LAMP: LYAPUNOV-INSPIRED POTENTIALS FOR LATENT MODEL PREDICTIVE CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent model predictive control ranks candidate actions using a learned model and a bootstrapped value function. During learning, errors in these components can distort candidate preferences, particularly when experience is limited. We introduce **LAMP**, a planning extension that learns an auxiliary state potential from reward-bin labels and uses it to penalize candidate trajectories. The potential uses a Lyapunov-inspired non-increase objective on sampled reward-improving transitions. It is trained on detached latents without Bellman-return targets. It guides planning through a scheduled hinge penalty. Our analysis characterizes the penalty's effects on candidate weights, conditions for reducing absolute and relative score errors, and limits of transferring reward supervision to planning queries. On 28 robotic control tasks from the DeepMind Control Suite, the reported results show improved learning performance over TD-MPC2, with the largest gains on Dog and Humanoid locomotion.

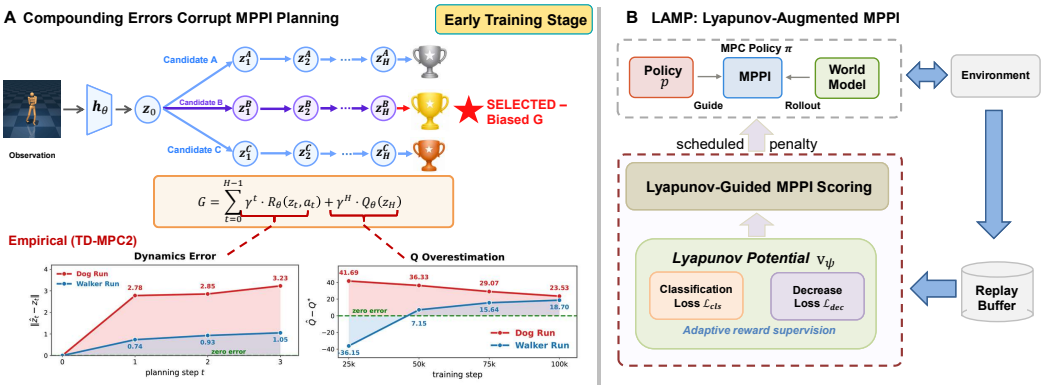


Figure 1: **LAMP and its motivation.** **(A)** Top: a hypothesized failure mode in which errors in learned rollouts and terminal values distort candidate preferences. Bottom: TD-MPC2 diagnostics on Dog Run and Walker Run. Latent-rollout error compares predicted latents with encodings of real states under the same action sequence. The value discrepancy compares $\hat{Q}$ with a Monte-Carlo continuation return under the current policy; the graphic's Q^* label denotes this empirical reference, not the optimal action value. These two examples motivate the design without establishing its mechanism. **(B)** An auxiliary potential learns from reward-bin labels and a sampled non-increase objective on detached latents. Its EMA target supplies a scheduled hinge penalty to MPPI candidate scores.

1 INTRODUCTION

Learning control policies from limited interaction is a central challenge in robotics. Model-based reinforcement learning addresses this challenge by using learned dynamics to support policy learning or online planning. TD-MPC and TD-MPC2 (Hansen et al., 2022; 2023) combine short latent rollouts with a learned terminal value, allowing receding-horizon control without a known dynamics model. Their practical appeal is that each action can be reconsidered using the latest model and

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.

observations. The same search procedure, however, can favor actions whose predicted returns are inaccurate.

The relevant error is a *difference between candidates*. A common offset in all predicted returns leaves the planner's softmax weights unchanged, whereas a smaller, candidate-dependent error can reverse a preference. Both rollout error and value miscalibration can contribute. Recent work also identifies a mismatch between planner-generated data and the policy used in value learning as a source of overestimation (Lin et al., 2025), and shows that search may fail even with accurate dynamics (Chang et al., 2026). These findings motivate interventions in how learned predictions are used, alongside improvements to the predictions themselves.

Figure 1A illustrates model and value errors during early TD-MPC2 training on Dog Run and Walker Run. These observations motivate our design; two diagnostic tasks do not establish a causal account of learning across the benchmark. We ask a narrower question: *can a state preference learned from immediate reward labels provide a useful additional signal for latent trajectory scoring, without changing the backbone's return-learning objectives?* Such a signal need not estimate the magnitude of the value error. It needs to distinguish candidates in a way that is useful for the control task, while avoiding excessive intervention.

We propose **LAMP** (**L**yapunov-**A**ugmented **M**odel-based **P**lanning). LAMP learns an auxiliary potential on detached latents using high- and low-reward transition labels. High-reward samples receive a low-potential target; low-reward samples receive a target above the planner's penalty threshold. A squared-norm network with positive-definite (PD) hidden layers and a sampled non-increase objective supply Lyapunov-inspired structural biases (Richards et al., 2018; Han et al., 2020). No goal state or equilibrium must be specified. At planning time, an EMA copy of the potential assigns a cumulative hinge penalty to each candidate rollout (Figure 1B). A warmup–ramp–decay schedule controls its influence during training.

Reward labels offer supervision without a Bellman-return target, but immediate reward is only a proxy for long-term utility. Necessary low-reward transitions may be penalized, and model error still enters through predicted latents. Our analysis distinguishes reward-bin calibration from absolute and relative score-error reduction. The PD construction and sampled decrease objective are design priors whose utility is assessed empirically, rather than stability certificates.

Our contributions are:

- **A reward-supervised planning extension.** LAMP combines a structured auxiliary potential with a scheduled penalty in TD-MPC2, retaining its world-model losses, value targets, and candidate sampler.
- **An analysis of candidate preferences.** We derive the penalty's effect on softmax weights and give distinct conditions for score calibration and pairwise score-error reduction, with proofs and explicit failure cases.
- **An evaluation in simulated continuous control.** The 28-task DMControl comparison and schedule/architecture ablations characterize LAMP on one backbone, with the strongest reported gains on high-dimensional locomotion.

2 Related Work

Models for control and policy learning. PETS (Chua et al., 2018) uses probabilistic dynamics ensembles for trajectory optimization, while PlaNet (Hafner et al., 2019) plans in a learned latent space. TD-MPC and TD-MPC2 combine latent planning with a terminal value (Hansen et al., 2022; 2023). A separate use of models is to generate experience for policy updates: Dyna (Sutton, 1990) established this approach, MBPO (Janner et al., 2019) uses short model rollouts, and DreamerV3 (Hafner et al., 2023) learns behavior in latent imagination. LAMP requires an explicit candidate-scoring interface; it is evaluated as an extension of TD-MPC2.

Overestimation and conservative planning. TD3 (Fujimoto et al., 2018) addresses actor–critic overestimation with clipped double Q-learning. CQL (Kumar et al., 2020) and MOPO (Yu et al., 2020) introduce conservatism in offline value learning and model-based policy optimization, respectively. More directly, TD-M(PC)^2 (Lin et al., 2025) constrains policy learning to reduce the

mismatch with planner-generated data. MRS.Q (Chang et al., 2026) uses the minimum over a value ensemble to address search-induced overestimation. EfficientTDMPC (Evers et al., 2026) combines multi-horizon return aggregation, ensemble estimates, and pessimistic reanalyze for policy targets. Dream-MPC (Spieler & Behnke, 2026) combines latent gradient-based planning with uncertainty regularization and action reuse. Thus neither modifying trajectory scores nor penalizing unreliable predictions is new by itself. LAMP's specific choice is a separate, reward-bin-supervised state potential, used in acting-time scoring while retaining the backbone's value targets. It does not estimate ensemble uncertainty or directly constrain the learned policy. These differences identify the method's scope; the present experiments do not establish superiority or complementarity to these recent alternatives.

Lyapunov-inspired structure. Lyapunov methods in RL use decrease conditions for safe learning, constraint satisfaction, or stability analysis (Perkins & Barto, 2002; Chow et al., 2018; 2019; Sikchi et al., 2021). Richards et al. (2018) learn a structured neural Lyapunov candidate for stability certification, while Han et al. (2020) use a Lyapunov critic in actor–critic control. LAMP borrows a squared-norm parameterization and a sampled decrease objective for a different role: ranking short learned rollouts. Its reward labels, unrestricted output projection, and masked non-increase loss do not satisfy the hypotheses needed for a classical certificate. The resulting potential is a learned planning preference.

Reward shaping and auxiliary learning. Potential-based reward shaping (Ng et al., 1999) uses a discounted difference of potentials to preserve optimal policies under its assumptions. LAMP instead subtracts a cumulative hinge penalty from planning scores; this term is not telescoping and can change the selected policy. It leaves the environment reward and TD targets unchanged. Unlike representation-learning auxiliaries such as CURL and SPR (Laskin et al., 2020; Schwarzer et al., 2020), the auxiliary's gradient stops at the latent input. This separates its optimization path from the encoder, while their data and prediction errors remain coupled.

3 PRELIMINARIES

3.1 LATENT-SPACE PLANNING WITH TD-MPC2

We consider an MDP $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$ and seek a policy π maximizing $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$. TD-MPC2 (Hansen et al., 2023) learns five components: an encoder $h_\theta : \mathcal{S} \to \mathcal{Z}$ ($\mathcal{Z} = \mathbb{R}^{512}$, SimNorm), latent dynamics $d_\theta : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$, reward predictor $R_\theta : \mathcal{Z} \times \mathcal{A} \to \mathbb{R}$, Q-ensemble $\{Q_\theta^i\}_{i=1}^5$, and policy prior $\pi_\theta : \mathcal{Z} \to \mathcal{A}$. At each step, MPPI (Williams et al., 2016) samples N action sequences, rolls each out for H steps in the latent space, and evaluates each trajectory by

$$G^{(j)} = \sum_{t=0}^{H-1} \gamma^t R_\theta(\mathbf{z}_t^{(j)}, \mathbf{a}_t^{(j)}) + \frac{\gamma^H}{|\mathcal{I}_{\text{sub}}|} \sum_{i \in \mathcal{I}_{\text{sub}}} Q_\theta^i(\mathbf{z}_H^{(j)}, \pi_\theta(\mathbf{z}_H^{(j)})), \tag{1}$$

where $\mathcal{I}_{\text{sub}} \subset \{1, \dots, 5\}$ with $|\mathcal{I}_{\text{sub}}| = 2$ is sampled for the terminal-value evaluation. TD-MPC2 uses the subsampled mean for planning and the subsampled minimum for TD targets (Hansen et al., 2023); the planned action is a softmax-weighted combination of the sampled sequences. For fixed candidate actions, their relative scores determine the softmax weights; a common score offset cancels. Candidate-dependent errors in Q_θ^i and d_θ can therefore change trajectory ranking and action selection. The analysis in Appendix D distinguishes score calibration from this relative-error requirement.

3.2 LYAPUNOV FUNCTIONS: BACKGROUND AND SCOPE

For a discrete-time system $x_{t+1} = f(x_t)$ with equilibrium $x^\star$, a classical Lyapunov argument uses a continuous function V that is zero at $x^\star$, positive elsewhere nearby, and non-increasing along trajectories. On a compact invariant sublevel set containing $x^\star$ in its interior, continuity of f and strict decrease away from $x^\star$ yield asymptotic stability within that set (Khalil, 2002). These are properties of the function and closed-loop dynamics together; positive-definite network weights alone are insufficient. LAMP borrows PD hidden layers and a sampled non-increase prior, but its positive output floor, unrestricted output projection, and reward-dependent mask do not establish the

classical conditions. In particular, high-reward latent regions are not verified equilibria. We therefore use "Lyapunov-inspired" to describe an inductive bias for scoring, with the precise structural scope stated in Lemma D.1 and Equation (37).

4 LAMP: LYAPUNOV-AUGMENTED MODEL-BASED PLANNING

LAMP adds two operations to TD-MPC2: learning a state potential from reward-labeled transitions (Section 4.1), and evaluating that potential along candidate rollouts (Section 4.2). The auxiliary's gradients stop at its latent inputs. Its training therefore leaves the backbone's model, reward, value, and policy losses unchanged; its effect on later experience comes through action selection. The complete update is given in Algorithm 1.

