RETHINKING THE PREDICTIVE LOSS IN LATENT WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent world models rely on predictive penalties to learn representations, meaning the choice of loss function dictates how prediction errors shape the latent space. The derivative of standard squared-error loss grows linearly with the error, amplifying the influence of large prediction errors. To address this, we investigate the element-wise log-cosh loss, which behaves quadratically for small errors but smoothly bounds the derivative for large ones. Through residual-space analysis on models with explicit representation regularization, we show that this attenuation improves local gradient interaction where it reduces gradient conflict more than cooperation. We applied this training-time loss replacement across multiple latent world model families without changing their underlying architectures or inference methods. On navigation, manipulation, and continuous control benchmarks, log-cosh consistently improved planning success over squared error and outperformed a locally curvature-matched quadratic control. On real-world driving data NAVSIM, log-cosh also improved perception and planning. We further study the learned representations through physical probing and action-space analyses. We observe that log-cosh makes task-relevant state information more accessible, and action-space analyses reveal closer alignment with the true-system cost landscape. These findings establish the predictive penalty as a fundamental component of representation design.

1 INTRODUCTION

Latent World Models. Latent world models encode high-dimensional observations into latent states and learn predictive relationships in the resulting representation space (LeCun, 2022; Assran et al., 2023; Garrido et al., 2024; Bardes et al., 2024; Assran et al., 2025; Mur-Labadia et al., 2026). Their usefulness depends on the information retained by that space and on how its geometry supports downstream tasks. In goal-conditioned planning, a predictor simulates future states under various actions, and a planner selects actions whose predicted outcomes are close to the goal (Pertsch et al., 2020; Wang et al., 2023; Zhang et al., 2026). Therefore, a useful world model must provide both accurate predictions and a spatial geometry that guides action search toward the desired outcome. Existing approaches learn these representations in various ways: from adapting pretrained features to end-to-end encoder training (Zhou et al., 2024; Destrade et al., 2025; Balestriero & LeCun, 2025). Some additionally impose explicit structure on the latent space through representation objectives (Wang et al., 2026; Maes et al., 2026; Kuang et al., 2026). These choices determine the space in which prediction errors are measured and where the model is subsequently used.

Learning with Predictive Penalties. Predictive models learn by penalizing discrepancies between predicted and target latent states (Doerr et al., 2018; Henaff et al., 2019; LeCun, 2022). Under squared-loss formulations, the derivative with respect to the residual grows linearly, giving larger errors proportionally greater influence (Efron, 1991; Chen & Paschalidis, 2018; Hu et al., 2025). This weighting dictates how individual errors update both the predictor and the underlying representation. Unlike regression to fixed labels, the targets here are not fixed; the model is updating the very representation that generates them (Gidaris & Komodakis, 2017; Müller et al., 2019). As a result, a small prediction error does not guarantee a useful model; it can occur when the representation simply discards task-relevant details: with total representation collapse being the extreme example. Thus, the predictive penalty directly shapes the learned latent space, even when the model architecture and the data remain unchanged.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Prediction and Representation Shaping. This raises a broader question: *How should prediction errors be weighted to support representation learning?* Explicit representation objectives provide a concrete setting to examine it. For example, Temporal Straightening (TS) enforces straighter trajectories (Wang et al., 2026), while LeWorldModel (LeMW) uses anti-collapse regularization (Maes et al., 2026). Both the predictive penalty and these structural objectives update shared parameters, meaning their gradients can either cooperate or conflict. By changing how individual prediction errors are weighted, we can alter the overall direction and magnitude of the predictive gradient. Under squared error, large residuals can severely amplify conflicting updates. While attenuating these large errors reduces this interference, it also weakens cooperative updates. Therefore, attenuation improves this local gradient interaction when it removes more conflict than the cooperation it sacrifices.

Our Approach and Contributions. Motivated by this gradient interaction, we investigate log-cosh as the training-time predictive penalty. Because log-cosh behaves quadratically near zero and has a derivative that smoothly saturates for large residuals, it provides a smooth, magnitude-dependent attenuation of residual contributions. We derive a residual-space decomposition that identifies the exact conditions under which this attenuation improves local gradient interaction with an explicit representation objective. We also compare log-cosh against a locally curvature-matched penalty $\frac{1}{2}r^2$, testing whether merely halving the squared error reproduces the benefits.

Figure 1 illustrates how quadratic and log-cosh penalties weight cooperative and conflicting residual contributions. Log-cosh attenuates both types at large residuals, so the resulting interaction depends on the balance between removed conflict and lost cooperation.

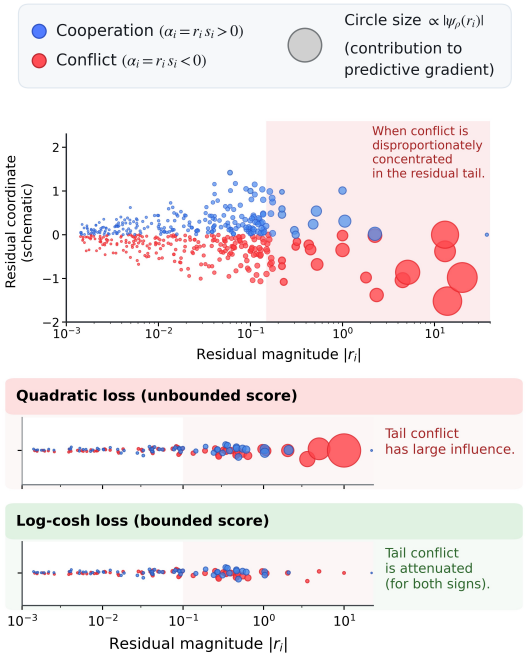


Figure 1: **Predictive penalties re-weight cooperation and conflict.** Schematic comparison of quadratic and log-cosh weighting. Blue and red denote cooperative and conflicting residual contributions. Log-cosh attenuates both types at large residual magnitudes; the resulting gradient interaction depends on the balance between removed conflict and lost cooperation.

We evaluate this loss replacement across multiple latent world model families, including V-JEPA-based models. Across navigation, manipulation, and continuous control benchmarks, log-cosh consistently matches or improves planning success over squared error, with gains holding even against the matched quadratic control. In these planning comparisons, inference-time procedures remain unchanged. We further evaluate this replacement on real-world driving data NAVSIM Dauner et al. (2024), where log-cosh improves both perception and planning. In both settings, it outperforms squared error and the locally curvature-matched quadratic, showing that simply halving squared error does not reproduce the observed gains. Furthermore, we examine task-relevant information through physical-state probing and study planning-cost landscapes through action-space analyses. Together, these results link the choice of predictive penalty to the quality of the learned representation and downstream perception and planning performance.

We summarize our contributions as follows:

- We re-frame predictive penalties as a core component of representation design, proposing log-cosh as a training-time replacement across latent world models.
- We characterize the exact theoretical conditions under which residual attenuation improves local gradient interaction with explicit representation objectives.
- We validate this replacement across multiple model families, demonstrating improvements through planning benchmarks (including real-world driving), controlled quadratic baselines, physical-state probing, and action-space analyses.

2 Related Work

Predictive Representations and Planning. Joint-Embedding Predictive Architectures (JEPA) instantiate energy-based learning by measuring the compatibility between observations through prediction in representation space (LeCun et al., 2006; LeCun, 2022; Assran et al., 2023). While originally developed for self-supervised image (I-JEPA) and video (V-JEPA) representation learning (Assran et al., 2023; Bardes et al., 2024), JEPA extends naturally to planning: action-conditioned latent world models use learned dynamics to predict future representations and evaluate candidate action sequences (Zhou et al., 2025; Sobal et al., 2025; Assran et al., 2025). Essentially, energy-based learning distinguishes the compatibility function optimized during inference from the objective used during training (LeCun et al., 2006). In our setting, this means that even when the inference-time planning cost retains a fixed quadratic form, its optimization landscape can still change depending on the predictive penalty used to train the representation. Because a predictive penalty can reshape a representation only through trainable representation parameters, the architectural distinction between predictor-only training on frozen features (e.g., DINO-WM (Zhou et al., 2025)) and joint encoder–predictor training (e.g., LeWM (Maes et al., 2026)) is central to our study. We specifically target this shared parameter block rather than attributing representation shifts to predictor-only training.

Constraints on the Representation. Jointly learning embeddings by matching predictions admits collapsed solutions unless additional structural constraints prevent them. To avoid collapse, VICReg enforces explicit variance and covariance constraints (Bardes et al., 2022), while LeWM employs SIGReg to encourage an isotropic Gaussian embedding distribution (Maes et al., 2026). Beyond distributional anti-collapse, other approaches impose dynamical or structural priors: TS constrains temporal geometry by penalizing trajectory curvature (Wang et al., 2026), and LpWM investigates sparse representations and their implications for modeling controlled dynamics (Kuang et al., 2026). Collectively, these works establish the importance of explicit representation regularizers. Rather than proposing a new regularizer, we examine a complementary design dimension: how the choice of predictive penalty interacts with a fixed representation objective through shared training parameters. Because objectives such as distributional anti-collapse and temporal straightness serve fundamentally different roles, we do not treat raw progress on either regularizer as a universal proxy for representation quality, focusing instead on their downstream effect on planning and latent geometry.

Robust Penalties and Gradient Interaction. Quadratic-to-linear penalties and bounded residual derivatives are well established in robust estimation (Huber, 1992; Charbonnier et al., 1997; Barron, 2019), and penalties with bounded derivatives such as ℓ_1 regression have already been shown to improve training stability in predictive architectures like V-JEPA (Bardes et al., 2024). When multiple objectives share parameters, methods such as PCGrad address optimization tension directly by inspecting task-gradient angles to project away conflict (Yu et al., 2020). Here, rather than modifying the optimizer, we investigate a more targeted question at the loss level: *when does a smooth, magnitude-dependent penalty make prediction more compatible with an explicit representation regularizer, and how should that claim be rigorously tested in latent planning?*

3 Predictive Penalties Shape Representation Learning

By framing predictive penalties as residual weighting rules, we analyze how they shape representation learning when optimized jointly with explicit structural objectives.

