/16)^2 = 196$ learnable query tokens. The resulting patch embeddings are then linearly projected to $16 \times 16 \times 3$ pixel patches and rearranged to produce a 224×224 RGB image. This decoder is used only as a diagnostic tool to visualize what visual information is retained in the `[CLS]` representation.

Planning solver. For planning, we use the Cross-Entropy Method (CEM). At each planning step, CEM samples 300 candidate action sequences and optimizes them for a maximum of 30 iterations in *PushT* and 10 iterations in the other environments. At each iteration, the top 30 trajectories are retained to update the sampling distribution, and the initial sampling variance is set to 1. The planning horizon is set to 5 steps, which corresponds to 25 environment timesteps due to the use of a frame skip of 5. We employ a receding-horizon Model Predictive Control (MPC) scheme with a horizon of 5, meaning that the entire optimized action sequence is executed before replanning. This configuration follows the setup used in [18].

E Environment & Dataset

a) **TwoRoom** is a simple continuous 2D navigation task introduced by Sobal et al. [22]. The environment consists of two rooms separated by a wall with a single door connecting them. The agent (represented as a red dot) must navigate from a random starting position in one room to a randomly sampled target location in the other room, which requires passing through the door. We collect 10,000 episodes with an average trajectory length of 92 steps. The data are generated using a simple noisy heuristic policy that first directs the agent toward the door along a straight-line path and then toward the target location once the agent has crossed into the other room. Each world model is trained on this dataset for 10 epochs.

b) **PushT** is a continuous 2D manipulation task in which an agent (represented as a blue dot) must push a T-shaped block to match a target configuration, with interactions restricted to pushing actions. We follow the same setup and dataset as Zhou et al. [18], which contains 20,000 expert episodes with an average length of 196 steps. However, we train each world model for only 10 epochs. Empirically, we observe that 10 epochs are sufficient to reach the best performance, matching the results reported in the DINO-WM paper.

c) **OGBench-Cube** is a continuous 3D robotic manipulation task in which a robotic arm with an end-effector must pick up a cube and place it at a target location. Originally introduced by Park et al. [49], we consider only the single-cube variant. We collect 10,000 episodes, each consisting of 200 steps. The data are generated using the data-collection heuristic provided in the benchmark library. Each world model is trained on this dataset for 10 epochs.

d) **Reacher** is a continuous control environment from the DeepMind Control Suite [51]. The task consists of controlling a two-joint robotic arm to reach a target location in a 2D plane. Following the setup used in DINO-WM, we consider the variant where success is defined by the perfect alignment of the arm joints with the target configuration required to reach the goal position. We train each world model for 10 epochs on a dataset of 10,000 episodes, each with 200 steps. The data are collected using a Soft Actor-Critic policy.

