UNCERTAINTY-GUIDED λ-SOFT VALUE ESTIMATION FOR SAMPLE-EFFICIENT MAXIMUM ENTROPY REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Maximum entropy actor-critic methods such as Soft Actor-Critic (SAC) are known for effective exploration and robustness, yet their learning progress hinges on a learned soft critic whose estimates are inaccurate early in training. Model-based reinforcement learning (MBRL) promises higher sample efficiency, but compounding model errors limit how far learned dynamics can be trusted. In this work, we use model-based rollouts to improve the sample efficiency of SAC. We introduce λ-soft value estimation, which recursively blends the critic's direct estimate with a model-based bootstrapped estimate at each step of a model rollout, and prove that under an exact model such estimates accelerate soft policy iteration, with the one-step backup of SAC as the slowest case. To set the per-step blending factor automatically, we propose uncertainty guidance, which balances critic-ensemble disagreement against dynamics-ensemble disagreement: the estimate relies on the model where the critic is uncertain and falls back to the critic for the rest of the rollout once only the model is uncertain. On four contact-rich MuJoCo tasks, λ-SAC is competitive in sample efficiency, wall-clock time, and robustness to tuning, as a single hyperparameter setting solves all tasks, and it reaches a final return on par with model-free methods.

1 INTRODUCTION

Model-free deep reinforcement learning (RL) has been applied successfully to complex control tasks in which gradient-based optimal control typically struggles, such as problems with strongly nonlinear dynamics or discontinuities from contact, e.g., contact-rich manipulation (OpenAI et al., 2019) and legged locomotion (Hwangbo et al., 2019). It learns the mapping from states to actions directly, which makes it broadly applicable, and the resulting lightweight policies are fast at inference, as required, e.g., in agile flight (Kaufmann et al., 2023). This generality comes at the cost of low sample efficiency. On-policy methods such as A3C (Mnih et al., 2016) and PPO (Schulman et al., 2017) restrict updates to recent rollouts to limit policy staleness, which keeps the update-to-data (UTD) ratio low. Off-policy actor-critics such as DDPG (Lillicrap et al., 2019), TD3 (Fujimoto et al., 2018), and SAC (Haarnoja et al., 2018) reuse older transitions from a replay buffer at a higher UTD ratio and are therefore more sample-efficient. Among them, SAC combines a stochastic maximum-entropy objective (Ziebart, 2010), which improves exploration and robustness, with a learned temperature (Haarnoja et al., 2019) that reduces tuning effort, which makes it a widely used baseline and the method we build on.

Model-based RL (MBRL) leverages a model of the environment to improve sample efficiency further (Reiter et al., 2025). The model can improve the actor, with MPC as an expert for policy regression (Carius et al., 2020; Hertneck et al., 2018) or as a guiding trajectory distribution (Levine & Koltun, 2013; Levine & Abbeel, 2014), with the actor as an initial guess for MPC (Reiter et al., 2024), with MPC as a safety filter (Wabersich & Zeilinger, 2021), or with a parameterized MPC trained closed-loop as a differentiable optimizer (Amos et al., 2019; Gros & Zanon, 2020). It can improve the value targets of the critic through model rollouts (Feinberg et al., 2018; Buckman et al., 2019), latent imagined rollouts (Hafner et al., 2020), or MPC (Lowrey et al., 2018; Bhardwaj et al., 2021), which also avoids the importance sampling that multi-step targets from replayed trajectories

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.

require (Munos et al., 2016). Finally, it can serve as an extra data source through Dyna-style rollouts written to the replay buffer (Sutton, 1990; Janner et al., 2021).

Model bias limits all of these uses: one-step prediction errors compound along a rollout, and policies or value estimates exploit them (Gu et al., 2016; Janner et al., 2021). Model-based value targets therefore trade model bias against the bias of the bootstrapped critic. Existing methods settle this trade-off with a fixed rollout horizon (Feinberg et al., 2018; Janner et al., 2021), a global blending schedule (Bhardwaj et al., 2021), or a per-node blend driven by critic uncertainty alone (Schoch & Ryll, 2026), although the reliability of both model and critic changes across states and over training.

In this work, we extend the soft policy iteration of SAC with model-based rollouts. Our λ-soft value estimate is a TD(λ)-style blend of model backup and soft critic bootstrap at every node of an imagined rollout, which generalizes the hard blend of Bhardwaj et al. (2021) and Schoch & Ryll (2026) to the maximum-entropy setting. A per-node blending factor λ_t, set from the disagreement of critic and dynamics ensembles, leans on the model where the critic is uncertain and bootstraps early where only the model is uncertain, so the effective rollout horizon adapts across states and over training. Our contributions are:

- **λ-soft policy iteration**, which carries one blend of model backup and soft critic bootstrap through both steps of SAC's scheme: the λ-soft estimate replaces the bootstrap in policy evaluation, and its differentiable form supplies multi-step soft policy gradients in policy improvement (Sections 4 and 5). Under an exact model, the evaluation step keeps SAC's soft fixed point and contracts faster than SAC's one-step backup (Section 5.1).
- **Uncertainty guidance**, which sets λ_t per node from the disagreement of the dynamics and the critic ensembles jointly (Section 5.3).

2 Related Work

Model rollouts as data or inside the target. Model-based methods branch short rollouts from states of real data (Sutton, 1990; Janner et al., 2021), with rewards from a learned or an analytical reward function, and differ in where the rollout is consumed. Dyna (Sutton, 1990) and MBPO (Janner et al., 2021) write the rollouts into the replay buffer as extra transitions and leave the critic target one-step. Value-expansion methods instead use the rollout inside the target, as do MPC-based blends that also act with the rollout (Lowrey et al., 2018; Bhardwaj et al., 2021; Schoch & Ryll, 2026). We follow the second line: each rollout produces one target and is discarded, so the buffer remains a sample of the true dynamics and a poor model affects a single update rather than the training set. Since the rollout actions come from the current policy, the multi-step target needs no importance correction, unlike multi-step targets from replay traces (Munos et al., 2016), and the critic keeps scoring the current policy, unlike critics conditioned on the executed action sequence (Tian et al., 2026).

Overestimation and model-based value estimation. Actor-critics such as DDPG (Lillicrap et al., 2019) overestimate values through their bootstrapped critic targets, which TD3 counters with clipped double-Q learning, delayed actor updates, and target-policy smoothing (Fujimoto et al., 2018), and SAC adopts the clipped double-Q minimum (Haarnoja et al., 2019). Model-based value estimation puts less weight on the bootstrapped critic by expanding the target over a model rollout: MVE over a fixed horizon (Feinberg et al., 2018), STEVE with one target per rollout length combined by inverse-variance weights (Buckman et al., 2019), Dreamer in a learned latent space (Hafner et al., 2020), and POLO with a local planner at the queried state (Lowrey et al., 2018). We bootstrap on the minimum over N critic heads, blend model backup and critic bootstrap at every node, and train the actor on the same estimate.

Choosing the rollout horizon. TD(λ) trades the variance of a long return against the bias of an early bootstrap (Sutton & Barto, 2018); with a learned model, the trade-off becomes model bias against critic bias. Bhardwaj et al. (2021) and Schoch & Ryll (2026) blend an MPC rollout with a learned critic at every node, the former with a scalar λ annealed during training, the latter with a critic attenuated per node by value-head disagreement. STEVE (Buckman et al., 2019) weights the $H+1$ candidate targets by their inverse variance, which requires one rollout per dynamics member and all member combinations per target. We instead set λ_t per node from the model and critic

disagreement that the rollout already evaluates at that node, and keep the two apart so that the gate cuts the rollout where the model is uncertain.

Model error along the rollout. Compounding model errors are countered by training the model on its own multi-step predictions (Talvitie, 2016) or by restricting its use through ensembles against model exploitation (Chua et al., 2018; Kurutach et al., 2018) or a global rollout-length schedule (Janner et al., 2021). Policy gradients through a model face the same error: SVG and MAAC differentiate the policy objective through learned dynamics over a fixed horizon (Heess et al., 2015; Clavera et al., 2020), and Dreamer weights the nodes with a constant λ (Hafner et al., 2020). We fit the model free-running over several steps and let λ_t weight each node in both the critic target and the actor gradient.

3 PRELIMINARIES

3.1 STANDARD MDP SETTING

We consider an infinite-horizon Markov decision process $(\mathcal{S}, \mathcal{A}, p_0, p, r, \gamma)$ (Sutton & Barto, 2018) and follow the notation of Haarnoja et al. (2018). States $\mathbf{s}_t \in \mathcal{S}$ and actions $\mathbf{a}_t \in \mathcal{A}$ are continuous, the initial state is drawn from p_0, and the unknown transition density p yields the next state $\mathbf{s}_{t+1} \sim p(\cdot \mid \mathbf{s}_t, \mathbf{a}_t)$. Each transition emits a bounded reward $r(\mathbf{s}_t, \mathbf{a}_t) \in [r_{\min}, r_{\max}]$, discounted by $\gamma \in [0, 1)$. A stochastic policy $\pi(\mathbf{a}_t \mid \mathbf{s}_t)$ induces the trajectory distribution ρ_π over $\tau = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \dots)$, with state and state-action marginals $\rho_\pi(\mathbf{s}_t)$ and $\rho_\pi(\mathbf{s}_t, \mathbf{a}_t)$. Standard RL maximizes the expected discounted return $\mathbb{E}_{\tau\sim\rho_\pi}\big[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\big]$.