3.1 Predictive Penalties in Representation Learning

Let $\hat{Z}$ and Z^+ denote the predicted and target latents after any native pre-loss normalization, and let $r = \mathrm{vec}[\hat{Z} - Z^+] \in \mathbb{R}^m$ be the prediction residuals. For models with an explicit representation objective, we consider a uniformly weighted predictive penalty combined with a structural regularizer:

$$\mathcal{L}_\rho = \underbrace{c \sum_{i=1}^{m} \rho(r_i)}_{\mathcal{P}_\rho} + \underbrace{\lambda \mathcal{R}}_{\mathcal{G}}, \tag{1}$$

where ρ is a scalar penalty applied element-wise to the residuals and $c > 0$ is a parameter-independent reduction factor. The weighted objective $\mathcal{G}$ imposes a structural preference on the representation,

such as trajectory straightening in Temporal Straightening (Wang et al., 2026) or anti-collapse regularization in LeWorldModel (Maes et al., 2026). We study the role of ρ with the same $\mathcal{R}$, c, and λ within each architecture; any training schedules are shared across penalties.

Prediction and representation shaping jointly update a set of shared parameters ϑ. To isolate the effect of ρ at a single optimization step, our local gradient analysis uses the same model state, minibatch, and stochastic realization for all penalties. Stopped targets are held constant, while differentiable target paths and native normalization are included in the derivatives. The predictive gradient on ϑ is

$$p_\rho \equiv \nabla_\vartheta \mathcal{P}_\rho = c \sum_{i=1}^{m} \rho'(r_i) \nabla_\vartheta r_i. \tag{2}$$

This equation shows that each residual r_i contributes $c\rho'(r_i)\nabla_\vartheta r_i$ to the predictive gradient. Because the residual-gradient directions vary across dimensions, their relative coefficients determine the net predictive gradient. Consequently, even when the representation objective is fixed, changing the predictive penalty can shift the balance between cooperating with or opposing that objective.

Under squared error, the derivative $\rho'(r_i) = 2r_i$ assigns linearly growing weights to larger residuals, allowing them to exert an increasingly dominant pull. This unbounded influence motivates a penalty that preserves local quadratic behavior for small errors while actively moderating the coefficients assigned to large errors.

3.2 Log-Cosh Prediction

We use element-wise log-cosh as the predictive penalty:

$$\mathcal{L}_{\text{lc}} = c \sum_{i=1}^{m} \log\cosh(r_i) + \lambda \mathcal{R}. \tag{3}$$

Near zero, the penalty behaves quadratically ($\log\cosh u = \frac{1}{2}u^2 + O(u^4)$). Its derivative, or *residual score*, is $\psi_{\text{lc}}(u) = \tanh u$. Consequently, the coefficient assigned to a small residual grows linearly, while that of a large residual smoothly saturates.

Residual-dependent Weighting. To formalize how this alters the gradient relative to a standard quadratic penalty, let $J = D_\vartheta r$ be the Jacobian and $\psi_\rho = \rho'$ be the element-wise derivative. By defining a weight function $w_\rho(u) = \psi_\rho(u)/u$ for each penalty in Table 1 (with the continuous extension $w_\rho(0) = \rho''(0)$), we construct a diagonal *Predictive Influence Operator*:

$$\mathsf{W}_\rho(r) = \operatorname{diag}(w_\rho(r_1), \ldots, w_\rho(r_m)), \qquad p_\rho = cJ^\top \mathsf{W}_\rho(r) r. \tag{4}$$

Each diagonal entry scales a single residual's contribution before $J^\top$ maps the weighted residuals into parameter space. For log-cosh, this weight is $w_{\text{lc}}(u) = \tanh(u)/u$. Because this value lies in $(0, 1]$ and strictly decreases as $|u|$ increases, larger residuals receive stronger attenuation. Since this attenuation is non-uniform across dimensions, it can change the magnitude and direction of the predictive gradient p_ρ. Finally, while the residual scores are strictly bounded, the resulting parameter gradient still scales with the Jacobian J.

Table 1: Predictive penalties and their residual weights. Its weight is defined continuously as $w_{\text{lc}}(0) = 1$.

Penalty	$\rho(u)$	$\psi_\rho(u)$	$w_\rho(u)$
Squared error	u^2	$2u$	2
Matched quadratic	$\frac{1}{2}u^2$	u	1
Log-cosh	$\log\cosh u$	$\tanh u$	$\tanh(u)/u$

A Locally Matched Quadratic Control. Table 1 distinguishes between uniform scaling and residual-dependent weighting. Replacing the standard u^2 penalty with $\frac{1}{2}u^2$ halves every predictive contribution (i.e., $p_{\text{mse}} = 2p_{\text{q}}$ at a given state). In contrast, log-cosh applies a tailored attenuation factor based on each residual's magnitude. Since $\rho''_{\text{lc}}(0) = \rho''_{\text{q}}(0) = 1$, the half-squared penalty matches the curvature of log-cosh at the origin. Comparing log-cosh against this control tests whether merely halving the predictive term reproduces the benefits.

Selectively attenuating large residuals shifts which errors dominate the predictive update. Its effect on local gradient interaction depends on the balance between the conflicting and cooperative contributions that are attenuated.

3.3 Interaction with Representation Shaping

Prediction can Reinforce or Oppose Shaping. Let $h = \nabla_\vartheta \mathcal{G}$ be the gradient of the weighted representation objective and define their inner product $A_\rho = \langle p_\rho, h \rangle$. To interpret this interaction, consider a local raw-gradient step on the shared parameters, $\vartheta^+ = \vartheta - \eta(p_\rho + h)$, with $\eta > 0$ and all other parameters held fixed. For differentiable $\mathcal{G}$, a first-order expansion gives

$$\mathcal{G}(\vartheta^+) - \mathcal{G}(\vartheta) = -\eta\big(\|h\|^2 + A_\rho\big) + o(\eta). \tag{5}$$

The term $\|h\|^2$ represents the first-order contribution of the representation objective acting alone. The predictive update reinforces this contribution when $A_\rho > 0$ (cooperation) and opposes it when $A_\rho < 0$ (conflict). Strict first-order descent requires $\|h\|^2 + A_\rho > 0$. This analysis concerns raw gradients, before optimizer preconditioning, momentum, or clipping.

Locating Cooperation and Conflict. By writing $s = Jh$, Equation 2 allows us to decompose the interaction into residual space:

$$A_\rho = c\langle \psi_\rho(r), s \rangle = c \sum_{i=1}^{m} \psi_\rho(r_i) s_i. \tag{6}$$

For the locally matched quadratic penalty, we define the coordinate-wise interaction as $\alpha_i = r_i s_i$, yielding the total interaction $A_{\mathrm{q}} = c \sum_i \alpha_i$. A positive α_i means that the negative-gradient update from coordinate i decreases $\mathcal{G}$ to first order (cooperation); conversely, a negative α_i identifies a conflict. Essentially, this sign depends on both the prediction residual and its sensitivity along the representation gradient h. Thus, while residual magnitude dictates how heavily log-cosh attenuates a coordinate, α_i dictates whether that coordinate was helping or hurting representation shaping.

When attenuation helps. Let $\delta_i = 1 - w_{\mathrm{lc}}(r_i)$ represent the fractional attenuation relative to the matched quadratic, where $\delta_i = 0$ at $r_i = 0$. Since log-cosh attenuates contributions of both signs, the net change is determined by their weighted balance.

Proposition 1 (Residual-space interaction)**.** *At the same model state and mini-batch realization, log-cosh and the matched quadratic satisfy*

$$A_{\mathrm{lc}} - A_{\mathrm{q}} = c\Bigg[\underbrace{\sum_{\alpha_i < 0} \delta_i |\alpha_i|}_{\textit{removed conflict}} \; - \; \underbrace{\sum_{\alpha_i > 0} \delta_i \alpha_i}_{\textit{lost cooperation}} \Bigg]. \tag{7}$$

Consequently, $A_{\mathrm{lc}} > A_{\mathrm{q}}$ if and only if the removed conflicting interaction exceeds the lost cooperative interaction.

Proof. Since $\psi_{\mathrm{lc}}(r_i) = (1-\delta_i) r_i$, Equation 6 gives $A_{\mathrm{lc}} = c \sum_i (1-\delta_i)\alpha_i$. Subtracting $A_{\mathrm{q}} = c \sum_i \alpha_i$ and separating the positive and negative terms yields Equation 7. □

Notably, log-cosh determines its weights based purely on residual magnitude; it does not observe the representation gradient to identify conflict. Therefore, suppressing these contributions makes prediction more compatible with representation shaping only if those heavily attenuated residuals harbor enough conflict. Any cooperative contributions at those same magnitudes are unavoidably penalized, and the proposition strictly accounts for both effects. As shown in Equation 5, an increase in A_ρ corresponds to a more favorable first-order contribution to decreasing $\mathcal{G}$. Figure 1 illustrates this conditional balance. While this result characterizes local gradient interaction on shared parameters, its broader consequences for learned representations and downstream tasks must be evaluated empirically.

Applying the replacement. We implement log-cosh as a direct, unit-scale replacement for the predictive penalty, preserving each model's native coordinate selection, normalization, loss reduction, and target-gradient conventions. In TS and LeWM, we keep architectures, auxiliary objectives and their weights, training protocols, and inference procedures fixed, including the squared latent goal cost. The driving JEPA objective uses its native masking, position weights, branch-wise reductions, and coefficient schedules, as detailed in the experimental protocol. The local analysis above assumes a fixed explicit representation objective $\mathcal{G}$, whereas the loss replacement can also be applied without one.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We first evaluate whether the predictive penalty improves planning across navigation, manipulation, and continuous control, following the same setting as in Temporal Straightening (TS) and LeWorld-Model (LeWM) (Wang et al., 2026; Maes et al., 2026). We then extend the comparison to perception and planning on real-world driving data. Physical-state probes and qualitative visualizations examine the learned representations and the costs used for planning.

Configurations and tasks. The TS study covers Wall, PointMaze–UMaze, PointMaze–Medium, and PushT, with a frozen DINOv2 backbone and a trainable projector or a ResNet trained from scratch. Each encoder is studied with global (1×384) and spatial ($14 \times 14 \times 8$) latents. The LeWM study covers TwoRoom, PushT, Reacher, and Cube. These families use temporal-curvature and distributional anti-collapse objectives, respectively.

