Latent-Space Planning with JEPA World Models

The Role of Action-Sensitivity and True-Position Evaluation in MiniGrid

Ankit Jha (IIT Patna)
Mentor: Dr. Dipasree Pal
Indian Statistical Institute, Kolkata

Abstract

World models based on Joint-Embedding Predictive Architectures (JEPA) promise sample-efficient, action-conditioned planning by predicting future representations rather than raw observations. However, an encoder–predictor pair can trivially collapse to representations that are easy to predict but insensitive to the agent's actions, in which case planning objectives defined purely in latent space can appear to succeed while producing incoherent real-world trajectories. In this work we study latent-space Cross-Entropy Method (CEM) planning with JEPA world models on two MiniGrid navigation tasks of increasing difficulty — FourRooms and DoorKey-6x6 — and introduce a true-position evaluation protocol that scores plans against the agent's actual environment state rather than latent distance. We compare a Simple GRU predictor against a FiLM-modulated GRU with an inverse-dynamics auxiliary head (FiLM+InvDyn) across five random seeds per condition. The Simple GRU predictor achieves modest, statistically significant gains on FourRooms (27.6% ± 15.3%, $p = 0.0224$) but fails on the harder DoorKey task (−4.4% ± 59.6%, $p = 0.8899$). Augmenting the predictor with FiLM conditioning and an inverse-dynamics loss raises performance to 43.4% ± 24.1% ($p = 0.0227$) on FourRooms and 77.2% ± 14.2% ($p = 0.0004$) on DoorKey. We trace this gap to an action-sensitivity ratio that improves from 0.11 in a historical, latent-space-evaluated checkpoint to 3.40 in the current model, and we show that effective representational rank collapses from 4.28 to 1.98 (out of 256 dimensions) between the easier and harder task. Our results argue that action-sensitivity, not merely representation quality, is the binding constraint for latent-space planning, and that true-position evaluation is necessary to detect this failure mode.

Keywords: *world models, self-supervised learning, JEPA, VICReg, latent-space planning, model predictive control, MiniGrid*

1 Introduction

Model-based reinforcement learning promises sample efficiency by allowing an agent to plan inside a learned model of the environment rather than acquiring experience purely through trial and error. A growing line of work builds such world models directly in a learned latent space using Joint-Embedding Predictive Architectures (JEPA), which train an encoder and a predictor to make future latent representations predictable from past representations and actions, without reconstructing raw pixels. This self-supervised formulation avoids wasting model capacity on irrelevant visual detail and has been argued to produce representations that are better suited to downstream planning and control.

A latent-space world model is only useful for planning, however, if its predicted latents actually track the consequences of the agent's actions. A predictor can satisfy a self-supervised objective such as VICReg by producing representations that are smooth and non-collapsed on average while remaining almost insensitive to which action was taken — a degenerate solution that is easy to reach because most self-supervised losses do not explicitly penalize action-invariance. When such a model is used inside a sampling-based planner like the Cross-Entropy Method (CEM), the planner can drive the latent-space objective to a favourable value while the corresponding sequence of actions in the real environment does nothing coherent at all. Because CEM planning is ordinarily scored by the same latent-space objective it optimizes, this failure mode can be invisible unless the evaluation is grounded in the environment's true state.

This paper reports a controlled empirical study of this issue. We train JEPA world models on two MiniGrid tasks — FourRooms, a purely navigational task, and DoorKey-6x6, which additionally requires picking up a key and opening a door — using two predictor architectures: (i) a Simple GRU predictor that concatenates an action embedding with the GRU hidden state, and (ii) a FiLM+InvDyn predictor that instead uses Feature-wise Linear Modulation (FiLM) to condition the GRU on the action and adds an inverse-dynamics auxiliary loss that predicts the action from consecutive latents. We evaluate all models with a true-position CEM planning test that scores plans by the agent's real position in the environment rather than latent distance, and we further quantify each predictor's action-sensitivity and the effective rank of its representation space.

Contributions.

- We introduce and motivate a true-position evaluation protocol for latent-space planning with JEPA world models, and present a case study showing that a checkpoint judged successful under a latent-space metric (+83.1% improvement) is in fact a near-total failure under true-position evaluation (−60.5%).
- We show, across five seeds per condition, that a Simple GRU predictor is statistically significant on FourRooms (27.6%, $p = 0.0224$) but statistically indistinguishable from no improvement on DoorKey (−4.4%, $p = 0.8899$).
- We show that a FiLM+InvDyn predictor recovers strong, statistically significant performance on both FourRooms (43.4%, $p = 0.0227$) and DoorKey (77.2%, $p = 0.0004$), and we connect this improvement to a nearly 30× increase in action-sensitivity ratio ($0.11 \rightarrow 3.40$).
- We report effective-rank analysis showing that the Simple GRU predictor's representation collapses from an effective rank of 4.28 on FourRooms to 1.98 on DoorKey (of 256 nominal dimensions), consistent with the harder task inducing a more degenerate, less action-sensitive latent space.

2 Related Work

Joint-Embedding Predictive Architectures.

JEPA-style self-supervised learning predicts the representation of one view or future state from another, rather than reconstructing raw inputs, and has been proposed as a path toward world models that are more robust to task-irrelevant detail (LeCun, 2022). Image-based instantiations such as I-JEPA (Assran

et al., 2023) and video-based extensions such as V-JEPA (Bardes et al., 2024) demonstrate that predicting in representation space yields transferable features without pixel-level reconstruction.

Self-supervised representation learning and collapse avoidance.