3.2 MAXIMUM-ENTROPY RL

Maximum-entropy RL maximizes the return subject to a minimum expected policy entropy $\mathcal{H}$ at every time step, which prevents the policy from collapsing (Haarnoja et al., 2019). Relaxing the constraint with the Lagrange multiplier $\alpha \geq 0$, the temperature, gives the soft objective

$$\mathcal{L}(\pi, \alpha) = \mathbb{E}_{\tau\sim\rho_\pi}\Big[\sum_{t=0}^{\infty} \gamma^t \big(\underbrace{r(\mathbf{s}_t, \mathbf{a}_t) - \alpha \log \pi(\mathbf{a}_t \mid \mathbf{s}_t)}_{\tilde{r}_t} - \alpha\mathcal{H}\big)\Big]. \tag{1}$$

The temperature controls the stochasticity of the policy, which improves exploration and robustness to perturbations, and $\alpha = 0$ recovers standard RL. At fixed α, the term $\alpha\mathcal{H}$ is a constant per step, so equation 1 is standard RL on the maximum-entropy stage reward $\tilde{r}_t$. For a fixed policy π, the corresponding soft Bellman backup (Haarnoja et al., 2018) with $r_t = r(\mathbf{s}_t, \mathbf{a}_t)$ reads

$$\mathcal{T}^\pi Q(\mathbf{s}_t, \mathbf{a}_t) := r_t + \gamma\, \mathbb{E}_{\mathbf{s}_{t+1}\sim p}\big[V(\mathbf{s}_{t+1})\big], \qquad V(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t\sim\pi}\big[Q(\mathbf{s}_t, \mathbf{a}_t) - \alpha \log \pi(\mathbf{a}_t \mid \mathbf{s}_t)\big], \tag{2}$$

and its repeated application converges to the soft action-value function Q^π_{soft} (Haarnoja et al., 2018, Lemma 1). The entropy of the action taken in $\mathbf{s}_t$ enters V, so Q contains only the entropy of later steps. Maximizing over the policy at each step yields the Boltzmann policy $\pi^* \propto \exp(Q/\alpha)$, whose value is a log-sum-exp over actions in place of $\max_{\mathbf{a}}$ (Ziebart, 2010; Haarnoja et al., 2017). This is what *soft* refers to throughout, and $\alpha \to 0$ recovers the hard maximum. From here on, V and Q denote the soft quantities; hard quantities appear only in Section 3.4.

3.3 SOFT ACTOR-CRITIC (SAC)

SAC optimizes equation 1 by approximate soft policy iteration (Haarnoja et al., 2018), and Haarnoja et al. (2019) additionally learn the temperature. A Gaussian policy π_ϕ and two soft Q-functions $Q_{\theta_1}, Q_{\theta_2}$ are trained on minibatches from a replay buffer $\mathcal{D}$, alternating single gradient steps on a critic, an actor and a temperature objective. Soft policy evaluation regresses Q_{θ_i} onto the soft Bellman target y_t, which samples equation 2 with one stored transition and evaluates V with delayed target networks $\bar{\theta}$,

$$J_Q(\theta_i) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})\sim\mathcal{D}}\Big[\big(Q_{\theta_i}(\mathbf{s}_t, \mathbf{a}_t) - y_t\big)^2\Big], \qquad y_t = r_t + \gamma\, V_{\bar{\theta}}(\mathbf{s}_{t+1}), \tag{3}$$

$$V_{\bar{\theta}}(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t\sim\pi_\phi}\Big[\min_{i\in\{1,2\}} Q_{\bar{\theta}_i}(\mathbf{s}_t, \mathbf{a}_t) - \alpha \log \pi_\phi(\mathbf{a}_t \mid \mathbf{s}_t)\Big]. \tag{4}$$

Soft policy improvement minimizes the KL divergence to the Boltzmann distribution of the critic via the reparameterization trick,

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D},\, \mathbf{a}_t \sim \pi_\phi}\Big[\alpha \log \pi_\phi(\mathbf{a}_t \mid \mathbf{s}_t) - \min_{i \in \{1,2\}} Q_{\theta_i}(\mathbf{s}_t, \mathbf{a}_t)\Big], \tag{5}$$

and the temperature follows dual gradient descent on equation 1,

$$J(\alpha) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D},\, \mathbf{a}_t \sim \pi_\phi}\big[-\alpha \log \pi_\phi(\mathbf{a}_t \mid \mathbf{s}_t) - \alpha \mathcal{H}\big]. \tag{6}$$

3.4 λ-Blended Returns

Schoch & Ryll (2026) and Bhardwaj et al. (2021) blend a local model-based rollout with a learned critic Q_θ to obtain better value estimates for planning and for value bootstrapping. The two components have complementary errors. A learned dynamics model captures nonlinear, potentially stiff contact dynamics, so a rollout can use the rewards along the way instead of relying on the critic alone, but its approximation errors accumulate with the rollout length. The critic covers the entire remaining horizon in a single evaluation at an error that does not grow with the horizon H, but it is biased while undertrained and unreliable off-distribution. A blending factor $\lambda \in [0, 1]$ trades one error source against the other. Along a rollout $(\mathbf{s}_t, \mathbf{a}_t)_{t=0}^{H-1}$, with actions from model predictive control (MPC) in Schoch & Ryll (2026) and next states from a nominal model, the blend is defined recursively for $t = 0, \ldots, H-2$ as

$$Q_t^\lambda = (1-\lambda)\, Q_{\theta,t} + \lambda\big(r_t + \gamma\, Q_{t+1}^\lambda\big), \qquad Q_{H-1}^\lambda = (1-\lambda)\, Q_{\theta,H-1} + \lambda\, r_{H-1}, \tag{7}$$

with $Q_{\theta,t} = Q_\theta(\mathbf{s}_t, \mathbf{a}_t)$. This blend is hard: it contains no entropy term, and Q_θ approximates the hard action-value function. $\lambda = 0$ returns the critic alone and $\lambda = 1$ the pure model rollout. Unrolling equation 7 expresses the estimate as the critic plus geometrically weighted Bellman residuals $B_t = r_t + \gamma Q_{\theta,t+1} - Q_{\theta,t}$ along the rollout,

$$Q_0^\lambda = Q_{\theta,0} + \lambda\Bigg(\sum_{k=0}^{H-2} (\lambda\gamma)^k B_k + (\lambda\gamma)^{H-1}\big(r_{H-1} - Q_{\theta,H-1}\big)\Bigg). \tag{8}$$

The blending factor can vary per rollout step, $\lambda_t \in [0, 1]$, in which case the residual of step k enters with the cumulative weight $\gamma^k \prod_{j \le k} \lambda_j$. A single small λ_j thus damps all later residuals, which makes λ_t a natural function of uncertainty. Schoch & Ryll (2026) exploit this by down-weighting the critic with a confidence kernel $e^{-\kappa\sigma}$ of the disagreement σ within a critic ensemble, and an iLQR-based MPC maximizes Q_0^λ over the action sequence.

4 λ-Soft Returns

We improve the soft policy iteration of SAC by infusing its value estimates with knowledge from a learned dynamics model. Two terms of SAC admit this: the bootstrap $V_{\bar\theta}(\mathbf{s}_{t+1})$ in the critic target in equation 3 of policy evaluation, and $Q_{\theta_i}(\mathbf{s}_t, \mathbf{a}_t)$ in the actor objective in equation 5 of policy improvement. Both replacements rest on the λ-soft estimate derived in this section.

We carry the blend in equation 7 over to the maximum-entropy setting by replacing the reward with the stage reward $\tilde r_t = r_t - \alpha \log \pi_t$ of equation 1, with $\pi_t = \pi(\mathbf{a}_t \mid \mathbf{s}_t)$, which turns the value into the soft value of equation 2. The stochastic version of equation 7 then reads

$$\begin{aligned} V^\lambda(\mathbf{s}_t) &= \mathbb{E}_{\mathbf{a}_t \sim \pi}\big[Q^\lambda(\mathbf{s}_t, \mathbf{a}_t) - \alpha \log \pi_t\big], \\ Q^\lambda(\mathbf{s}_t, \mathbf{a}_t) &= (1-\lambda_t)\, Q_{\theta,t} + \lambda_t\big(r_t + \gamma\, \mathbb{E}_{\mathbf{s}_{t+1} \sim p}\big[V^\lambda(\mathbf{s}_{t+1})\big]\big), \end{aligned} \tag{9}$$

where the blending factor $\lambda_t \in [0, 1]$ may differ per node. As in Schoch & Ryll (2026), we unroll this recursion along a rollout of H nodes $(\mathbf{s}_t, \mathbf{a}_t)_{t=0}^{H-1}$, which contains $H-1$ model transitions because the last action $\mathbf{a}_{H-1}$ enters only the bootstrap $Q_{\theta,H-1} - \alpha \log \pi_{H-1}$. The rollout branches from an anchor state $\mathbf{s}_0$ of real data (Sutton, 1990; Janner et al., 2021), and its later nodes follow ρ_π, i.e., $\mathbf{a}_t \sim \pi(\cdot \mid \mathbf{s}_t)$ for $t = 0, \ldots, H-1$ and $\mathbf{s}_{t+1} \sim p(\cdot \mid \mathbf{s}_t, \mathbf{a}_t)$ for $t = 0, \ldots, H-2$, where a learned dynamics model (Section 5.4) stands in for p in practice. With $\mathbb{E}$ denoting the expectation over the rollout, the soft one-step Bellman residual

$$B_t = r_t + \gamma\, \mathbb{E}\big[Q_{\theta,t+1} - \alpha \log \pi_{t+1}\big] - Q_{\theta,t} \tag{10}$$

replaces the hard residual of equation 8, and the recursion unrolls into the TD(λ)-style estimator (Appendix A)

$$V^{\lambda}(\mathbf{s}_0) = \mathbb{E}\left[Q_{\theta,0} - \alpha \log \pi_0 + \sum_{t=0}^{H-2} w_t \, B_t\right], \qquad w_t = \gamma^t \prod_{j \le t} \lambda_j. \tag{11}$$

The weight w_t is a cumulative product, so a single small λ_j damps every later residual, not just its own. This cut-off property, inherited from equation 8, lets the estimator truncate a rollout at the first unreliable node. The rollout stops at $\mathbf{s}_{H-1}$, so $\lambda_{H-1} = 0$ and the last node enters B_{H-2} as a pure bootstrap, the usual closure of a fixed-horizon rollout (Feinberg et al., 2018). For $\lambda_t \equiv 0$, the estimate reduces to $V^{\lambda=0}(\mathbf{s}_0) = \mathbb{E}\big[Q_{\theta,0} - \alpha \log \pi_0\big]$, the one-step soft TD target of SAC (Haarnoja et al., 2018; 2019). For $\lambda_t \equiv 1$, the intermediate $Q_{\theta,t}$ cancel and only the terminal critic bootstraps, which gives a maximum-entropy model value expansion in the style of Feinberg et al. (2018) (equation A9 in Appendix A). The same weights measure how far the estimator reaches. The critic term at $\mathbf{s}_0$ enters with weight one and every rollout step with w_t, which defines the effective horizon $h_{\text{eff}} = 1 + \sum_{t=0}^{H-2} w_t \in [1, H]$. For $\lambda_t \equiv 0$, $h_{\text{eff}} = 1$ recovers the model-free one-step estimate, and $\lambda_t \equiv 1$ with $\gamma = 1$ gives $h_{\text{eff}} = H$. Thus $h_{\text{eff}} - 1$ is the number of model steps the estimator effectively uses.