Controlled comparison and reporting. We compare r^2, $\frac{1}{2}r^2$, and log-cosh. In TS and LeWM, we retain the architecture, representation objective and its coefficient, training budget, checkpoint rule, and planner, including the squared latent goal cost. TS uses gradient-based planning while LeWM uses the cross-entropy method (CEM). We report open-loop and model-predictive control (MPC) separately, with mean and standard deviation over three seeds and 50 test episodes per seed. Open-loop execution follows a plan without intermediate feedback, whereas MPC replans using updated observations. These settings distinguish performance with a fixed action sequence from performance with feedback during execution. Driving results use the separate protocol in Section D.

4.2 PLANNING PERFORMANCE

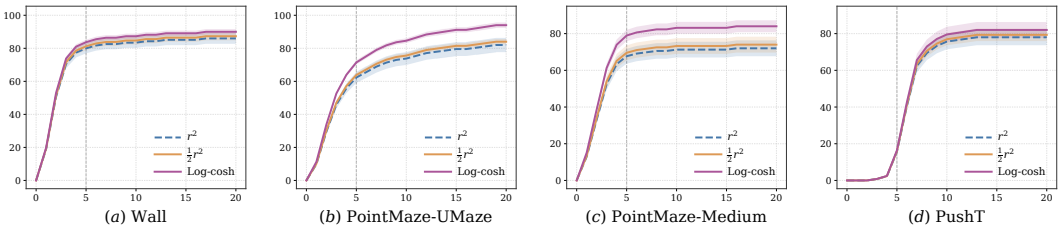
Temporal Straightening. Table 2 shows improvements over r^2 across both encoders, latent shapes, and execution modes. The encoder comparison covers both adaptation of frozen visual features and learning an encoder from scratch, while the latent shapes vary the representation used for planning. With DINOv2 spatial latents, open-loop success rises from 82% to 94% on U-Maze and from 72% to 84% on Medium: gains of 12 percentage points each. Gains also persist under MPC, including configurations where the quadratic baseline is already close to the success-rate ceiling. Log-cosh matches or exceeds the locally matched quadratic across all reported configurations.

Table 2: Goal-reaching success rates (%) for TS using the gradient descent (GD) planner. Values are mean ± standard deviation over three data sampling seeds, with 50 test episodes per seed. The highest mean performance within each encoder and latent-shape configuration is highlighted in **bold**.

Encoder	Shape	Penalty	Wall		PointMaze–UMaze		PointMaze–Medium		PushT	
			Open-loop	MPC	Open-loop	MPC	Open-loop	MPC	Open-loop	MPC
DINOv2 (patch) + proj	1×384	r^2	36.67 ± 3.40	51.33 ± 4.11	30.67 ± 3.40	90.00 ± 3.27	16.67 ± 4.11	69.33 ± 3.40	2.67 ± 2.49	10.00 ± 1.63
		$\frac{1}{2}r^2$	38.67 ± 2.49	54.00 ± 3.27	34.00 ± 3.27	92.67 ± 1.89	19.33 ± 4.11	73.33 ± 2.49	4.00 ± 1.63	12.00 ± 1.63
		Log-cosh	**42.00** ± 3.27	**56.67** ± 4.11	**38.67** ± 3.40	**96.00** ± 0.00	**22.67** ± 5.73	**78.00** ± 2.83	**5.33** ± 3.40	**14.67** ± 0.94
ResNet (from scratch)	1×384	r^2	78.67 ± 7.36	98.00 ± 1.63	42.00 ± 5.89	80.00 ± 3.27	44.67 ± 6.18	94.00 ± 1.63	14.67 ± 4.11	42.00 ± 6.53
		$\frac{1}{2}r^2$	81.33 ± 5.73	99.33 ± 0.94	45.33 ± 5.73	83.33 ± 0.94	48.00 ± 6.53	96.00 ± 1.63	16.67 ± 4.11	45.33 ± 5.73
		Log-cosh	**84.00** ± 7.12	**100.00** ± 0.00	**52.00** ± 6.53	**86.67** ± 0.94	**54.00** ± 7.12	**98.00** ± 0.00	**19.33** ± 3.40	**48.67** ± 4.99
DINOv2 (patch) + proj	$14^2 \times 8$	r^2	86.00 ± 3.27	98.00 ± 1.63	82.00 ± 4.32	95.33 ± 2.49	72.00 ± 4.32	94.67 ± 2.49	78.00 ± 4.32	80.67 ± 2.49
		$\frac{1}{2}r^2$	87.33 ± 2.49	99.33 ± 0.94	84.00 ± 1.63	98.00 ± 1.63	74.00 ± 4.32	96.67 ± 1.89	79.33 ± 3.40	82.67 ± 3.77
		Log-cosh	**90.00** ± 1.63	**100.00** ± 0.00	**94.00** ± 1.63	**100.00** ± 0.00	**84.00** ± 3.27	**98.67** ± 0.94	**82.00** ± 4.32	**85.33** ± 4.99
ResNet (from scratch)	$14^2 \times 8$	r^2	80.00 ± 3.27	98.67 ± 0.94	53.33 ± 6.60	94.67 ± 2.49	69.33 ± 4.11	96.00 ± 1.63	66.00 ± 3.27	84.00 ± 4.32
		$\frac{1}{2}r^2$	82.00 ± 3.27	**100.00** ± 0.00	57.33 ± 6.60	96.00 ± 1.63	73.33 ± 2.49	97.33 ± 0.94	68.00 ± 2.83	87.33 ± 3.77
		Log-cosh	**84.67** ± 2.49	**100.00** ± 0.00	**64.67** ± 8.38	**98.67** ± 1.89	**80.67** ± 0.94	**99.33** ± 0.94	**70.67** ± 0.94	**91.33** ± 2.49

Figure 2 provides the corresponding open-loop success curves for the DINOv2 spatial configuration.

LeWorldModel. LeWM tests the replacement with a different representation objective and planner. In Table 3, log-cosh improves over both quadratic penalties on all four tasks under open-loop execution and MPC. For example, relative to r^2, Reacher open-loop success increases from 80.67% to 86.67%, and Cube MPC success from 82.67% to 88.67%. The matched quadratic does not improve every task: on PushT, its means are slightly lower than those of r^2, whereas log-cosh improves both execution modes. Together with TS, these results extend the observed gains across different representation objectives and search procedures.



(a) Wall (b) PointMaze-UMaze (c) PointMaze-Medium (d) PushT

Figure 2: Open-loop success curves with DINOv2 spatial latents ($14 \times 14 \times 8$) on Wall, PointMaze–UMaze, PointMaze–Medium, and PushT.

Table 3: LeWM planning success rates (%) on navigation, manipulation, and continuous control tasks. Results are mean ± standard deviation over three seeds, with 50 test episodes per seed; the best mean in each column is in **bold**.

Predictive penalty	TwoRoom		PushT		Reacher		Cube	
	Open-loop	MPC	Open-loop	MPC	Open-loop	MPC	Open-loop	MPC
r^2	90.67 ± 2.49	96.67 ± 1.89	91.33 ± 2.49	96.67 ± 2.49	80.67 ± 4.11	90.67 ± 3.40	70.67 ± 4.11	82.67 ± 4.11
$\frac{1}{2}r^2$	91.33 ± 1.89	97.33 ± 1.89	90.67 ± 3.40	95.33 ± 2.49	82.67 ± 3.40	92.67 ± 2.49	72.67 ± 3.77	84.67 ± 3.40
Log-cosh	**94.67** ± 1.89	**98.67** ± 0.94	**93.33** ± 2.49	**98.67** ± 0.94	**86.67** ± 3.40	**95.33** ± 1.89	**74.67** ± 3.40	**88.67** ± 2.49

Locally Matched Quadratic. Across both model families, $\frac{1}{2}r^2$ does not reproduce the gains of log-cosh. Thus, merely halving the predictive term is insufficient to explain the observed improvements.

Longer-horizon Planning. The longer-horizon evaluation tests whether the gains can still hold when plans extend beyond the standard evaluation horizon (Table 4). Log-cosh improves over both quadratic controls for both encoders and execution modes, with gains of 4–6 percentage points over r^2 on PushT and Medium. PushT remains challenging: log-cosh achieves 12.67–14.67% success in open-loop execution and 33.33–36.67% under MPC. Thus, the gains persist at longer horizons while leaving substantial room for improving the complete planning system.

Table 4: Longer-horizon success rates (%). Values are mean ± population standard deviation over three seeds, with 50 test episodes per seed. The best mean in each configuration is in **bold**.

Encoder	Penalty	PushT		PointMaze–Medium	
		Open-loop	MPC	Open-loop	MPC
DINO	r^2	10.67 ± 2.49	28.67 ± 6.60	64.67 ± 4.99	86.67 ± 4.11
	$\frac{1}{2}r^2$	12.67 ± 3.40	30.67 ± 5.73	67.33 ± 5.73	89.33 ± 3.40
	Log-cosh	**14.67** ± 4.11	**33.33** ± 5.73	**70.67** ± 5.73	**92.67** ± 3.40
ResNet	r^2	8.67 ± 3.40	30.67 ± 7.72	62.67 ± 6.60	87.33 ± 3.77
	$\frac{1}{2}r^2$	10.67 ± 4.11	33.33 ± 6.80	65.33 ± 5.25	90.67 ± 4.11
	Log-cosh	**12.67** ± 5.73	**36.67** ± 6.60	**68.67** ± 6.80	**93.33** ± 2.49

4.3 Driving Perception and Planning

To evaluate the generalizability of our method beyond simulated benchmarks, we extend our analysis to real-world autonomous driving perception and planning. We adopt a convolutional JEPA architecture inspired by V-JEPA 2.1 (Mur-Labadia et al., 2026), comprising a HENet-TinyM (Xia et al., 2024) encoder, an EMA (Assran et al., 2023) target encoder, and a spatiotemporal predictor. For self-supervised training, we use eight-frame clips from the NAVSIM (Dauner et al., 2024) `trainval` split and compute prediction residuals after channel-wise LayerNorm (see Section D for implementation details). Following standard large-scale pretraining practice, we evaluate the model both with and without prior pretraining on separate data. Downstream performance is measured across five perception MAE metrics alongside planning ADE and FDE (in meters), testing whether the relative ranking of the predictive penalties persists across both pretraining regimes.

Table 5: Driving perception and planning with and without pretraining on separate data. Lower is better for all metrics. The best result within each pretraining setting is highlighted in **bold**. Perception categories are vehicle (Veh.), pedestrian (Ped.), bicycle (Bic.), traffic cone (Cone), and barrier.

