FICReg: Forward-Inverse Consistency Regularization for Latent World Models

Sungwon Seo [1] **Jean Seong Bjorn Choe** [2] **Jong-Kook Kim** [1]

Abstract

Joint Embedding Predictive Architecture (JEPA) based world models learn dynamics efficiently in latent space. However, their forward predictors are typically supervised to match target states, without an explicit mechanism to verify that predictions preserve the causal effect of actions. While Inverse Dynamics Models (IDMs) have been used to learn action-relevant representations, they are typically trained jointly with the encoder on ground-truth states, risking co-adaptation and shortcut solutions. We propose the Forward-Inverse Consistency Regularizer (FICReg), which repurposes the IDM as a frozen consistency judge for the forward predictor: the IDM is trained on ground-truth transitions but applied to predicted states with its weights frozen, so that gradients flow exclusively into the predictor. This controlled gradient path improves the dynamics learned by the forward model—pushing it to produce states from which the executed action is recoverable—without distorting the encoder or the IDM itself. Preliminary experiments on four continuous control environments suggest that FICReg benefits tasks where actions play a decisive causal role in state transitions.

1. Introduction

World models learn dynamics of the environment to predict future states and plan actions (Ha & Schmidhuber, 2018). Recently, Joint Embedding Predictive Architecture (JEPA) based approaches (LeCun et al., 2022) have advanced this direction by learning dynamics entirely in latent space, avoiding the cost of pixel-level reconstruction and enabling offline planning from pre-collected trajectories without reward labels. In this setting, the quality of the learned forward predictions is critical, as planning relies entirely on the world model's dynamics without access to reward signals.

Among JEPA-based methods (Assran et al., 2023; Bardes et al., 2023; Assran et al., 2025; Mur-Labadia et al., 2026; Nam et al., 2026), Planning with Latent Dynamics Models (PLDM) (Sobal et al., 2025) first demonstrated that end-to-end latent dynamics models can achieve competitive performance on planning benchmarks, though its multi-term VICReg-based objective requires careful balancing of competing gradients across components. LeWorldModel (LeWM) (Maes et al., 2026b) proposed a simple yet effective approach that combines Sketched Isotropic Gaussian regularization (SIGReg) (Balestriero & LeCun, 2025) for collapse-free representation learning with a forward dynamics model, demonstrating strong performance on offline, reward-free planning and manipulation tasks. However, the forward predictor in LeWM is supervised only to match target states via mean squared error. While actions are provided as input through Adaptive Layer Normalization (AdaLN), the training objective does not explicitly verify that predicted transitions reflect the causal effect of those actions. The predictor can minimize the prediction loss by approximating target states without faithfully encoding how actions drive state changes, potentially underutilizing action information in the learned dynamics.

A natural approach to enforce action-awareness is to introduce an Inverse Dynamics Model (IDM) that predicts the action taken between two consecutive states. Prior work such as the Intrinsic Curiosity Module (ICM) (Pathak et al., 2017) uses an IDM as a representation learning objective: the inverse model is jointly trained with the encoder on ground-truth state pairs to encourage action-relevant features. However, prior work has observed that when all components share gradients, the encoder may co-adapt with the forward model and learn overly simplified representations (Burda et al., 2018). More fundamentally, the IDM in ICM shapes *what the encoder learns*, not *how the forward model predicts*.

We take a different approach. Rather than using the IDM to train the encoder, we repurpose it as a *frozen consistency judge* that directly regularizes the forward predictor. The key idea is a two-stage gradient design within a single optimization step:

[1]School of Electrical Engineering, Korea University, Seoul, Republic of Korea [2]Independent Researcher. Correspondence to: Jong-Kook Kim <jongkook@korea.ac.kr>.

Accepted at ICML 2026 Workshop on Decision-Making from Offline Datasets to Online Adaptation: Black-Box Optimization to Reinforcement Learning.