5 λ-Soft Policy Iteration

We learn three components: a reparameterized Gaussian policy π_ϕ as in SAC, an ensemble of N soft Q-functions Q_{θ_i} reduced by $\min_i$, and an ensemble of E one-step dynamics models $\hat{f}_j$ on the simulator state $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}})$ of generalized positions and velocities. Each ensemble serves twice: the rollout steps with the member mean of $\hat{f}_j$ and the estimator bootstraps on $\min_i Q_{\theta_i}$, while the spread across members provides the disagreements σ_f and σ_Q (Section 5.3).

5.1 Policy Evaluation

Policy evaluation replaces the soft Bellman target of equation 3 with one real Bellman step onto the λ-soft estimate of equation 11 at the next state, computed with the target networks $\bar{\theta}$ (Haarnoja et al., 2018),

$$J_Q(\theta_i) = \mathbb{E}_{(\mathbf{s}_0, \mathbf{a}_0, r_0, \mathbf{s}_1) \sim \mathcal{D}}\Big[\big(Q_{\theta_i}(\mathbf{s}_0, \mathbf{a}_0) - y\big)^2\Big], \qquad y = r_0 + \gamma \, V^{\lambda}_{\bar{\theta}}(\mathbf{s}_1). \tag{12}$$

The anchor transition comes from the replay buffer, so the first step carries no model error and is never gated: it enters with weight one, and only the $H - 1$ model steps of the rollout branched at $\mathbf{s}_1$ carry the weights w_t. The rollout actions come from the current policy π_ϕ rather than from the behavior policy that filled the buffer, so the multi-step target is on-policy by construction and needs no importance correction (Munos et al., 2016).

Proposition 1 (Soft fixed point) *Under an exact dynamics model and for any fixed $\lambda_t \in [0, 1]$, the λ-soft estimate in equation 11 is a convex combination of soft n-step returns, and the target operator of equation 12 is a γ-contraction with fixed point Q^{π}_{soft}.*

The proof (Appendix B) writes equation 11 as a convex combination of repeated soft Bellman backups, so λ_t sets their weights but never the fixed point. The fixed point is the soft action value rather than the hard one because each n-step return in the combination collects $-\alpha \log \pi_t$ at every node it passes, i.e., each backup is the soft backup $\mathcal{T}^{\pi}$ of equation 2.

Proposition 2 (Contraction rate) *Under the assumptions of Proposition 1, the target operator of equation 12 contracts with modulus $\gamma\big(1 - (1 - \gamma)\sum_t w_t\big) \in [\gamma^H, \gamma]$.*

The contraction is therefore faster than SAC's whenever the rollout carries weight, $\lambda_0 > 0$, which the uncertainty guidance in equation 15 always ensures since $\lambda_t \ge c_f > 0$; it matches SAC only in the limit $\lambda_t \to 0$. At constant, state-independent λ and $H \to \infty$, the modulus becomes $\gamma(1 - \lambda)/(1 - \gamma\lambda)$, the modulus of the λ-policy-iteration operator of Bertsekas & Tsitsiklis (1996, Sec. 2.3.1), of which equation 12 is the per-node, finite-horizon form (Appendix C).

5.2 Policy Improvement

Policy improvement replaces the Boltzmann projection in equation 5 with the same estimator on the online critic. The rollout is anchored at a replay state $\mathbf{s}_0 \sim \mathcal{D}$ and starts with the policy's own action $\mathbf{a}_0 \sim \pi_\phi(\cdot \mid \mathbf{s}_0)$, so the model already predicts $\mathbf{s}_1$. As in SAC, the actor maximizes the soft value, written as the loss

$$J_\pi(\phi) = -\mathbb{E}_{\mathbf{s}_0 \sim \mathcal{D},\, \mathbf{a}_t \sim \pi_\phi}\big[V_\theta^\lambda(\mathbf{s}_0)\big]. \tag{13}$$

For $\lambda_t \equiv 0$ or $H = 1$, the loss reduces to $\mathbb{E}\big[\alpha \log \pi_0 - Q_{\theta,0}\big]$, SAC's actor loss in equation 5, and for $\lambda_t \equiv 1$ the actor maximizes the full-horizon soft return over the $H-1$ model transitions in equation A9. The actions $\mathbf{a}_t$ are reparameterized and the dynamics model is differentiable, so $\nabla_\phi J_\pi$ flows through the entire rollout and yields a multi-step soft policy gradient. Policy evaluation uses the λ-soft estimate as a regression target without gradient, whereas policy improvement differentiates through it. The weights w_t are treated as constants: they shape the estimator but are not part of the objective.

5.3 Uncertainty Guidance

Both learned components, the dynamics model and the critic, carry approximation errors. We train each as an ensemble and use the disagreement between its members as an error proxy (Lakshminarayanan et al., 2017). For an ensemble output $\mathbf{v} \in \mathbb{R}^N$, the disagreement statistic

$$\sigma(\mathbf{v}) = \frac{\mathrm{std}(\mathbf{v})}{\|\mathbf{v}\|_2/\sqrt{N}} \in [0, 1] \tag{14}$$

is dimensionless, so its scale does not depend on the reward scale. Without the factor $\sqrt{N}$, the std of the unit-normalized members $\mathbf{v}/\|\mathbf{v}\|_2$ would shrink with $1/\sqrt{N}$ as the ensemble grows, whereas σ stays in $[0, 1]$ for any N. We compute σ_Q over the N critic heads and σ_f over the E member predictions per state dimension, averaged over dimensions, both at the node to which λ_t applies. The confidence kernel $c(\sigma; \kappa) = e^{-\kappa\sigma} \in (0, 1]$ maps the disagreements to confidences $c_f = c(\sigma_f; \kappa_f)$ and $c_Q = c(\sigma_Q; \kappa_Q)$, which set the per-node blending factor

$$\lambda_t = 1 - (1 - c_f)\, c_Q. \tag{15}$$

Here $1 - c_f$ is the cut that the model requests and c_Q the share of it that the critic grants. If the model is uncertain and the critic confident, λ_t drops and the estimate bootstraps early; if both are uncertain, λ_t stays high and the rollout continues; if both are confident, $\lambda_t = 1$ and the rollout reaches the horizon. Since $\lambda_t \geq c_f$, the model confidence is a floor that the critic can only lift. $\kappa_Q = 0$ reduces equation 15 to the pure model gate $\lambda_t = c_f$, and $\kappa_f = 0$ trusts the model everywhere with $\lambda_t = 1$. Through the cumulative product w_t, a low λ_t at one node shortens the effective horizon for the rest of the rollout.

5.4 Dynamics Model Learning

The dynamics model has to be differentiable for equation 13, batched on the GPU with one rollout per minibatch element, refittable online at a small fraction of the cost of an environment step, accurate over the $H-1$ model transitions of a rollout rather than asymptotically, and it has to expose σ_f for equation 15. We use an ensemble of E small MLPs. Each member maps the standardized input $(\mathbf{s}, \mathbf{a})$, where the observation $\mathbf{s}$ is $\mathbf{x}$ without the root coordinates that Gymnasium omits, to the standardized one-step delta $\Delta\mathbf{x} = \mathbf{x}_{t+1} - \mathbf{x}_t$. The rollout carries the full $\mathbf{x}$ because it evaluates the analytic reward function and termination predicate of the environment on each predicted transition $(\mathbf{x}_t, \mathbf{a}_t, \mathbf{x}_{t+1})$, and the forward-progress reward reads the root position that the observation omits; only the dynamics are learned. The state is Markov, so the model needs no recurrence, and multi-step accuracy comes from the training objective instead. The members are fit free-running over windows of L consecutive transitions: only the first state and the stored actions are given, and every later node receives the member's own prediction,

$$J_f = \frac{1}{L}\sum_{k=1}^{L} \left\| \frac{\hat{\mathbf{x}}_k - \mathbf{x}_k}{\mathrm{std}(\Delta\mathbf{x})} \right\|^2, \qquad \hat{\mathbf{x}}_k = \hat{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{a}_{k-1}), \quad \hat{\mathbf{x}}_0 = \mathbf{x}_0. \tag{16}$$