Pretrained	Loss	Perception: MAE ↓					Planning ↓	
		Veh.	Ped.	Bic.	Cone	Barrier	ADE/m	FDE/m
✓	r^2	2.93	3.79	0.053	1.02	0.158	7.91	14.21
	$\frac{1}{2}r^2$	2.94	3.81	0.053	1.02	0.158	7.95	14.28
	Log-cosh	**2.65**	**3.43**	**0.048**	**0.91**	**0.133**	**7.15**	**12.85**
✗	r^2	3.08	3.98	0.055	1.07	0.165	8.36	15.16
	$\frac{1}{2}r^2$	3.09	4.00	0.055	1.07	0.166	8.40	15.24
	Log-cosh	**2.71**	**3.50**	**0.049**	**0.93**	**0.136**	**7.30**	**13.12**

Table 6: Physical-state probing on PushT with frozen LeWM encoders. Lower MSE and higher correlation r are better. MSE variability is measured across three probe initialization seeds.

Method	Agent Location				Block Location				Block Angle			
	Linear		MLP		Linear		MLP		Linear		MLP	
	MSE ↓	r ↑	MSE ↓	r ↑	MSE ↓	r ↑	MSE ↓	r ↑	MSE ↓	r ↑	MSE ↓	r ↑
LeWorldModel with r^2	0.0513 ± 0.0051	0.9743	0.0041 ± 0.0005	0.9979	0.0287 ± 0.0035	0.9857	0.0011 ± 0.0002	0.9991	0.1823 ± 0.0173	0.9053	0.0207 ± 0.0021	0.9897
LeWorldModel with $\frac{1}{9}r^2$	0.0497 ± 0.0047	0.9751	0.0038 ± 0.0004	0.9981	0.0263 ± 0.0033	0.9871	0.0010 ± 0.0002	0.9992	0.1733 ± 0.0167	0.9097	0.0193 ± 0.0023	0.9907
LeWorldModel with Log-cosh	**0.0383** ± 0.0039	**0.9817**	**0.0024** ± 0.0003	**0.9993**	**0.0203** ± 0.0027	**0.9903**	**0.0007** ± 0.0001	**0.9995**	**0.1373** ± 0.0137	**0.9277**	**0.0143** ± 0.0017	**0.9933**

In both pretraining regimes, log-cosh achieves the lowest error across all seven metrics (Table 5). Compared to squared error (r^2), log-cosh reduces planning ADE/FDE from $7.91/14.21$ m to $7.15/12.85$ m under pretraining, and from $8.36/15.16$ m to $7.30/13.12$ m from scratch, while lowering perception MAEs by 9.4%–15.8% and 10.9%–17.6%, respectively. Although relative ADE/FDE reductions are more pronounced when training from scratch, pretrained log-cosh attains the best absolute performance. Meanwhile, the matched quadratic control remains close to r^2, confirming once again that uniform loss rescaling cannot account for the improvement. This consistent ordering across perception and planning metrics extends our findings beyond simulated goal-reaching tasks.

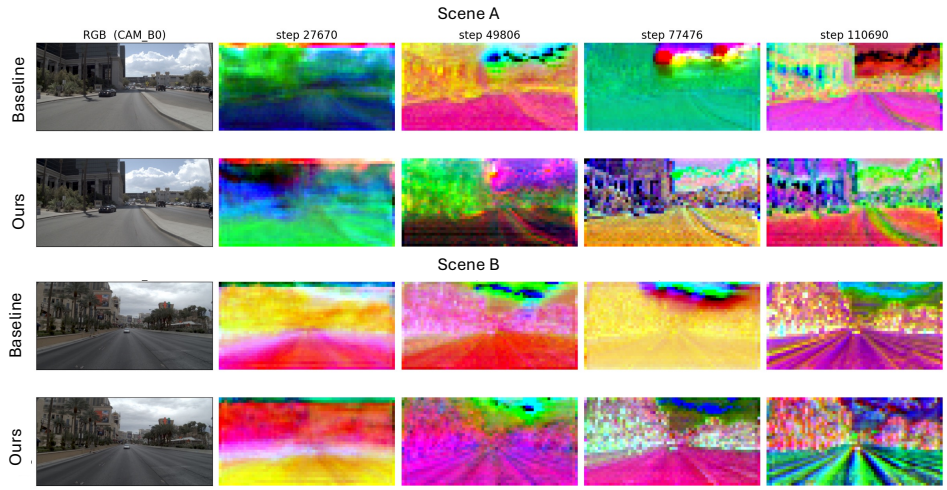


Figure 3: Visualization of the latent features. Baseline and Ours refers to r^2 and log-cosh, respectively.

4.4 Physical Structure of the Latent Space

Downstream planning success reflects the combined performance of the learned representation, the dynamics predictor, and the action-search procedure. We use frozen-encoder probes to directly evaluate which physical quantities are captured in the latent space. Following the LeWM protocol (Maes et al., 2026), we train linear and MLP probes to recover agent location, block location, and block orientation on PushT (Table 6; details in Section E.1). Evaluating both linear and nonlinear readouts tests whether the relative ordering of the predictive penalties holds regardless of probe expressiveness.

Relative to the matched quadratic control, log-cosh reduces linear-probe MSE by 22.9%, 22.8%, and 20.8% for agent location, block location, and block orientation, respectively, and the MLP probes preserve this exact ranking. These gains confirm that log-cosh makes physical state quantities more accessible across readout capacities, while we examine the native latent distances used directly by the planner separately below.

4.5 Latent Goal-Distance Geometry

Because a learned probe can recover physical state even when raw latent distances offer poor planning guidance, we directly inspect the squared latent distances to fixed goals in PointMaze–UMaze and PointMaze–Medium (Figure 4). In these mazes, where visually proximate states may be separated by walls, the heatmaps complement probing by revealing how the planner's native distance metric varies around obstacles and near the goal. These maps visualize goal distance at each planning state, and we observe that the log-cosh learned features appear more consistent with the ground truth.



Figure 4: Latent goal-distance heatmaps for PointMaze–UMaze (left four panels) and PointMaze–Medium (right four panels). Each group displays (from left to right) ground truth, r^2, $\frac{1}{2}r^2$, and log-cosh. White regions indicate maze walls, the goal is denoted by the tiny yellow star, and darker blue tones reflect shorter distances to the goal.

4.6 Action-Space Planning Geometry

Action search optimizes the composition of the learned representation, dynamics model, and goal cost. Figures 5 and 6 offer complementary views of this composite objective across a two-dimensional action-coefficient plane (a_1, a_2): the trajectory panels trace the optimization paths, while the contour panels reveal the underlying cost landscape. Together, they link the geometry of the learned objective directly to the behavior of action search.

Optimization Trajectories. Figure 5 compares optimization paths under the learned costs and a true-dynamics reference, illustrating how the cost landscape guides action search. We note that the log-cosh dynamics behave more straightforwardly.

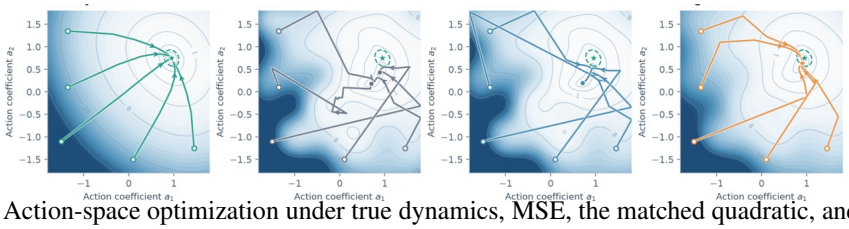


Figure 5: Action-space optimization under true dynamics, MSE, the matched quadratic, and log-cosh (left to right). Paths show optimization from multiple starting points in the (a_1, a_2) plane; shading and contours show the corresponding costs.

Planning-cost Contours. Figure 6 compares the location and shape of learned and reference cost contours. In these examples, log-cosh contours follow the reference more closely around the low-cost region. These qualitative observations complement the planning results without independently establishing the local gradient-interaction mechanism.

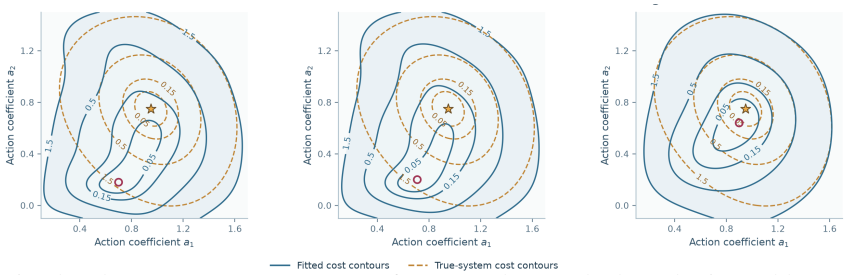


Figure 6: Fitted and true-system cost contours for MSE, the matched quadratic, and log-cosh (left to right). Solid blue curves denote fitted costs; dashed orange curves denote the true-system reference at the labeled levels.

5 Conclusion

This paper demonstrates that the choice of predictive penalty fundamentally shapes the quality of representations learned for latent planning. Replacing squared error with a log-cosh penalty yields consistent performance gains across both trajectory-curvature regularization (TS) and SIGReg anti-collapse regularization (LeWM), and extends effectively to real-world autonomous driving perception and planning. Together, our matched quadratic controls, physical latent probing, and geometric visualizations confirm that log-cosh provides superior guidance for action optimization specifically through non-linear residual attenuation, rather than uniform gradient rescaling.

AI USE STATEMENT

We used OpenAI Codex to assist with literature review, mathematical exposition, experimental-design discussion, and manuscript editing. Earlier exploratory drafts included AI-generated hypothetical numerical examples. This working draft marks illustrative values and unresolved provenance explicitly. The authors are responsible for checking all final empirical claims against experiment records and for the accuracy, originality, and integrity of the submission.

REFERENCES

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15619–15629, 2023.

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.

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

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *ICLR*, 2022.

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

Jonathan T Barron. A general and adaptive robust loss function. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4326–4334. IEEE, 2019.

Pierre Charbonnier, Laure Blanc-Féraud, Gilles Aubert, and Michel Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on image processing*, 6(2): 298–311, 1997.

Ruidi Chen and Ioannis Ch Paschalidis. A robust learning approach for regression models based on distributionally robust optimization. *Journal of Machine Learning Research*, 19(13):1–48, 2018.

Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, et al. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. *Advances in Neural Information Processing Systems*, 37:28706–28719, 2024.

Matthieu Destrade, Oumayma Bounou, Quentin Le Lidec, Jean Ponce, and Yann LeCun. Value-guided action planning with jepa world models. *arXiv preprint arXiv:2601.00844*, 2025.

Andreas Doerr, Christian Daniel, Martin Schiegg, Nguyen-Tuong Duy, Stefan Schaal, Marc Toussaint, and Trimpe Sebastian. Probabilistic recurrent state-space models. In *International conference on machine learning*, pp. 1280–1289. PMLR, 2018.

Bradley Efron. Regression percentiles using asymmetric squared error loss. *Statistica Sinica*, pp. 93–125, 1991.

Quentin Garrido, Mahmoud Assran, Nicolas Ballas, Adrien Bardes, Laurent Najman, and Yann LeCun. Learning and leveraging world models in visual representation learning. *arXiv preprint arXiv:2403.00504*, 2024.

Spyros Gidaris and Nikos Komodakis. Detect, replace, refine: Deep structured prediction for pixel wise labeling. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 7187–7196. IEEE, 2017.

Mikael Henaff, Alfredo Canziani, and Yann LeCun. Model-predictive policy learning with uncertainty regularization for driving in dense traffic. *arXiv preprint arXiv:1901.02705*, 2019.

Rui Hu, Yifan Zhang, Zhuoran Li, and Longbo Huang. Beyond squared error: Exploring loss design for enhanced training of generative flow networks. In *International Conference on Learning Representations*, volume 2025, pp. 20764–20787, 2025.

Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pp. 492–518. Springer, 1992.

Yilun Kuang, Yash Dagade, Quentin Le Lidec, Lucas Maes, Randall Balestriero, and Yann LeCun. Lpwm: A case for sparse representations in world models. *arXiv preprint arXiv:2608.22764*, 2026.

Yann LeCun. A path towards autonomous machine intelligence. OpenReview, 2022. URL `https://openreview.net/forum?id=BZ5a1r-kVsf`. Version 0.9.2.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press, 2006. URL `https://cs.nyu.edu/~yann/research/ebm/`.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, and Nicolas Ballas. V-jepa 2.1: Unlocking dense features in video self-supervised learning. In *European Conference on Computer Vision*, pp. 671–689. Springer, 2026.

Karl Pertsch, Oleh Rybkin, Frederik Ebert, Shenghao Zhou, Dinesh Jayaraman, Chelsea Finn, and Sergey Levine. Long-horizon visual planning with goal-conditioned hierarchical predictors. *Advances in Neural Information Processing Systems*, 33:17321–17333, 2020.

Uladzislau Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim G. J. Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 43905–43941. Curran Associates, Inc., 2025. URL `https://proceedings.neurips.cc/paper_files/paper/2025/file/3e7cf447f21cd11c846463affefce665-Paper-Conference.pdf`.

Ying Wang, Oumayma Bounou, Gaoyue Zhou, Randall Balestriero, Tim G. J. Rudner, Yann LeCun, and Mengye Ren. Temporal straightening for latent planning. In *International Conference on Machine Learning (ICML)*, 2026.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Shawn Ma, and Yitao Liang. Describe, explain, plan and select: interactive planning with llms enables open-world multi-task agents. *Advances in Neural Information Processing Systems*, 36:34153–34189, 2023.

Zhongyu Xia, Zhiwei Lin, Xinhao Wang, Yongtao Wang, Yun Xing, Shengxiang Qi, Nan Dong, and Ming-Hsuan Yang. Henet: Hybrid encoding for end-to-end multi-task 3d perception from multi-view cameras. In *European Conference on Computer Vision*, pp. 376–392. Springer, 2024.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33: 5824–5836, 2020.

Jiahan Zhang, Muqing Jiang, Nanru Dai, Taiming Lu, Arda Uzunoglu, Shunchi Zhang, Yana Wei, Jiahao Wang, Vishal Patel, Paul Liang, et al. World-in-world: World models in a closed-loop world. In *International Conference on Learning Representations*, volume 2026, pp. 55660–55699, 2026.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

Gaoyue Zhou, Hengkai Pan, Yann Lecun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135. PMLR, 13–19 Jul 2025.

A Experimental Protocols

A.1 Controlled Loss Replacement

The TS and LeWM comparisons change the coordinatewise predictive penalty among r^2, $\frac{1}{2}r^2$, and $\log\cosh r$. The residual r includes the normalization used by the corresponding model. We retain each family's masks, reductions, architecture, representation objective and its coefficient, training budget, checkpoint rule, and planner. In particular, the planning cost remains the original squared latent goal distance. The half-scaled quadratic matches the second derivative of log-cosh at zero and tests whether reducing the predictive term by a constant factor accounts for the improvement. The driving model uses the separate two-branch objective in Section D.

Model families. TS uses one-step latent prediction from observation/action history, a stopped-gradient predictive target, temporal-curvature regularization, and gradient-based action optimization. We evaluate a frozen DINOv2 backbone with a trainable projector and a ResNet trained from scratch, each with global 1×384 and spatial $14 \times 14 \times 8$ latents. LeWM uses an end-to-end encoder, a teacher-forced causal predictor, and SIGReg distributional regularization. Gradients pass through both the predicted and target encoder paths; its planner uses the cross-entropy method (CEM). These differences are retained within their respective comparisons.

Training and evaluation. TS training runs for 20 epochs; LeWM results use the epoch-10 checkpoint. The task sets are Wall, U-Maze, Medium, and PushT for TS, and TwoRoom, PushT, Reacher, and Cube for LeWM. Each condition in the main planning tables uses three seeds and 50 test episodes per seed. TS uses data-sampling seeds, as specified in the main table. Open-loop execution follows the planned action sequence without intermediate feedback, whereas MPC replans using updated observations. We report the two execution modes separately.

A.2 Reporting and Comparisons

Planning tables retain the reported mean and standard deviation. The longer-horizon table specifically uses population standard deviation. Probe variability is across three probe-initialization seeds and is distinct from world-model training variability. Driving results are reported as scalar metrics, without inferred error bars.

All supplementary gains below are arithmetic summaries of the displayed results. For success rates in percent, the gain in percentage points is $S_{\mathrm{lc}} - S_b$. For an error metric E, the relative reduction against baseline b is $100(E_b - E_{\mathrm{lc}})/E_b$. Task-macro success is the unweighted mean of the four task means for one model configuration and one execution mode.

B Complete Planning Results

The planning, driving, and probe tables in this appendix reorganize the corresponding main-paper results without changing their numerical values.

B.1 Temporal Straightening

The following tables reorganize the main TS results by encoder and latent shape, retaining every mean and standard deviation. U-Maze and Medium denote PointMaze–UMaze and PointMaze–Medium. Log-cosh improves over r^2 in all 32 task–configuration–mode comparisons. Against $\frac{1}{2}r^2$, it improves in 31 comparisons and ties at 100% success for ResNet spatial latents on Wall under MPC. These are comparisons of reported means, not significance tests.

Table 7: TS success (%) with DINOv2, 1×384. Mean $\pm$ standard deviation over three data-sampling seeds, 50 episodes per seed. Best means are bold.

	Open-loop			MPC		
Task	r^2	$\frac{1}{2}r^2$	Log-cosh	r^2	$\frac{1}{2}r^2$	Log-cosh
Wall	36.67 $\pm$ 3.40	38.67 $\pm$ 2.49	**42.00** $\pm$ 3.27	51.33 $\pm$ 4.11	54.00 $\pm$ 3.27	**56.67** $\pm$ 4.11
U-Maze	30.67 $\pm$ 3.40	34.00 $\pm$ 3.27	**38.67** $\pm$ 3.40	90.00 $\pm$ 3.27	92.67 $\pm$ 1.89	**96.00** $\pm$ 0.00
Medium	16.67 $\pm$ 4.11	19.33 $\pm$ 4.11	**22.67** $\pm$ 5.73	69.33 $\pm$ 3.40	73.33 $\pm$ 2.49	**78.00** $\pm$ 2.83
PushT	2.67 $\pm$ 2.49	4.00 $\pm$ 1.63	**5.33** $\pm$ 3.40	10.00 $\pm$ 1.63	12.00 $\pm$ 1.63	**14.67** $\pm$ 0.94

Table 8: TS success (%) with ResNet, 1×384. Mean $\pm$ standard deviation over three data-sampling seeds, 50 episodes per seed. Best means are bold.

	Open-loop			MPC		
Task	r^2	$\frac{1}{2}r^2$	Log-cosh	r^2	$\frac{1}{2}r^2$	Log-cosh
Wall	78.67 $\pm$ 7.36	81.33 $\pm$ 5.73	**84.00** $\pm$ 7.12	98.00 $\pm$ 1.63	99.33 $\pm$ 0.94	**100.00** $\pm$ 0.00
U-Maze	42.00 $\pm$ 5.89	45.33 $\pm$ 5.73	**52.00** $\pm$ 6.53	80.00 $\pm$ 3.27	83.33 $\pm$ 0.94	**86.67** $\pm$ 0.94
Medium	44.67 $\pm$ 6.18	48.00 $\pm$ 6.53	**54.00** $\pm$ 7.12	94.00 $\pm$ 1.63	96.00 $\pm$ 1.63	**98.00** $\pm$ 0.00
PushT	14.67 $\pm$ 4.11	16.67 $\pm$ 4.11	**19.33** $\pm$ 3.40	42.00 $\pm$ 6.53	45.33 $\pm$ 5.73	**48.67** $\pm$ 4.99

Table 9: TS success (%) with DINOv2, $14 \times 14 \times 8$. Mean $\pm$ standard deviation over three data-sampling seeds, 50 episodes per seed. Best means are bold.