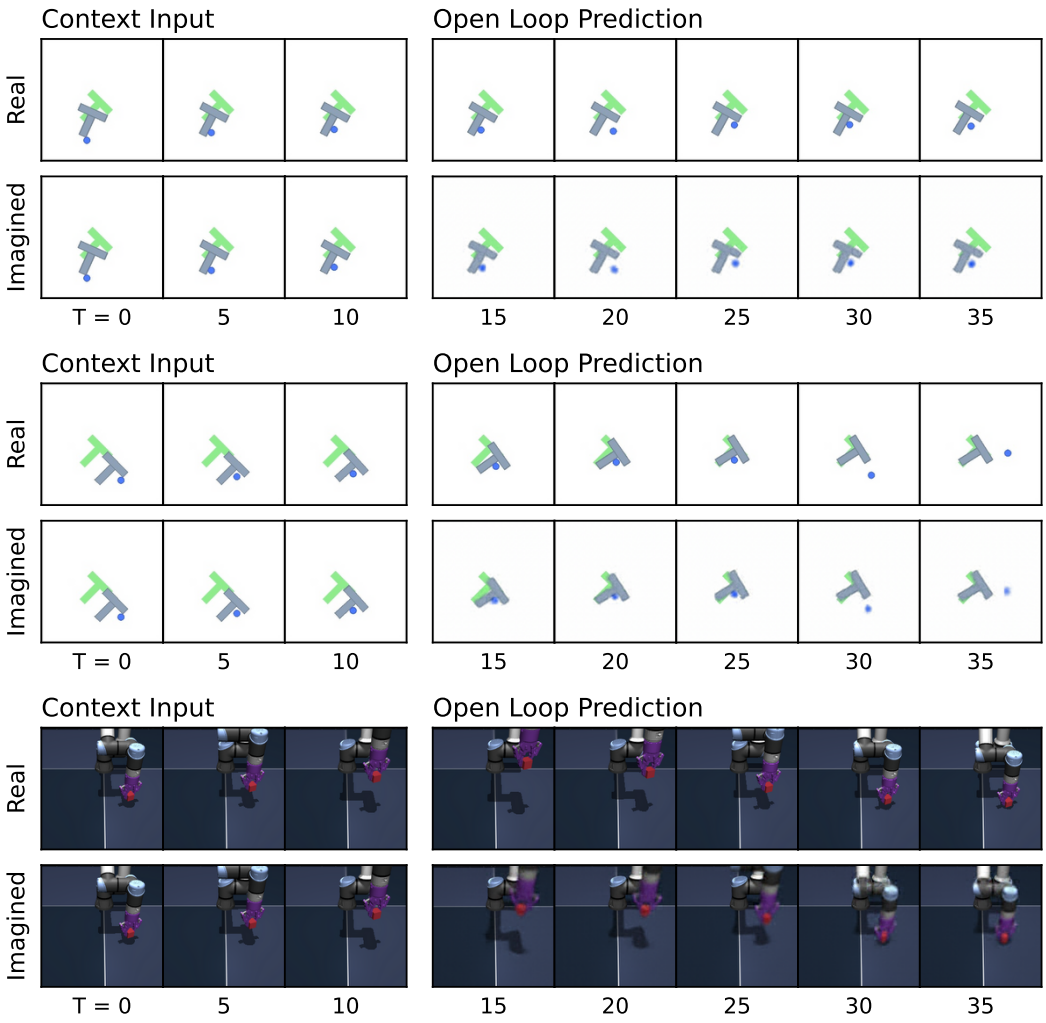

Figure 9: **Additional predictor rollouts on PushT (top) and OGBench-Cube (bottom).** Same setup as Fig. 7: three context frames are encoded into latent representations, and the predictor autoregressively generates future latent states conditioned on the action sequence. All predictions are decoded using a decoder not used during training. On PushT, the imagined trajectory closely tracks the real one, accurately capturing both agent and block motion. On OGBench-Cube, the model preserves the overall scene layout and cube displacement but loses finer details such as end-effector orientation at longer horizons, consistent with the lower probing accuracy on rotational quantities reported in Tab. 4.

F Evaluation Details

F.1 Control

We evaluate LeWM on goal-conditioned control tasks in the three environments introduced previously. Control performance is measured using two parameters: the evaluation budget and the distance to the goal. The evaluation budget corresponds to the maximum number of actions the agent is allowed to execute in the environment. The goal distance determines how far in the future the goal state is sampled relative to the initial state. During evaluation, trajectories are sampled from the offline dataset. The initial state is chosen by randomly sampling a state from a trajectory in the dataset, while the goal state corresponds to a state occurring several timesteps later in the same trajectory. This ensures that the goal is reachable and consistent with the dataset dynamics. In *TwoRoom*, the evaluation budget is set to 50 steps, and the goal state is sampled 25 timesteps in the future. In *PushT*, the evaluation budget is 50 steps and the goal is sampled 25 timesteps in the future. In *OGBench-Cube* and *Reacher*, the evaluation budget is 50 steps, and the goal is sampled 25 timesteps in the future.

F.2 Probing

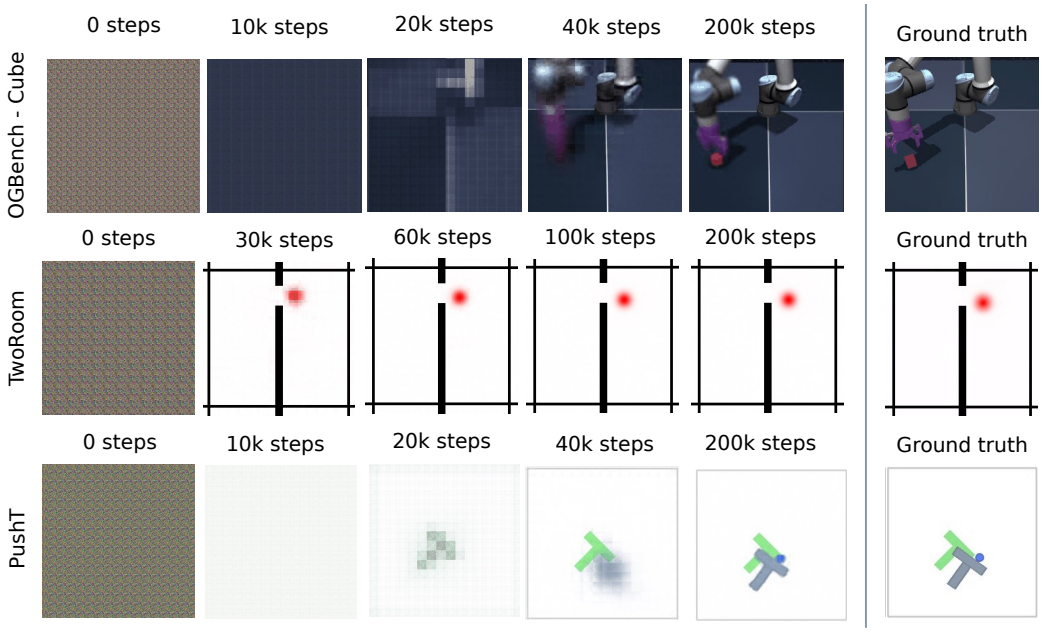

Figure 10: **Decoder visualization during training.** As training progresses, the latent representation increasingly captures the information required to reconstruct the visual scene, even though no reconstruction loss is used during training. Early in training, the decoded images correspond to slow features, a phenomenon previously reported [21].

We use probing to analyze the information contained in the learned latent representations across the three environments. Specifically, we train both linear and non-linear probes to predict physical quantities from the latent embeddings. Linear probes evaluate whether the information is linearly accessible in the latent space, while non-linear probes assess whether the information is present but potentially entangled.

For each probe, we report the mean squared error (MSE) and the Pearson correlation coefficient between the predicted and ground-truth quantities.