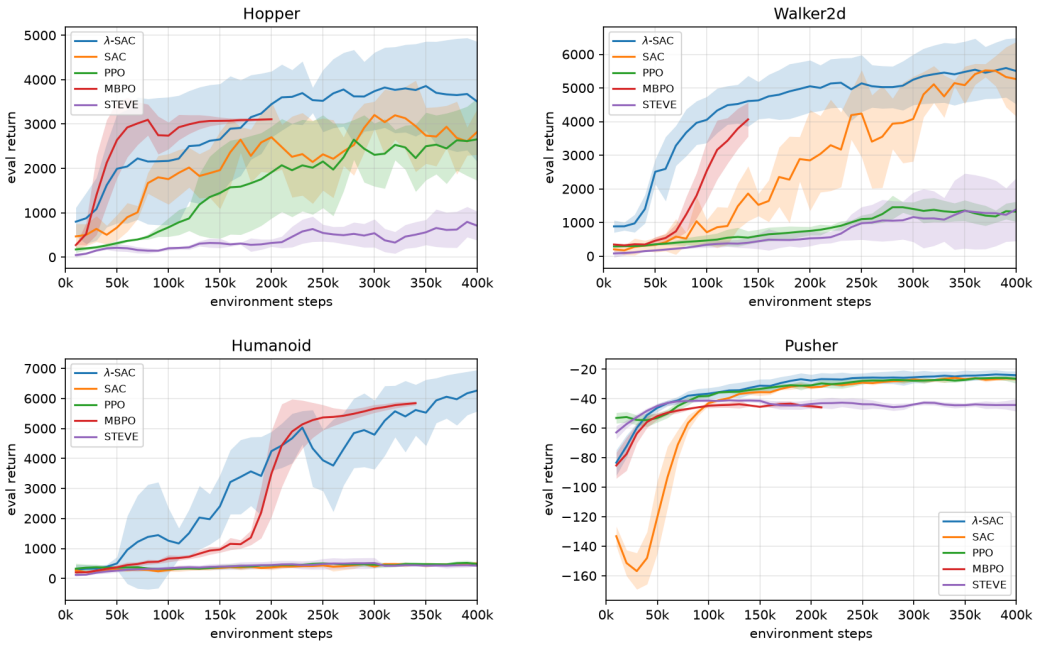

Figure 1: Evaluation return of λ-SAC, SAC, PPO, MBPO, and STEVE over environment steps.

We choose L on the order of H, the rollout length that policy iteration uses; $\mathrm{std}(\Delta \mathbf{x})$ normalizes the loss per dimension and is recomputed at every refit. The members disagree because of their independent initialization and because each member draws its own windows. We read σ_f on the standardized deltas rather than on the predicted states, since the delta is what the members disagree about and its near-zero mean lets σ_f approach the upper bound of equation 14.

6 Results

We test three claims: λ-SAC is more sample-efficient than model-free baselines, it needs less wall-clock time than model-based baselines, and it solves all tasks with a single hyperparameter setting. We evaluate on the contact-rich MuJoCo tasks Hopper, Walker2d, Humanoid, and Pusher from Gymnasium (Towers et al., 2025), against the model-free SAC with learned temperature (Haarnoja et al., 2019) and PPO (Schulman et al., 2017), and the model-based MBPO (Janner et al., 2021) and STEVE (Buckman et al., 2019). On all tasks, λ-SAC runs the full method with $H = 5$, $N = E = 4$, and a UTD ratio of 1. λ-SAC captures its update step in a CUDA graph, whereas the baselines run their public implementations, so wall-clock differences include implementation effects. All methods run on normalized observations. Every 10k of the 400k environment steps, we evaluate the deterministic policy over 5 episodes; curves show the mean and standard deviation over 5 seeds. Appendix D details the hardware, the baseline implementations, and all hyperparameters.

6.1 Performance Evaluation

Figure 1 shows the return over environment steps, and Figure 2 over training time. λ-SAC solves all four tasks and converges in the fewest steps on Walker2d and Humanoid. On Hopper, it converges faster than SAC and PPO, but slower than MBPO. On higher dimensional tasks such as Humanoid, Walker2D or Pusher, λ-SAC converges on par-with, or faster than the model-free and the model-based baselines. SAC needs more than 1M steps to solve Humanoid, PPO solves only Hopper and Pusher, and STEVE solves no task within the budget of 4e5 steps.

Figure 2 compares the training time on one seed per method. SAC and PPO complete the step budget in the shortest time on all tasks, and MBPO and STEVE in the longest. λ-SAC spends more time per environment step than the model-free methods but stays within the same order of magnitude;

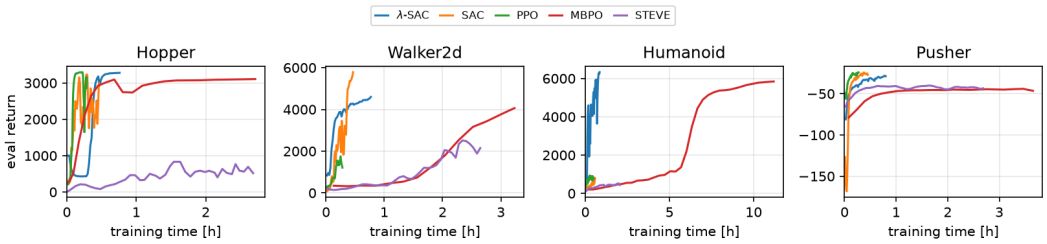

Figure 2: Evaluation return of λ-SAC, SAC, PPO, MBPO, and STEVE over training time, for one seed per method. A single seed does not represent the return of a method; Figure 1 shows the return across seeds.

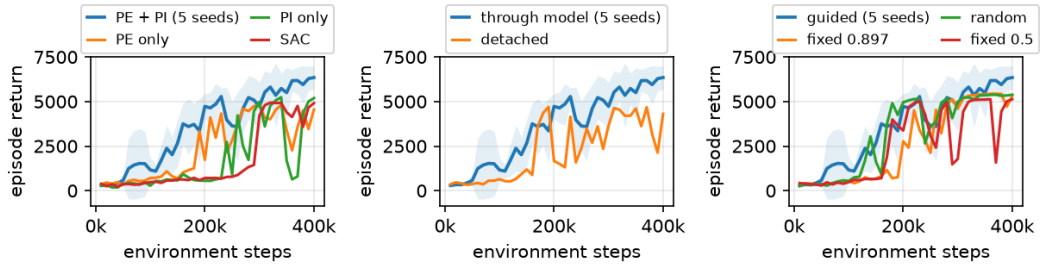

Figure 3: Ablations on Humanoid. Left: λ-soft estimate in policy evaluation (PE), policy improvement (PI), both, or neither (turning λ-SAC into quasi SAC). Middle: actor gradient through the model rollout or with detached model states. Right: guided λ_t against random λ_t and fixed λ matched in effective horizon.

it also converges an order of magnitude faster than MBPO on Walker2d and Humanoid. On these two high-dimensional tasks, λ-SAC is thus the only method that reaches a high return both in few environment steps and in short training time.

A single configuration thus solves all tasks without retuning, with a final return on par with SAC and PPO and above both on Humanoid. On the high-dimensional Walker2d and Humanoid, λ-SAC is the most sample-efficient method, and on all tasks it needs less wall-clock time than both model-based baselines.

6.2 Ablation on soft policy iteration

We ablate λ-SAC on Humanoid with one seed per variant. Each variant changes one setting of the configuration in Table 1. Figure 3 shows three of these ablations, and Appendix E shows all of them together with wall-clock times and training losses.

The first ablation uses the λ-soft estimate in only one step of soft policy iteration. Both single-step variants learn faster than SAC, and using the estimate in both steps learns fastest (Figure 3, left). Of the two single-step variants, the one in policy evaluation gains more. We attribute this to the actor gradient in equation 13, which passes through the critic, so that a better critic also improves the actor update.

The second ablation asks where the gain in policy improvement comes from. It detaches the model states, so the actor no longer gets a gradient through the dynamics. The return then drops to about that of the PE-only variant (Figure 3, middle). The gain in policy improvement therefore comes from the multi-step gradient through the rollout, not from querying the policy at more nodes.

The rollout horizon trades these gains against model error (Appendix E.3). With $H = 2$, λ-SAC learns slowest. $H = 10$ rises later than $H = 5$ and catches up once its dynamics loss approaches that of $H = 5$. A longer horizon speeds up the contraction of Proposition 2, but as long as the model is poor, it also carries more model error into the estimate.

6.3 Ablation on the uncertainty guidance

We compare the guided λ_t with two uninformed choices. The random variant draws λ_t uniformly from $[0, 1]$, and the fixed variants apply a constant λ without guidance. The fixed values 0.897 and 0.5 match the effective horizon of the guided and the random run. Among the four runs, the guided run learns fastest (Figure 3, right). In particular, it beats fixed 0.897 at the same effective horizon. Where the rollout is cut thus matters, not how far it reaches on average. Fixed 0.897 trusts the rollout also where the model is wrong, and it has the highest critic loss of the four runs (Appendix E.4). The guided run, in contrast, reaches the lowest dynamics loss, although λ_t does not enter the model loss. Its policy collects better data, the better data improve the model fit, and the better fit lets the guidance lengthen the rollout. The same link between return and dynamics loss appears in the ablations on ensemble sizes and UTD ratios (Appendices E.6 to E.8). Of the two confidence scales, κ_f sets the effective horizon throughout training, while κ_Q can only lift λ_t above the floor c_f and does so mainly early, while both ensembles still disagree; the return degrades only when a large κ_f cuts the rollout at most nodes (Appendix E.5).

7 Limitations

Propositions 1 and 2 assume an exact dynamics model and give no bound on the shift of the fixed point under the learned model $\hat{f}$. The guidance likewise sees model error only locally: σ_f is a one-step quantity at each node and does not measure the error that compounds along the rollout, so the setting of κ_f chosen at $H = 5$ need not necessarily carry over to other horizons.

Each update adds model rollouts to policy evaluation and policy improvement, and the dynamics ensemble is refit on top of the SAC update, so the wall-clock time relative to SAC depends on how efficiently both are implemented.

8 Conclusion and Future Work

We presented λ-SAC, a model-based maximum-entropy actor-critic built on λ-soft policy iteration. A single estimator blends model backup and soft critic bootstrap at every node of a model rollout: policy evaluation regresses the critic onto it, and policy improvement differentiates through it. Under an exact model, the policy evaluation step contracts with a modulus in $[\gamma^H, \gamma]$ and thus guaranteed faster than SAC's one-step backup. The uncertainty guidance sets λ_t per node from the disagreement of the dynamics and critic ensembles, and the cumulative product w_t turns it into a soft cutoff of the rollout. On four contact-rich MuJoCo tasks, λ-SAC is a competitive model-based method in sample efficiency, wall-clock time, and robustness to tuning, as a single hyperparameter setting solves all tasks.