	Open-loop			MPC		
Task	r^2	$\frac{1}{2}r^2$	Log-cosh	r^2	$\frac{1}{2}r^2$	Log-cosh
Wall	86.00 $\pm$ 3.27	87.33 $\pm$ 2.49	**90.00** $\pm$ 1.63	98.00 $\pm$ 1.63	99.33 $\pm$ 0.94	**100.00** $\pm$ 0.00
U-Maze	82.00 $\pm$ 4.32	84.00 $\pm$ 1.63	**94.00** $\pm$ 1.63	95.33 $\pm$ 2.49	98.00 $\pm$ 1.63	**100.00** $\pm$ 0.00
Medium	72.00 $\pm$ 4.32	74.00 $\pm$ 4.32	**84.00** $\pm$ 3.27	94.67 $\pm$ 2.49	96.67 $\pm$ 1.89	**98.67** $\pm$ 0.94
PushT	78.00 $\pm$ 4.32	79.33 $\pm$ 3.40	**82.00** $\pm$ 4.32	80.67 $\pm$ 2.49	82.67 $\pm$ 3.77	**85.33** $\pm$ 4.99

Table 10: TS success (%) with ResNet, $14 \times 14 \times 8$. Mean $\pm$ standard deviation over three data-sampling seeds, 50 episodes per seed. Best means are bold.

	Open-loop			MPC		
Task	r^2	$\frac{1}{2}r^2$	Log-cosh	r^2	$\frac{1}{2}r^2$	Log-cosh
Wall	80.00 $\pm$ 3.27	82.00 $\pm$ 3.27	**84.67** $\pm$ 2.49	98.67 $\pm$ 0.94	**100.00** $\pm$ 0.00	**100.00** $\pm$ 0.00
U-Maze	53.33 $\pm$ 6.60	57.33 $\pm$ 6.60	**64.67** $\pm$ 8.38	94.67 $\pm$ 2.49	96.00 $\pm$ 1.63	**98.67** $\pm$ 1.89
Medium	69.33 $\pm$ 4.11	73.33 $\pm$ 2.49	**80.67** $\pm$ 0.94	96.00 $\pm$ 1.63	97.33 $\pm$ 0.94	**99.33** $\pm$ 0.94
PushT	66.00 $\pm$ 3.27	68.00 $\pm$ 2.83	**70.67** $\pm$ 0.94	84.00 $\pm$ 4.32	87.33 $\pm$ 3.77	**91.33** $\pm$ 2.49



Figure 7: Long-Horizon Open-Loop GD Planning on PushT.

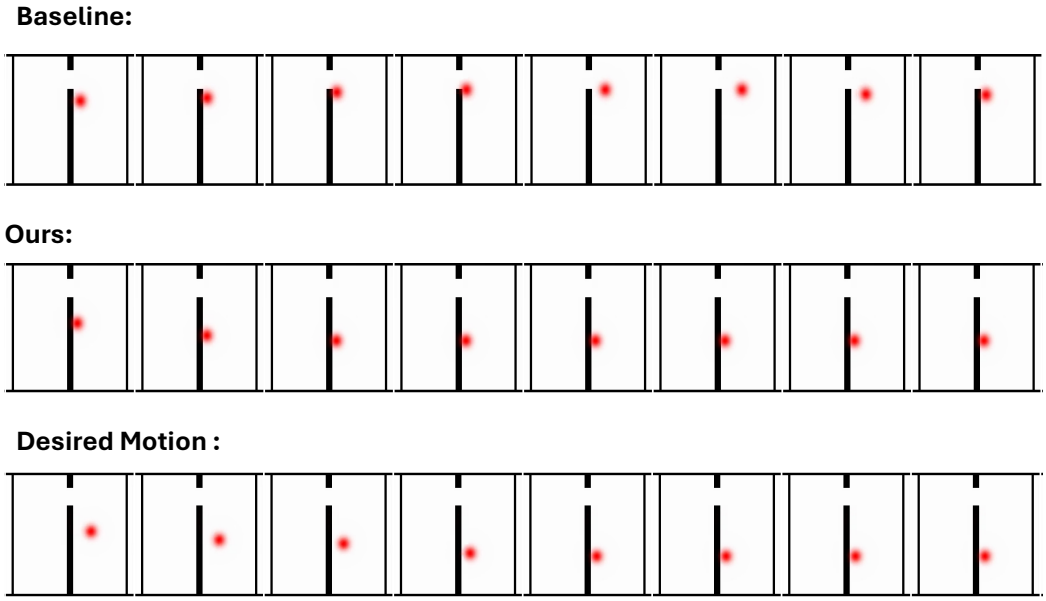


Figure 8: Open-Loop Planning Trajectories of Wall.

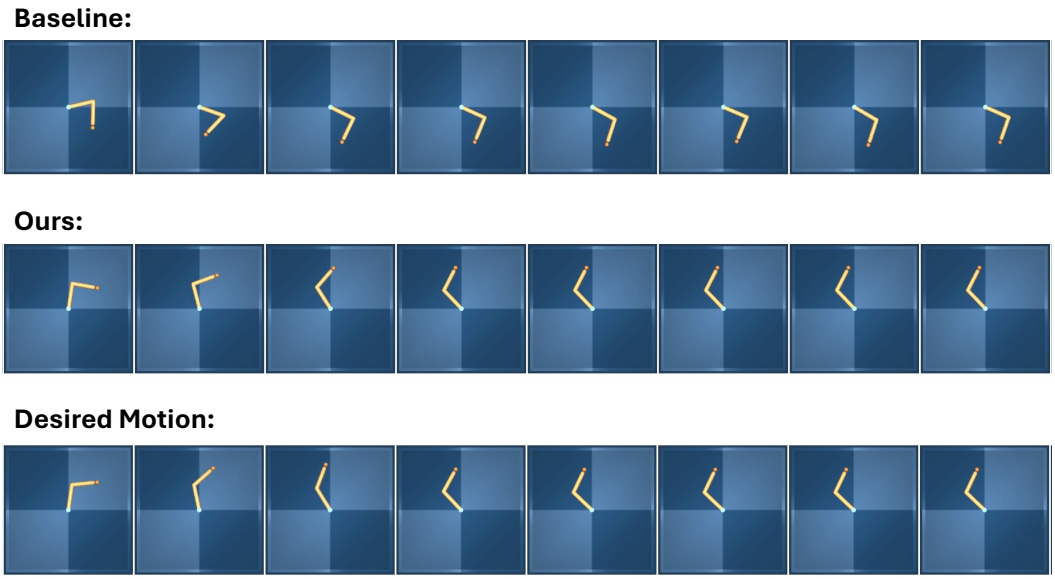


Figure 9: Open-Loop GD Planning on Reacher.

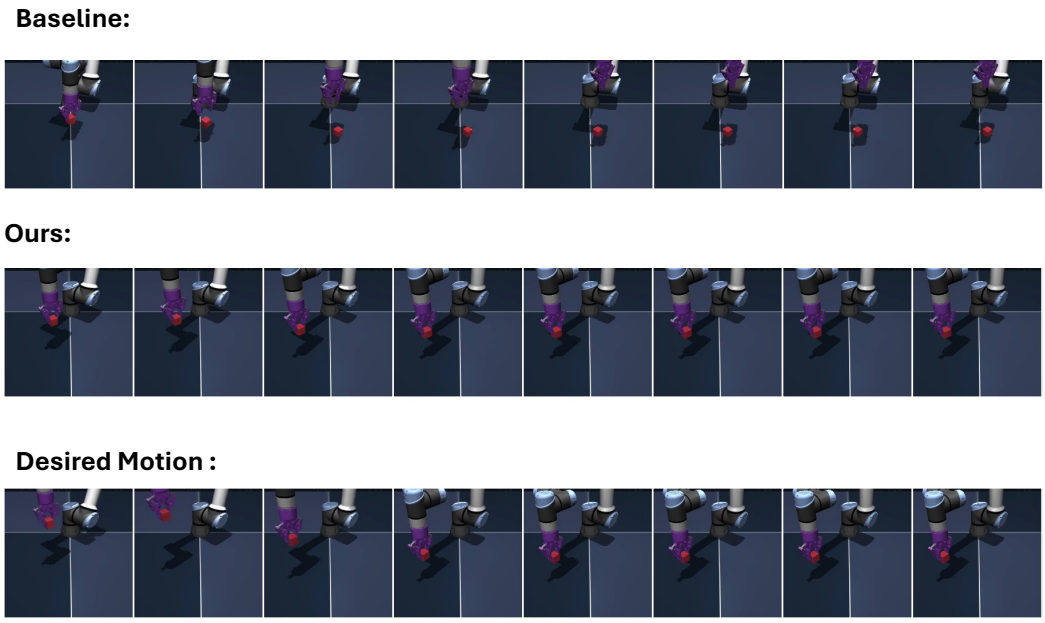


Figure 10: Predictor rollout on OGBench-Cube.

B.2 LeWorldModel and Longer-Horizon Planning

Log-cosh improves the LeWM mean success rate over both quadratic controls on all four tasks in both execution modes. Relative to r^2, gains range from 2 to 6 percentage points, including Reacher open-loop (80.67% to 86.67%) and Cube MPC (82.67% to 88.67%). The improvement therefore extends across navigation, manipulation, and continuous control.

Table 11: LeWM success (%). Mean ± standard deviation over three seeds, with 50 episodes per seed. Best means are bold.

	Open-loop			MPC		
Task	r^2	$\frac{1}{2}r^2$	Log-cosh	r^2	$\frac{1}{2}r^2$	Log-cosh
TwoRoom	90.67 ± 2.49	91.33 ± 1.89	**94.67** ± 1.89	96.67 ± 1.89	97.33 ± 1.89	**98.67** ± 0.94
PushT	91.33 ± 2.49	90.67 ± 3.40	**93.33** ± 2.49	96.67 ± 2.49	95.33 ± 2.49	**98.67** ± 0.94
Reacher	80.67 ± 4.11	82.67 ± 3.40	**86.67** ± 3.40	90.67 ± 3.40	92.67 ± 2.49	**95.33** ± 1.89
Cube	70.67 ± 4.11	72.67 ± 3.77	**74.67** ± 3.40	82.67 ± 4.11	84.67 ± 3.40	**88.67** ± 2.49

For longer-horizon planning, the gains persist for both encoders and both modes, as shown below. Against r^2, improvements range from 4 to 6 percentage points; against $\frac{1}{2}r^2$, they range from 2 to 3.34 points. The absolute PushT scores remain low, showing that the loss replacement does not eliminate the difficulty of longer-horizon manipulation.

Table 12: Longer-horizon success (%) with DINO. Mean ± population standard deviation over three seeds, 50 episodes per seed. Best means are bold.