A central risk in joint-embedding methods is representational collapse, where the encoder maps all inputs to a near-constant output that trivially satisfies the training objective. VICReg (Bardes et al., 2022) addresses this directly with an explicit variance-preservation term alongside invariance and covariance-decorrelation terms, and is the loss family adopted in this work. Related approaches include BYOL (Grill et al., 2020) and Barlow Twins (Zbontar et al., 2021), which use architectural asymmetry or cross-correlation redundancy reduction, respectively, to the same end.

World models and latent-space planning.

Learning a compact latent dynamics model for planning and control has a long history, from recurrent state-space models used for pixel-based control (Ha & Schmidhuber, 2018) to Dreamer-style agents that plan entirely within a learned latent space (Hafner et al., 2020; Hafner et al., 2023). These works typically supervise the latent model with reconstruction or reward prediction; our setting instead relies purely on a self-supervised joint-embedding objective, which removes the reconstruction signal but reintroduces the risk of action-insensitive collapse that reconstruction-based losses implicitly discourage.

Sampling-based planning.

The Cross-Entropy Method (Rubinstein, 1997; de Boer et al., 2005) is a population-based optimization algorithm widely used for model-predictive control in learned world models due to its simplicity and robustness to non-convex, non-differentiable objectives (Chua et al., 2018). We use CEM as the planner throughout, and vary only the world model and the evaluation metric.

Feature-wise conditioning and inverse dynamics.

FiLM (Perez et al., 2018) conditions intermediate network activations on an auxiliary input via learned affine transformations, and has been used extensively to inject conditioning signals more expressively than simple concatenation. Predicting the action that produced a transition from its endpoints — an inverse-dynamics objective — has separately been used to encourage representations to retain controllable, action-relevant information (Pathak et al., 2017; Shelhamer et al., 2017). Our FiLM+InvDyn predictor combines both ideas specifically to counter action-insensitive collapse in a JEPA predictor.

Benchmarks.

MiniGrid (Chevalier-Boisvert et al., 2023) provides a family of lightweight, procedurally structured gridworld tasks, including FourRooms and DoorKey, that isolate navigation and compositional object-interaction sub-tasks and are widely used for evaluating sample-efficient RL and planning methods.

3 Methodology

3.1 JEPA World Model

Our world model consists of an online encoder E_θ, a target (EMA) encoder E_ξ, and an action-conditioned predictor P. At each timestep t, the observation o_t is encoded to a 256-dimensional latent

$z_t = E_\theta(o_t)$ by a 4-layer convolutional encoder with adaptive pooling. Given z_t and the action a_t taken at time t, the predictor produces the next-latent estimate $\hat{z}_{t+1} = P(z_t, a_t)$, which is trained to match the target encoder's representation of the true next observation, $z_{t+1} = E_\xi(o_{t+1})$. The target encoder's parameters are an exponential moving average (EMA) of the online encoder's parameters, with the EMA decay annealed from 0.996 to 0.9995 over training, following standard practice in joint-embedding self-supervised learning to prevent representational collapse.

$$z_t = E_\theta(o_t), \quad \hat{z}_{t+1} = P(z_t, a_t), \quad z_{t+1} = E_\xi(o_{t+1}) \tag{1}$$

$$\xi \leftarrow m\xi + (1 - m)\theta, \quad m : 0.996 \rightarrow 0.9995 \tag{2}$$

Here E_θ is the online (trainable) encoder with parameters θ, E_ξ is the EMA target encoder with parameters ξ, and P is the action-conditioned predictor. Equation (2) is the EMA update applied after every optimizer step: because the target only drifts slowly toward the online weights, it cannot be driven to a trivial constant by the predictor's own gradients — this is the mechanism that keeps the self-supervised target non-degenerate.

3.2 Predictor Architectures

We compare two predictor designs that share the same encoder and training data, isolating the effect of how the action is injected into the recurrent dynamics.

Simple GRU (final architecture).

The action a_t is embedded to a 32-dimensional vector and concatenated with the current latent z_t before being passed through a GRU cell with 256-dimensional hidden state to produce $\hat{z}_{t+1}$. This is the more parameter-efficient of the two designs and is the primary architecture evaluated in this study.

FiLM+InvDyn (ablation / comparison).

Rather than concatenating the action embedding, this variant uses it to compute per-channel scale and shift parameters (γ, β) via Feature-wise Linear Modulation, which modulate the GRU hidden state multiplicatively and additively at each step. In addition to the primary JEPA prediction loss, this variant is trained with an auxiliary inverse-dynamics loss that predicts a_t from the pair (z_t, z_{t+1}), encouraging the representation to retain information sufficient to distinguish actions from their effects.

3.3 Self-Supervised Objective

The encoder and predictor are trained jointly with a VICReg-style objective applied to the predicted and target embeddings, comprising an invariance (similarity) term that pulls $\hat{z}_{t+1}$ toward z_{t+1}, a variance term that keeps the standard deviation of each embedding dimension above a threshold across the batch to prevent collapse, and a covariance term that decorrelates different embedding dimensions to encourage the network to spread information across the representation. Writing $Z \in \mathbb{R}^{N \times d}$ for a batch of N target embeddings and $\hat{Z}$ for the corresponding batch of predicted embeddings ($d = 256$), the three VICReg terms are defined as follows.

Invariance (similarity) term.

$$L_{sim} = (1/N) \sum_{i=1..N} \|\hat{z}_{(i)t+1} - z_{(i)t+1}\|_2^2 \tag{3}$$

Equation (3) is the mean squared error between each predicted latent and its corresponding target latent; minimizing it is what makes the predictor's rollout track the encoder's future representation at all.