Two directions can extend this work. The first improves the performance of λ-SAC in its current setting: adaptive UTD ratios can spend more compute on critic and dynamics training while the data distribution shifts and less while it stays the same, e.g., through prioritized sampling of critic training data (Schaul et al., 2016) and by steering the number of model refits based on the monitored prediction error of the dynamics model. The second broadens the application to pretrained latent world models, where the method runs on the latent state $\mathbf{z}$ of a frozen video encoder such as V-JEPA 2 (Assran et al., 2025) instead of the simulator state. A fixed PCA compresses the encoder tokens into $\mathbf{z}$, and the dynamics ensemble and the uncertainty guidance in equation 14 and equation 15 operate on $\mathbf{z}$ as they do on the simulator state. A task-progress estimator pretrained on language and video (Zhang et al., 2025; Rocamonde et al., 2024), distilled into a head $\Phi(\mathbf{z})$, replaces the analytic reward of the rollout through the potential-based term $\gamma\Phi(\mathbf{z}_{t+1}) - \Phi(\mathbf{z}_t)$ (Ng et al., 1999). Since the encoder is frozen, $\mathbf{z}$ is computed once per real frame, stored in the replay buffer, and does not drift during training.

AI USE STATEMENT

In this work, we used generative AI tools for software development and visualization: debugging, testing, and question answering on algorithms, training methodologies, and the training framework, writing configuration-file parsers and plotting scripts, refactoring code, and plotting results from training runs. We have not used generative AI tools for automated research or web searches aimed at deriving new algorithms. Additionally, we used generative AI tools for formatting LaTeX equations and for condensing and paraphrasing text in the manuscript. We have reviewed all AI-assisted work. AI tools were used only in the supporting roles listed above: the authors reviewed every generated artifact, and each served solely to support the authors' own work. AI-written code was reviewed and tested for correctness by the authors before being used in any experiment. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work involves no human participants and processes no personal or sensitive data; all data are generated in simulation. We are not aware of any security, dual-use, or legal concerns, and the authors declare no conflicts of interest or external sponsorship.

REPRODUCIBILITY STATEMENT

The appendix provides complete proofs of our theoretical results and details the software and hardware setup, the evaluation protocol, and all hyperparameters. We plan to publicly release the code, including the method implementation, training and evaluation scripts, and configuration files, after the review process.

REFERENCES

Brandon Amos, Ivan Dario Jimenez Rodriguez, Jacob Sacks, Byron Boots, and J. Zico Kolter. Differentiable mpc for end-to-end planning and control, 2019. URL https://arxiv.org/abs/1810.13400.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba, Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-jepa 2: Self-supervised video models enable understanding, prediction and planning, 2025. URL https://arxiv.org/abs/2506.09985.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Optimization and Computation Series. Athena Scientific, Belmont, Massachusetts, 1996. ISBN 978-1-886529-10-6.

Mohak Bhardwaj, Sanjiban Choudhury, and Byron Boots. Blending MPC & value function approximation for efficient reinforcement learning, 2021. URL https://arxiv.org/abs/2012.05909.

Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion, 2019. URL https://arxiv.org/abs/1807.01675.

Jan Carius, Farbod Farshidian, and Marco Hutter. Mpc-net: A first principles guided policy search. *IEEE Robotics and Automation Letters*, 5(2):2897–2904, April 2020. ISSN 2377-3774. doi: 10.1109/lra.2020.2974653. URL http://dx.doi.org/10.1109/LRA.2020.2974653.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models, 2018. URL https://arxiv.org/abs/1805.12114.

Ignasi Clavera, Violet Fu, and Pieter Abbeel. Model-augmented actor-critic: Backpropagating through paths, 2020. URL https://arxiv.org/abs/2005.08068.

Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I. Jordan, Joseph E. Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning, 2018. URL https://arxiv.org/abs/1803.00101.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods, 2018. URL https://arxiv.org/abs/1802.09477.

Sebastien Gros and Mario Zanon. Data-driven economic nmpc using reinforcement learning. *IEEE Transactions on Automatic Control*, 65(2):636–648, February 2020. ISSN 2334-3303. doi: 10.1109/tac.2019.2913768. URL http://dx.doi.org/10.1109/TAC.2019.2913768.

Shixiang Gu, Timothy P. Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. *CoRR*, abs/1603.00748, 2016. URL http://arxiv.org/abs/1603.00748.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies, 2017. URL https://arxiv.org/abs/1702.08165.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290.

Tuomas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning, 2019. URL https://arxiv.org/abs/1812.11103.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination, 2020. URL https://arxiv.org/abs/1912.01603.

Nicolas Heess, Greg Wayne, David Silver, Timothy Lillicrap, Yuval Tassa, and Tom Erez. Learning continuous control policies by stochastic value gradients, 2015. URL https://arxiv.org/abs/1510.09142.

Michael Hertneck, Johannes Kohler, Sebastian Trimpe, and Frank Allgower. Learning an approximate model predictive controller with guarantees. *IEEE Control Systems Letters*, 2(3):543–548, 2018. ISSN 2475-1456. doi: 10.1109/lcsys.2018.2843682. URL http://dx.doi.org/10.1109/LCSYS.2018.2843682.

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), January 2019. ISSN 2470-9476. doi: 10.1126/scirobotics.aau5872. URL http://dx.doi.org/10.1126/scirobotics.aau5872.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization, 2021. URL https://arxiv.org/abs/1906.08253.

Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, August 2023.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization, 2018. URL https://arxiv.org/abs/1802.10592.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles, 2017. URL https://arxiv.org/abs/1612.01474.

Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/file/c7c9344b5a3c0533e29fa69ce807cf08-Paper.pdf.

Sergey Levine and Vladlen Koltun. Guided policy search. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pp. III–1–III–9. JMLR.org, 2013.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019. URL https://arxiv.org/abs/1509.02971.

Kendall Lowrey, Aravind Rajeswaran, Sham M. Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. *CoRR*, abs/1811.01848, 2018. URL http://arxiv.org/abs/1811.01848.

Hamid Reza Maei and Richard S. Sutton. Gq(lambda): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In *Proceedings of the 3d Conference on Artificial General Intelligence (2010)*, pp. 100–105. Atlantis Press, 2010/06. ISBN 978-90-78677-36-9. doi: 10.2991/agi.2010.22. URL https://doi.org/10.2991/agi.2010.22.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016. URL https://arxiv.org/abs/1602.01783.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and efficient off-policy reinforcement learning, 2016. URL https://arxiv.org/abs/1606.02647.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pp. 278–287, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1558606122.

OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation, 2019. URL https://arxiv.org/abs/1808.00177.

Luis Pineda, Brandon Amos, Amy Zhang, Nathan O. Lambert, and Roberto Calandra. Mbrl-lib: A modular library for model-based reinforcement learning. *Arxiv*, 2021. URL https://arxiv.org/abs/2104.10159.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL https://jmlr.org/papers/v22/20-1364.html.

Rudolf Reiter, Andrea Ghezzi, Katrin Baumgärtner, Jasper Hoffmann, Robert D. McAllister, and Moritz Diehl. Ac4mpc: Actor-critic reinforcement learning for nonlinear model predictive control, 2024. URL https://arxiv.org/abs/2406.03995.

Rudolf Reiter, Jasper Hoffmann, Dirk Reinhardt, Florian Messerer, Katrin Baumgärtner, Shamburaj Sawant, Joschka Boedecker, Moritz Diehl, and Sebastien Gros. Synthesis of model predictive control and reinforcement learning: Survey and classification, 2025. URL https://arxiv.org/abs/2502.02133.

Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning, 2024. URL https://arxiv.org/abs/2310.12921.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2016. URL https://arxiv.org/abs/1511.05952.

Philipp Schoch and Markus Ryll. Optimal control with learned critics under unmodeled state dependencies. In *Conference on Robot Learning (CoRL)*, 2026.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, pp. 216–224, 1990.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Erik Talvitie. Self-correcting models for model-based reinforcement learning, 2016. URL https://arxiv.org/abs/1612.06018. AAAI Conference on Artificial Intelligence, 2017.

Dong Tian, Onur Celik, and Gerhard Neumann. Chunking the critic: A transformer-based soft actor-critic with n-step returns, 2026. URL https://arxiv.org/abs/2503.03660.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments, 2025. URL https://arxiv.org/abs/2407.17032.

Kim P. Wabersich and Melanie N. Zeilinger. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems, 2021. URL https://arxiv.org/abs/1812.05506.

Jiahui Zhang, Yusen Luo, Abrar Anwar, Sumedh Anand Sontakke, Joseph J Lim, Jesse Thomason, Erdem Biyik, and Jesse Zhang. Rewind: Language-guided rewards teach robot policies without new demonstrations, 2025. URL https://arxiv.org/abs/2505.10911.

Brian D. Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010. URL https://api.semanticscholar.org/CorpusID:11919065.

A Residual Form of the λ-Soft Estimate

This appendix expands the λ-soft recursion in equation 9 into the explicit form in equation 11. Schoch & Ryll (2026, App. A) perform the same expansion for the hard blend, from equation 7 to equation 8; the derivation below differs by the per-node blending factor λ_t and the entropy term.