	Open-loop			MPC		
Task	r^2	$\frac{1}{2}r^2$	Log-cosh	r^2	$\frac{1}{2}r^2$	Log-cosh
PushT	10.67 ± 2.49	12.67 ± 3.40	**14.67** ± 4.11	28.67 ± 6.60	30.67 ± 5.73	**33.33** ± 5.73
Medium	64.67 ± 4.99	67.33 ± 5.73	**70.67** ± 5.73	86.67 ± 4.11	89.33 ± 3.40	**92.67** ± 3.40

Table 13: Longer-horizon success (%) with ResNet. Mean ± population standard deviation over three seeds, 50 episodes per seed. Best means are bold.

	Open-loop			MPC		
Task	r^2	$\frac{1}{2}r^2$	Log-cosh	r^2	$\frac{1}{2}r^2$	Log-cosh
PushT	8.67 ± 3.40	10.67 ± 4.11	**12.67** ± 5.73	30.67 ± 7.72	33.33 ± 6.80	**36.67** ± 6.60
Medium	62.67 ± 6.60	65.33 ± 5.25	**68.67** ± 6.80	87.33 ± 3.77	90.67 ± 4.11	**93.33** ± 2.49

B.3 TASK-MACRO SUMMARIES

Table 14 summarizes the four standard-horizon tasks within each configuration. These equally weighted summaries supplement the individual task results and are calculated from the same reported means. In particular, the DINOv2 spatial configuration gains 8.00 percentage points in open-loop success over r^2 and 6.34 over the matched quadratic. The stronger performance of spatial configurations is an observed configuration comparison; their dimensionality also differs from that of global latents.

Table 14: Equal-task macro success (%) and log-cosh gains (percentage points). Each row averages four task means; OL denotes open-loop. No new experiments or macro error bars are introduced.

		Macro success (%)			Log-cosh gain (pp)	
Configuration	Mode	r^2	$\frac{1}{2}r^2$	Log-cosh	vs. r^2	vs. $\frac{1}{2}r^2$
DINOv2, 1×384	OL	21.67	24.00	27.17	5.50	3.17
DINOv2, 1×384	MPC	55.17	58.00	61.34	6.17	3.34
ResNet, 1×384	OL	45.00	47.83	52.33	7.33	4.50
ResNet, 1×384	MPC	78.50	81.00	83.34	4.84	2.34
DINOv2, $14 \times 14 \times 8$	OL	79.50	81.17	87.50	8.00	6.34
DINOv2, $14 \times 14 \times 8$	MPC	92.17	94.17	96.00	3.83	1.83
ResNet, $14 \times 14 \times 8$	OL	67.17	70.17	75.17	8.01	5.01
ResNet, $14 \times 14 \times 8$	MPC	93.34	95.17	97.33	4.00	2.17
LeWM	OL	83.34	84.34	87.34	4.00	3.00
LeWM	MPC	91.67	92.50	95.34	3.67	2.84

C REPRESENTATION-OBJECTIVE ABLATIONS

C.1 SIGREG COEFFICIENT SWEEP

To examine sensitivity to the representation-objective weight, we retain log-cosh and vary the SIGReg coefficient k in a single-run TwoRoom sweep. The remaining training and planning protocol is fixed, and each coefficient is evaluated on 50 CEM planning episodes. Table 15 reports the observed success rates. Among the tested values, $k = 0.03$ performs best at 94%, versus 28% without SIGReg. Increasing k further reduces success in this sweep. The result supports retaining the representation objective and choosing its weight jointly with the predictive objective; it does not establish a multi-seed optimum or a loss-by-weight interaction.

Table 15: TwoRoom SIGReg sweep with log-cosh. Each entry is one run evaluated on 50 episodes; these are not the multi-seed means in the main table.

k	0	0.03	0.09	0.27	0.50	0.81
Success (%)	28	**94**	82	64	50	42

C.2 TEMPORAL-CURVATURE ABLATION

The recorded TS ablation with log-cosh reports open-loop gains of 56 percentage points on Wall and 78 points on U-Maze when temporal curvature is enabled. These are the available on/off difference summaries; absolute condition scores and their uncertainty are not reconstructed from the main table. They complement the SIGReg sweep by indicating that changing the predictive penalty does not remove the benefit of a representation-shaping objective.

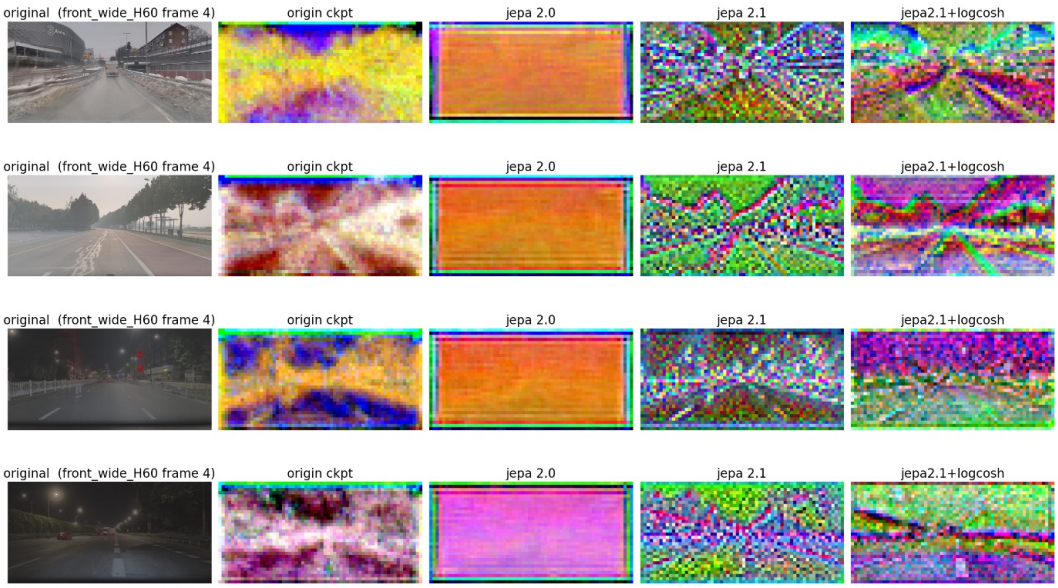


Figure 11: Feature visualization. This panel is reserved for a comparison of r^2 and log-cosh on the same scenes.

D Driving Experiment Details

D.1 Self-Supervised Data and Architecture

D.2 Physical Structure of the Latent Space

Data and clips. For the NAVSIM self-supervised stage, training uses images from the NAVSIM trainval split, based on OpenScene/nuPlan. This stage uses neither human annotations nor images from navtest. The source split contains 1,310 driving logs, 19,376 scenes, and 723,019 keyframes sampled at 2 Hz, with eight camera views per keyframe. We construct non-overlapping, single-camera clips of $T = 8$ frames within each scene. Adjacent frames are separated by $0.5\,\mathrm{s}$ with a tolerance of $\pm 0.15\,\mathrm{s}$, giving a clip span of $3.5\,\mathrm{s}$. Clips do not cross scene boundaries, and incomplete clips are discarded. This yields 708,416 clips from 18,920 scenes, comprising 5,667,328 images (98.0% of the source images). The camera streams share an encoder, and a batch can contain clips from different cameras. Images are resized bilinearly to 1024×512 (width $\times$ height) and use ImageNet normalization, without cropping, flipping, or color augmentation.

Architecture. The model follows a JEPA architecture with a context encoder, an exponential-moving-average (EMA) target encoder, and a spatiotemporal predictor. The HENet-TinyM context encoder has approximately 7.6 million parameters, four stages with channel counts $64/128/192/384$ and block counts $4/3/8/6$, depthwise separable convolutional blocks, and space-to-depth downsampling. It processes frames independently and produces 384-channel features at stride 32, giving a $T \times H_f \times W_f = 8 \times 16 \times 32$ grid. Temporal mixing is performed by the predictor. To restrict information flow from masked regions, masked input pixels are zeroed, features are masked again after convolution, pooling, and normalization, and Sparse BatchNorm computes statistics only over visible positions. The feature-grid mask is upsampled to the intermediate resolutions.

The predictor has three blocks, each consisting of a $3 \times 3 \times 3$ depthwise convolution, a $1 \times 1 \times 1$ pointwise convolution, channel-wise LayerNorm, and GELU. A learned mask token fills masked context positions. Two independent linear heads predict masked target features and reconstruct visible target features, respectively. The predictor has approximately 0.77 million parameters. The target encoder has the same architecture as the context encoder, processes the unmasked clip, and receives no gradients. After each parameter update, its parameters follow

$$\theta_{\mathrm{tgt}} \leftarrow m\theta_{\mathrm{tgt}} + (1 - m)\theta_{\mathrm{ctx}}, \tag{8}$$

with m increasing linearly from 0.996 to 1.0 during training.

Tube masking. For each clip, four spatial rectangles are sampled on the feature grid. Each covers 15–20% of the spatial grid, with an aspect ratio sampled uniformly from 0.75 to 1.5. Their union defines the masked set, and each rectangle is repeated across all eight frames. At least ten cells per frame remain visible. The reported masked fraction is approximately 43–49%.

D.3 Training Objective and Optimization

Two-branch objective. Let h denote target features, and let z_b denote the output of head $b \in \{\mathrm{pred}, \mathrm{ctx}\}$. For each cell i, the predicted and target features are separately normalized across their $C = 384$ channels using LayerNorm without affine parameters. The specified objective uses a common scalar penalty ρ in both branches:

$$d_b(i) = \frac{1}{C} \sum_{c=1}^{C} \rho(\mathrm{LN}(z_b)_{c,i} - \mathrm{LN}(h)_{c,i}) \,. \tag{9}$$

For masked cells $\mathcal{M}$ and visible cells $\mathcal{V}$, the branch losses are

$$\mathcal{L}_{\mathrm{pred}} = \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} d_{\mathrm{pred}}(i), \qquad \mathcal{L}_{\mathrm{ctx}} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} w_i d_{\mathrm{ctx}}(i), \tag{10}$$

where

$$w_i = \left[\max\left(1, \min_{j \in \mathcal{M}} \|p_i - p_j\|_2\right) \right]^{-1/2}, \qquad p_i = (t_i, y_i, x_i). \tag{11}$$

The total loss is $\mathcal{L}_{\mathrm{pred}} + \lambda \mathcal{L}_{\mathrm{ctx}}$, where λ increases linearly from zero to 0.05 over the first 20,000 updates and then remains fixed. The context loss is reduced by $|\mathcal{V}|$, rather than by the sum of its position weights. With this shared-ρ formulation, a change of penalty affects both branches.