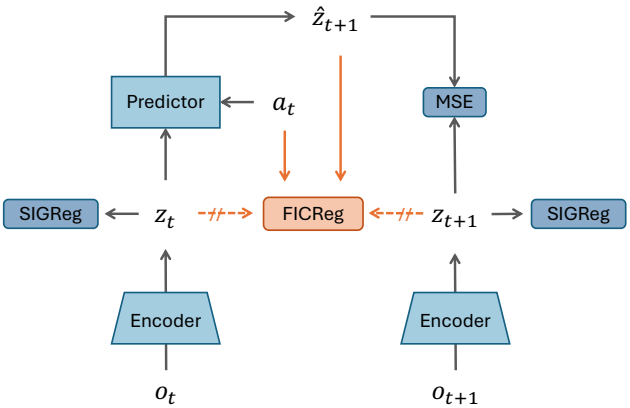


Figure 1. Overview of the FICReg architecture.

1. The IDM is trained on ground-truth state transitions with detached encoder outputs, learning to recover actions from real dynamics without influencing the encoder.

2. The same IDM is then frozen and applied to *predicted* transitions. Gradients flow exclusively into the forward predictor, pushing it to produce next states from which the executed action is recoverable.

This separation ensures that the IDM remains anchored to real transition dynamics while providing a learned consistency metric for the predictor. Because all gradient paths are isolated via selective detaching and freezing, the auxiliary losses do not interfere with the baseline encoder or predictor, making the method a plug-and-play module that requires no hyperparameter re-tuning of the base model.

We call this approach the Forward-Inverse Consistency Regularizer (FICReg). Preliminary experiments on four continuous control environments suggest that FICReg benefits tasks where actions play a decisive causal role in state transitions, while preserving baseline performance elsewhere. Linear probing further indicates that FICReg improves the linear decodability of action-relevant state variables in the learned representations.

2. Forward-Inverse Consistency Regularizer

FICReg leverages a frozen inverse dynamics model (IDM) to improve dynamics learning in JEPA-based world models. The method builds on LeWorldModel (Maes et al., 2026b), which learns latent dynamics end-to-end through state prediction and a Gaussian regularizer. The key addition is training an IDM exclusively on ground-truth transitions, and then reusing it as a frozen judge that regularizes the forward predictor—with gradient paths carefully controlled so that the encoder and IDM remain unaffected.

2.1. Baseline: LeWorldModel

LeWorldModel (LeWM) consists of an encoder f_θ that maps pixel observations to latent representations $z_t = f_\theta(o_t)$, and a forward predictor g_ϕ that estimates the next latent state: $\hat{z}_{t+1} = g_\phi(z_{1:t}, a_t)$. The predictor conditions on actions through Adaptive Layer Normalization (AdaLN). LeWM is trained fully end-to-end without stop-gradient or a target encoder. With no stop-gradient on the target z_{t+1}, the prediction loss trains both the encoder and predictor jointly, incentivizing the encoder to produce representations that are predictable by the forward model. The training objective combines a state prediction loss and a collapse-prevention regularizer:

$$\mathcal{L}_{\text{base}} = \underbrace{\|\hat{z}_{t+1} - z_{t+1}\|^2}_{\mathcal{L}_{\text{pred}}} + \lambda \cdot \underbrace{\frac{1}{M}\sum_{m=1}^{M} T(\mathbf{Z}\mathbf{u}^{(m)})}_{\mathcal{L}_{\text{sigreg}}} \quad (1)$$

where $\mathcal{L}_{\text{pred}}$ supervises the predictor to match the encoder's output for the true next observation, and $\mathcal{L}_{\text{sigreg}}$ (SIGReg) (Balestriero & LeCun, 2025) encourages the batch of latent embeddings $\mathbf{Z} \in \mathbb{R}^{N \times B \times d}$ to follow an isotropic Gaussian distribution by applying a univariate normality test T to random projections $\mathbf{u}^{(m)}$. By matching all one-dimensional marginals to a Gaussian, SIGReg promotes feature diversity and prevents the encoder from mapping all observations to a constant representation.

While this combination produces a stable world model, the dynamics learning relies solely on next-state regression. $\mathcal{L}_{\text{pred}}$ measures the distance to the target state but does not verify that the predicted state preserves action-relevant transition structure. The predictor can minimize $\mathcal{L}_{\text{pred}}$ by approximating the target without ensuring that the action's causal effect is faithfully encoded.