The rollout pairs $(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi$ start at the anchor $\mathbf{s}_0$ and follow the state-action trajectory distribution under π. We use the shorthand $r_t = r(\mathbf{s}_t, \mathbf{a}_t)$, $Q_{\theta,t} = Q_\theta(\mathbf{s}_t, \mathbf{a}_t)$, $\pi_t = \pi(\mathbf{a}_t \mid \mathbf{s}_t)$, $Q_t^\lambda = Q^\lambda(\mathbf{s}_t, \mathbf{a}_t)$ and $V_t^\lambda = V^\lambda(\mathbf{s}_t)$. Without subscript, $\mathbb{E}$ denotes the expectation over the rollout under ρ_π, conditioned on the arguments of the quantity it defines. For $t = 0, \dots, H-2$, the recursion of equation 9 in shorthand, closed at the last node by $\lambda_{H-1} = 0$, reads

$$Q_t^\lambda = (1-\lambda_t)\, Q_{\theta,t} + \lambda_t \big(r_t + \gamma\, \mathbb{E}\big[V_{t+1}^\lambda\big]\big), \qquad Q_{H-1}^\lambda = Q_{\theta,H-1}. \tag{A1}$$

We define the deviation of the blended estimate from the learned critic as

$$\delta_t = Q_t^\lambda - Q_{\theta,t}. \tag{A2}$$

Subtracting $Q_{\theta,t}$ from both sides of equation A1 and expanding V_{t+1}^λ by its definition gives

$$\begin{aligned} \delta_t &= (1-\lambda_t)\, Q_{\theta,t} + \lambda_t \big(r_t + \gamma\, \mathbb{E}\big[V_{t+1}^\lambda\big]\big) - Q_{\theta,t} \\ &= \lambda_t \big(r_t + \gamma\, \mathbb{E}\big[Q_{t+1}^\lambda - \alpha \log \pi_{t+1}\big] - Q_{\theta,t}\big). \end{aligned} \tag{A3}$$

Rewriting the expectation with $Q_{t+1}^\lambda = Q_{\theta,t+1} + \delta_{t+1}$ from equation A2 yields

$$\delta_t = \lambda_t \big(r_t + \gamma\, \mathbb{E}\big[Q_{\theta,t+1} + \delta_{t+1} - \alpha \log \pi_{t+1}\big] - Q_{\theta,t}\big). \tag{A4}$$

With the soft one-step Bellman residual

$$B_t = r_t + \gamma\, \mathbb{E}\big[Q_{\theta,t+1} - \alpha \log \pi_{t+1}\big] - Q_{\theta,t}, \tag{A5}$$

equation A4 becomes a recursion in δ alone,

$$\delta_t = \lambda_t\big(B_t + \gamma\,\mathbb{E}\big[\delta_{t+1}\big]\big). \tag{A6}$$

Since $\delta_{H-1} = 0$ by equation A1, unrolling equation A6 gives

$$\delta_t = \mathbb{E}\left[\sum_{k=t}^{H-2} \gamma^{k-t}\Big(\prod_{j=t}^{k} \lambda_j\Big) B_k\right], \tag{A7}$$

where the weights remain inside the expectation because λ_j is state-dependent. Setting $t = 0$ and substituting back $V^\lambda(\mathbf{s}_0) = \mathbb{E}\big[Q_{\theta,0} + \delta_0 - \alpha \log \pi_0\big]$ yields

$$V^\lambda(\mathbf{s}_0) = \mathbb{E}\left[Q_{\theta,0} - \alpha \log \pi_0 + \sum_{k=0}^{H-2} w_k\, B_k\right], \qquad w_k = \gamma^k \prod_{j \le k} \lambda_j. \tag{A8}$$

Sampling the rollout gives equation 11 in the main text.

Two special cases connect equation A8 to known estimators. For $\alpha = 0$ and a constant $\lambda_t \equiv \lambda$, the weights become $w_k = \lambda(\lambda\gamma)^k$ and B_k reduces to the hard residual of equation 8, which recovers its sum. The terminal term of equation 8 is absent because equation A1 closes with a pure bootstrap, whereas the MPC rollout of Schoch & Ryll (2026) collects a reward at its last node. For $\lambda_t \equiv 1$ at all nodes before the last, the weights become $w_k = \gamma^k$, the intermediate $Q_{\theta,k}$ telescope, and

$$V^{\lambda=1}(\mathbf{s}_0) = \mathbb{E}\left[\sum_{k=0}^{H-2} \gamma^k r_k + \gamma^{H-1} Q_{\theta,H-1} - \alpha \sum_{k=0}^{H-1} \gamma^k \log \pi_k\right], \tag{A9}$$

which is the maximum-entropy model value expansion of Section 4.

B Proposition 1

The analysis assumes an exact transition model. For $n = 0, \ldots, H-1$ steps along the rollout, we define the soft n-step return

$$G^{(n)} = \sum_{t=0}^{n-1} \gamma^t\big(r_t - \alpha \log \pi_t\big) + \gamma^n\big(Q_{\theta,n} - \alpha \log \pi_n\big), \qquad G^{(0)} = Q_{\theta,0} - \alpha \log \pi_0, \tag{A10}$$

which consists of n steps of the maximum-entropy stage reward followed by a soft bootstrap. Consecutive returns differ by the sampled soft Bellman residual of equation A5,

$$G^{(n+1)} - G^{(n)} = \gamma^n\Big[\big(r_n - \alpha \log \pi_n\big) + \gamma\big(Q_{\theta,n+1} - \alpha \log \pi_{n+1}\big) - \big(Q_{\theta,n} - \alpha \log \pi_n\big)\Big] = \gamma^n B_n, \tag{A11}$$

since the two $-\alpha \log \pi_n$ terms cancel.

Convex combination. Substituting equation A11 into equation A8 turns each residual into a difference of consecutive returns, $w_k B_k = c_k\big(G^{(k+1)} - G^{(k)}\big)$: the factor γ^k of equation A11 cancels the discount in $w_k = \gamma^k \prod_{j \le k} \lambda_j$ and leaves the bare product $c_k = \prod_{j \le k} \lambda_j$, which satisfies $c_0 \ge c_1 \ge \cdots \ge 0$ for $\lambda_j \in [0, 1]$. For a sequence (λ_t) fixed in advance, i.e., independent of the rollout, these weights are deterministic and move out of the expectation (Appendix C extends this to the guided λ_t), so summation by parts gives

$$V^\lambda(\mathbf{s}_0) = \mathbb{E}\Big[\underbrace{G^{(0)} + \sum_{k=0}^{H-2} c_k\big(G^{(k+1)} - G^{(k)}\big)}_{Q^\lambda(\mathbf{s}_0,\mathbf{a}_0) - \alpha \log \pi_0}\Big] = \sum_{n=0}^{H-1} \beta_n\, \mathbb{E}\big[G^{(n)}\big], \tag{A12}$$

with the weights

$$\beta_0 = 1 - c_0, \qquad \beta_n = c_{n-1} - c_n \quad (1 \le n \le H-2), \qquad \beta_{H-1} = c_{H-2}. \tag{A13}$$

The β_n are non-negative, since c_k is non-increasing, and sum to one. Hence equation A8 is a convex combination of $G^{(0)}, \ldots, G^{(H-1)}$ for any fixed $\lambda_t \in [0, 1]$.

Contraction. The recursion in equation A1 maps the critic to the blended estimate. Applying it recursively along the rollout (Bertsekas & Tsitsiklis, 1996, Prop. 2.7) defines the operator $\mathcal{M}^\lambda$ with input Q_θ and output Q^λ,

$$\big(\mathcal{M}^\lambda Q_\theta\big)(\mathbf{s}_0, \mathbf{a}_0) := Q^\lambda(\mathbf{s}_0, \mathbf{a}_0) = \sum_{n=0}^{H-1} \beta_n \big(\mathcal{T}^\pi\big)^n Q_\theta(\mathbf{s}_0, \mathbf{a}_0), \tag{A14}$$

a convex combination of soft Bellman backups, where $\mathcal{T}^\pi$ is the maximum-entropy policy evaluation operator in equation 2 (Haarnoja et al., 2018). Each $(\mathcal{T}^\pi)^n$ has the fixed point Q^π_{soft} (Haarnoja et al., 2018, Lemma 1), so $\mathcal{M}^\lambda$ shares this fixed point and is Lipschitz with modulus $\kappa \le 1$ (Appendix C). The target in equation 12 applies one real Bellman step on the blended estimate, $\mathcal{T}^\pi \circ \mathcal{M}^\lambda$, which is therefore a γ-contraction with fixed point Q^π_{soft} for any (λ_t).

Minimum over the critic heads. The implementation bootstraps on $\min_i Q_{\theta_i}$ (Section 5) and regresses every head $Q_i = Q_{\theta_i}$ onto the target built from it. Since $|\min_i a_i - \min_i b_i| \le \max_i |a_i - b_i|$, this ensemble operator contracts in $\max_i \|Q_i - Q'_i\|_\infty$ with the same modulus as $\mathcal{T}^\pi \circ \mathcal{M}^\lambda$ and has the fixed point $Q_i = Q^\pi_{\text{soft}}$ for all i.

C Proposition 2

C.1 Derivation of the λ-soft operator's contraction rate

The operator $\mathcal{M}^\lambda$ of equation A14 is the per-node, finite-horizon form of the λ-policy-iteration operator of Bertsekas & Tsitsiklis (1996, Sec. 2.3.1). Since $\mathcal{T}^\pi$ is a γ-contraction (Haarnoja et al., 2018, Lemma 1), $(\mathcal{T}^\pi)^n$ contracts at rate γ^n, and the triangle inequality gives

$$\big\|\mathcal{M}^\lambda Q - \mathcal{M}^\lambda Q'\big\|_\infty \le \sum_{n=0}^{H-1} \beta_n \gamma^n \big\|Q - Q'\big\|_\infty.$$

The modulus is thus $\kappa = \sum_n \beta_n \gamma^n$, the β-weighted average of the per-depth rates. With the conventions $c_{-1} = 1$ and $c_{H-1} = 0$, the latter following from $\lambda_{H-1} = 0$, the three cases of equation A13 read $\beta_n = c_{n-1} - c_n$ for all n, and the modulus evaluates to

$$\begin{aligned}
\kappa := \sum_{n=0}^{H-1} \beta_n \gamma^n &= \sum_{n=0}^{H-1} c_{n-1}\gamma^n - \sum_{n=0}^{H-1} c_n \gamma^n \\
&= 1 + \gamma \sum_{k=0}^{H-2} c_k \gamma^k - \sum_{k=0}^{H-2} c_k \gamma^k \\
&= 1 - (1-\gamma) \sum_{k=0}^{H-2} c_k \gamma^k \\
&= 1 - (1-\gamma) \sum_{k=0}^{H-2} w_k.
\end{aligned} \tag{A15}$$

Here $w_k = \gamma^k c_k$ is the weight that the residual of node k carries in equation A8, so κ is set by the residual mass $\sum_k w_k$. The target in equation 12 starts with one real transition from the replay buffer (Section 5.1), so it applies $\mathcal{T}^\pi \circ \mathcal{M}^\lambda$, and the algorithm contracts at rate $\gamma\kappa$.