Variance term.

$$Lvar = (1/d)\ \Sigma j{=}1..d\ max(0,\ \gamma - S(z{\cdot},j)),\quad S(x) = \sqrt{(Var(x) + \varepsilon)} \tag{4}$$

where $z_{\cdot,j}$ is the j-th coordinate of the embedding taken across the batch, $\gamma = 1$ is the target standard deviation, and ε is a small constant added for numerical stability. Equation (4) is a hinge loss that penalizes any embedding dimension whose batch standard deviation falls below γ, which is precisely what stops the encoder–predictor pair from collapsing to a near-constant output.

Covariance term.

$$C(Z) = 1/(N-1)\ \Sigma i\ (z(i) - \bar{z})(z(i) - \bar{z})^{T},\quad Lcov = (1/d)\ \Sigma j{\neq}k\ [C(Z)]^{2}j,k \tag{5}$$

where $\bar{z}$ is the batch mean embedding. C(Z) is the empirical covariance matrix of the embedding dimensions and, in Equation (5), Lcov penalizes its off-diagonal entries, discouraging different dimensions from encoding redundant information. This is what encourages the network to spread information across many dimensions rather than a few, which is directly reflected later in the effective-rank diagnostic of Equation (14) in Section 3.6.

Combined objective.

$$L = \lambda sim\ Lsim + \lambda var\ Lvar + \lambda cov\ Lcov,\quad \lambda sim = 15,\ \lambda var = 40,\ \lambda cov = 5 \tag{6}$$

For the FiLM+InvDyn predictor, an additional inverse-dynamics cross-entropy term is added with weight 15, encouraging action-recoverability from consecutive latents; we refer to the value of this term reported in training logs as InvLoss.

Inverse-dynamics auxiliary loss (FiLM+InvDyn only).

$$\hat{p}(i) = softmax(f\phi([z(i)t\ ;\ z(i)t{+}1])),\quad Linv = -(1/N)\ \Sigma i\ \Sigma k\ y(i)k\ log\ \hat{p}(i)k \tag{7}$$

$$Ltotal = L + \lambda inv\ Linv,\quad \lambda inv = 15 \tag{8}$$

$f\phi$ is a small MLP classification head over the A = 7 discrete actions, [zt ; zt+1] denotes concatenation of consecutive latents, and y(i) is the one-hot true action. Equation (7) is a standard cross-entropy loss: it is minimized only if the action that produced a transition can be recovered from its endpoints, which directly penalizes the action-insensitive collapse this paper diagnoses via Equation (13) in Section 3.6 and reports empirically in Section 5.4. Equation (8) simply adds this auxiliary term to the base VICReg objective of Equation (6).

3.4 Latent-Space Planning with CEM

At evaluation time, the agent plans by sampling candidate action sequences of length equal to the planning horizon (6 steps) from a diagonal Gaussian over a relaxed action representation, rolling each sequence forward through the predictor to obtain a sequence of predicted latents, and scoring each candidate by the latent distance between the final predicted latent and the encoded goal representation. We use a continuous relaxation of the discrete action space: each action is represented as a 7-dimensional logit vector, and CEM samples from a diagonal Gaussian over these logits; the final action executed in the environment is obtained by applying softmax and sampling from the resulting categorical distribution. The Cross-Entropy Method iteratively re-fits the sampling distribution to the best-scoring (elite) candidates: we use 200 samples, 20 elites, and 5 CEM iterations per planning step,

and only the first action of the optimized sequence is executed before replanning (receding-horizon control).

Formally, at planning step t the agent optimizes an action sequence a0:H−1 of horizon H = 6 by rolling the predictor forward from the current latent and scoring the resulting trajectory against the encoded goal representation zg = Eθ(og):

$$\hat{z}_0 = z_t, \quad \hat{z}_{k+1} = P(\hat{z}_k, a_k) \ (k = 0, \ldots, H-1), \quad J(a_{0:H-1}) = -\|\hat{z}_H - z_g\|_2 \tag{9}$$

CEM maintains a diagonal Gaussian sampling distribution N(μ, Σ) over the action sequence and, at each of I = 5 iterations, draws M = 200 candidates, evaluates J on each via Equation (9), and refits the distribution to the K = 20 top-scoring elites $\mathscr{E}$:

$$a^{(m)}_{0:H-1} \sim N(\mu, \Sigma), \ m = 1, \ldots, M \qquad \mathscr{E} = \text{argtop}{-K}\, J(a^{(m)}_{0:H-1}) \tag{10}$$

$$\mu \leftarrow (1/K) \sum_{e \in \mathscr{E}} a_e, \qquad \Sigma \leftarrow (1/K) \sum_{e \in \mathscr{E}} \text{diag}[(a_e - \mu)(a_e - \mu)^T] \tag{11}$$

After the final iteration, only the first action of the resulting mean sequence, μ_0, is executed in the environment before the whole procedure (Equations 9–11) is repeated from the new state (receding-horizon control).

3.5 True-Position vs. Latent-Space Evaluation

A plan can minimize latent distance to the goal representation while the corresponding real trajectory makes no progress toward the goal, if the predictor's latents are not sufficiently sensitive to the conditioning action. To detect and avoid reporting this failure mode, we define our primary evaluation metric — true-position CEM improvement — as the percentage improvement in the agent's actual distance to the goal in environment coordinates, comparing the CEM-planned policy against a fixed baseline, averaged over 20 planning trials per seed. All results reported as “CEM improvement” in this paper use this true-position metric unless explicitly labelled otherwise; Section 5.3 presents a direct case-study comparison against the latent-space metric.