The probed variables differ across environments. In *TwoRoom*, we probe the 2D position of the agent (Tab. 3). In *PushT*, we probe both the state of the agent and the state of the block (Tab. 1). In *OGBench-Cube*, we probe the position of the cube and the position of the robot end-effector (Tab. 4).

Table 3: **Physical Latent Probing results on TwoRoom.** Although LeWM underperforms PLDM in downstream planning on this environment, it matches or outperforms PLDM across all probing metrics, and both methods substantially outperform DINO-WM on the linear probe. This suggests that the learned latent space captures the underlying physical state equally well and that the planning gap is not due to a less informative representation but rather to other factors such as the dynamics model or the planning procedure itself.

	Agent Position			
	Linear		MLP	
Model	MSE ↓	r ↑	MSE ↓	r ↑
DINO-WM	0.488 ± 0.451	0.824	0.000 ± 0.000	0.999
PLDM	0.008 ± 0.041	0.996	0.000 ± 0.000	1.000
LeWM	0.008 ± 0.018	0.996	0.000 ± 0.000	1.000

Table 4: **Physical latent probing results on OGBench-Cube.** LeWM matches or outperforms PLDM on most properties and achieves the best results on positional quantities such as block position and end-effector position. DINO-WM retains a clear advantage on dynamic and rotational properties (joint velocity, end-effector yaw), likely because such quantities benefit from the richer visual priors learned during large-scale pretraining. All three methods struggle to recover block orientation (quaternion and yaw), suggesting that fine-grained rotational information remains difficult to encode in compact latent spaces regardless of the training strategy.

		Linear		**MLP**	
Property	**Model**	MSE ↓	r ↑	MSE ↓	r ↑
Joint Position	DINO-WM	0.960 ± 1.150	0.808	0.200 ± 0.967	0.870
	PLDM	0.372 ± 1.172	0.695	0.340 ± 1.164	0.728
	LeWM	0.352 ± 1.173	0.706	0.330 ± 1.157	0.742
Joint Velocity	DINO-WM	0.792 ± 0.748	0.763	0.263 ± 0.683	0.852
	PLDM	1.016 ± 0.905	0.115	0.661 ± 0.830	0.536
	LeWM	1.021 ± 0.902	0.095	0.818 ± 0.899	0.386
End-Effector Position	DINO-WM	0.024 ± 0.010	0.996	0.004 ± 0.003	0.999
	PLDM	0.052 ± 0.073	0.974	0.013 ± 0.029	0.993
	LeWM	0.018 ± 0.025	0.991	0.003 ± 0.004	0.998
End-Effector Yaw	DINO-WM	3.317 ± 1.016	0.828	0.167 ± 0.168	0.917
	PLDM	0.996 ± 0.165	0.056	0.985 ± 0.207	0.117
	LeWM	0.980 ± 0.295	0.124	0.952 ± 0.369	0.213
Gripper	DINO-WM	0.114 ± 0.095	0.943	0.038 ± 0.060	0.982
	PLDM	0.234 ± 0.169	0.876	0.066 ± 0.111	0.967
	LeWM	0.121 ± 0.111	0.938	0.048 ± 0.079	0.976
Block Position	DINO-WM	0.085 ± 0.029	0.991	0.007 ± 0.007	0.998
	PLDM	0.031 ± 0.023	0.985	0.003 ± 0.004	0.999
	LeWM	0.007 ± 0.010	0.997	0.002 ± 0.003	0.999
Block Quaternion	DINO-WM	1.596 ± 10.457	0.257	0.769 ± 8.046	0.411
	PLDM	1.021 ± 12.600	0.066	0.989 ± 12.140	0.218
	LeWM	1.019 ± 12.596	0.087	0.963 ± 11.450	0.224
Block Yaw	DINO-WM	4.223 ± 2.530	0.176	0.916 ± 0.278	0.304
	PLDM	0.996 ± 0.088	0.061	0.989 ± 0.140	0.106
	LeWM	0.996 ± 0.094	0.062	0.973 ± 0.199	0.164
Overall	DINO-WM	1.162 ± 1.579	0.725	0.290 ± 1.202	0.799
	PLDM	0.611 ± 1.875	0.464	0.503 ± 1.809	0.600
	LeWM	0.592 ± 1.874	0.477	0.525 ± 1.714	0.584

F.3 Violation-of-expectation

We evaluate physical understanding using the violation-of-expectation (VoE) framework across three environments. In each environment, we generate three types of trajectories: an unperturbed reference trajectory, a trajectory containing a visual perturbation, and a trajectory containing a physical perturbation. Visual perturbations correspond to abrupt color changes of an object, while physical perturbations correspond to teleporting objects to random positions, thereby violating physical continuity. Examples of trajectories are shown in Figure 11.

TwoRoom. In the *TwoRoom* environment, the agent is controlled by an expert policy that navigates toward a goal position. We generate three trajectories: (1) an unperturbed trajectory, (2) a trajectory where the color of the agent changes midway through the episode, and (3) a trajectory where the agent is teleported to a random position at the same timestep. The resulting surprise signals for PLDM and DINO-WM are shown in the left panels of Figures 12 and 14, respectively.