Optimization. The NAVSIM self-supervised recipe uses AdamW with $\beta = (0.9, 0.95)$ and a peak learning rate of 6×10^{-4}. The learning rate increases linearly for one epoch and then decays to zero with a cosine schedule. Weight decay increases from 0.04 to 0.4 with a cosine schedule, and the global gradient norm is clipped at 1.0. Training uses FP32 and data parallelism over sixteen A100 80 GB GPUs, with four clips per device and 64 clips (512 frames) per global batch. Ten epochs comprise 110,690 updates, with 11,069 updates per epoch.

Initialization and downstream evaluation. The driving comparison considers r^2, $\frac{1}{2}r^2$, and log-cosh both with pretraining and when training from scratch. The downstream metrics are perception MAE for vehicle, pedestrian, bicycle, traffic cone, and barrier, and planning ADE and FDE in metres.

D.4 Perception and Planning Results

Table 16 lists the complete driving results, retaining the pretraining distinction. Log-cosh lowers all five perception MAEs and both planning errors against either quadratic in both conditions. For pretrained models, ADE decreases from 7.91 to 7.15 m and FDE from 14.21 to 12.85 m against r^2. Training from scratch gives larger relative planning reductions: ADE decreases from 8.36 to 7.30 m and FDE from 15.16 to 13.12 m. The perception results and trajectory errors are separate downstream measurements; a lower MAE alone does not establish the cause of an ADE or FDE improvement.

Table 17 makes the scale of improvement explicit without averaging metrics with different meanings. Compared with r^2, the five perception reductions range from 9.4% to 15.8% with pretraining and from 10.9% to 17.6% from scratch. Improvements also hold relative to $\frac{1}{2}r^2$, so the advantage is not reproduced by halving the quadratic penalty.

Table 16: Complete driving results. The five MAEs measure perception; ADE and FDE measure planning in metres. Lower is better. Best results within each initialization condition are bold.

		Perception: MAE ↓					Planning ↓	
Initialization	Loss	Vehicle	Ped.	Bicycle	Cone	Barrier	ADE	FDE
pretrained	r^2	2.93	3.79	0.053	1.02	0.158	7.91	14.21
pretrained	$\frac{1}{2}r^2$	2.94	3.81	0.053	1.02	0.158	7.95	14.28
pretrained	Log-cosh	**2.65**	**3.43**	**0.048**	**0.91**	**0.133**	**7.15**	**12.85**
From scratch	r^2	3.08	3.98	0.055	1.07	0.165	8.36	15.16
From scratch	$\frac{1}{2}r^2$	3.09	4.00	0.055	1.07	0.166	8.40	15.24
From scratch	Log-cosh	**2.71**	**3.50**	**0.049**	**0.93**	**0.136**	**7.30**	**13.12**

Table 17: Relative error reductions (%) of log-cosh, calculated separately for each driving metric and each quadratic baseline from the displayed results.

	pretrained		From scratch	
Metric	vs. r^2	vs. $\frac{1}{2}r^2$	vs. r^2	vs. $\frac{1}{2}r^2$
Vehicle MAE	9.6	9.9	12.0	12.3
Pedestrian MAE	9.5	10.0	12.1	12.5
Bicycle MAE	9.4	9.4	10.9	10.9
Traffic-cone MAE	10.8	10.8	13.1	13.1
Barrier MAE	15.8	15.8	17.6	18.1
ADE (m)	9.6	10.1	12.7	13.1
FDE (m)	9.6	10.0	13.5	13.9

E PHYSICAL-STATE PROBING AND VISUALIZATION

E.1 PHYSICAL-STATE PROBE PROTOCOL

The PushT evaluation uses a frozen encoder with linear and MLP probes for agent location, block location, and block orientation. The protocol uses a fixed train–validation–test split, scales spatial targets to $[-1, 1]$, and encodes orientation as $(\cos\theta, \sin\theta)$. Probes are trained with MSE using Adam and validation-based stopping. The reported MSE variability is across three probe-initialization seeds; it does not describe variation across independently trained world models.

E.2 COMPLETE PROBE RESULTS

The tables below retain both MSE and the reported correlation r for all readouts. Log-cosh gives lower MSE and higher correlation for all six target–readout combinations than either quadratic. The gains under both readout classes indicate that the measured physical quantities are more accessible to these probes. This does not by itself determine whether the planner's native latent distance has suitable geometry.

Table 18: Linear probe results on PushT. MSE ↓ and correlation r ↑. The reported ± values describe variability across three probe-initialization seeds; best values are bold.

	r^2		$\frac{1}{2}r^2$		Log-cosh	
Target	MSE	r	MSE	r	MSE	r
Agent location	0.0513 ± 0.0051	0.9743	0.0497 ± 0.0047	0.9751	**0.0383** ± 0.0039	**0.9817**
Block location	0.0287 ± 0.0035	0.9857	0.0263 ± 0.0033	0.9871	**0.0203** ± 0.0027	**0.9903**
Block orientation	0.1823 ± 0.0173	0.9053	0.1733 ± 0.0167	0.9097	**0.1373** ± 0.0137	**0.9277**

Table 19: MLP probe results on PushT. MSE ↓ and correlation r ↑. The reported ± values describe variability across three probe-initialization seeds; best values are bold.

	r^2		$\frac{1}{2}r^2$		Log-cosh	
Target	MSE	r	MSE	r	MSE	r
Agent location	0.0041 ± 0.0005	0.9979	0.0038 ± 0.0004	0.9981	**0.0024** ± 0.0003	**0.9993**
Block location	0.0011 ± 0.0002	0.9991	0.0010 ± 0.0002	0.9992	**0.0007** ± 0.0001	**0.9995**
Block orientation	0.0207 ± 0.0021	0.9897	0.0193 ± 0.0023	0.9907	**0.0143** ± 0.0017	**0.9933**

Table 20: Relative probe-MSE reductions (%) of log-cosh from the displayed means. No new runs or uncertainties are inferred.

Target	Readout	vs. r^2	vs. $\frac{1}{2}r^2$
Agent location	Linear	25.3	22.9
Agent location	MLP	41.5	36.8
Block location	Linear	29.3	22.8
Block location	MLP	36.4	30.0
Block orientation	Linear	24.7	20.8
Block orientation	MLP	30.9	25.9

E.3 SCOPE OF THE VISUALIZATIONS

The main-paper visualizations examine complementary quantities. The maze heatmaps show squared latent distances from encoded observed states to a goal, without dynamics rollout error. The action-space figures examine a planning objective that composes the encoder, learned dynamics, and goal cost, and compare it with a true-dynamics reference. These visualizations are qualitative; the numerical planning and probe results provide the corresponding task and readout measurements.

F Analytical Details

Stable evaluation and scalar weights. An algebraically equivalent, numerically stable expression for log-cosh is

$$\log\cosh r = |r| + \mathrm{softplus}(-2|r|) - \log 2, \qquad \mathrm{softplus}(x) = \log(1 + e^x). \tag{12}$$

This avoids forming $\cosh r$ directly. Relative to the locally matched penalty $\frac{1}{2}r^2$, log-cosh multiplies each residual score by $w(r) = \tanh(r)/r$, with the continuous extension $w(0) = 1$. The fractional attenuation is $\delta(r) = 1 - w(r)$.

Table 21: Analytically computed scalar values, rounded to six decimal places. These are evaluations of the stated functions, not measurements from trained models.

r	$\tanh(r)$	$w(r)$	$\delta(r)$
0	0.000000	1.000000	0.000000
0.5	0.462117	0.924234	0.075766
1	0.761594	0.761594	0.238406
2	0.964028	0.482014	0.517986
3	0.995055	0.331685	0.668315
5	0.999909	0.199982	0.800018

Simultaneous local descent. The main analysis studies how prediction contributes to decreasing the representation objective. A related criterion asks when the combined raw-gradient direction decreases both objectives to first order.

Proposition 2 (Raw-gradient common descent)**.** *At a fixed parameter state and common minibatch realization, let $p = \nabla_\vartheta \mathcal{P}$ and $h = \nabla_\vartheta \mathcal{G}$ be nonzero gradients of locally differentiable objectives, where $\mathcal{G} = \lambda\mathcal{R}$ includes the representation weight, with other parameters fixed and stopped targets held constant. Define*

$$\Gamma = \frac{\|h\|_2}{\|p\|_2}, \qquad C = \frac{\langle p, h\rangle}{\|p\|_2\|h\|_2}.$$

The direction $v = -(p + h)$ is a strict first-order descent direction for both objectives if and only if

$$1 + \Gamma C > 0 \qquad \textit{and} \qquad \Gamma + C > 0. \tag{13}$$

Proof. The directional derivatives are

$$D\mathcal{P}[v] = -\|p\|_2^2(1 + \Gamma C), \qquad D\mathcal{G}[v] = -\|p\|_2\|h\|_2(\Gamma + C).$$

Since both norms are positive, both derivatives are strictly negative exactly under the stated inequalities. □

This criterion concerns gradients before optimizer preconditioning, momentum, or clipping. It gives a local directional condition, without asserting a property of Adam trajectories or reporting measured gradient interactions.

G Compute and Implementation Cost

The described NAVSIM self-supervised training configuration runs for approximately 25 hours on sixteen A100 80 GB GPUs. Table 22 separates this recorded setup from quantities calculated from its batch size, update count, and approximate duration. These estimates apply to that training stage, not to the complete project or the unspecified pretraining and downstream stages.

Table 22: Compute for the described NAVSIM self-supervised run.

Quantity	Value	Basis
Devices	16 A100 80 GB	Recorded
Numerical precision	FP32	Recorded
Global batch	64 clips / 512 frames	Recorded
Epochs / updates per epoch	10 / 11,069	Recorded
Total updates	110,690	Recorded
Wall-clock duration	≈ 25 h	Recorded
Total frame presentations	56,673,280	Recorded
GPU-hours	≈ 400	Recorded
Average time per update	≈ 0.81 s	Recorded
Aggregate training throughput	≈ 630 frames/s	Recorded
Aggregate clip throughput	≈ 78.7 clips/s	Recorded

The elementwise loss replacement introduces zero additional trainable model parameters and requires no additional model forward pass. It changes the arithmetic used to evaluate the residual penalty and its derivative. This structural observation does not imply zero runtime or memory overhead; no measured overhead percentage is inferred from it.