Concretely, let d(s) denote the Euclidean distance, in environment grid coordinates, between the agent's position at state s and the goal position. For trial r, let d(r)base and d(r)CEM be this distance at the end of the fixed-baseline rollout and the CEM-planned rollout respectively. The true-position CEM improvement for a seed is

$$\text{Impr} = (1/R) \sum_{r=1..R} [\ (d(r)_{\text{base}} - d(r)_{\text{CEM}}) / d(r)_{\text{base}}\ ] \times 100\%, \quad R = 20 \tag{12}$$

Equation (12) is defined purely in terms of the agent's real (x, y) position in the grid — never in terms of latent distance — which is exactly what allows it to catch the failure mode of Section 5.3, where a checkpoint minimizes latent distance to the goal while making no real progress toward it. The fixed baseline itself is a random action policy — the same uniformly random policy used for data collection in Section 4 — so d(r)base in Equation (12) is the agent's final distance to the goal after taking random actions for the episode, and Impr therefore measures how much of the gap between random behaviour and the goal the CEM-planned policy closes.

3.6 Action-Sensitivity and Effective Rank Diagnostics

To diagnose whether a predictor's failure under true-position evaluation stems from action-insensitive collapse, we compute an action-sensitivity ratio, defined as the average latent displacement induced by changing the conditioning action relative to the average latent displacement induced by holding the action fixed and re-sampling only stochastic factors; a ratio near zero indicates the predictor's output is nearly independent of the action. We additionally compute the effective rank of the predictor's latent representation — the exponential of the entropy of the normalized singular value spectrum of a batch of predicted embeddings — as a scalar summary of how many of the 256 nominal latent dimensions are actually used.

Action-sensitivity ratio.

$$S = \square_{z,a \neq a'}[\|P(z,a) - P(z,a')\|_2] \;/\; \square_{z,a}[\|P(z,a) - P'(z,a)\|_2] \tag{13}$$

The numerator of Equation (13) averages the latent displacement produced by changing only the conditioning action a on a fixed z (where a ≠ a′); the denominator averages the latent displacement between two forward passes with the same z and the same action a but resampled stochastic factors (e.g. dropout), denoted by P and P′, i.e. pure noise. S near zero means the predictor's output does not depend on which action was taken — the numerator and denominator are of the same order — while S ≫ 1 means the action signal dominates incidental noise, which is the regime a planner needs to be in for latent distance to be a meaningful planning objective.

Effective rank.

$$Z = U\Sigma V^T, \quad \hat{p}i = \sigma i \,/\, \Sigma k{=}1..r\ \sigma k, \quad erank(Z) = exp(-\ \Sigma i{=}1..r\ \hat{p}i \log \hat{p}i) \tag{14}$$

where σ1,…,σr are the singular values of a batch of predicted embeddings Z (r = min(N, d)). Equation (14) normalizes the singular values into a distribution p̂ and exponentiates their Shannon entropy, giving a single number between 1 (all variance on one axis) and r (variance spread equally over every axis) — a soft, continuous count of how many of the 256 nominal latent dimensions the predictor is actually using, which is why the values of 4.28 and 1.98 reported in Table 5 are so small relative to 256.

4 Experimental Setup

Environments.

We evaluate on two tasks from the MiniGrid suite (Chevalier-Boisvert et al., 2023): MiniGrid-FourRooms-v0, a purely navigational task requiring the agent to traverse four connected rooms to reach a goal, and MiniGrid-DoorKey-6x6-v0, a compositional task in which the agent must locate a key, pick it up, unlock a door, and then reach the goal — introducing a long-horizon dependency absent from FourRooms.

Data collection.

For each environment, we collect 400 rollouts using a random policy, with a training context/prediction horizon of 4 steps and a maximum of 128 steps per episode, used to construct the (ot, at, ot+1) transition tuples on which the JEPA objective is trained.

Training.

Each (environment, predictor) condition is trained for 15,000 steps with the AdamW optimizer (learning rate 3×10^{-4}, weight decay 1×10^{-6}), batch size 128, using the VICReg objective of Section 3.3 (with the added inverse-dynamics term for FiLM+InvDyn). Every condition is repeated over 5 random seeds to characterize variance across initialization and data ordering.

Planning evaluation.

Trained models are evaluated with the true-position CEM planning test described in Section 3.5, using a planning horizon of 6, 200 CEM samples, 20 elites, 5 CEM iterations, and 20 independent planning trials per seed. Table 1 summarizes the full experimental configuration.

Table 1: Experimental configuration.

Hyperparameter / Setting	Value
Environments	MiniGrid-FourRooms-v0, MiniGrid-DoorKey-6x6-v0
Encoder	CNN (4 conv layers + adaptive pooling) → 256-d embedding
Action embedding dim.	32
Predictor hidden size	256
Number of actions	7
Context / prediction horizon (training)	4
Rollouts collected	400 (random policy)
Batch size	128
Optimizer	AdamW, lr = 3×10^{-4}, weight decay = 1×10^{-6}
EMA target schedule	0.996 → 0.9995
Loss weights (VICReg)	λsim = 15, λvar = 40, λcov = 5
Auxiliary loss weight (InvDyn / contrastive)	15
Training steps	15,000
Max steps / episode	128
Planning horizon (CEM)	6
CEM samples / elites / iterations	200 / 20 / 5
Evaluation trials per seed	20
Random seeds per condition	5

Statistical testing.