4.1 LYAPUNOV POTENTIAL LEARNING

Architecture. The auxiliary potential $v_\psi : \mathcal{Z} \to \mathbb{R}_{>0}$ is parameterized as

$$v_\psi(\mathbf{z}) = \|\phi_\psi(\mathbf{z})\|_2^2 + \varepsilon_{\text{out}}, \tag{2}$$

where ϕ_ψ is composed of two width-512 PD hidden layers followed by a width-256 linear output layer, with `tanh` activations throughout. For each PD layer, the weight matrix $\mathbf{W}_l \in \mathbb{R}^{d_l \times d_{l-1}}$ is reconstructed from two factor matrices:

$$\mathbf{W}_l \;=\; \begin{bmatrix} \mathbf{G}_l^{(1)\top}\mathbf{G}_l^{(1)} + \varepsilon_{\text{pd}}\,\mathbf{I}_{d_{l-1}} \\ \mathbf{G}_l^{(2)} \end{bmatrix}, \qquad \mathbf{G}_l^{(1)} \in \mathbb{R}^{d_{l-1}\times d_{l-1}}, \quad \mathbf{G}_l^{(2)} \in \mathbb{R}^{(d_l - d_{l-1})\times d_{l-1}}, \tag{3}$$

with $\varepsilon_{\text{pd}} = 10^{-6}$. For the square hidden layers used here, this restricts each linear map to be symmetric positive definite, with no hidden linear kernel; the bias-free $\tanh$ stack preserves the zero-preimage property (Richards et al., 2018). The learnable factors and unrestricted $512 \to 256$ output projection together determine ϕ_ψ. The output floor $\varepsilon_{\text{out}} = 10^{-6}$ guarantees positivity, separately from the PD hidden-layer constraint. It does not prevent a constant potential: setting the output projection to zero gives $v_\psi \equiv \varepsilon_{\text{out}}$. Lemma D.1 gives the precise norm and continuity bounds.

Reward-bin classification loss. Supervision comes from a binary **reward-bin** label. For a transition with immediate reward r_t, we assign label "HI" when $r_t \geq r_{\text{hi}}$ and "LO" when $r_t \leq r_{\text{lo}}$; in-between transitions are ignored. The thresholds adapt to an exponential moving average of recent rewards:

$$r_{\text{hi}} \;=\; \bar{r}_{\text{ema}} + \sigma_{\text{hi}} \cdot \hat{\sigma}_{\text{ema}}, \quad r_{\text{lo}} \;=\; \bar{r}_{\text{ema}} - \sigma_{\text{lo}} \cdot \hat{\sigma}_{\text{ema}}, \tag{4}$$

with $\bar{r}_{\text{ema}}, \hat{\sigma}_{\text{ema}}$ tracking the buffer's reward mean and standard deviation. The classifier regresses v_ψ toward fixed targets v_{hi} and v_{lo} with $v_{\text{hi}} < c < v_{\text{lo}}$ (specific values in Table 1), where c is the planner threshold of Section 4.2:

$$\begin{aligned} \ell_{\text{cls}}(t) \;=\; \tfrac{1}{2}\Big[& \mathbf{1}[r_t \geq r_{\text{hi}}] \cdot \big((v_\psi(\mathbf{z}_t) - v_{\text{hi}})^2 + (v_\psi(\mathbf{z}_{t+1}) - v_{\text{hi}})^2\big) \\ & + \mathbf{1}[r_t \leq r_{\text{lo}}] \cdot \big((v_\psi(\mathbf{z}_t) - v_{\text{lo}})^2 + (v_\psi(\mathbf{z}_{t+1}) - v_{\text{lo}})^2\big)\Big]. \end{aligned} \tag{5}$$

Equation (5) defines the loss for one transition; $\mathcal{L}_{\text{cls}} := \mathbb{E}_t[\ell_{\text{cls}}(t)]$ is its batch expectation. Both endpoints share the band label of r_t. This is supervision on transition occurrences, not a deterministic label for each state: the same latent may receive conflicting labels from different samples. The reversed ordering $v_{\text{hi}} < c < v_{\text{lo}}$ realizes the Lyapunov-style inversion in which high-reward states sit at low potential, with the planner threshold c of Section 4.2 separating the two bins. The anchors fix the supervised reference scale relative to c; they do not calibrate the potential in units of return error. As written, both indicators activate if thresholds coincide and the reward equals them, producing competing targets. If no indicator or decrease mask activates, its corresponding batch loss is zero. These sparse-label cases limit the information available to the auxiliary. Auxiliary training begins at T_{update} environment steps.

Reward-improving decrease loss. Neighboring transitions can assign different reward-bin targets to the same latent endpoint. To restrict the decrease prior to reward-improving samples, a **triple-mask** gate requires that (i) r_t is in the HI band, (ii) r_{t+1} is also in that band, and (iii) $r_{t+1} > r_t$:

$$\mathbb{M}(t) \;:=\; \mathbf{1}[r_t \geq r_{\mathrm{hi}}] \cdot \mathbf{1}[r_{t+1} \geq r_{\mathrm{hi}}] \cdot \mathbf{1}[r_{t+1} > r_t]. \tag{6}$$

The decrease objective is then

$$\mathcal{L}_{\mathrm{dec}} \;=\; \mathbb{E}_{(\mathbf{z}_t, \mathbf{z}_{t+1})}\Big[\, \mathbb{M}(t) \cdot \max\big(0,\; v_\psi(\mathbf{z}_{t+1}) - v_{\bar{\psi}}(\mathbf{z}_t)\big)\Big], \tag{7}$$

where $v_{\bar{\psi}}$ is an exponential-moving-average target with rate τ_{EMA}, held fixed during the gradient step. This regularizes positive increments on the active mask, but permits equal endpoint potentials and does not imply strict progress or stability (Equation (37)). The decrease loss is additionally disabled for the first K_{warm} Lyapunov updates (**decrease warmup**), so that classification can begin fitting its anchors before the decrease prior is imposed.

Training objective.

$$\mathcal{L}_{\mathrm{lyap}} = \lambda_{\mathrm{cls}}\, \mathcal{L}_{\mathrm{cls}} + \lambda_{\mathrm{dec}}\, \mathcal{L}_{\mathrm{dec}} + \lambda_{\mathrm{reg}}\, \mathbb{E}[v_\psi^2]. \tag{8}$$

The λ_{dec} coefficient controls the relative weight of the decrease objective (Appendix A.5); the trailing magnitude regularizer discourages large outputs on sampled latents. Neither term imposes a uniform bound on unseen inputs or future updates. v_ψ is trained on the *detached* latent states already computed during world-model updates, so it shares each batch with TD-MPC2 and adds no extra forward passes through h_θ or d_θ.

The complete auxiliary update is given in Algorithm 1 in Appendix A.

4.2 Lyapunov-Guided MPPI Scoring

Penalized MPPI score. We augment the MPPI trajectory value of Equation (1) with a hinge penalty on the latent rollout:

$$G^{(j)} \leftarrow G^{(j)} - \lambda\, \alpha \sum_{k=1}^{H} \max\big(0,\; v_{\bar{\psi}}(\mathbf{z}_k^{(j)}) - c\big), \tag{9}$$

with $\mathbf{z}_0^{(j)} := h_\theta(\mathbf{s}_t)$ and $\mathbf{z}_{k+1}^{(j)} := d_\theta(\mathbf{z}_k^{(j)}, \mathbf{a}_k^{(j)})$. The sum starts at $k = 1$ since $\mathbf{z}_0^{(j)}$ is shared across candidates and its offset cancels under the MPPI softmax. Unlike the reward sum, the penalty is *undiscounted*, assigning equal weight to potential excess at each rollout depth. The penalty uses the EMA target $v_{\bar{\psi}}$ (not the live v_ψ), matching the decrease objective's RHS and smoothing parameter updates; the target can nevertheless lag the live network (Lemma D.2). The hinge form $\max(0, v_{\bar{\psi}} - c)$ encodes a soft prior: trajectories below c incur no penalty, while those traversing high-$v_{\bar{\psi}}$ regions are penalized in proportion to their excess. At $k = H$ the penalty and the bootstrapped $Q_\theta^i(\mathbf{z}_H^{(j)}, \pi_\theta(\mathbf{z}_H^{(j)}))$ in Equation (1) act on different quantities—a state-level scalar versus a state-action expected return. Different targets may provide complementary information, but do not guarantee it.

Schedule. All step counts below are in environment steps, matching the x-axis convention of Figures 2 and 3. The strength α follows a four-phase schedule (warmup $\rightarrow$ ramp $\rightarrow$ decay $\rightarrow$ plateau) with milestones $(T_{\mathrm{enable}}, T_{\mathrm{ramp}}, T_{\mathrm{decay}})$ and floor α_{end}; the formal schedule is given in Equation (16), with values for λ, c, and the milestones listed in Table 1. The schedule gives the auxiliary time to fit its labels, then reduces its influence as training proceeds while retaining a residual penalty. This is an empirical design, not a consequence of a monotone optimal schedule; Proposition D.10 gives an explicit sufficient condition for decreasing MSE-optimal strength at fixed checkpoints. After each Lyapunov update, factor matrices are projected to $\|\mathbf{G}_l^{(1)}\|_F \leq c_F$, bounding the hidden-layer norms. The full network's Lipschitz bound also depends on its output projection (Lemma D.1).

4.3 WHAT CHANGES IN THE PLANNER?

Candidate preferences. Fix a checkpoint and a common set of N candidates with raw scores G_j, penalties P_j, and temperature $T > 0$. Write $\beta = \lambda\alpha$ and $w_j^\beta \propto \exp((G_j - \beta P_j)/T)$. Then

$$w_j^\beta = \frac{w_j^0 \exp(-\beta P_j/T)}{\sum_i w_i^0 \exp(-\beta P_i/T)}, \qquad \log \frac{w_i^\beta / w_j^\beta}{w_i^0 / w_j^0} = -\frac{\beta(P_i - P_j)}{T}. \tag{10}$$

Only penalty differences affect these weights. Consequently a useful intervention need not match the absolute magnitude of Q-error, and a constant potential cannot change preferences. This identity applies on a fixed support, including a fixed elite set; subsequent elite selection and resampling can also change the support.

Absolute and relative score error. Let $\Delta_j = G_j - G^\pi_{\mathrm{ref}}(q_j)$, where G^π_{ref} is the true candidate return under a specified continuation policy (Equation (17)). For a fixed query distribution, subtracting βP reduces MSE precisely when $\beta^2\mathbb{E}[P^2] < 2\beta\mathbb{E}[\Delta P]$ (Proposition D.5). For relative scores, instead let Var_u and Cov_u denote moments under the uniform distribution over the fixed candidates. The mean squared error in all pairwise score gaps is $2\mathrm{Var}_u(\Delta - \beta P)$. Its change equals

$$2\beta^2 \mathrm{Var}_u(P) - 4\beta \mathrm{Cov}_u(\Delta, P). \tag{11}$$

Thus relative-error reduction needs positive *centered* alignment and a suitable strength (Proposition D.8). Reducing either average error does not ensure correct ordering of every pair. Proposition D.7 states the additional margin conditions.

Design implications and scope. Reward-bin training supplies a possible source of candidate variation, not a direct estimate of Δ. Transferring that supervision to planning queries requires coverage and target tracking (Proposition D.4); PD layers do not provide these conditions. Increasing β cannot increase the weighted mean penalty on a fixed support (Proposition D.8), but may favor short-term reward over necessary low-reward transitions. The schedule controls this tradeoff empirically. The analysis explains which quantities matter for scoring; it does not establish that the reported learning curves satisfy the alignment conditions or that averaging candidate actions improves closed-loop return.