C.2 Contraction rates of concrete instantiations of the operator

Three instantiations of equation A15 recover known rates:

- $\lambda_t \equiv 0$: $w_k = 0$, so $\kappa = 1$ and $\gamma\kappa = \gamma$, the one-step rate of SAC.
- $\lambda_t \equiv 1$: $w_k = \gamma^k$, so $\kappa = \gamma^{H-1}$ and $\gamma\kappa = \gamma^H$, the rate of the full H-step target of anchor transition and $H-1$ model transitions.

- constant λ and $H \to \infty$: $w_k = \lambda(\lambda\gamma)^k$, so $\kappa = (1-\lambda)/(1-\gamma\lambda)$ and $\gamma\kappa = \gamma(1-\lambda)/(1-\gamma\lambda)$, the modulus of the λ-policy-iteration operator of Bertsekas & Tsitsiklis (1996, Sec. 2.3.1).

The rate $\gamma\kappa$ therefore ranges over $[\gamma^H, \gamma]$: under an exact model, blending can only accelerate the contraction relative to SAC, and equation A15 is the per-node, finite-horizon form of the TD(λ) rate.

Guided λ_t. Within a backup, the critic and the dynamics model are fixed, so the guidance in equation 15 is a fixed function $\lambda_t = \lambda(\mathbf{s}_t, \mathbf{a}_t)$ of the node, i.e., a state-dependent λ (Maei & Sutton, 2010/06). The weights β_n then vary with the rollout and no longer move out of the expectation in equation A12, but the fixed point is kept: λ_t depends only on node t, not on the next state that the residual B_t samples, and at Q^π_{soft} every residual has zero mean. The expression in equation A15 holds on every rollout, and its average over rollouts at the worst anchor gives the modulus, so $\gamma\kappa \in [\gamma^H, \gamma]$ is unchanged. Across backups, λ changes with the critic and the model, so every backup applies a different operator. All of them share the fixed point Q^π_{soft} and contract with at least γ, so the iteration converges to it.

D Experimental Setup

All experiments run on a consumer-grade notebook with an NVIDIA RTX 5080 mobile GPU, an AMD Ryzen 9 9955HX CPU, and 64 GB of RAM. λ-SAC captures its update step in a single CUDA graph, and both λ-SAC and SAC simulate the environment with MuJoCo on a single CPU thread.

D.1 λ-SAC Settings

Table 1 lists all hyperparameters of λ-SAC, which builds on the SAC implementation of Stable-Baselines3 (Raffin et al., 2021); all tasks share them.

Table 1: Hyperparameters of λ-SAC, used for all tasks.

SAC	
Discount γ	0.99
Replay buffer size	10^6
Batch size	256
Steps before the first update	1000
Update interval	every environment step
Gradient steps per update	1
Initial temperature α	1, learned
Normalization	observations and rewards
λ-soft policy iteration	
Rollout horizon H	5
Critics N	4, reduced by $\min_i$
Policy evaluation	λ-soft target equation 12
Policy improvement	λ-soft objective equation 13
Gradient through the rollout	yes
Uncertainty guidance	
Critic confidence κ_Q	5
Model confidence κ_f	0.5
Dynamics model	
Ensemble members E	4
Refit interval	every 100 environment steps
Gradient steps per refit	100

D.2 Network Architectures and Optimization

Table 2 lists the networks of λ-SAC. Actor and critics are the Stable-Baselines3 defaults for SAC; each of the N critics is a separate MLP. The temperature α is trained with Adam at a learning rate of $3 \cdot 10^{-4}$. The E dynamics members share no weights and are evaluated in one batched matrix multiplication; weights and biases are initialized uniformly in $\pm 1/\sqrt{d_{\text{in}}}$, with d_{in} the input width of the layer. Each member draws its own windows of $L = 5$ consecutive transitions uniformly from the replay buffer and drops windows that contain an episode end. Every refit continues from the previous weights and Adam state and recomputes the standardization of input and delta over all stored transitions. Model rollouts propagate the mean of the members.

Table 2: Network architectures and optimizer settings of λ-SAC.

	Actor	Critic ($\times N$)	Dynamics ($\times E$)
Input	observation	observation, action	standardized $(\mathbf{s}, \mathbf{a})$
Hidden layers	2×256	2×256	3×128
Activation	ReLU	ReLU	SiLU
Output	tanh-Gaussian	Q	standardized $\Delta\mathbf{x}$
Optimizer	Adam	Adam	Adam
Learning rate	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	10^{-3}
Gradient clipping	none	none	10
Batch size	256	256	512 windows
Loss	equation 13	equation 12	equation 16

D.3 Settings for other baselines

SAC uses the Stable-Baselines3 implementation (Raffin et al., 2021) with its default settings, two critics, and a UTD ratio of 1. PPO uses the Stable-Baselines3 implementation with its default settings, which amount to a UTD ratio of 0.156. MBPO uses the unmodified implementation of mbrl-lib (Pineda et al., 2021) with its default settings, leading to a UTD of 20. STEVE uses a public implementation[1], tuned for the v1 versions of the tasks, which we adapted to the v5 versions; it runs also with its default settings and a UTD ratio of 4.

E Ablation Details

All ablations run on Humanoid for 400k environment steps, with one seed per variant. Each variant changes one setting of the configuration in Table 1. Return curves show the evaluation return, slightly smoothed with an exponential moving average of weight 0.2. Loss panels show the actor loss, the critic loss per critic, the dynamics fit loss equation 16, and the critic disagreement σ_Q of equation 14. During the ablations, several runs shared the GPU. Wall-clock times are therefore comparable only between runs of the same batch, and we report only those of direct relevance to the analysis, namely the PE/PI mode and the two UTD ratios. The reported times are not comparable with those in Section 6, where each method did run sequentially.

E.1 Policy Evaluation and Policy Improvement

This ablation varies the step of soft policy iteration that uses the λ-soft estimate. The estimate enters policy evaluation equation 12 only (PE only), policy improvement equation 13 only (PI only), both steps (PE + PI), or neither step. The last variant is SAC with $N = 4$ critics. PE + PI learns fastest, followed by PE only, PI only, and SAC (Figure 4). This order holds over environment steps and over wall-clock time. The dynamics model is trained the same way in all four variants, and yet its loss follows the same order.

The two steps use the estimate in different ways. In policy improvement, the estimate gives the actor a multi-step gradient through the model rollout. In policy evaluation, it gives the critic a better target.

[1] https://github.com/tensorflow/models/tree/archive/research/steve

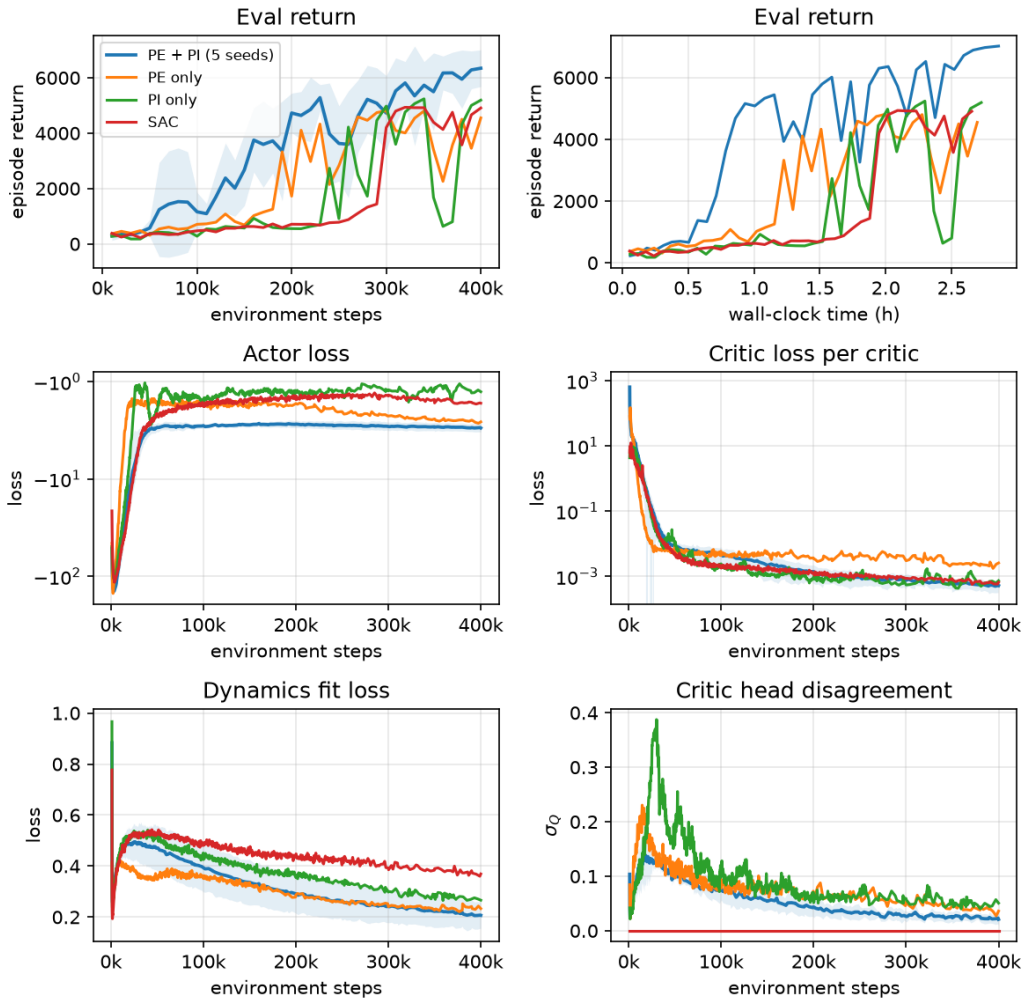

Figure 4: λ-soft estimate in policy evaluation (PE), policy improvement (PI), both, or neither (SAC). Top: return over environment steps and wall-clock time. Middle and bottom: training losses; the SAC variant does not compute σ_Q.