For each of the four (environment, predictor) conditions, we report the mean and standard deviation of true-position CEM improvement across the 5 seeds and test whether the mean improvement is significantly different from zero using a one-sample t-test (4 degrees of freedom); we consider $p < 0.05$ significant.

5 Results

5.1 Main Results

Table 2 reports the statistical summary across all four (environment, predictor) conditions. The Simple GRU predictor achieves a significant improvement on FourRooms (27.6% ± 15.3%, t = 3.616, p = 0.0224) but is not distinguishable from zero improvement on DoorKey (−4.4% ± 59.6%, t = −0.147, p = 0.8899), with a standard deviation exceeding the magnitude of the mean itself. The FiLM+InvDyn predictor is significant on both environments, more than doubling the FourRooms improvement relative to Simple GRU (43.4% ± 24.1%, p = 0.0227) and turning the DoorKey result from a non-significant near-zero effect into the strongest result in the study (77.2% ± 14.2%, t = 10.904, p = 0.0004).

Table 2: True-position CEM improvement by environment and predictor architecture, aggregated over 5 seeds.

Environment	Predictor	Mean (%)	Std (%)	t	p-value	Status
FourRooms	Simple GRU	27.6	15.3	3.616	0.0224	Significant
DoorKey	Simple GRU	−4.4	59.6	−0.147	0.8899	Not significant
FourRooms	FiLM+InvDyn	43.4	24.1	3.601	0.0227	Significant
DoorKey	FiLM+InvDyn	77.2	14.2	10.904	0.0004	Significant

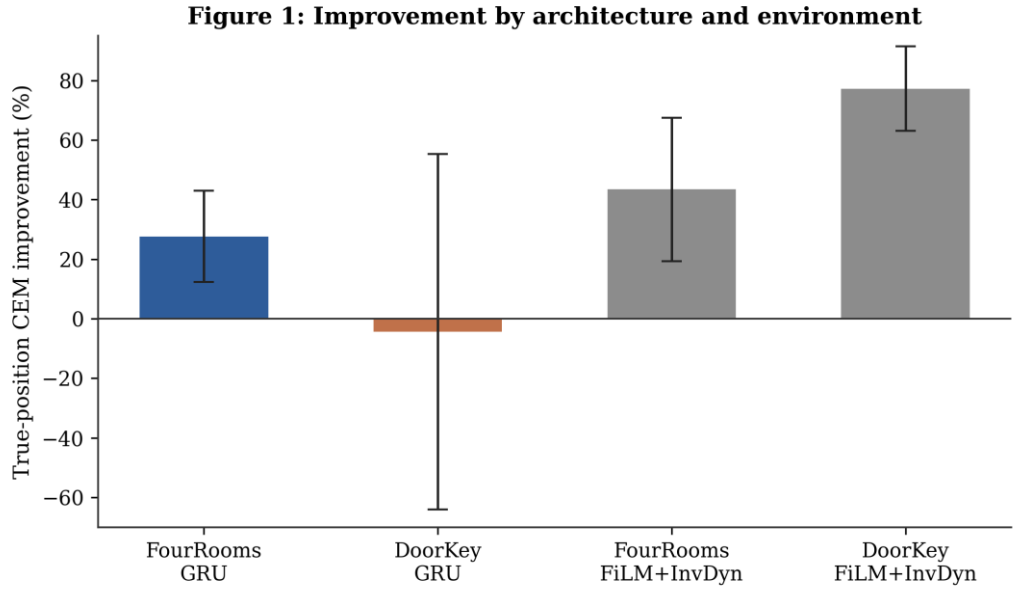

Figure 1: True-position CEM improvement (%) by architecture and environment, mean ± std over 5 seeds.

Figure 1 visualizes these four conditions together with error bars (± 1 standard deviation), making clear both the ordering FiLM+InvDyn > Simple GRU within each environment and the substantially larger variance of the Simple GRU predictor on DoorKey.

5.2 Seed-Level Variability

Table 3 and Figures 2–3 report per-seed results for the Simple GRU predictor. On FourRooms, all five seeds achieve a positive improvement (range 8.4% to 51.0%), consistent with the significant, moderate-variance result in Table 2. On DoorKey, by contrast, three seeds are strongly positive (29.0%, 43.4%,

36.5%) while two seeds are negative, including one severe failure at −116.7% — i.e., the planned policy performs far worse than the fixed baseline — illustrating that the Simple GRU predictor's behaviour on the harder task is not merely noisy but bimodal, alternating between reasonable planning and outright divergence.

Table 3: Seed-wise true-position CEM improvement (%).

Condition	Seed 0	Seed 1	Seed 2	Seed 3	Seed 4	Mean ± Std
FourRooms + Simple GRU	51.0	34.4	13.7	30.2	8.4	27.6 ± 15.3
DoorKey + Simple GRU	29.0	−116.7	43.4	−14.1	36.5	−4.4 ± 59.6
FourRooms + FiLM+InvDyn	3.9	59.5	35.4	75.3	42.7	43.4 ± 24.1
DoorKey + FiLM+InvDyn	57.5	70.0	100.0	75.0	83.3	77.2 ± 14.2