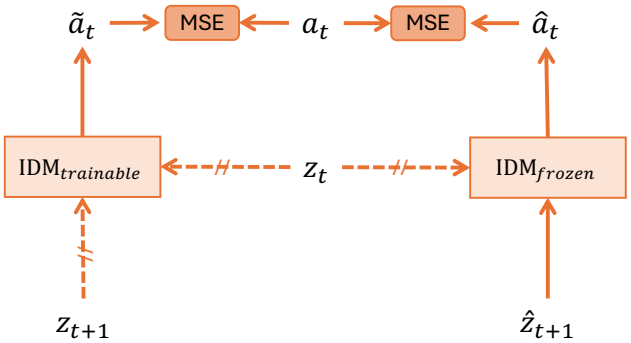


Figure 2. The forward-inverse consistency losses and gradient flow. Dashed arrows indicate detached (no-gradient) paths. $\tilde{a}_t$ and $\hat{a}$ denote actions recovered by the trainable and frozen IDM, respectively.

2.2. Forward via Inverse Dynamics Model

Two additional loss terms enforce forward-inverse consistent dynamics. Both are computed in a single forward pass alongside the baseline losses and optimized in one gradient step. Importantly, LeWM's core design principle—pure end-to-end learning without stop-gradients—is fully preserved for the encoder-predictor path. The `detach` operations introduced here apply only to the auxiliary IDM inputs, not to the main representation learning pipeline. This selective use of stop-gradients serves a different purpose from the heuristic stabilization techniques that LeWM avoids: it physically isolates the IDM's gradient path to prevent the co-adaptation identified in prior joint-training approaches (Burda et al., 2018), while leaving the baseline learning dynamics untouched.

Ground-Truth IDM Loss. The IDM is trained to recover the action a_t from two consecutive ground-truth encoder outputs:

$$\mathcal{L}_{\text{idm}} = \|\text{IDM}(\bar{z}_t, \bar{z}_{t+1}) - a_t\|^2 \tag{2}$$

where $\bar{z}$ denotes a detached encoder output. Detaching both inputs prevents the encoder from reshaping its representations to trivially expose action information, following the gradient isolation principle described above. The IDM trains only on ground-truth transitions and never on predicted states to avoid overfitting to artifacts of the predictor rather than learning genuine transition dynamics.

Forward via IDM Consistency Loss. The core contribution is a consistency loss that reuses the frozen IDM as a learned metric over predicted transitions:

$$\mathcal{L}_{\text{fwd}} = \|\text{IDM}_{\text{frozen}}(\bar{z}_t, \hat{z}_{t+1}) - a_t\|^2 \tag{3}$$

Here $\bar{z}_t$ is the detached current state from the encoder and $\hat{z}_{t+1}$ is the predicted next state from the forward predictor. The IDM weights are frozen, so gradients flow only into the predictor through $\hat{z}_{t+1}$. This pushes the predictor to produce next states from which the executed action is recoverable—enforcing that predictions lie on the forward-inverse consistent manifold learned by the IDM, without contaminating either the IDM or the encoder.

The two losses play complementary roles: $\mathcal{L}_{\text{idm}}$ teaches the IDM what real transitions look like; $\mathcal{L}_{\text{fwd}}$ uses that knowledge to judge whether the predictor's outputs are consistent with real dynamics.

2.3. Joint Optimization

The full objective combines all four losses in a single backward pass:

$$\mathcal{L} = \mathcal{L}_{\text{pred}} + \lambda \cdot \mathcal{L}_{\text{sigreg}} + \mu \cdot \mathcal{L}_{\text{idm}} + \gamma \cdot \mathcal{L}_{\text{fwd}} \tag{4}$$

where λ, μ, and γ weight the SIGReg, IDM, and forward-inverse consistency terms respectively.

Algorithm 1 FICReg Training Step

```
def ficreg(obs, acts, lambd, mu, gamma):
    z = encoder(obs)     # (B, T, D)
    z_pred = predictor(z[:, :-1], acts)

    # Baseline losses
    L_pred = mse(z_pred, z[:, 1:])
    L_sig = mean(SIGReg(z.transpose(0, 1)))

    # IDM on ground-truth(detached encoder)
    a_gt = idm(z[:, :-1].detach(),
               z[:, 1:].detach())
    L_idm = mse(a_gt, acts)

    # Frozen IDM judges predictions
    with frozen(idm):
        a_pred = idm(z[:, :-1].detach(),
                     z_pred)

    L_fwd = mse(a_pred, acts)

    return (L_pred + lambd * L_sig
            + mu * L_idm + gamma * L_fwd)
```

The encoder's learning dynamics remain identical to the baseline: both $\mathcal{L}_{\text{idm}}$ and $\mathcal{L}_{\text{fwd}}$ detach their encoder inputs, so the only gradient paths to the encoder come from $\mathcal{L}_{\text{pred}}$ and $\mathcal{L}_{\text{sigreg}}$, exactly as in LeWM. The IDM is not pretrained. Instead, it is updated jointly with the predictor in every step, but gradient-path separation ensures each loss updates only its intended parameters.