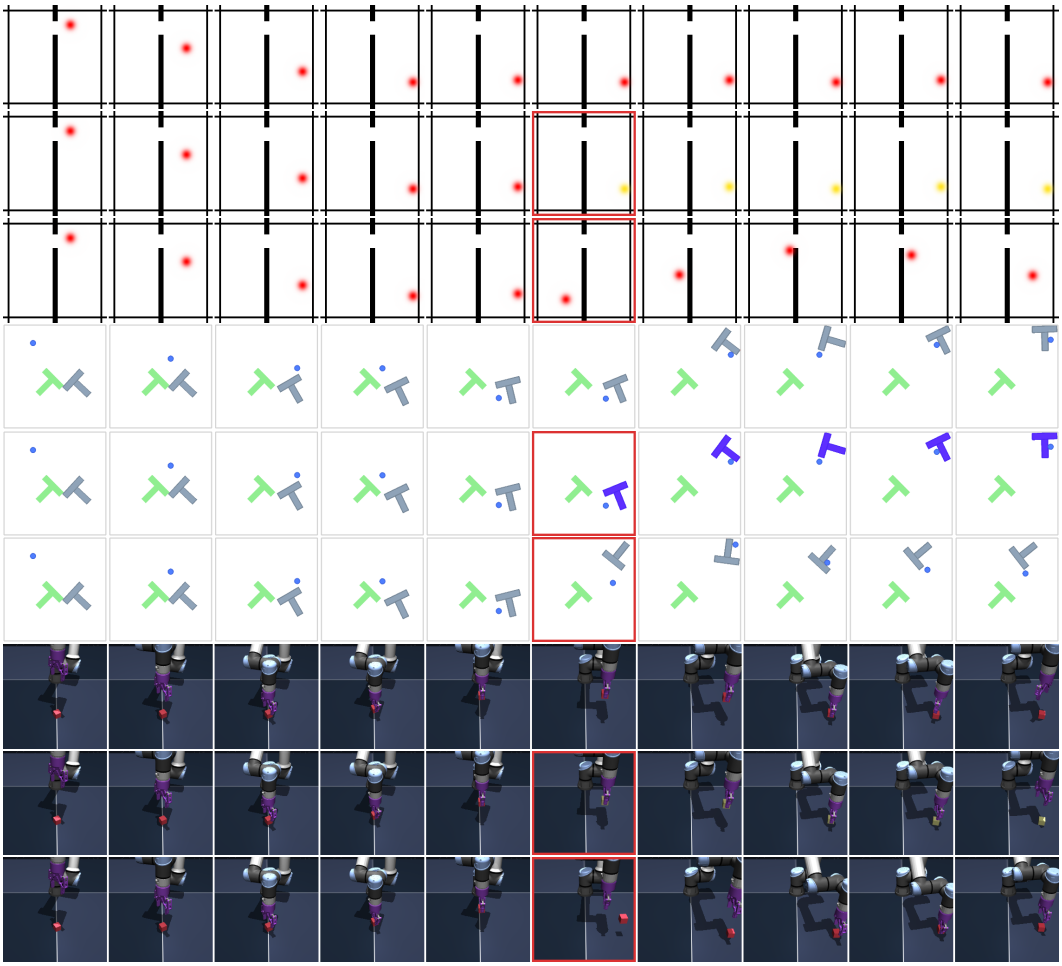

Figure 11: **Example of trajectories used for the Violation of Expectation experiments** (Sec. 5.2). For each environment, the first row corresponds to the unperturbed trajectory, the second row corresponds to a trajectory where a visual perturbation occurs and the third row displays trajectories where the state of the system is randomly reset in the middle of the trajectory. The frame where the perturbation occurs is highlighted in red.

PushT. In the *PushT* environment, the agent is controlled by a random policy biased toward interacting with the block. As before, we construct three trajectories: (1) an unperturbed trajectory, (2) a trajectory where the color of the block changes abruptly during the episode, and (3) a trajectory where both the agent and the block are teleported to random positions at the perturbation timestep. The corresponding surprise signals for PLDM and DINO-WM are shown in the center panels of Figures 12 and 14.

OGBench-Cube. In the *OGBench-Cube* environment, the agent follows an expert policy that picks up the cube and places it at a target position. We again consider three trajectories: (1) an unperturbed trajectory, (2) a trajectory where the cube's color changes during the episode, and (3) a trajectory where the cube is teleported to a random position midway through the trajectory. The resulting surprise signals for PLDM and DINO-WM are shown in the right panels of Figures 12 and 14.

G Ablations.

Training variance. To assess the stability of training, we retrain the model using multiple random seeds. As shown in Tab. 5, the resulting performance exhibits consistently high success rates with low variance across runs, indicating that the training procedure is stable and reproducible.

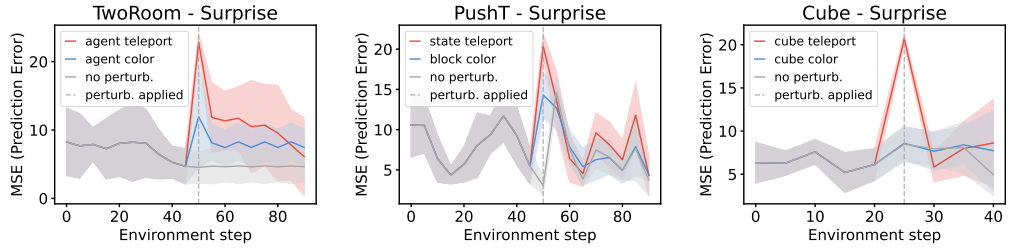

Figure 12: **Violation-of-expectation evaluation with PLDM.** From left to right: *TwoRoom*, *PushT*, and *OGBench-Cube*. Surprise is plotted over time for unperturbed, visually perturbed, and physically perturbed trajectories. In *TwoRoom* and *PushT*, the model assigns significantly higher surprise to both visual and physical perturbations. In *OGBench-Cube*, the increase in surprise is weaker and not consistently significant.