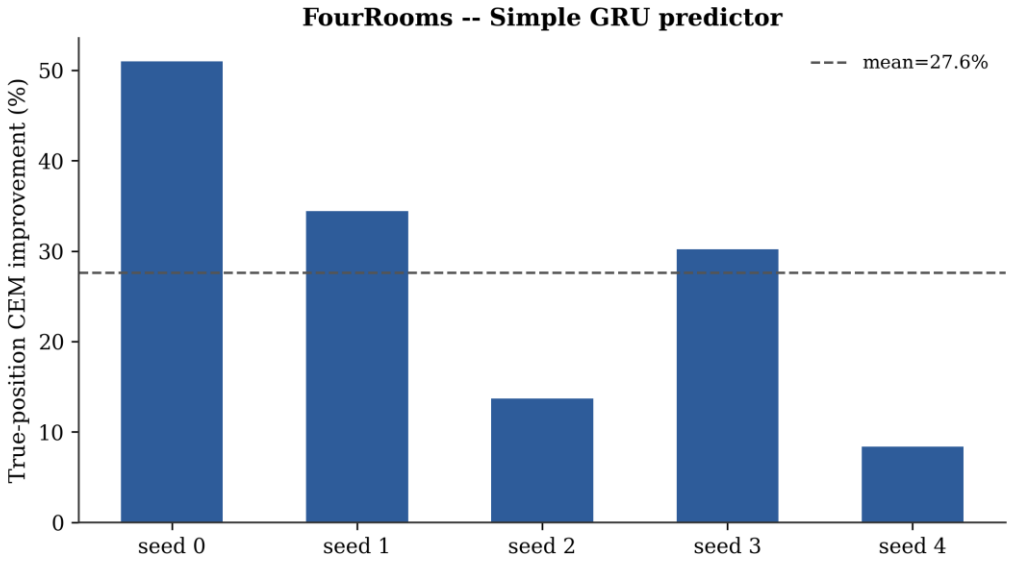

Figure 2: FourRooms, Simple GRU predictor — per-seed true-position CEM improvement.

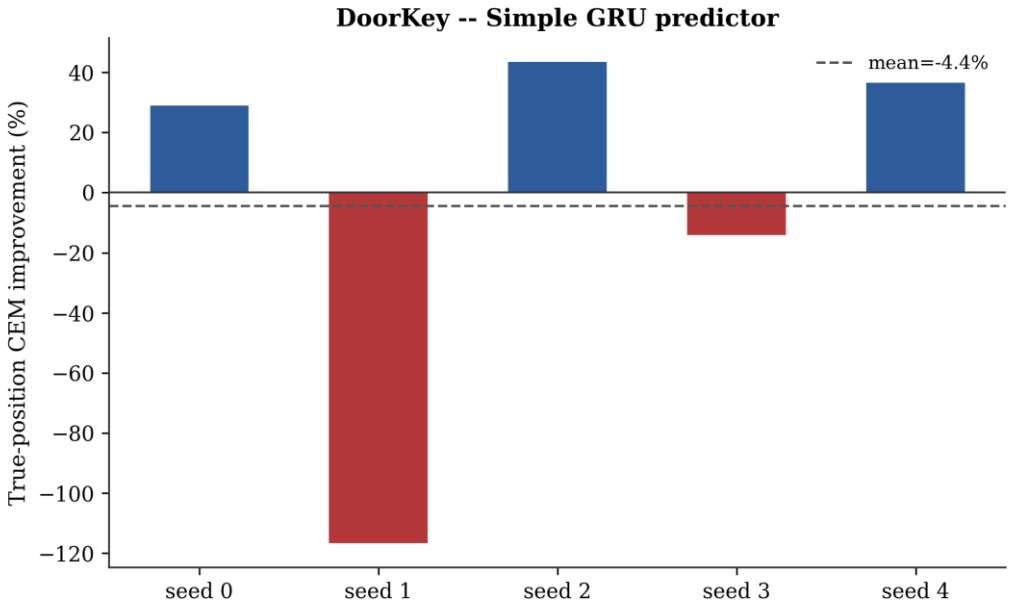

Figure 3: DoorKey, Simple GRU predictor — per-seed true-position CEM improvement.

5.3 Case Study: Latent-Space vs. True-Position Evaluation

To directly illustrate why latent-distance is an unreliable planning-success metric, Table 4 compares a historical, pre-fix checkpoint evaluated under both metrics. Under the latent-space metric, this checkpoint appears highly successful, with a planned-policy improvement of +83.1% over baseline. Under true-position evaluation, the same checkpoint's plans are revealed to make the agent's real-world outcome −60.5% worse than baseline — a directly contradictory conclusion. This checkpoint's action-sensitivity ratio was measured at 0.11, indicating that its predicted latents are nearly invariant to the conditioning action; CEM was therefore free to minimize latent distance to the goal by an essentially arbitrary action sequence. The current, fixed model achieves an action-sensitivity ratio of 3.40, roughly a 30× increase, and its true-position results (Table 2) are consistent with genuine planning improvement rather than metric exploitation.

Table 4: Latent-space vs. true-position evaluation on a historical pre-fix checkpoint, and current action-sensitivity for comparison.

Checkpoint	Latent-space CEM impr. (%)	True-position CEM impr. (%)	Action-sensitivity ratio
Historical (pre-fix)	+83.1	−60.5	0.11
Current (fixed) — reported in Table 2	—	—	3.40

This case study motivates our use of true-position evaluation as the sole reported metric throughout Sections 5.1–5.2, and more broadly suggests that any latent-space planning result should be accompanied by an environment-grounded sanity check before being interpreted as evidence of successful planning.