Relation to reward prediction and scale. The existing reward head predicts a state-action reward, whereas v_ψ aggregates transition-endpoint labels into a state-only preference with a non-increase regularizer. This difference in supervision motivates the extra head; it does not establish that the head is necessary. The potential's scale is also distinct from return units. For fixed G_j, replacing (v, c, β) by $(sv, sc, \beta/s)$ for $s > 0$ leaves every penalized score unchanged. Likewise, scaling (G_j, β, T) by a common positive factor leaves the weights in Equation (10) unchanged. Reward-bin normalization alone therefore does not make a fixed penalty coefficient insensitive to changes in return scale. These are scoring identities, not invariance claims about retraining the networks.

4.4 INTERPRETING AND CONTROLLING THE AUXILIARY PREFERENCE

What reward-bin supervision identifies. The two anchors define a state preference through the frequencies of observed labels. Freeze distinct reward thresholds and consider only the classification objective, with an unrestricted scalar predictor and without decrease or magnitude regularization. Let $\rho_{\mathrm{lo}}(z)$ be the probability of a low-reward label conditional on latent z and on the occurrence being labeled. Where labeled data have positive conditional mass, the population least-squares solution is

$$v^*_{\mathrm{cls}}(z) = v_{\mathrm{hi}} + (v_{\mathrm{lo}} - v_{\mathrm{hi}})\rho_{\mathrm{lo}}(z). \tag{12}$$

This follows by completing the conditional squared loss (Proposition D.11). Its hinge activates precisely when $\rho_{\mathrm{lo}}(z) > (c - v_{\mathrm{hi}})/(v_{\mathrm{lo}} - v_{\mathrm{hi}})$. With the anchors in Table 1, this threshold is $2/3$. Thus the idealized classifier encodes a frequency of low-reward occurrences among labeled samples, rather than the probability of failure or the expected return. Conflicting endpoint labels yield an intermediate optimum and a nonzero irreducible classification loss; additional optimization cannot remove that ambiguity. The finite network, shared parameters, regularizers, and EMA can all move the learned output away from this pointwise solution.

For control, the distinction matters when reward depends strongly on the action. A latent visited by both successful and unsuccessful actions inherits their mixture in the replay distribution. Likewise, assigning a transition's label to its successor does not identify the reward of every action available there. The model rollout makes the trajectory penalty action-dependent through its predicted successors, but the potential remains state-only. Its useful regime therefore requires local state preferences that remain informative across the actions and states considered by the planner.

A bound on the change in candidate weights. On the fixed support of Equation (10), let $R_P = \max_j P_j - \min_j P_j$. The total-variation distance satisfies

$$\mathrm{TV}(w^\beta, w^0) := \tfrac{1}{2} \sum_j |w_j^\beta - w_j^0| \le \min\{1, \beta R_P/(4T)\}. \tag{13}$$

Proposition D.12 proves the bound by differentiating the weights and bounding the mean absolute deviation of the penalties. The ratio $\beta R_P/T$ therefore controls a worst-case change in weights on the current candidates; an equal penalty has no effect regardless of its magnitude. If first-step candidate actions have diameter D_a in a chosen norm, their weighted means differ by at most $D_a\,\mathrm{TV}(w^\beta, w^0)$. This quantifies the local size of an intervention, without predicting its benefit. The bound applies before changes to the candidate support and gives no safety guarantee for executing the averaged action.

5 Experiments

We evaluate learning performance on the DeepMind Control Suite, followed by comparisons of the penalty schedule and network parameterization. These experiments assess the complete learning system; the score-error conditions in Section 4.3 are analytical conditions, not measurements from these learning curves.

5.1 Experimental Setup

Environments. We evaluate on the **DeepMind Control Suite** (DMControl) (Tunyasuvunakool et al., 2020), comprising 28 continuous-control tasks. Within these we examine a **7-task high-dimensional locomotion subset**—the four Dog and three Humanoid tasks—to study the reported performance on more demanding control problems. The per-seed training budget is $2{\times}10^6$ *environment steps*; the full task list and the precise environment-step convention are given in Appendix B.1.

Baselines. We compare LAMP against three continuous-control references spanning the dominant design points: **TD-MPC2** (Hansen et al., 2023), the direct architectural backbone of LAMP; **DreamerV3** (Hafner et al., 2023), a world-model baseline that learns behavior in imagination; and **SAC** (Haarnoja et al., 2018), a model-free maximum-entropy reference. Per-baseline algorithmic and implementation details are deferred to Appendix B.2.

Protocol and implementation. We run 5 seeds per task and report mean episode return with shaded 95% Student-t confidence intervals; no temporal smoothing is applied. LAMP is a drop-in extension of TD-MPC2 with shared hyperparameters across all 28 tasks; the full configuration is in Appendix A.5.

5.2 Core Empirical Results

Benchmark performance. Across the 28 DMControl tasks (Figure 2), the reported curves favor LAMP on aggregate, with task-dependent gains. Simple tasks (e.g., walker-stand, cup-catch) saturate early, leaving little room for separation: LAMP performs on par with TD-MPC2. On medium-difficulty tasks (e.g., cheetah-run, quadruped-run), LAMP converges earlier and ends higher than TD-MPC2, while DreamerV3 and SAC trail both. The largest gains concentrate on the high-dimensional locomotion subset—Dog and Humanoid embodiments—where TD-MPC2 has its slowest convergence and highest inter-seed variation. This pattern motivates examining the locomotion subset separately, without identifying the cause of its gains.

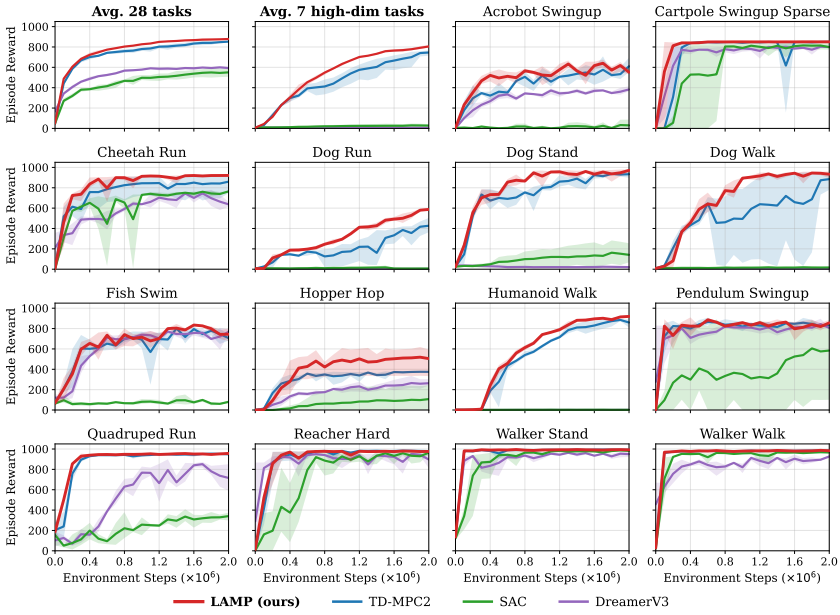


Figure 2: **DMControl tasks.** Comparing LAMP to baselines on DMControl tasks. In the top left, we present the average performance of all 28 tasks and the 7 high-dimensional locomotion tasks. Mean and 95% CIs over 5 seeds. Training curves for all tasks are provided in Appendix C.

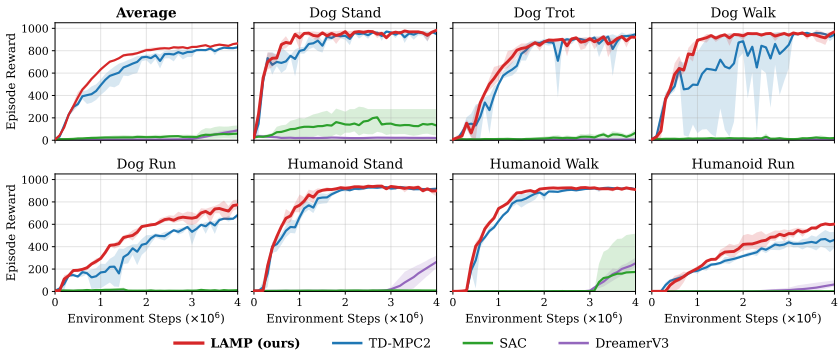


Figure 3: **High-dimensional locomotion tasks.** Comparison of LAMP with baselines on the 7 most challenging high-dimensional locomotion tasks; the environment steps are extended to 4M for a comprehensive comparison. In the top left, we present the results averaged over all 7 tasks. Mean and 95% CIs over 5 seeds.

High-dimensional locomotion. On the 7 high-dimensional locomotion tasks (Figure 3)—Dog Run, Stand, Trot, Walk and Humanoid Run, Stand, Walk—the aggregate curve shows earlier learning and higher final return for LAMP than TD-MPC2; the size and timing of the differences vary across tasks. DreamerV3 and SAC remain below the MPC-based methods in the aggregate comparison. These results support a benefit on this subset, but do not isolate model error, value error, or reward preference as its cause.

Summary. Figures 2 and 3 support improved aggregate learning performance for the evaluated LAMP configuration, with the clearest separation on high-dimensional locomotion. The results concern episode return, not direct calibration of candidate scores or OOD detection. The two-task diagnostics in Figure 1A provide motivation, rather than a measured relationship between error severity and improvement across all tasks.

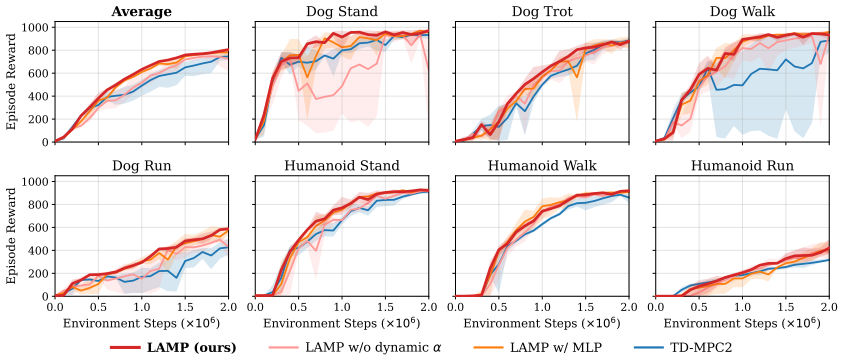


Figure 4: **Ablation studies on the high-dimensional locomotion subset.** Comparison of full LAMP against *LAMP w/o dynamic* α (constant penalty, no dynamic schedule) and *LAMP w/ MLP* (plain MLP ϕ_ψ in place of the PD-parameterized network). Mean and 95% CIs over 5 seeds. The curves compare learning trajectories under these two changes to the full configuration.

Wall-clock overhead. On a single NVIDIA RTX 4090, reaching 500k environment steps takes 13% longer for LAMP than TD-MPC2 on walker-walk and humanoid-run—modest training-time overhead relative to the data-efficiency gains on the high-dimensional locomotion subset.

5.3 Ablation Studies

We compare two variants on the high-dimensional locomotion subset. **LAMP w/o dynamic** α replaces the warmup–ramp–decay schedule with a constant penalty strength, disabling the scheduled reduction in auxiliary strength. **LAMP w/ MLP** swaps the PD-parameterized ϕ_ψ for a width- and depth-matched plain MLP, removing the PD hidden-layer constraint. Positivity depends on the output form in Equation (2), not on whether the hidden layers are PD. All other settings match Section 5.1.

Effect of the schedule on α**.** Figure 4 shows that the constant-α variant is competitive early on, with later differences in return and variability across tasks. Reducing the penalty limits its late-training influence, a possible explanation of this pattern. The comparison does not establish optimality of the schedule or the moment condition in Proposition D.10.

Effect of the PD parameterization. The MLP comparison examines the effect of the hidden-layer parameterization. The performance differences do not prove that PD layers prevent collapse or identify OOD states: even the full network can be constant (Lemma D.1). These comparisons concern the schedule and hidden layers; they do not separately identify the contribution of the decrease loss.