3. Experiments

3.1. Environments

FICReg is evaluated on four environments spanning navigation and manipulation tasks with varying action complexity (Figure 4 in the Appendix).

In **PushT**, an agent must push a T-shaped block to a target pose in a continuous 2D workspace. The agent's action directly determines how the block moves, making the action–state relationship highly causal. **TwoRoom** is a 2D navigation environment consisting of two connected rooms, where the agent must navigate to a goal position. In **Reacher**, a two-joint arm is controlled via joint torques to reach a target in a 2D plane. **OGBench-Cube** requires a robotic arm to manipulate a cube to a target position in a 3D environment.

3.2. Implementation

FICReg is added directly on top of the LeWM architecture, inheriting all baseline hyperparameters without modification; only the auxiliary loss weights (μ and γ) are introduced. This isolates the effect of FICReg from any confounding changes in training configuration. All experiments are run

Table 1. Success rates across environments (epoch 10). Mean ± Std over 3 seeds. Baseline (LeWM) values are from our re-run.

	TwoRoom	PushT	Reach.	OGB.
LeWM	87.3±1.9	88.7±2.5	78.7±3.4	**75.3**±2.5
FICReg	87.3±0.9	**94.0**±3.3	**81.3**±4.1	72.0±1.6

with 3 seeds. Success rates are reported at epoch 10 following the evaluation protocol of LeWM. Both methods are trained and evaluated using our re-run of the official LeWM code with identical hyperparameters.

3.3. Results and Analysis

Table 1 summarizes success rates across four environments. The results are discussed as preliminary observations in the context of FICReg's design motivation.

Action-causal tasks. On PushT, where actions directly determine object motion, FICReg achieves 94.0% compared to 88.7% for the baseline. This is consistent with our hypothesis: when the action–state-transition relationship is strongly causal, enforcing forward-inverse consistency provides a useful inductive bias for the predictor. On OGBench-Cube, FICReg obtains 72.0% compared to 75.3% for the baseline. This environment involves complex contact dynamics and a higher-dimensional action space, making it difficult for a simple MLP-based IDM to accurately capture the inverse dynamics. Consequently, when the frozen IDM provides an unreliable consistency signal, the regularizer may misguide the predictor rather than improving it. Investigating more expressive IDM architectures or environment-adaptive loss weighting remains a promising direction to address this limitation.

Navigation and simple control. On TwoRoom, both methods achieve 87.3%. On Reacher, FICReg obtains 81.3% versus 78.7%, a modest improvement. These results suggest that when the baseline already captures the action–state relationship adequately, the additional consistency signal provides limited but non-negative impact, which is consistent with the gradient-path separation design that isolates FICReg from the baseline learning dynamics.

Representation quality. To examine whether FICReg affects the learned representations beyond task performance, linear and MLP probes are trained to decode physical quantities (agent location, block location, block angle) from PushT latent states (Table 2), reporting Pearson correlation r. Linear probing shows improved decodability of agent location (r: 0.968 $\rightarrow$ 0.986) with reduced variance across seeds, while block location and angle remain comparable. MLP probes achieve near-perfect correlation for both methods. This suggests that FICReg does not alter the information content of the representations but encourages a latent geometry in which action-relevant state variables are more linearly accessible. The selective improvement on agent location, the variable most directly driven by actions, is consistent with this interpretation.

Table 2. Probing correlation (r) on PushT (mean ± std over 3 seeds).

Property	Probe	LeWM	FICReg
Agent Location	Linear	0.9679 ± 0.0069	$\mathbf{0.9863 \pm 0.0012}$
	MLP	0.9995 ± 0.0000	$\mathbf{0.9996 \pm 0.0000}$
Block Location	Linear	0.9845 ± 0.0041	$\mathbf{0.9867 \pm 0.0005}$
	MLP	0.9999 ± 0.0000	0.9999 ± 0.0000
Block Angle	Linear	$\mathbf{0.9036 \pm 0.0064}$	0.9020 ± 0.0077
	MLP	0.9971 ± 0.0000	$\mathbf{0.9971 \pm 0.0002}$

Learning curve. Figure 3 shows the full training trajectory on PushT over 15 epochs. FICReg consistently outperforms the baseline across most of the training, rather than only at the snapshot used for Table 1. This confirms that the performance gap reflects a sustained improvement in dynamics learning, not a favorable evaluation point.