5.4 Effective Rank Analysis

Table 5 reports the effective rank of the Simple GRU predictor's representation on each environment. On FourRooms, the effective rank is 4.28 ± 0.36 out of a nominal 256 dimensions; on DoorKey it falls further to 1.98 ± 0.23. In both cases the model uses a strikingly small number of effective dimensions

relative to the embedding size, and the harder DoorKey task — where Simple GRU also fails under true-position evaluation — exhibits the more severe collapse, consistent with a link between representational collapse, low action-sensitivity, and planning failure.

Table 5: Effective rank of the Simple GRU predictor's latent representation.

Environment	Predictor	Effective rank (of 256)
FourRooms	Simple GRU	4.28 ± 0.36
DoorKey	Simple GRU	1.98 ± 0.23

5.5 Training Dynamics (DoorKey, FiLM+InvDyn)

For the best-performing condition (DoorKey, FiLM+InvDyn), training loss decreases steadily across all five seeds, from an initial range of roughly 28–46 at step 3,000 to 15.8–34.8 at step 12,000, while the inverse-dynamics auxiliary loss (InvLoss) remains stable in a narrow band around 0.94–1.06 throughout training, indicating that the action-recoverability signal is maintained rather than traded off against the primary JEPA loss as training proceeds. Notably, final training loss is not monotonically predictive of final planning accuracy: seed 2 achieves both the lowest final loss (15.843) and the highest planning accuracy (100.0%), but seed 0, despite a middling final loss (31.453), still achieves a respectable 57.5%, suggesting that planning success depends on qualitative properties of the learned representation (e.g., action-sensitivity) beyond the scalar loss value alone.

6 Discussion

6.1 Action-Sensitivity Is the Binding Constraint

Taken together, our results support a specific mechanistic account of when latent-space planning with a JEPA world model succeeds or fails, and Equations (1)–(14) give that account a precise vocabulary. The Simple GRU predictor injects the action only by concatenation before a single GRU update; nothing in its training objective (Equation 6) directly penalizes a solution where this concatenated action embedding is effectively ignored, since Lsim, Lvar and Lcov are all satisfiable by a predictor that is smooth and non-collapsed in aggregate while barely depending on at. On FourRooms, a task whose dynamics are dominated by short, locally-consequential movements, this vulnerability is apparently not exploited by the optimizer and the predictor remains usable. On DoorKey, a task with longer-horizon, more heterogeneous action consequences (pick up, unlock, navigate), the same predictor's representation instead collapses toward a low effective rank (1.98/256, Equation 14) and low action-sensitivity (implicitly $S \approx 0$ in the sense of Equation 13), and true-position planning (Equation 12) fails outright. The FiLM+InvDyn predictor addresses this on two complementary fronts: FiLM conditioning gives the action a more expressive, per-channel influence on the predicted representation than concatenation, while the inverse-dynamics term Linv (Equation 7) is, unlike Lsim, Lvar, and Lcov, a loss that is only minimized when at is recoverable from (zt, zt+1) — it directly rewards the action-sensitivity that the base VICReg objective leaves unconstrained. Consistent with this account, the combination raises the DoorKey result from a non-significant −4.4% to a highly significant 77.2%, and improves FourRooms as well, and the measured action-sensitivity ratio rises roughly 30× (0.11 → 3.40, Section 5.3) alongside it. This co-movement of a training-time diagnostic (Equation 13) with a downstream planning outcome is what lets us call the fix a fix, rather than an unexplained correlation,

and suggests the underlying vulnerability — an objective that never explicitly rewards action-sensitivity — is a general property of action-conditioned JEPA predictors rather than a quirk of DoorKey specifically.

6.2 Latent-Distance Metrics Can Be Misleading

The case study in Section 5.3 is, in our view, the paper's most important cautionary finding: a checkpoint that appears to succeed by +83.1% under a latent-space evaluation is a −60.5% failure under Equation (12)'s true-position evaluation. The mechanism is visible directly in Equation (9): the planner's objective J is defined entirely in terms of $\hat{z}H$, the predictor's own rollout, compared against a goal latent zg — there is no term anywhere in J that references the agent's actual (x, y) position. When the predictor is action-insensitive ($S \approx 0.11$ for this checkpoint, far below the $S = 3.40$ of the current model), CEM can still drive J toward zero, because $\hat{z}H$ depends only weakly on which actions were actually chosen; it is minimizing a quantity that has been almost entirely disconnected from the environment. Because the CEM planner in the latent-space setting is optimizing and being evaluated with the same objective, an action-insensitive predictor creates a self-consistent but vacuous success signal, and the evaluation cannot detect the failure because it is blind to precisely the information the predictor is blind to. This is not a hypothetical risk specific to our setup: any planner that is scored by the same latent objective it optimizes is structurally unable to distinguish genuine progress from exploitation of an under-constrained predictor. We therefore argue that an environment-grounded evaluation metric such as Equation (12), accompanied by an action-sensitivity diagnostic such as Equation (13), should be treated as close to mandatory whenever latent-space planning results are reported — not merely as a nice-to-have robustness check.