6 Conclusion

LAMP augments latent MPC with a reward-supervised state potential and a scheduled trajectory penalty. It offers an additional candidate preference while retaining TD-MPC2's model and return-learning objectives. The analysis separates reward-bin calibration, absolute score error, and relative candidate error, explaining why a downward score correction alone does not ensure better control. The label-mixture analysis identifies the idealized supervised target, while a finite-candidate bound quantifies changes in weights and action means as the penalty strength increases. The reported DMControl results support the utility of this design on the evaluated backbone, particularly for Dog and Humanoid locomotion. They do not establish that the potential detects OOD states or that the decrease prior independently causes the gains. LAMP is most plausibly useful when local reward structure informs short-horizon trajectory quality; delayed rewards, model extrapolation, and interaction with other conservative planners remain important limits (Appendix E).

AI use statement

We used large language models (LLMs) solely to assist with language editing and polishing of author-written text, including grammar, wording, clarity, and readability. The authors reviewed all AI-assisted revisions and take full responsibility for the final content of this paper, including its methods, results, and claims.

REFERENCES

Wei-Di Chang, Mikael Henaff, Brandon Amos, Gregory Dudek, and Scott Fujimoto. The surprising difficulty of search in model-based reinforcement learning. *arXiv preprint arXiv:2601.21306*, 2026.

Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A Lyapunov-based approach to safe reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Yinlam Chow, Ofir Nachum, Aleksandra Faust, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*, 2019.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Thomas Evers, Cristian Meo, Wendelin Bohmer, Justin Dauwels, and Yaniv Oren. EfficientTDMPC: Improved MPC objectives for sample-efficient continuous control. *arXiv preprint arXiv:2605.16692*, 2026.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019.

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

Minghao Han, Lixian Zhang, Jun Wang, and Wei Pan. Actor-critic reinforcement learning for control with stability guarantee. *IEEE Robotics and Automation Letters*, 5(4):6217–6224, 2020.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

Hassan K. Khalil. *Nonlinear Systems*. Prentice Hall, Upper Saddle River, NJ, 3 edition, 2002.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In *International conference on machine learning*, pp. 5639–5650. PMLR, 2020.

Haotian Lin, Pengcheng Wang, Jeff Schneider, and Guanya Shi. TD-M(PC)^2: Improving temporal difference MPC through policy constraint. *arXiv preprint arXiv:2502.03550*, 2025.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 278–287. Morgan Kaufmann, 1999.

Theodore J Perkins and Andrew G Barto. Lyapunov design for safe reinforcement learning. *Journal of Machine Learning Research*, 3(Dec):803–832, 2002.

Spencer M Richards, Felix Berkenkamp, and Andreas Krause. The Lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Conference on robot learning*, pp. 466–476. PMLR, 2018.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

Harshit Sikchi, Wenxuan Zhou, and David Held. Lyapunov barrier policy optimization. *arXiv preprint arXiv:2103.09230*, 2021.

Jonathan Spieler and Sven Behnke. Dream-MPC: Gradient-based model predictive control with latent imagination. *arXiv preprint arXiv:2605.04568*, 2026.

Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pp. 216–224. Elsevier, 1990.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 1433–1440. IEEE, 2016.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based offline policy optimization. *Advances in neural information processing systems*, 33:14129–14142, 2020.

A Implementation Details

This section consolidates the implementation choices that were abbreviated in Sections 4 and 5.1. We focus on what is needed to reproduce LAMP from a TD-MPC2 codebase; per-equation derivations are not repeated here.

A.1 World-model backbone (TD-MPC2)

The world model is the unmodified TD-MPC2 stack of Hansen et al. (2023): encoder $h_\theta : \mathcal{S} \to \mathcal{Z}$ ($\mathcal{Z} = \mathbb{R}^{512}$ with SimNorm normalization), latent dynamics d_θ, reward predictor R_θ, a five-network Q-ensemble $\{Q_\theta^i\}_{i=1}^5$ trained with subsampled clipped double-Q targets, and a stochastic policy prior π_θ. LAMP introduces no change to these modules, to the consistency, reward, value, or termination losses, or to the planner's MPPI sampler; we refer to Hansen et al. (2023) for the full specification of these components. The Lyapunov auxiliary update of Algorithm 1 runs on *detached* latent rollouts $\mathbf{z}_{0:H}$ that are already materialized inside the TD-MPC2 update for the consistency loss, and shares the same minibatch as the world-model and actor-critic updates; this couples the auxiliary's training distribution to the world model's without adding any forward pass through h_θ or d_θ.

Algorithm 1: Lyapunov potential update

Input: detached latent batch $\mathbf{z}_{0:H-1}$, rewards $r_{0:H-1}$, parameters ψ, target $\bar{\psi}$
Globals : counter k_{lyap}, decrease warmup K_{warm}

Form transition tuples $\mathcal{T} = \{(\mathbf{z}_t, \mathbf{z}_{t+1}, r_t, r_{t+1}) : t = 0, \dots, H-2\}$
Update reward EMA; compute thresholds $r_{\text{hi}}, r_{\text{lo}}` // Equation (4)
foreach *transition in* $\mathcal{T}$ **do**
  $\mathbb{M}(t) \leftarrow \mathbf{1}[r_t \geq r_{\text{hi}}] \cdot \mathbf{1}[r_{t+1} \geq r_{\text{hi}}] \cdot \mathbf{1}[r_{t+1} > r_t]$ `// Equation (6)`
end
$\mathcal{L}_{\text{cls}} \leftarrow$ classification loss `// Equation (5)`
if $k_{\text{lyap}} \geq K_{\text{warm}}$ **then**
  $\mathcal{L}_{\text{dec}} \leftarrow$ decrease loss `// Equation (7)`
else
  $\mathcal{L}_{\text{dec}} \leftarrow 0$
end
$\mathcal{L}_{\text{lyap}} \leftarrow \lambda_{\text{cls}} \mathcal{L}_{\text{cls}} + \lambda_{\text{dec}} \mathcal{L}_{\text{dec}} + \lambda_{\text{reg}} \mathbb{E}[v_\psi^2]$ `// Equation (8)`
Backpropagate $\mathcal{L}_{\text{lyap}}$ through ψ only; clip gradients and take an Adam step
Project the PD factors to their Frobenius balls `// Equation (14)`
$\bar{\psi} \leftarrow (1 - \tau_{\text{EMA}})\bar{\psi} + \tau_{\text{EMA}}\psi$
$k_{\text{lyap}} \leftarrow k_{\text{lyap}} + 1$

A.2 Lyapunov auxiliary architecture

Layer construction. Following Richards et al. (2018), every PD layer reconstructs its weight matrix from two factor matrices $\{\mathbf{G}_l^{(1)}, \mathbf{G}_l^{(2)}\}$ as in Equation (3) of Section 4.1. Both PD hidden layers use width 512, so $\mathbf{G}_l^{(2)}$ is empty for those layers and $\mathbf{W}_l$ reduces to $\mathbf{G}_l^{(1)\top}\mathbf{G}_l^{(1)} + \varepsilon_{\text{pd}}\mathbf{I}$ with $\mathbf{G}_l^{(1)} \in \mathbb{R}^{512\times512}$ and $\varepsilon_{\text{pd}} = 10^{-6}$. The $512 \to 256$ output stage is a standard linear projection (not PD-constrained); the strict-positivity floor $v_\psi \geq \varepsilon_{\text{out}}$ is supplied by the squared ℓ_2 form in Equation (2). The $\tanh$ activation has trivial zero preimage and is globally 1-Lipschitz. Together with the PD layers, the first property holds for the bias-free hidden stack (Richards et al., 2018), but need not hold after the unrestricted output projection. Bounds for the full network must include the output-layer norm (Lemma D.1). The PD parameterization admits only non-decreasing widths $d_l \geq d_{l-1}$; width-512 hidden layers are the simplest choice within this admissible set, and dimension reduction is delegated to the non-PD output layer to avoid breaking the block structure.

Initialization. We initialize each $\mathbf{G}_l^{(1)}$ entry-wise from $\mathcal{N}(0, 0.01)$ and use PyTorch defaults for the linear output stage. Under this scheme the expected spectral norm of $\mathbf{G}_l^{(1)^\top}\mathbf{G}_l^{(1)}$ is $\mathcal{O}(d \cdot 0.01^2) = \mathcal{O}(5\times10^{-2})$ at d=512, so each $\mathbf{W}_l$ at step zero is a full-rank symmetric positive-definite matrix of small magnitude. In the small-activation regime, replacing the hidden $\tanh$ maps by their linearizations yields a local squared-affine approximation to v_ψ; the actual network is non-linear and its sublevel sets need not be ellipsoids or bounded target neighborhoods. Total learnable parameter count is ≈ 0.66M: the two PD hidden layers contribute $2\times512^2 \approx 0.52$M and the linear output projection $512\times256 \approx 0.13$M.

A.3 Training pipeline integration

Optimizer and schedule. v_ψ is trained by Adam with learning rate 3×10^{-4} and weight decay 10^{-5}, separate from the world-model and actor-critic optimizers. Gradients are clipped to a global norm of 10.0 before each step. Updates begin at $T_{\text{update}} = 12\,\text{k}$ environment steps (corresponding to 6 k internal Lyapunov updates; the conversion factor is the DMControl action repeat of 2); classification supervision generated before this point is discarded. The decrease loss $\mathcal{L}_{\text{dec}}$ is held at zero for the first $K_{\text{warm}} = 150$ Lyapunov updates after T_{update}, so that the classification and magnitude terms train the auxiliary before the decrease prior is imposed.

Frobenius projection. After every Adam step, each PD-layer factor matrix is hard-projected to the Frobenius ball $\|\mathbf{G}_l^{(1)}\|_F \le c_F$:

$$\mathbf{G}_l^{(1)} \;\leftarrow\; \mathbf{G}_l^{(1)} \cdot \min\bigl(1,\; c_F/\|\mathbf{G}_l^{(1)}\|_F\bigr). \tag{14}$$

Combined with Equation (3), this bounds the hidden-layer spectral norms. It does not constrain the non-PD output projection or guarantee a uniform output scale across training. Lemma D.1 includes the output parameters in a bound at each checkpoint. Classification anchors, rather than this projection, provide the supervised reference scale relative to c.

EMA target network. The decrease loss in Equation (7) and the planner penalty in Equation (9) both consume an EMA-averaged copy $v_{\bar\psi}$ of v_ψ, parameterized by ψ and updated after every Adam step on ψ as

$$\bar\psi \;\leftarrow\; (1-\tau_{\text{EMA}})\,\bar\psi + \tau_{\text{EMA}}\,\psi, \qquad \tau_{\text{EMA}} = 0.05. \tag{15}$$

The live network v_ψ enters $\mathcal{L}_{\text{cls}}$, while the target $v_{\bar\psi}$ supplies the right-hand side of the decrease objective and the value scored inside MPPI. The target is detached during a gradient update, but remains coupled to the live network through EMA. This smooths parameter changes without guaranteeing small output changes; Lemma D.2 bounds tracking error under explicit parameter-drift and sensitivity assumptions.

Reuse of latents. The detached latent rollout $\mathbf{z}_{0:H}$ on which v_ψ is trained is the same rollout already produced by the world-model update for the consistency loss; the auxiliary therefore adds no forward pass through h_θ or d_θ. Only the v_ψ forward pass and its gradient are extra cost, which is dominated by the 2×512^2 matrix multiplications in the PD stack. Wall-clock overhead relative to TD-MPC2 is $< 15\%$ on all DMControl tasks we measured.

Numerical robustness. If any element of ψ becomes non-finite after a step (rare; we have not observed it on the reported DMControl runs), ψ and $\bar\psi$ are reinitialized from the same scheme as at step zero. This safeguard is active throughout training but contributed nothing in the experiments reported in Sections 5.2 and 5.3.