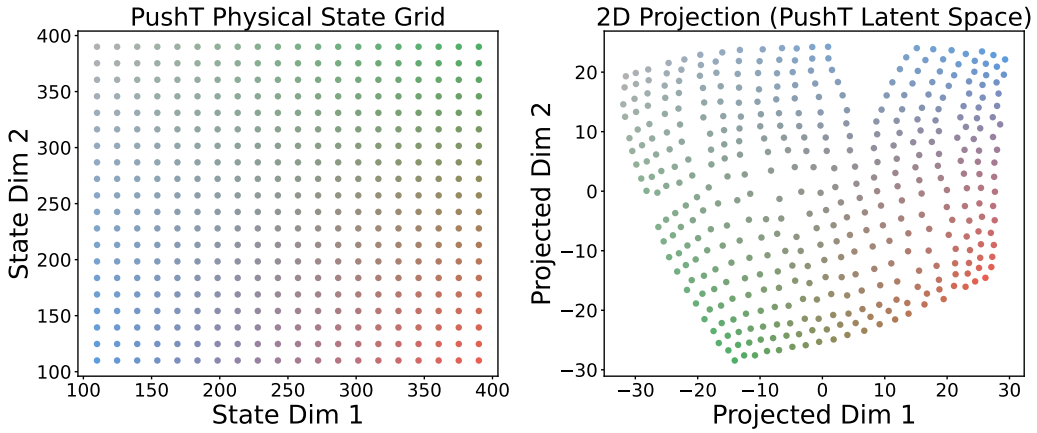

Figure 13: **Visualization of the latent space** obtained with LeWM for the PushT environment. On the left, the grid of states is obtained by moving the agent and the block in the x-y plane. On the right, the embeddings of these states are visualized using a t-SNE.

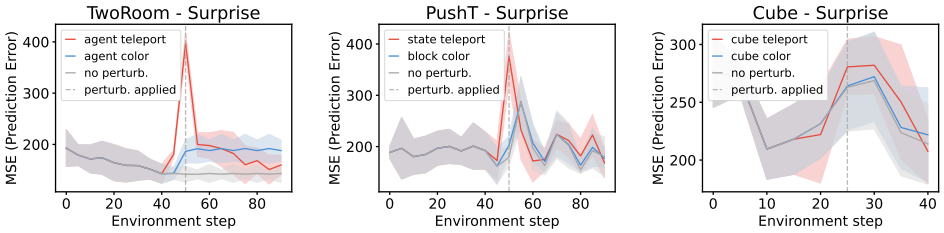

Figure 14: **Violation-of-expectation evaluation with DINO-WM.** From left to right: *TwoRoom*, *PushT*, and *OGBench-Cube*. Surprise is plotted over time for unperturbed, visually perturbed, and physically perturbed trajectories. While the model detects both perturbations in *TwoRoom* and *PushT*, surprise does not increase significantly for either perturbation in *OGBench-Cube*.

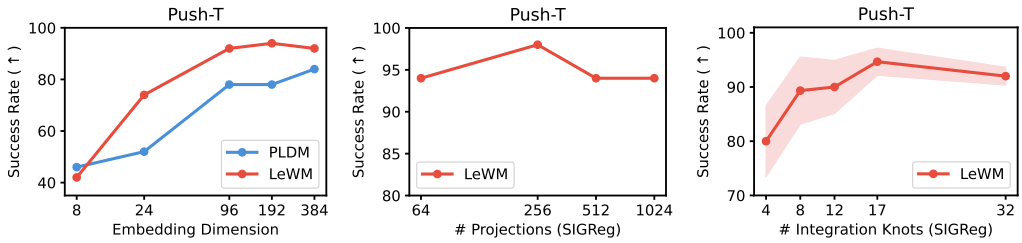

Figure 15: **Ablation studies of key design choices in LeWM. Left:** effect of the embedding dimension; performance improves with larger embeddings but quickly saturates beyond a certain threshold. **Center:** effect of the number of random projections used in SIGReg; performance remains stable, indicating that this parameter is not critical. **Right:** effect of the number of integration knots used to compute the SIGReg loss; results are similarly insensitive to this parameter.

Table 5: **Training Variance.** We report the mean success rate across three training seeds and the corresponding variance, evaluated over the same set of 50 trajectories on Push-T. The goal configuration is reachable within 25 steps, and we allow a planning budget of 50 steps. PLDM exhibits higher variance compared to DINO-WM and LeWM.

Model	**Push-T** (SR ↑)
DINO-WM	92.0 ± 1.63
PLDM	78.0 ± 5.0
LeWM (ours)	$\mathbf{96.0 \pm 2.83}$

Embedding dimensions. We study the impact of the embedding dimensionality on performance. As shown in Fig. 15, performance drops when the embedding dimension falls below a certain threshold (around 184), while increasing the dimension beyond this value yields diminishing returns and leads to performance saturation.