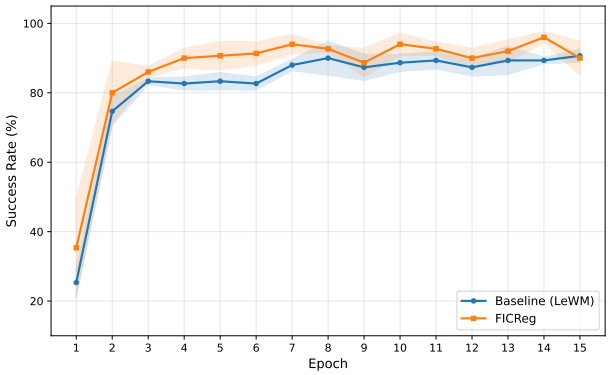


Figure 3. Learning curve of PushT.

4. Conclusion

FICReg repurposes the IDM from a representation learning objective into a frozen judge that directly regularizes the forward predictor. By training the IDM on ground-truth transitions and freezing it when evaluating predictions, with gradients flowing only into the predictor, FICReg improves the dynamics learned by the forward model without distorting the encoder or the IDM. Preliminary results show clear benefit on PushT, where actions are strongly causal, and linear probing suggests that FICReg encourages more linearly accessible action-relevant structure in the learned representations. Promising directions include more expressive IDM architectures to extend applicability to contact-rich environments, as well as ablations on the IDM freezing strategy and sensitivity analysis of the loss weights.

References

Assran, M., Duval, Q., Misra, I., Bojanowski, P., Vincent, P., Rabbat, M., LeCun, Y., and Ballas, N. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15619–15629, 2023.

Assran, M., Bardes, A., Fan, D., Garrido, Q., Howes, R., Muckley, M., Rizvi, A., Roberts, C., Sinha, K., Zholus, A., et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Balestriero, R. and LeCun, Y. Lejepa: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025.

Bardes, A., Garrido, Q., Ponce, J., Chen, X., Rabbat, M., LeCun, Y., Assran, M., and Ballas, N. V-jepa: Latent video prediction for visual representation learning. 2023.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

LeCun, Y. et al. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1): 1–62, 2022.

Maes, L., Lidec, Q. L., Haramati, D., Massaudi, N., Scieur, D., LeCun, Y., and Balestriero, R. stable-worldmodel-v1: Reproducible world modeling research and evaluation, 2026a. URL https://arxiv.org/abs/2602.08968.

Maes, L., Lidec, Q. L., Scieur, D., LeCun, Y., and Balestriero, R. Leworldmodel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026b.

Mur-Labadia, L., Muckley, M., Bar, A., Assran, M., Sinha, K., Rabbat, M., LeCun, Y., Ballas, N., and Bardes, A. V-jepa 2.1: Unlocking dense features in video self-supervised learning. *arXiv preprint arXiv:2603.14482*, 2026.

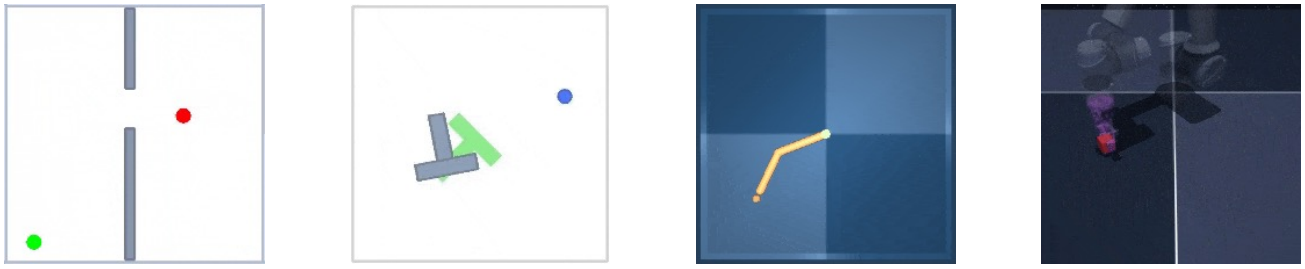
Nam, H., Lidec, Q. L., Maes, L., LeCun, Y., and Balestriero, R. Causal-jepa: Learning world models through object-level latent interventions. *arXiv preprint arXiv:2602.11389*, 2026.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Sobal, V., Zhang, W., Cho, K., Balestriero, R., Rudner, T. G. J., and LeCun, Y. Learning from reward-free offline data: A case for planning with latent dynamics models. *arXiv preprint arXiv:2502.14819*, 2025.