A.4 Planning-time penalty

Step-counter semantics. All step counts in this section are reported in environment steps, in line with Section 4.2 and the x-axis convention of Figures 2 and 3. Internally, the agent maintains an update counter incremented once per gradient step (see Algorithm 1); this counter advances at half the rate of environment steps under DMControl's action repeat of 2, so an internal value of N corresponds to $2N$ environment steps. The 2×10^6 environment-step training budget reported

in Section 5.1 therefore corresponds to 1×10^6 internal updates and lies far inside the $\alpha_{\text{end}} = 0.1$ plateau.

Schedule, formalized. Letting t denote the environment step, the schedule used in Equation (9) is

$$\alpha(t) \;=\; \begin{cases} 0, & t < T_{\text{enable}}, \\ \dfrac{t - T_{\text{enable}}}{T_{\text{ramp}}}, & T_{\text{enable}} \le t < T_{\text{enable}} + T_{\text{ramp}}, \\ 1 - (1 - \alpha_{\text{end}})\,\text{clip}\Big(\dfrac{t - T_{\text{enable}} - T_{\text{ramp}}}{T_{\text{decay}}},\, 0,\, 1\Big), & t \ge T_{\text{enable}} + T_{\text{ramp}}, \end{cases} \tag{16}$$

with T_{enable}=30 k, T_{ramp}=4 k, T_{decay}=200 k, and α_{end}=0.1, all expressed in environment steps (equivalently T_{enable}=15 k, T_{ramp}=2 k, T_{decay}=100 k internal updates). The plateau begins at $t = T_{\text{enable}} + T_{\text{ramp}} + T_{\text{decay}} = 234$ k environment steps and is held flat through the end of training.

Interaction with the MPPI softmax. The MPPI softmax temperature is unchanged from Hansen et al. (2023); the Lyapunov term enters Equation (9) as a per-trajectory scalar offset, leaving the temperature operating on its original return scale. Because $\mathbf{z}_0^{(j)}$ is shared across candidates, the k=0 contribution to the penalty is a constant offset that cancels under the softmax, which is why the sum in Equation (9) starts at k=1.

A.5 HYPERPARAMETERS

Table 1 groups every LAMP-specific hyperparameter. Hyperparameters not listed below are inherited unchanged from TD-MPC2 (Hansen et al., 2023) and are not re-tuned per task.

Table 1: Complete LAMP hyperparameter specification. All values are constant across the 28 DM-Control tasks; entries marked "TD-MPC2 default" inherit the published settings of Hansen et al. (2023).

Component	Symbol	Value	Notes
Auxiliary architecture			
PD hidden widths	d_1, d_2	$512, 512$	non-decreasing widths required by PD construction
Output projection width	d_3	256	non-PD linear stage; defines $\phi_\psi : \mathcal{Z} \to \mathbb{R}^{256}$
Activation	—	$\tanh$	trivial nullspace, 1-Lipschitz
PD regularizer	$\varepsilon_{\mathrm{pd}}$	10^{-6}	Equation (3)
Output floor	$\varepsilon_{\mathrm{out}}$	10^{-6}	Equation (2)
Parameter count	—	$\approx 0.66\,\mathrm{M}$	two PD layers + linear output
Auxiliary training			
Classification weight	λ_{cls}	5.0	Equation (8)
Decrease weight	λ_{dec}	0.5	Equation (8)
Magnitude regularizer	λ_{reg}	10^{-6}	Equation (8)
Decrease warmup	K_{warm}	150 updates	auxiliary update counter, not env steps; $\mathcal{L}_{\mathrm{dec}}$ disabled before this
Auxiliary start	T_{update}	12 k env steps	$= 6\,\mathrm{k}$ internal updates; no auxiliary updates before this
Optimizer	—	Adam	separate from world-model optimizer
Learning rate	—	3×10^{-4}	
Weight decay	—	10^{-5}	
Gradient clip (norm)	—	10.0	global before Adam step
Frobenius cap	c_F	3.0	hard projection on $\|\mathbf{G}_l^{(1)}\|_F$ via Equation (14)
Target EMA rate	τ_{EMA}	0.05	Equation (15)
Reward binning			
Hi-band sigma	σ_{hi}	0.5	Equation (4)
Lo-band sigma	σ_{lo}	0.5	Equation (4)
Reward EMA rate	—	0.99	momentum on $\bar{r}_{\mathrm{ema}}, \hat{\sigma}_{\mathrm{ema}}$
Hi-band anchor	v_{hi}	$0.5\,c$	target in Equation (5)
Lo-band anchor	v_{lo}	$1.25\,c$	target in Equation (5)
Planning penalty			
Penalty weight	λ	0.4	Equation (9)
Threshold	c	1.0	Equation (9); fixed, not re-estimated
Schedule milestones	—	$0/30\mathrm{k}/34\mathrm{k}/234\mathrm{k}$ env steps	warmup / ramp start / ramp end / plateau start
Schedule enable	T_{enable}	30 k env steps	start of ramp ($= 15\,\mathrm{k}$ internal updates)
Schedule ramp duration	T_{ramp}	4 k env steps	linear ramp $0 \to 1$ ($= 2\,\mathrm{k}$ internal updates)
Schedule decay duration	T_{decay}	200 k env steps	linear decay $1 \to \alpha_{\mathrm{end}}$ ($= 100\,\mathrm{k}$ internal updates)
Schedule plateau coefficient	α_{end}	0.1	residual strength
TD-MPC2 settings (not modified)			
Latent dimension	—	512	TD-MPC2 default
Planning horizon	H	3	TD-MPC2 default
Q-ensemble size	—	5	TD-MPC2 default
Subsampled clipped double Q	$\lvert\mathcal{I}_{\mathrm{sub}}\rvert$	2	TD-MPC2 default
MPPI candidates / temperature	—	—	TD-MPC2 default
Discount γ	—	—	TD-MPC2 default

B Environments and Baselines

B.1 DeepMind Control Suite

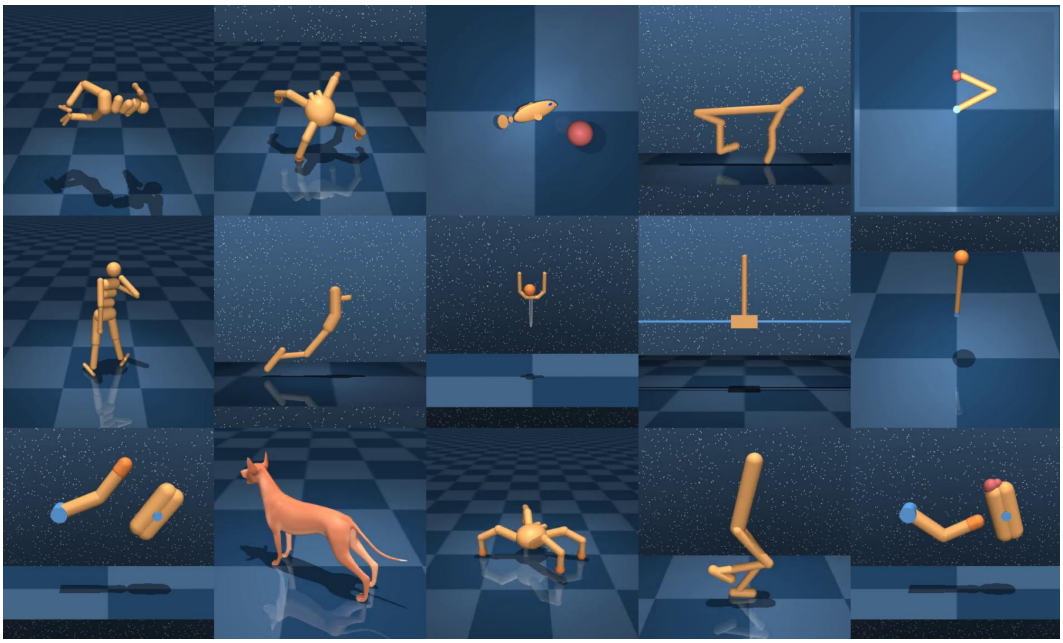


Figure 5: Snapshots of the 28 DeepMind Control Suite tasks evaluated in this paper.

The 28 evaluation tasks span 11 domains (Acrobot, Ball-in-cup, Cartpole, Cheetah, Cup, Dog, Finger, Fish, Hopper, Humanoid, Pendulum, Point-mass, Quadruped, Reacher, Swimmer, Walker), with continuous state observations of dimension between 4 (Cartpole) and 223 (Dog), continuous action spaces of dimension between 1 (Cartpole) and 38 (Dog), and a fixed episode length of 1,000 steps before action repeat. All tasks are run with action repeat 2, matching the convention used by TD-MPC2 (Hansen et al., 2023) and DreamerV3 (Hafner et al., 2023); this halves the inference frequency to 500 control decisions per episode and explains the 2×10^6 environment-step training budget translating to 1×10^6 agent decisions per seed. We use the standard *state-based* (proprioceptive features) observation modality across all tasks, identical to the configuration of TD-MPC2 in Hansen et al. (2023); pixel observations are not used by any of the four methods compared. The seven Dog and Humanoid tasks are isolated as the high-dimensional locomotion subset analyzed in Section 5.2; this subset matters because, as established in Section 1, joint Q-overestimation and latent-dynamics error have their longest early-training window precisely on these embodiments. The remaining 21 tasks span the easier-to-saturate end of DMControl and are reported in aggregate (and per-task in Appendix C) to show that LAMP does not regress on tasks where the failure mode it targets is mild.

B.2 Baselines

All four methods (LAMP and the three baselines) train for the same $2{\times}10^6$ environment-step budget per seed, are evaluated periodically using 10 deterministic rollouts per evaluation checkpoint, and use 5 random seeds; learning curves report mean and 95% Student-t confidence intervals matching Section 5.1. No temporal smoothing is applied to any curve.

SAC (Haarnoja et al., 2018). Soft actor-critic is a model-free off-policy maximum-entropy actor-critic that interleaves a soft Q-update against an entropy- regularized Bellman target with an actor update that maximizes $\mathbb{E}_\pi[Q(\mathbf{s}, \mathbf{a}) - \alpha \log \pi(\mathbf{a} \mid \mathbf{s})]$ under an automatically tuned entropy coefficient. We use the SAC implementation distributed with the TD-MPC2 codebase (Hansen et al., 2023), which follows the original SAC formulation of Haarnoja et al. (2018); it adopts standard architectural defaults (two-layer MLP critics, double-Q targets, Polyak averaging) and the same

DMControl wrapper as TD-MPC2, and no algorithmic changes are applied beyond matching the per-task evaluation protocol.

DreamerV3 (Hafner et al., 2023). DreamerV3 trains a recurrent state-space world model by reconstruction and runs an actor-critic in imagined latent rollouts: the actor is updated by REINFORCE with a learned baseline on the imagined returns, the critic by λ-target regression, and the world model under reconstruction, dynamics, and reward heads with the symlog/twohot transformations introduced by Hafner et al. (2023). We use the official DreamerV3 reference implementation released by the authors (Hafner et al., 2023) with default hyperparameters and the default DMControl proprio configuration; the 2×10^6 environment-step budget and 5-seed protocol match LAMP.

TD-MPC and TD-MPC2 (Hansen et al., 2022; 2023). TD-MPC2 is the direct backbone of LAMP and is described in Section 3.1; the comparison reported in Section 5.2 is therefore a controlled drop-in study—all hyperparameters of the world model, planner, actor, and Q-ensemble are held identical to TD-MPC2's defaults, and the only difference between the two methods is the addition of the auxiliary v_ψ and the penalty term in Equation (9). For completeness we note the design progression from TD-MPC (Hansen et al., 2022) to TD-MPC2 (Hansen et al., 2023): the latent space is expanded from 50 to 512 dimensions and SimNorm-normalized so that latent magnitudes remain bounded across tasks of differing state scale; an ensemble of five Q-networks replaces a single double-Q pair and is read out via subsampled clipped double-Q targets, removing a major source of TD-MPC's value-overestimation on high-dimensional embodiments; and the consistency loss is re-weighted to dominate the world-model update. We use the official TD-MPC2 reference implementation (Hansen et al., 2023) with default hyperparameters; LAMP modifies only the auxiliary potential and MPPI scoring as described in Section 4.