Number of projections in SIGReg. We study the impact of the number of projections used in SIGReg. As shown in Fig. 15, varying the number of projections has little effect on performance in downstream control tasks. This suggests that the method is largely insensitive to this hyperparameter, and therefore it does not require careful tuning. In practice, this leaves λ as the only effective hyperparameter to optimize.

Weight of SIGReg regularization. We analyze the effect of the SIGReg regularization weight λ. As shown in Fig.16, the method achieves high performance across a wide range of values for λ. In particular, for $\lambda \in [0.01, 0.2]$, the success rate remains above 80%. This indicates that the approach is robust to the choice of this parameter. Moreover, since λ is the only effective hyperparameter, it can be tuned efficiently, for instance via a simple bisection search.

Predictor Size. We analyze the effect of the predictor size on performance. As shown in Tab. 6, the best results are obtained with a ViT-S predictor. Reducing the predictor to a ViT-T model leads to a drop in performance, while increasing the size to ViT-B does not provide additional gains and slightly degrades performance. This suggests that ViT-S offers the best trade-off between model capacity and optimization stability for this task.

Table 6: Effect of the predictor size on planning performance in the Push-T environment. We report the success rate (SR). The ViT-S predictor achieves the best performance.

pred. size	**Push-T** (SR ↑)
tiny	80.67 ± 6.54
small	$\mathbf{96.0 \pm 2.83}$
base	86.7 ± 3.06

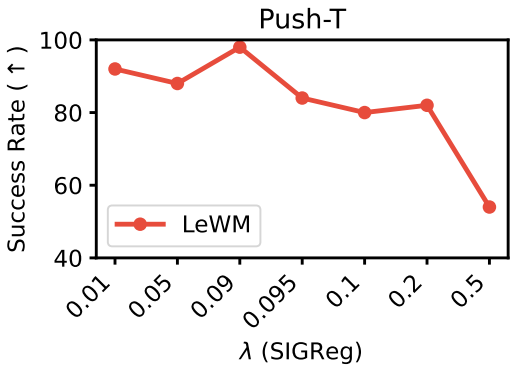

Figure 16: **Effect of the SIGReg regularization weight λ on Push-T planning performance.** Success rate remains above 80% across a wide range of values ($\lambda \in [0.01, 0.2]$), peaking near $\lambda = 0.09$. Performance degrades sharply only at $\lambda = 0.5$, where the regularizer dominates the prediction loss and hinders dynamics modeling. Since λ is the only effective hyperparameter of LeWM, the SIGReg loss coefficient is easy to tune via a simple bisection search.

Table 7: Effect of adding a reconstruction loss during training. We report the success rate (SR) on the Push-T planning task. The model trained without the decoder loss achieves higher performance.

	Push-T (SR ↑)
LeWM w/o decoder loss	$\mathbf{96.0 \pm 2.83}$
LeWM with decoder loss	86.0 ± 7.54

Decoder. We study the impact of adding a reconstruction loss during training. As shown in Tab. 7, incorporating a decoder and a reconstruction objective does not improve downstream control performance. In fact, performance slightly decreases compared to the model trained without a decoder. This suggests that the JEPA training objective already captures the information necessary for planning, while the reconstruction loss may encourage the model to encode additional visual details that are not relevant for control.

Architecture. We study the impact of encoder architecture on LeWM performance by replacing the ViT encoder with a ResNet-18 backbone. As shown in Tab. 8, LeWM achieves competitive performance with both architectures, suggesting that it is agnostic to the choice of vision encoder used during training, though ViT retains a modest advantage.

Table 8: **Encoder Architecture Effect.** We report the success rate (SR) on the Push-T planning task. LeWM achieves competitive performance across encoder architectures, with ViT holding a slight edge.

	Push-T (SR ↑)
LeWM ViT	$\mathbf{96.0 \pm 2.83}$
LeWM ResNet-18	94.0 ± 3.27

Predictor Dropout. We analyze the effect of applying dropout in the predictor during training. As shown in Tab. 9, introducing a small amount of dropout significantly improves downstream control performance. In particular, a dropout rate of 0.1 achieves the highest success rate, while both lower and higher values lead to worse performance. This suggests that moderate dropout helps regularize the predictor and improves generalization, whereas excessive dropout degrades the quality of the learned dynamics.

Planning Solver. We compare planning performance across diverse solvers. As shown in Tab. 10.

Table 9: Effect of predictor dropout during training on Push-T planning performance. We report the success rate (SR). A small amount of dropout ($p = 0.1$) yields the best results.

p	**Push-T** (SR ↑)
0.0	78 ± 6.54
0.1	$\mathbf{96.0 \pm 2.83}$
0.2	85.33 ± 5.74
0.5	66.67 ± 4.11

Table 10: **Planning Solver Performance.** We report the success rate (SR) on the Push-T planning task.

	LeWM	**PLDM**
CEM	$\mathbf{96.0 \pm 2.83}$	78.0 ± 5.0
SGD (β_1)	26 ± 4.32	4.67 ± 0.06
RMSProp (β_2)	67.33 ± 2.49	49.33 ± 8.26
Adam (β_1, β_2)	84 ± 7.12	80 ± 3.27

H Temporal Latent Path Straightening.