A. Environments

Figure 4 illustrates the four evaluation environments. The environments are selected to cover a range of action–state complexity: from simple 2D navigation (TwoRoom) to 3D robotic manipulation (OGBench-Cube). All environments have continuous action spaces and are evaluated in the offline, reward-free setting. Dataset details and collection procedures follow (Maes et al., 2026b).



(a) TwoRoom *(b)* PushT *(c)* Reacher *(d)* OGBench-Cube

Figure 4. Evaluation environments. (a) TwoRoom: 2D navigation between two connected rooms. (b) PushT: pushing a T-shaped block to a target pose. (c) Reacher: a two-joint arm reaching a target location. (d) OGBench-Cube: a robotic arm manipulating a cube to a target position.

B. Implementation and Evaluation Details

The implementation builds on the official LeWorldModel implementation using the `stable-worldmodel` (Maes et al., 2026a) framework. All baseline hyperparameters follow (Maes et al., 2026b) without modification; only the IDM loss weight μ and the forward-via-IDM loss weight γ are introduced.

Training. Observations are 224×224 pixels with a frame-skip of 5. Each batch contains 128 sub-trajectories of 4 frames. Training uses AdamW (lr $= 5 \times 10^{-5}$, weight decay $= 10^{-3}$) with gradient clipping at 1.0. The SIGReg weight is $\lambda = 0.09$. All parameters including the IDM are optimized with a single AdamW optimizer. The same offline datasets as LeWM are used.

FICReg weights. The IDM loss weight is $\mu = 0.1$ and the forward-via-IDM loss weight is $\gamma = 0.1$. These are preliminary values; further tuning may yield additional improvements.

Architecture. The encoder is a ViT-Tiny (patch size 14, embedding dimension 192). The predictor is a ViT-S with 6 layers, 16 heads, and MLP dimension 2048. History length is 3 for PushT, Reacher, and OGBench-Cube, and 1 for TwoRoom. The IDM is a 3-layer MLP (hidden dimension 512, GELU activations) that takes the concatenation of two state embeddings as input:

```
class InverseDynamicsModel(nn.Module):
    def __init__(self, emb_dim, action_dim, hidden_dim=512, dropout=0.0):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(2 * emb_dim, hidden_dim),
            nn.GELU(),
            nn.Dropout(dropout),
            nn.Linear(hidden_dim, hidden_dim),
            nn.GELU(),
            nn.Dropout(dropout),
            nn.Linear(hidden_dim, action_dim),
        )

    def forward(self, z_t, z_tp1):
        """
        z_t, z_tp1: (B, T, D)
        returns: (B, T, action_dim)
        """
        return self.net(torch.cat([z_t, z_tp1], dim=-1))
```

Evaluation. Following the LeWM protocol, each evaluation episode pairs a randomly chosen initial state with a goal state from the same offline trajectory, separated by a fixed number of timesteps. For TwoRoom, the budget is 150 steps with goals 100 timesteps ahead. For PushT, Reacher, and OGBench-Cube, the budget is 50 steps with goals 25 timesteps ahead. Each method is evaluated on the same set of 50 trajectories per seed. Planning uses CEM with MPC: 300 candidate sequences, 30 iterations for PushT and 10 for other environments, top-30 elite selection, and a planning horizon of 5 steps.

Probing. Linear and MLP probes are trained on frozen encoder representations to predict physical quantities from the latent embeddings. Linear probes test whether physical quantities are directly decodable from the latent space, while MLP probes test whether the information is encoded in a potentially nonlinear form. Pearson correlation r between predicted and ground-truth values is reported. On PushT, the probed quantities are agent location, block location, and block angle.