C All Training Curves

The complete per-task curves in Figure 6 complement the aggregate views in Section 5.2. They make task-dependent differences and overlap visible; an aggregate gain does not imply improvement on every task or at every point in training.

D Conditional Analysis of the Planning Penalty

This appendix separates three questions: what the parameterization guarantees, when reward-bin supervision transfers to planning queries, and when a nonnegative penalty improves a return estimate or its ranking. The results are conditional statements at a fixed training checkpoint. They do not assume that low reward identifies out-of-distribution (OOD) states, and do not establish stability, safety, or improved learning performance. In particular, the distribution-transfer and error-alignment conditions below are not consequences of the training objectives.

D.1 Reference returns and the evaluation distribution

Fix the world model, live and target auxiliary networks, and a continuation policy π. A query $q = (s_0, a_{0:H-1})$ consists of a true initial state and an open-loop candidate action sequence. Its reference return is defined in the original MDP:

$$G^{\pi}_{\mathrm{ref}}(q) := \mathbb{E}_T\left[\sum_{k=0}^{H-1}\gamma^k r(S_k,a_k) + \gamma^H V^{\pi}(S_H)\,\middle|\, S_0 = s_0, a_{0:H-1}\right]. \tag{17}$$

Here V^π uses the same reward, discount, and termination convention as the reference return; π is specified and need not be optimal. Bounded rewards and $0 \le \gamma < 1$ suffice for existence. This definition does not require the encoder to be injective or its latent representation to be Markov. A Monte-Carlo estimate must use the same continuation policy and account for any truncated tail.

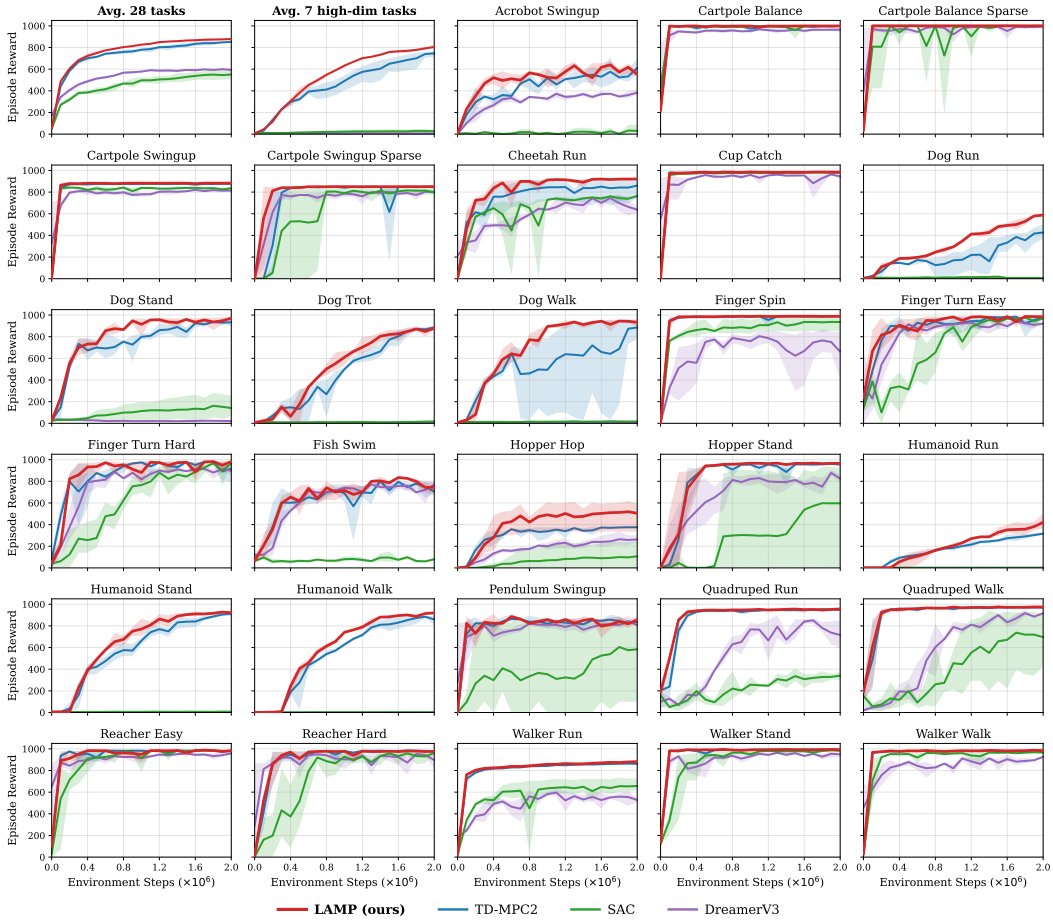


Figure 6: Per-task learning curves on all 28 DMControl tasks plus two aggregate panels (28-task average and 7-task high-dimensional locomotion subset). Mean and 95% Student-t confidence intervals over 5 seeds. The y-axis is episode return on a uniform $[0, 1050]$ range; the x-axis is environment steps in units of 10^6, evaluated up to 2×10^6 environment steps. No temporal smoothing is applied.

Write $\hat{z}_0 = h_\theta(s_0)$ and $\hat{z}_{k+1} = d_\theta(\hat{z}_k, a_k)$. Let $\widehat{G}(q)$ be the unpenalized score in Equation (1), including its actual Q-ensemble readout. Define

$$p(z) := (v_{\bar{\psi}}(z) - c)_+, \quad P(q) := \sum_{k=1}^{H} p(\hat{z}_k), \quad \Delta(q) := \widehat{G}(q) - G^{\pi}_{\mathrm{ref}}(q), \quad \Delta_\beta := \Delta - \beta P, \tag{18}$$

where $(x)_+ = \max(0, x)$ and $\beta = \lambda\alpha \geq 0$ is fixed during this comparison. Unless otherwise stated, expectations are over a specified, fixed distribution of queries and any ensemble or action randomness in their scores, conditional on the checkpoint. The same queries are used before and after adding the penalty. We assume finite second moments when discussing mean-squared error (MSE) or variance. No independence between P and Δ is required. Comparing two separately trained agents changes this distribution and is not covered by these fixed-checkpoint identities.

D.2 Structural properties and target-network tracking

For the two square PD hidden layers, write $W_l = G_l^\top G_l + \varepsilon_{\mathrm{pd}} I$ and $h(z) = \tanh(W_2 \tanh(W_1 z))$. The output projection has matrix A and bias b; set $b = 0$ if absent. The bounds below apply both to $\phi(z) = Ah(z) + b$ and to an additional final $\tanh$, since that activation is 1-Lipschitz and does not increase the norm.

Lemma D.1 (Positivity and checkpoint-dependent bounds)**.** *Each W_l is positive definite, and the bias-free hidden map h has trivial zero preimage. Nevertheless $v(z) = \|\phi(z)\|^2 + \varepsilon_{\text{out}}$ need not distinguish any two states. On $\|z\| \le Z_{\max}$, define*

$$a_l := \|G_l\|_F^2 + \varepsilon_{\text{pd}}, \qquad L_\phi := \|A\|_2 a_1 a_2, \qquad B_\phi := \|A\|_2 \min\{a_1 a_2 Z_{\max}, \sqrt{d_2}\} + \|b\|_2. \tag{19}$$

Then $\varepsilon_{\text{out}} \le v(z) \le V_{\max} := B_\phi^2 + \varepsilon_{\text{out}}$, and v is $L_v := 2B_\phi L_\phi$-Lipschitz on this set. The constants use the parameters of the network being evaluated, including the output layer; for planning, these are the target parameters.

Proof. For $x \neq 0$, $x^\top W_l x = \|G_l x\|^2 + \varepsilon_{\text{pd}}\|x\|^2 > 0$. Thus W_l is invertible. Coordinatewise $\tanh(u) = 0$ if and only if $u = 0$, proving the hidden-map statement by composition. Setting $A = 0$ and $b = 0$ gives the counterexample $v(z) \equiv \varepsilon_{\text{out}}$, even with both hidden matrices PD. Next, $\|W_l\|_2 \le \|G_l\|_2^2 + \varepsilon_{\text{pd}} \le a_l$. Since $\tanh$ is 1-Lipschitz and vanishes at zero, $\|h(z)\| \le a_1 a_2 \|z\|$; its coordinate bound also gives $\|h(z)\| \le \sqrt{d_2}$. The triangle inequality gives B_ϕ, and composition gives L_ϕ. Finally,

$$|v(z) - v(z')| \le (\|\phi(z)\| + \|\phi(z')\|)\|\phi(z) - \phi(z')\| \le 2B_\phi L_\phi \|z - z'\|.$$

□

Projection of G_l yields $a_l \le c_F^2 + \varepsilon_{\text{pd}}$ for a projected network, but gives no uniform bound on $\|A\|_2$ or $\|b\|_2$ across training. For SimNorm with m probability-simplex groups, $\|z\| \le \sqrt{m}$, since the squared norm of each group is at most one. These observations provide finite checkpoint bounds, not a training-uniform non-collapse or Lipschitz guarantee.

Lemma D.2 (EMA tracking under bounded parameter drift)**.** *Let $\bar{\psi}_n = (1-\tau)\bar{\psi}_{n-1} + \tau\psi_n$, $0 < \tau \le 1$, with $\|\psi_n - \psi_{n-1}\| \le d$. Then*

$$\|\bar{\psi}_n - \psi_n\| \le (1-\tau)^n \|\bar{\psi}_0 - \psi_0\| + \frac{(1-\tau)d}{\tau}\big(1 - (1-\tau)^n\big). \tag{20}$$

If $v_\psi(z)$ is K-Lipschitz in parameters between the live and target networks, uniformly over the query set, their output difference is at most K times the right-hand side.

Proof. Writing $e_n = \bar{\psi}_n - \psi_n$ gives $e_n = (1-\tau)(e_{n-1} + \psi_{n-1} - \psi_n)$. Apply the triangle inequality and sum the resulting geometric series. The output bound follows from the assumed parameter Lipschitz condition. □

The lemma requires bounds on actual parameter drift and parameter sensitivity; it does not assert that output tracking error is $O(\tau)$. In particular, smaller τ can increase lag. If the target factors are initialized within the same Frobenius ball as the projected live factors, convexity of that ball preserves the target factor bound under EMA; otherwise target norms must be bounded separately.

D.3 From reward-bin supervision to a planning penalty

Freeze the reward thresholds at the checkpoint. Let μ be a source distribution of latent–label pairs (Z, Y) used for classification, where $Y \in \{\text{HI}, \text{LO}, \text{MID}\}$. Labels attach to sampled transition endpoints, not to a deterministic reward function of Z: the same latent can occur with different labels. Let $a_{\text{HI}} = v_{\text{hi}} < c < a_{\text{LO}} = v_{\text{lo}}$. For $y \in \{\text{HI}, \text{LO}\}$, denote

$$\pi_y := \mu(Y = y) > 0, \qquad \varepsilon_y := \mathbb{E}_\mu[(v_\psi(Z) - a_y)^2 \mathbf{1}\{Y = y\}], \qquad \mu_y := \mathcal{L}_\mu(Z \mid Y = y). \tag{21}$$

These are distributional losses; a small minibatch loss alone does not establish them. If both reward indicators activate at a coincident threshold, the two supervised occurrences can be represented separately in μ with the weights of the classification loss. The incompatible anchors then preclude assuming an arbitrarily small residual.

Assumption D.3 (Explicit source-to-query transfer)**.** Let ν be a specified joint query distribution of (Z, Y), with the same label semantics. For each labeled class having positive ν mass, its conditional latent law ν_y is absolutely continuous with respect to μ_y, with $d\nu_y/d\mu_y \le C_y < \infty$ almost surely. Also $\|v_{\bar{\psi}} - v_\psi\|_{L^2(\nu_y)} \le \delta_y$.