6.3 Limitations

Several limitations bound the scope of our conclusions. First, our study covers two MiniGrid environments and two predictor architectures; while the contrast between them is informative, both are comparatively small-scale gridworld tasks and it remains to be seen whether the same action-insensitivity failure mode and FiLM+InvDyn remedy generalize to higher-dimensional, continuous-control, or pixel-realistic domains. Second, per-seed variance is large, particularly for Simple GRU on DoorKey (std = 59.6%, larger in magnitude than the mean), so individual-seed outcomes should be interpreted with caution even though the aggregate significance test is informative; with only 5 seeds per condition, the one-sample t-test of Section 4 also has limited power, and its normality assumption is not separately verified. Third, our inverse-dynamics auxiliary loss and FiLM conditioning are introduced jointly as a single ablation condition (FiLM+InvDyn) rather than independently, so we cannot yet attribute the observed improvement to either component in isolation; a factorial ablation isolating FiLM-only and InvDyn-only variants is a natural next step. Fourth, the action-sensitivity ratio of Equation (13) and the effective rank of Equation (14) are themselves estimated from finite batches, so their absolute values (though not, we expect, the order-of-magnitude contrasts we report) carry some estimation noise, and neither diagnostic is, by itself, guaranteed to be a sufficient condition for good planning; they are best interpreted as necessary correlates observed alongside, rather than proofs of, the causal mechanism proposed in Section 6.1. Finally, the CEM hyperparameters of Equations (9)–(11) — horizon, sample count, elite count, and iteration count — were fixed across both environments and predictors rather than tuned per condition, so some of the residual DoorKey variance for Simple GRU may reflect a planner configuration that is better matched to FourRooms than to the harder task.

7 Conclusion and Future Work

We studied latent-space CEM planning with JEPA world models on two MiniGrid tasks and showed that a Simple GRU predictor, while adequate on the easier FourRooms task (27.6%, p = 0.0224), fails outright on the harder DoorKey task (−4.4%, p = 0.8899). We showed that this failure coincides with a collapse in effective representational rank (1.98/256) and can be substantially remedied by a FiLM+InvDyn predictor that reaches 77.2% (p = 0.0004) on DoorKey and 43.4% (p = 0.0227) on FourRooms, alongside a roughly 30× increase in action-sensitivity ratio (0.11 → 3.40). A direct case study further showed that latent-space evaluation can report a large positive improvement (+83.1%) for a checkpoint that true-position evaluation reveals to be a severe failure (−60.5%), underscoring the necessity of environment-grounded evaluation for latent-space planning research.

Future work includes:

(i) a factorial ablation separating the contributions of FiLM conditioning and the inverse-dynamics loss;

(ii) extending the true-position evaluation protocol and action-sensitivity diagnostic to continuous-control and higher-dimensional visual domains;

(iii) exploring whether action-sensitivity can be directly regularized during training, rather than encouraged only indirectly via an auxiliary loss, to reduce the DoorKey seed-to-seed variance observed with the Simple GRU predictor; and

(iv) studying whether effective rank and action-sensitivity are predictive of planning success early in training, which would allow these diagnostics to be used for early stopping or model selection without requiring full downstream planning evaluation.

Acknowledgements

We thank the reviewers of this internal report for their comments. All experiments were run on GPU resources provided via Google Colab and Kaggle; total compute for the reported experiments was approximately 50 GPU-hours.

References

[1] Assran, M., Duval, Q., Misra, I., Bojanowski, P., Vincent, P., Rabbat, M., LeCun, Y., & Ballas, N. (2023). Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture. CVPR.

[2] Bardes, A., Ponce, J., & LeCun, Y. (2022). VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning. ICLR.

[3] Bardes, A., Garrido, Q., Ponce, J., Chen, X., Rabbat, M., LeCun, Y., Assran, M., & Ballas, N. (2024). Revisiting Feature Prediction for Learning Visual Representations from Video (V-JEPA). arXiv preprint.

[4] Chevalier-Boisvert, M., Dai, B., Towers, M., Perez-Vicente, R., Willems, L., Lahlou, S., Pal, S., Castro, P. S., & Terry, J. (2023). Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks. NeurIPS Datasets and Benchmarks Track.

[5] Chua, K., Calandra, R., McAllister, R., & Levine, S. (2018). Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. NeurIPS.

[6] de Boer, P.-T., Kroese, D. P., Mannor, S., & Rubinstein, R. Y. (2005). A Tutorial on the Cross-Entropy Method. Annals of Operations Research, 134(1), 19–67.

[7] Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z., Azar, M. G., et al. (2020). Bootstrap Your Own Latent (BYOL): A New Approach to Self-Supervised Learning. NeurIPS.

[8] Ha, D., & Schmidhuber, J. (2018). World Models. arXiv preprint / NeurIPS workshop.

[9] Hafner, D., Lillicrap, T., Norouzi, M., & Ba, J. (2020). Mastering Atari with Discrete World Models (DreamerV2). ICLR.

[10] Hafner, D., Pasukonis, J., Ba, J., & Lillicrap, T. (2023). Mastering Diverse Domains through World Models (DreamerV3). arXiv preprint.

[11] LeCun, Y. (2022). A Path Towards Autonomous Machine Intelligence. OpenReview preprint.

[12] Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017). Curiosity-Driven Exploration by Self-Supervised Prediction. ICML.

[13] Perez, E., Strub, F., de Vries, H., Dumoulin, V., & Courville, A. (2018). FiLM: Visual Reasoning with a General Conditioning Layer. AAAI.

[14] Rubinstein, R. Y. (1997). Optimization of Computer Simulation Models with Rare Events. European Journal of Operational Research, 99(1), 89–112.

[15] Shelhamer, E., Mahmoudieh, P., Argus, M., & Darrell, T. (2017). Loss is its own Reward: Self-Supervision for Reinforcement Learning. ICLR Workshop.

[16] Zbontar, J., Jing, L., Misra, I., LeCun, Y., & Deny, S. (2021). Barlow Twins: Self-Supervised Learning via Redundancy Reduction. ICML.