The temporal straightening hypothesis, introduced by Hénaff et al. [43], posits that we represent complex temporal dynamics as smooth, approximately straight trajectories in our representation spaces. This principle has since found applications beyond neuroscience: Internò et al. [44] leverage temporal straightness measured from DINOv2 features to discriminate AI-generated videos from real ones, demonstrating that this geometric property carries a meaningful signal about the nature of the underlying dynamics, and Wang et al. [45] shows it can be beneficial for planning.

During training on PushT, we record, for curiosity, the temporal straightness of LeWM's latent trajectories. Given a sequence of latent embeddings $\mathbf{z}_{1:T} \in \mathbb{R}^{B \times T \times D}$, we define the temporal velocity vectors as $\mathbf{v}_t = \mathbf{z}_{t+1} - \mathbf{z}_t$. The path straightening measure is defined as the mean pairwise cosine similarity between consecutive velocities:

$$\mathcal{S}_{\text{straight}} = \frac{1}{B(T-2)} \sum_{i=1}^{B} \sum_{t=1}^{T-2} \frac{\langle \mathbf{v}_t^{(i)}, \mathbf{v}_{t+1}^{(i)} \rangle}{\|\mathbf{v}_t^{(i)}\| \, \|\mathbf{v}_{t+1}^{(i)}\|}. \tag{9}$$

A value of $\mathcal{S}_{\text{straight}}$ close to 1 indicates that consecutive velocities are nearly collinear, meaning the latent trajectory approaches a straight line. Interestingly, we observe that temporal straightening emerges naturally over the course of training without any training term explicitly encouraging it (Fig. 17).

We hypothesize that this emerges because SIGReg is applied independently at each time step but not across the temporal dimension, leaving the temporal structure unconstrained. This allows the encoder to converge toward a form of *temporal collapse*, where successive embeddings evolve along increasingly linear paths. Rather than being detrimental, this implicit bias appears to benefit downstream performance, as shown in Fig. 6. Notably, LeWM achieves higher temporal straightness than PLDM despite having no explicit regularizer encouraging it, whereas PLDM employs a regularizer on consecutive latent states that directly promotes temporal smoothness.

I Training Curves

We visualize several training curves comparing the optimization dynamics of LeWM (Fig. 18) and PLDM (Fig. 19). In contrast to PLDM, whose objective contains multiple regularization terms, LeWM uses a single regularization term in addition to the prediction loss, making the training dynamics easier to interpret and analyze.

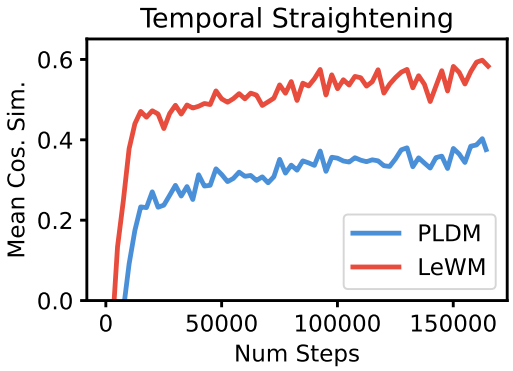

Figure 17: **Temporal Latent Straightening on Push-T.** Mean cosine similarity between consecutive latent velocity vectors (Eq. 9) over training. Higher values indicate straighter latent trajectories. PLDM explicitly encourages temporal regularity through a dedicated temporal smoothness loss ($\mathcal{L}_{\text{time-sim}}$), yet LeWM achieves substantially straighter latent paths as a purely emergent phenomenon, without any temporal regularization term in its objective.