This is a sufficient coverage and tracking condition, not an asserted property of OOD states or of the experiments. A joint query label can be defined using true transitions aligned with predicted latents for analysis; it is not observed by the planner. If μ is just a finite empirical buffer and query latents lie off its support, the absolute continuity condition generally fails. One then needs a justified population/generalization model or direct query-distribution error bounds; replay loss cannot replace either.

Proposition D.4 (Conditional calibration of the hinge penalty)**.** *Under Assumption D.3, put* $e_y := \sqrt{C_y \varepsilon_y / \pi_y} + \delta_y$ *and* $a := v_{\mathrm{lo}} - c > 0$*. For classes of positive query mass,*

$$\mathbb{E}_{\nu_{\mathrm{HI}}} p(Z) \le e_{\mathrm{HI}}, \qquad |\mathbb{E}_{\nu_{\mathrm{LO}}} p(Z) - a| \le e_{\mathrm{LO}}, \tag{22}$$

$$\mathbb{P}_{\nu_{\mathrm{HI}}}(v_{\bar{\psi}}(Z) \ge c) \le \min\left\{1, \frac{e_{\mathrm{HI}}^2}{(c - v_{\mathrm{hi}})^2}\right\}, \quad \mathbb{P}_{\nu_{\mathrm{LO}}}(v_{\bar{\psi}}(Z) \le c) \le \min\left\{1, \frac{e_{\mathrm{LO}}^2}{(v_{\mathrm{lo}} - c)^2}\right\}. \tag{23}$$

Suppose these conditions hold for each of the H query-step marginals, with common upper bounds e_y, and $0 \le p(Z) \le p_{\max}$. Let $N_y = \sum_{k=1}^{H} \mathbf{1}\{Y_k = y\}$ for any joint labeling with those marginals. Then

$$(a - e_{\mathrm{LO}})_+ \mathbb{E}N_{\mathrm{LO}} \le \mathbb{E}P \le (a + e_{\mathrm{LO}})\mathbb{E}N_{\mathrm{LO}} + e_{\mathrm{HI}}\mathbb{E}N_{\mathrm{HI}} + p_{\max}\mathbb{E}N_{\mathrm{MID}}. \tag{24}$$

Zero-mass classes contribute zero. No independence across steps is assumed.

Proof. The density bound gives

$$\|v_\psi - a_y\|^2_{L^2(\nu_y)} \le C_y \|v_\psi - a_y\|^2_{L^2(\mu_y)} = C_y \varepsilon_y / \pi_y.$$

Minkowski's inequality and target tracking imply $\|v_{\bar{\psi}} - a_y\|_{L^2(\nu_y)} \le e_y$. The function $u \mapsto (u-c)_+$ is 1-Lipschitz, with values zero and a at the HI and LO anchors. Thus $\mathbb{E}_{\nu_y}|p(Z) - (a_y - c)_+| \le \mathbb{E}_{\nu_y}|v_{\bar{\psi}}(Z) - a_y| \le e_y$ by Cauchy–Schwarz, proving Equation (22). Each threshold-crossing event requires an anchor error at least its stated margin; Markov's inequality on the squared error gives Equation (23). For each step, condition on its label. Use the LO lower bound and nonnegativity for the lower inequality, and the HI, LO, and MID upper bounds for the upper inequality. Sum over steps and use $\sum_k \mathbb{P}(Y_k = y) = \mathbb{E}N_y$. □

The dependence on class mass π_y, transfer factor C_y, and EMA error δ_y is essential. The result controls reward-bin calibration in expectation, not every candidate or all OOD regions. It also does not show that a well-calibrated penalty corrects a return error: that requires the separate alignment condition below.

D.4 When does subtracting the penalty improve a score?

Proposition D.5 (Exact bias, MSE, and variance conditions)**.** *Under the setup of Appendix D.1, write $b = \mathbb{E}\Delta$, $S = \mathbb{E}P$, $A_1 = \mathbb{E}[\Delta P]$, and $A_2 = \mathbb{E}[P^2]$. For any deterministic $\beta \ge 0$,*

$$\mathbb{E}\Delta_\beta = b - \beta S, \tag{25}$$

$$\mathbb{E}\Delta_\beta^2 - \mathbb{E}\Delta^2 = \beta^2 A_2 - 2\beta A_1, \tag{26}$$

$$\mathrm{Var}(\Delta_\beta) - \mathrm{Var}(\Delta) = \beta^2 \mathrm{Var}(P) - 2\beta \mathrm{Cov}(\Delta, P). \tag{27}$$

If $b > 0$ and $S > 0$, the absolute mean bias strictly decreases if and only if $0 < \beta S < 2b$. If $A_2 > 0$, a positive strength strictly reduces MSE if and only if $A_1 > 0$ and $0 < \beta < 2A_1/A_2$. The MSE-minimizing strength in $[0, \lambda]$ is

$$\beta^* = \mathrm{clip}(A_1/A_2, 0, \lambda). \tag{28}$$

If $A_2 = 0$, $P = 0$ almost surely and all strengths give the same score.

Proof. Substitute $\Delta_\beta = \Delta - \beta P$, expand the square, and take expectations. Center both random variables before the same expansion to obtain Equation (27). For $b > 0$, $|b - \beta S| < b$ is equivalent to $0 < \beta S < 2b$. For $A_2 > 0$ and $\beta > 0$, the MSE difference is negative exactly when $\beta A_2 < 2A_1$. Completing the square yields $A_2(\beta - A_1/A_2)^2 + \mathbb{E}\Delta^2 - A_1^2/A_2$, whose constrained minimum is the stated projection. If $A_2 = 0$, nonnegativity of P^2 gives $P = 0$ almost surely. □

Thus a downward score shift alone does not establish improvement. For instance, $\Delta = 1$ and $\beta P = 3$ increase absolute error. For nonpositive mean bias, any positive mean penalty cannot reduce its magnitude. Positive A_1 is an error-alignment requirement, not a consequence of positive-definite weights or reward classification. Likewise, variance reduction requires the covariance condition in Equation (27); it does not follow from MSE reduction.

Proposition D.6 (Bounded penalty variance)**.** *For a fixed target network satisfying Lemma D.1, set* $p_{\max} = (V_{\max} - c)_+$. *Then*

$$0 \le P \le H p_{\max}, \qquad \mathrm{Var}(\beta P) \le \frac{\beta^2 H^2 p_{\max}^2}{4}. \tag{29}$$

Proof. The range follows by summing H bounded nonnegative terms. For $X = \beta P \in [0, M]$, $M = \beta H p_{\max}$, one has $X^2 \le MX$. Hence $\mathrm{Var} X \le M\mathbb{E}X - (\mathbb{E}X)^2 \le M^2/4$ by completing the square. □

This bound requires no independence between rollout terms. It has no explicit Bellman resolvent factor because it bounds a finite-horizon penalty at a fixed checkpoint. It is not a comparison with Q-estimation variance, and does not imply independence from the discount through training. An unconditional bound over training randomness would also need uniform output-layer bounds. Nonzero TD-target variance, by itself, supplies no universal lower bound on the variance of the fitted Q-function.

D.5 Candidate ranking and softmax reweighting

Proposition D.7 (Pairwise ranking and fixed-candidate guarantees)**.** *Fix a finite candidate set and realized scores. Put* $g_j = G^{\pi}_{\mathrm{ref}}(q_j)$, $\tilde{g}_j = \widehat{G}(q_j) - \beta P_j$, *and* $e_j = \tilde{g}_j - g_j$. *The corrected pairwise gap is*

$$\tilde{g}_i - \tilde{g}_j = \widehat{G}(q_i) - \widehat{G}(q_j) + \beta(P_j - P_i). \tag{30}$$

If $g_i > g_j$ *but the uncorrected gap is* $-d \le 0$, *the pair is strictly ordered correctly precisely when* $\beta(P_j - P_i) > d$. *If there is a common offset* κ *with* $\max_j |e_j - \kappa| \le \eta$, *then any maximizer* $j_\beta \in \arg\max_j \tilde{g}_j$ *satisfies*

$$\max_j g_j - g_{j_\beta} \le 2\eta, \tag{31}$$

and every pair with true gap greater than 2η *is ordered correctly. For temperature* $T > 0$, *let* $w_j^\beta \propto \exp(\tilde{g}_j/T)$ *and* $w_j^* \propto \exp(g_j/T)$ *on this same support. Then*

$$\log \frac{w_i^\beta / w_j^\beta}{w_i^* / w_j^*} = \frac{e_i - e_j}{T}, \qquad \left| \log \frac{w_i^\beta / w_j^\beta}{w_i^* / w_j^*} \right| \le \frac{2\eta}{T}. \tag{32}$$

Proof. Subtract the two corrected scores to obtain Equation (30) and its sign condition. If $j^* \in \arg\max_j g_j$, maximality of $\tilde{g}_{j_\beta}$ implies $g_{j^*} - g_{j_\beta} \le e_{j_\beta} - e_{j^*} \le 2\eta$. Similarly, $g_i - g_j > 2\eta$ implies $\tilde{g}_i - \tilde{g}_j > 0$. In the softmax odds, each normalizing constant cancels, leaving $(e_i - e_j)/T$, bounded by $2\eta/T$. □

These results concern relative errors. Subtracting the same penalty from every candidate leaves ranking and weights unchanged, even if it reduces MSE. Conversely, a downward correction can worsen a correct ordering if it penalizes the better candidate more. Neither Proposition D.4 nor an average MSE reduction provides the uniform centered-error bound required in Proposition D.7. The fixed-support analysis also does not cover changes in elite selection, subsequent candidate resampling, or the return of a weighted average of actions in nonlinear dynamics. Consequently, Equation (31) is a candidate-selection statement, not a closed-loop MPPI performance guarantee.

Proposition D.8 (Centered errors and penalty reweighting)**.** *Fix* $N \ge 1$ *candidates, finite scores* G_j, *reference returns* g_j, *and penalties* P_j, *independently of the strength* β. *Set* $\Delta_j = G_j - g_j$ *and let* $\mathbb{E}_u$, Var_u, *and* Cov_u *be moments under the uniform distribution over these candidates. Define*

$$\mathcal{R}(\beta) := \frac{1}{N^2} \sum_{i,j=1}^{N} \big[(G_i - \beta P_i) - (G_j - \beta P_j) - (g_i - g_j) \big]^2. \tag{33}$$

Then $\mathcal{R}(\beta) = 2\mathrm{Var}_u(\Delta - \beta P)$ *and*

$$\mathcal{R}(\beta) - \mathcal{R}(0) = 2\beta^2 \mathrm{Var}_u(P) - 4\beta \mathrm{Cov}_u(\Delta, P). \tag{34}$$

For $\mathrm{Var}_u(P) > 0$*, a positive strength improves this criterion if and only if* $\mathrm{Cov}_u(\Delta, P) > 0$ *and* $0 < \beta < 2\mathrm{Cov}_u(\Delta, P)/\mathrm{Var}_u(P)$*. Its minimizer on* $[0, \lambda]$ *is* $\mathrm{clip}(\mathrm{Cov}_u(\Delta, P)/\mathrm{Var}_u(P), 0, \lambda)$*. If* $\mathrm{Var}_u(P) = 0$*, every pairwise gap is unchanged.*

For the weights w^β *of Equation* (10) *on this same support and fixed* $T > 0$*,*

$$\frac{d}{d\beta} \sum_j w_j^\beta P_j = -\frac{\mathrm{Var}_{w^\beta}(P)}{T} \leq 0. \tag{35}$$

This monotonicity concerns the penalty, not the true return.