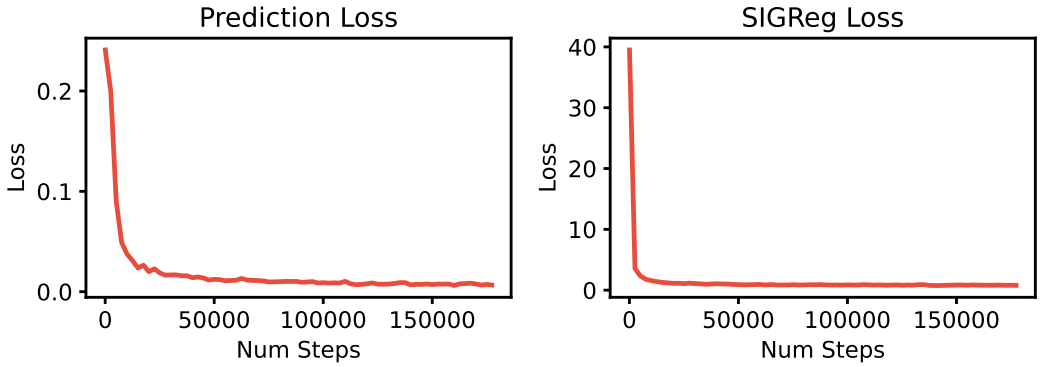

Figure 18: **Push-T Training curves for LeWM.**

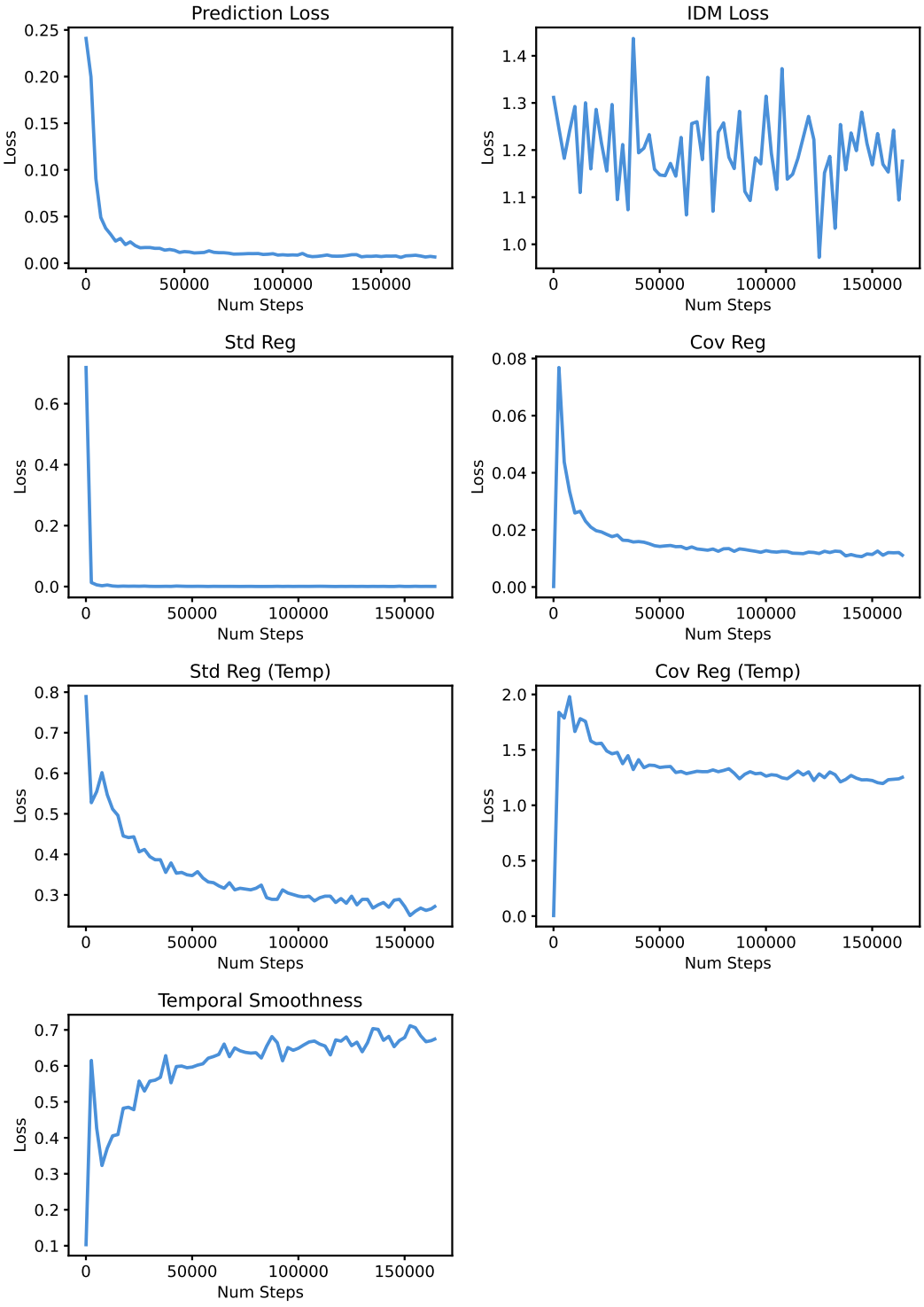

Figure 19: **Push-T Training curves for PLDM.**

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The four main claims, stable end-to-end pixel-based JEPA with a single effective hyperparameter, competitive control with a 15M-parameter model, up to $48\times$ faster planning, and physical structure in the latent space, are each substantiated in the article.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Limitations are discussed at the end of the conclusion, covering short planning horizons, reliance on offline datasets with sufficient coverage, weakened SIGReg in low-diversity environments, and dependence on action labels.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: The paper does not introduce new theoretical results; it relies on existing results which are properly cited.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: The architecture, training objective, and pseudo-code are fully described, with hyperparameters, dataset details, and planning setup provided.

Guidelines:

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

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

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

   Answer: [Yes]

   Justification: The material includes the forward pass and the core module components of LeWM (cf. URL in abstract); the full training and evaluation code along with the datasets will be released upon publication.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Training and evaluation details (architecture, hyperparameters, batch size, frame skip, planning horizon, CEM settings) are provided, with baseline hyperparameter selection described in Appendices.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Planning success rates are reported with standard deviations across multiple seeds, probing results include standard deviations, and violation-of-expectation results report paired t-test significance ($p < 0.01$).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The paper indicates that LeWM (15M parameters) is trained on a single GPU in a few hours; planning runtime is reported.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

Justification: The paper presents foundational research on world model learning evaluated in simulated control environments, with no direct path to specific societal applications or deployments.

Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The released model is a small world model trained on simulated control environments and poses no high risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All existing assets used (DINOv2, OGBench, DeepMind Control Suite, PushT dataset, baselines DINO-WM and PLDM) are properly cited in the paper and used in accordance with their respective licenses.

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The released code components in the supplemental material are documented; full code and datasets, with accompanying documentation, will be released upon publication.

    Guidelines:

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

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

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

    Answer: [N/A]

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

    Guidelines:

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

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

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

    Answer: [N/A]

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

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: LLMs are not used as part of the core methodology of this research.

Guidelines:

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