Proof. For $e_j = \Delta_j - \beta P_j$, expansion gives $N^{-2} \sum_{i,j} (e_i - e_j)^2 = 2\mathbb{E}_u[e^2] - 2(\mathbb{E}_u e)^2 = 2\mathrm{Var}_u(e)$. Expanding this variance gives Equation (34). The improvement interval and clipped optimizer follow by minimizing the quadratic; zero variance makes P constant on the support. For the last claim, divide the penalized and unpenalized softmax expressions to obtain Equation (10), then differentiate: $\frac{d}{d\beta} w_j^\beta = -w_j^\beta (P_j - \sum_i w_i^\beta P_i)/T$. Multiply by P_j and sum to obtain $-(\sum_j w_j^\beta P_j^2 - (\sum_j w_j^\beta P_j)^2)/T$. □

The criterion removes common score error, which has no effect on ranking. Its alignment requirement differs from the uncentered $\mathbb{E}[\Delta P]$ in Proposition D.5. For example, take two candidates with $g = (1, 0)$, $G = (3, 1)$, $P = (3, 0)$, and $\beta = 1$. The corrected scores are $(0, 1)$: MSE falls from $5/2$ to 1, yet the correct ordering is reversed. This also shows why a score-MSE improvement cannot alone explain an improvement in planning. The centered criterion controls average gap error, not every gap; the margin conditions in Proposition D.7 provide separate sufficient conditions for uniform ordering.

D.6 Model sensitivity and the scope of decrease regularization

Proposition D.9 (Sensitivity to rollout error)**.** *Suppose* $v_{\bar{\psi}}$ *is* L_v*-Lipschitz on a set containing both predicted latents* $\hat{z}_k$ *and encoded true states* z_k^{env} *under the same actions. With* $P_{\mathrm{env}} = \sum_{k=1}^H p(z_k^{\mathrm{env}})$*,*

$$|P - P_{\mathrm{env}}| \leq L_v \sum_{k=1}^H \|\hat{z}_k - z_k^{\mathrm{env}}\|. \tag{36}$$

Proof. The hinge is 1-Lipschitz. Apply the Lipschitz bound on $v_{\bar{\psi}}$ at each pair of latents, then the triangle inequality for the sum. □

Thus the penalty remains sensitive to the same learned rollout; it does not bypass model error. Its supervision avoids a Bellman-return target, which is a different property. Shared data and latents also preclude asserting statistical independence from the world model.

The decrease objective in Equation (7) has a similarly limited, but precise implication. On the training transition distribution, let $\mathbb{P}(\mathbb{M} = 1) = p_M > 0$, $\mathcal{L}_{\mathrm{dec}} \leq \ell$, and $|v_{\bar{\psi}}(Z) - v_\psi(Z)| \leq \delta$ on the active mask. Since the positive-part function is 1-Lipschitz,

$$\mathbb{E}[(v_\psi(Z') - v_\psi(Z))_+ \mid \mathbb{M} = 1] \leq \ell/p_M + \delta. \tag{37}$$

Indeed, pointwise the left integrand is at most $(v_\psi(Z') - v_{\bar{\psi}}(Z))_+ + \delta$; take the conditional expectation and use the definition of $\mathcal{L}_{\mathrm{dec}}$. There is no bound outside the mask or when $p_M = 0$. Even zero loss permits equal potentials at both endpoints and therefore does not imply strict progress, attraction to an equilibrium, or stability.

D.7 What the analysis says about the schedule

Proposition D.10 (A sufficient condition for decreasing optimal strength)**.** *Consider a family of fixed-checkpoint query distributions indexed by t, with differentiable moments $A_1(t) > 0$, $A_2(t) > 0$ as in Proposition D.5. For fixed λ, the pointwise MSE-optimal strength is $\beta^*(t) = \mathrm{clip}(A_1(t)/A_2(t), 0, \lambda)$. If $A_1'(t)A_2(t) - A_1(t)A_2'(t) \le 0$ throughout an interval, then $\beta^*(t)$ is non-increasing on that interval.*

Proof. The pointwise optimizer follows from Equation (28). The derivative of the unclipped ratio is $(A_1'A_2 - A_1A_2')/A_2^2$, which is nonpositive under the hypothesis. Clipping by fixed endpoints preserves monotonicity. □

Decreasing bias or undercoverage separately is insufficient for the derivative condition. Even two positive decreasing moments can have an increasing ratio: $A_1(t) = e^{-t}$ and $A_2(t) = e^{-2t}$ give $A_1/A_2 = e^t$. The optimizer here is local to a specified evaluation distribution, not optimal for the evolving data-collection process. LAMP's warmup–ramp–decay–plateau schedule remains an empirical design; the analysis neither derives its milestones nor guarantees that its strength stays in the MSE-improving interval.

D.8 Label ambiguity and the size of a planning intervention

Proposition D.11 (Classification under conflicting endpoint labels)**.** *Freeze thresholds with $r_{\mathrm{lo}} < r_{\mathrm{hi}}$ and consider the distribution of transition-endpoint occurrences used in Equation* (5)*. At a latent z with positive conditional labeled mass, let $\rho = \mathbb{P}(Y = \mathrm{LO} \mid Z = z, Y \neq \mathrm{MID})$. For an unrestricted scalar prediction u, the conditional loss is*

$$\begin{aligned} L_z(u) &= (1-\rho)(u - v_{\mathrm{hi}})^2 + \rho(u - v_{\mathrm{lo}})^2 \\ &= \big(u - v^*_{\mathrm{cls}}(z)\big)^2 + \rho(1-\rho)(v_{\mathrm{lo}} - v_{\mathrm{hi}})^2, \end{aligned} \tag{38}$$

where v^*_{cls} is given by Equation (12)*. It is the unique pointwise minimizer. Its hinge penalty is positive if and only if $\rho > (c - v_{\mathrm{hi}})/(v_{\mathrm{lo}} - v_{\mathrm{hi}})$. If $0 < \rho < 1$, the minimum conditional loss is strictly positive. These statements describe the classification term alone.*

Proof. The conditional target A equals v_{hi} or v_{lo}, with probabilities $1 - \rho$ and ρ. Using $\mathbb{E}[(u - A)^2] = (u - \mathbb{E}A)^2 + \mathrm{Var}(A)$ gives Equation (38), since $\mathbb{E}A = v_{\mathrm{hi}} + (v_{\mathrm{lo}} - v_{\mathrm{hi}})\rho$ and $\mathrm{Var}(A) = \rho(1-\rho)(v_{\mathrm{lo}} - v_{\mathrm{hi}})^2$. The strictly convex square yields the unique minimizer. Solving $v^*_{\mathrm{cls}}(z) > c$ gives the hinge condition, because $v_{\mathrm{lo}} > v_{\mathrm{hi}}$. Multiplication by a positive conditional labeled mass does not change the minimizer; at zero mass the classification objective imposes no condition on u. Finally, $0 < \rho < 1$ and distinct anchors make the residual positive. □

The endpoint distribution counts both endpoints with their training weights; it is not a distribution of state labels intrinsic to the environment. The proposition assumes distinct thresholds to obtain mutually exclusive bands. At coincident thresholds, overlapping labels can instead be represented as separate weighted occurrences, but their frequencies no longer describe mutually exclusive reward events. Neither construction removes network capacity constraints or the coupling introduced by the other terms in Equation (8).

Proposition D.12 (A bound on candidate-weight and action-mean changes)**.** *Fix finite raw scores and penalties, $T > 0$, and a common candidate support as in Proposition D.8. For $\beta \ge 0$, let $R_P = \max_j P_j - \min_j P_j$. Then Equation* (13) *holds. If the associated first actions are a_j and $D_a = \max_{i,j} \|a_i - a_j\|$, then*

$$\left\| \sum_j w_j^\beta a_j - \sum_j w_j^0 a_j \right\| \le D_a \, \mathrm{TV}(w^\beta, w^0). \tag{39}$$

Proof. At strength b, differentiation as in Proposition D.8 gives $\partial_b w_j^b = -w_j^b(P_j - \mathbb{E}_{w^b}P)/T$. The fundamental theorem of calculus and triangle inequality yield

$$\tfrac{1}{2}\sum_j |w_j^\beta - w_j^0| \le \frac{1}{2T}\int_0^\beta \mathbb{E}_{w^b}|P - \mathbb{E}_{w^b}P| \, db.$$

For any distribution supported on an interval of length R_P, Cauchy–Schwarz gives $\mathbb{E}|P - \mathbb{E}P| \leq \sqrt{\mathrm{Var}(P)} \leq R_P/2$; the last inequality follows by shifting the interval to $[0, R_P]$ and using the variance argument of Proposition D.6. Integration gives $\beta R_P/(4T)$, and total variation is also at most one.

For the action bound, set $d_j = w_j^\beta - w_j^0$. Its positive and negative parts both have total mass $t = \frac{1}{2}\sum_j |d_j|$. If $t = 0$, the action means agree. Otherwise normalize the two parts to probability vectors p^+ and p^-. Their action means a^+ and a^- satisfy

$$\|a^+ - a^-\| = \left\| \sum_{i,j} p_i^+ p_j^- (a_i - a_j) \right\| \leq \sum_{i,j} p_i^+ p_j^- D_a = D_a.$$

The original mean difference is $t(a^+ - a^-)$, proving Equation (39). □

Both bounds concern a single reweighting operation on fixed actions. They neither bound changes after resampling nor translate action closeness into return improvement, constraint satisfaction, or closed-loop stability in an unknown nonlinear environment.

D.9 Scope and implications

The analysis gives a sequence of distinct requirements: accurate reward-bin prediction must transfer to planning queries; the resulting penalty must align with return-estimation error; its strength must avoid excessive correction; and its relative errors must support better candidate ordering. The proofs identify these requirements without asserting that the reported learning curves verify them. The square-norm floor guarantees positivity only, the decrease prior controls an average positive increment on its active mask, and finite-horizon boundedness does not imply variance dominance over Q. No OOD-detection, policy-invariance, Lyapunov stability, regret, or sample-complexity guarantee is claimed. Reward-poor but necessary transitions and rarely visited high-return states can be penalized undesirably; such cases fall outside a successful error-alignment interpretation even when classification loss is small.

E Limitations

Reward information and task structure. The potential learns a preference from immediate reward bins, not a long-term value function or a probability of model error. Low-reward transitions may be necessary to reach a valuable state, and rare high-return states may be poorly represented in its training data. Sparse or delayed rewards can leave the classification bins and decrease mask uninformative. In particular, coincident thresholds can activate both anchors under the stated non-strict inequalities. The sampled decrease loss permits constant potentials and does not demonstrate progress toward a goal.

Distribution shift and theoretical scope. Detached gradients separate optimization paths but do not remove model error from latent inputs. Both the training distribution and latent coordinates change as the backbone learns. The analysis freezes a checkpoint and specifies conditions for query coverage, EMA tracking, and error alignment; the reported return curves do not verify these conditions. Fixed-candidate results do not cover repeated replanning or the return of a weighted average of actions under nonlinear dynamics. No OOD-detection, Lyapunov stability, safety, or sample-complexity guarantee follows from the architecture.

Attribution and comparisons. The evaluation compares the complete extension with TD-MPC2, DreamerV3, and SAC. The available ablations vary the schedule and hidden-layer parameterization, but do not independently vary the decrease loss, control for every use of the additional parameters, or compare against a penalty constructed directly from the existing reward head. They therefore support a comparison of configurations rather than a unique causal account of the gains. The recent policy-constraint, ensemble-pessimism, and planning alternatives discussed in Section 2 are not experimental baselines here. Performance relative to, and combinations with, those methods remain unresolved.

Application and scale. Evaluation is limited to state-observation DMControl tasks on a single TD-MPC2 backbone. Simulated locomotion results do not establish transfer to visual observations, real robots, manipulation, or safety-critical control. The measured training-time overhead does not characterize worst-case action-selection latency. The reward-bin thresholds adapt to reward statistics, but the fixed penalty weight is not automatically invariant to reward rescaling: scaling return estimates changes their balance with an unchanged penalty. The common hyperparameter setting used here does not establish robustness to other reward scales, planning horizons, or candidate temperatures.