The actor benefits from this better critic as well, because its gradient passes through the critic. We believe this is why policy evaluation gains more than policy improvement.

E.2 Gradient through the Rollout

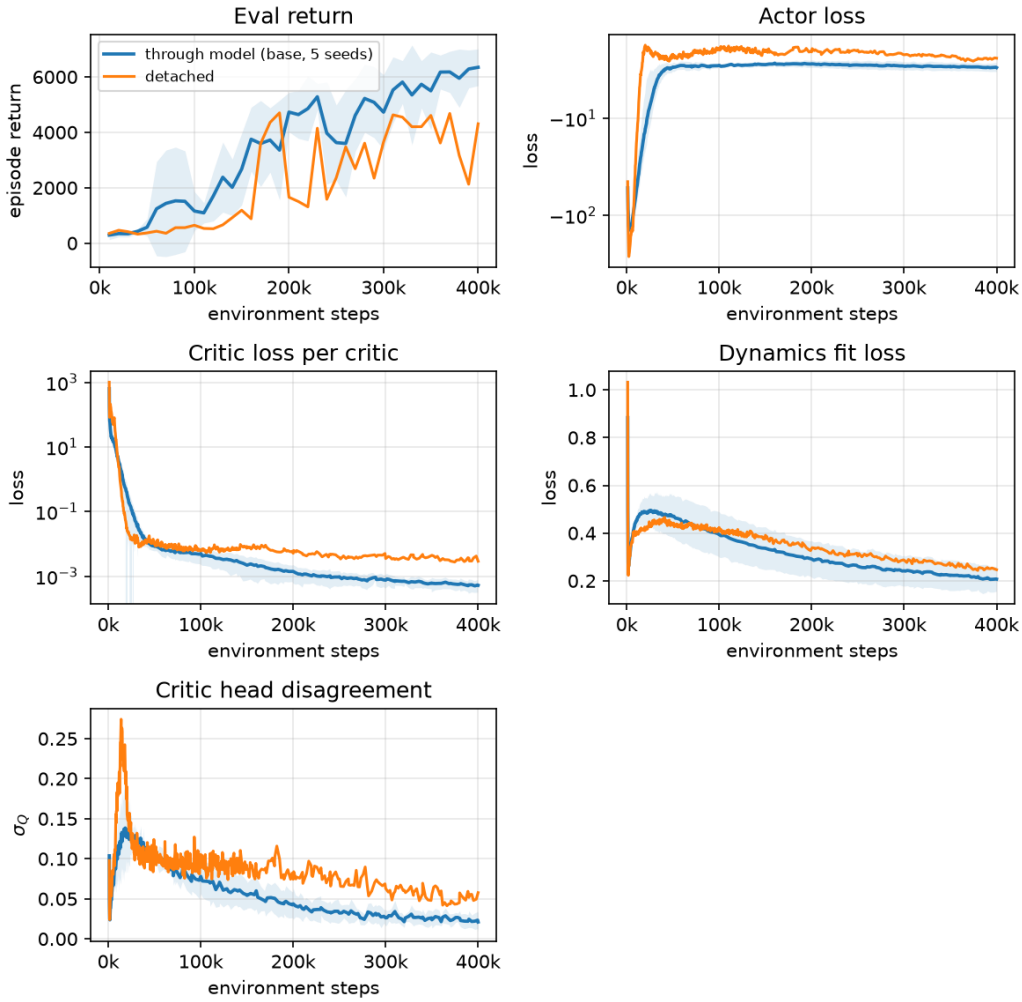

Figure 5: Actor gradient through the model rollout against detached model states in policy improvement: return and training losses.

This ablation detaches the model states in policy improvement. The actor then gets no gradient through the dynamics, and the rollout contributes only through the actions that the policy takes at its nodes. With detached states, the return rises later and stays below that of the full method over the whole budget (Figure 5). Its curve is close to the PE-only curve in Figure 4. The dynamics loss, the critic loss, and σ_Q also end higher.

Without the gradient through the rollout, λ-SAC relies mostly on the better critic from policy evaluation. Querying the policy at additional nodes adds little on its own. The gain in policy improvement thus comes from the multi-step gradient through the model.

E.3 Rollout Horizon

This ablation varies the rollout horizon H between 2, 5, and 10. With $H = 2$, λ-SAC learns slowest and does not reach the plateau of the other two within the budget (Figure 6). $H = 10$ rises later than $H = 5$. It catches up once its dynamics loss approaches that of $H = 5$, and from then on both improve at the same rate and plateau at the same return.

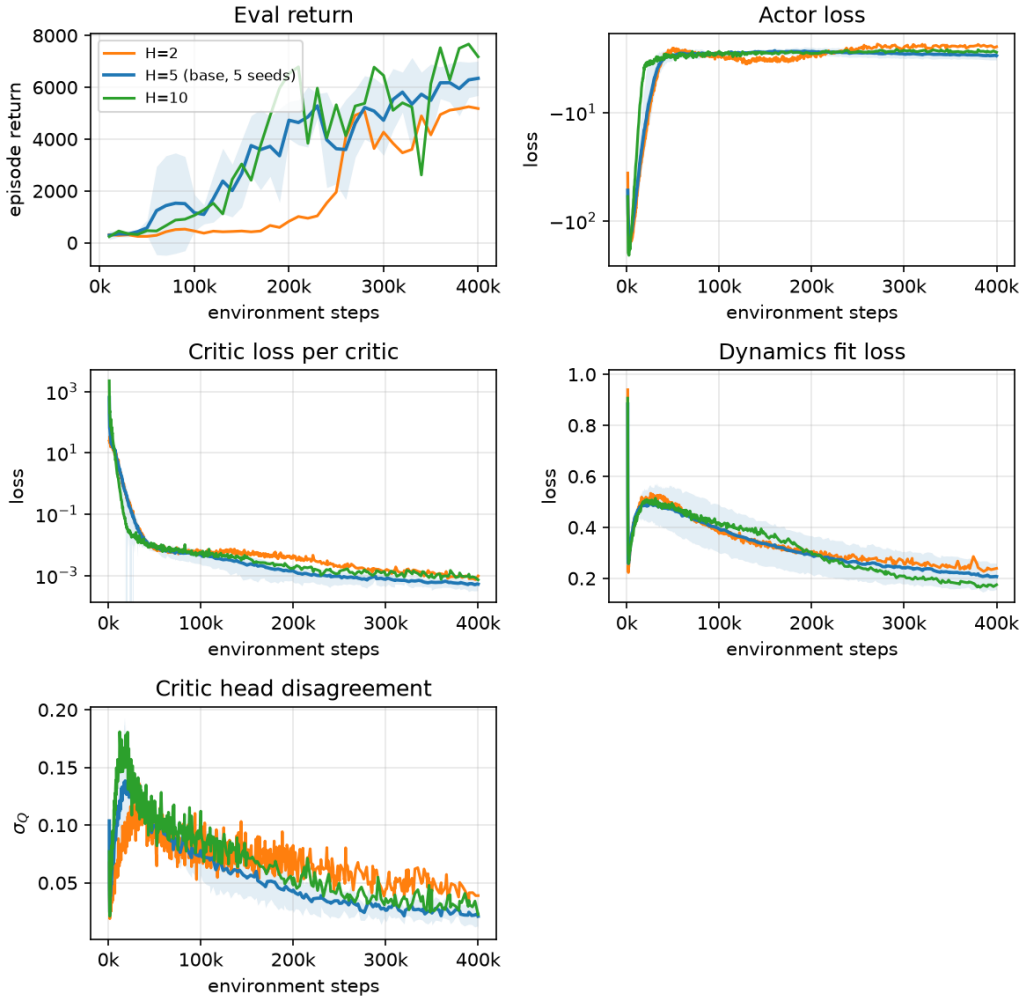

Figure 6: Rollout horizon $H \in \{2, 5, 10\}$: return and training losses.

A longer horizon speeds up the contraction (Proposition 2). Early in training, however, the model is poor, and a longer rollout carries more of its error into the estimate. The guidance limits this error, but it was not tuned for $H = 10$, and $H = 10$ therefore starts slower. Nevertheless, the guidance keeps the run learning. The policy keeps collecting data, the dynamics loss drops, and $H = 10$ catches up with $H = 5$.

E.4 Uncertainty Guidance against Uninformed λ

This ablation replaces the guided λ_t with an uninformed one. The random variant draws λ_t uniformly from $[0, 1]$ at every node. The fixed variants disable the guidance and apply a constant λ. Their values match the effective horizon of the other two runs: fixed 0.897 matches the guided run, and fixed 0.5 the random run. The effective-horizon panel of Figure 7 confirms this match. The guided h_{eff} starts lower and reaches that of fixed 0.897 as its dynamics loss drops. The guided run learns fastest. Fixed 0.897 and random end at a similar return, and fixed 0.5 ends lowest. In the losses, fixed 0.897 has the highest critic loss, and the guided run has the lowest dynamics loss.

At the same effective horizon, the guided run thus beats fixed 0.897. Where λ is placed matters, not how far the rollout reaches on average. A fixed high λ trusts the rollout everywhere, also where the model is wrong. The critic then fits targets that carry model error, which raises its loss. The guidance

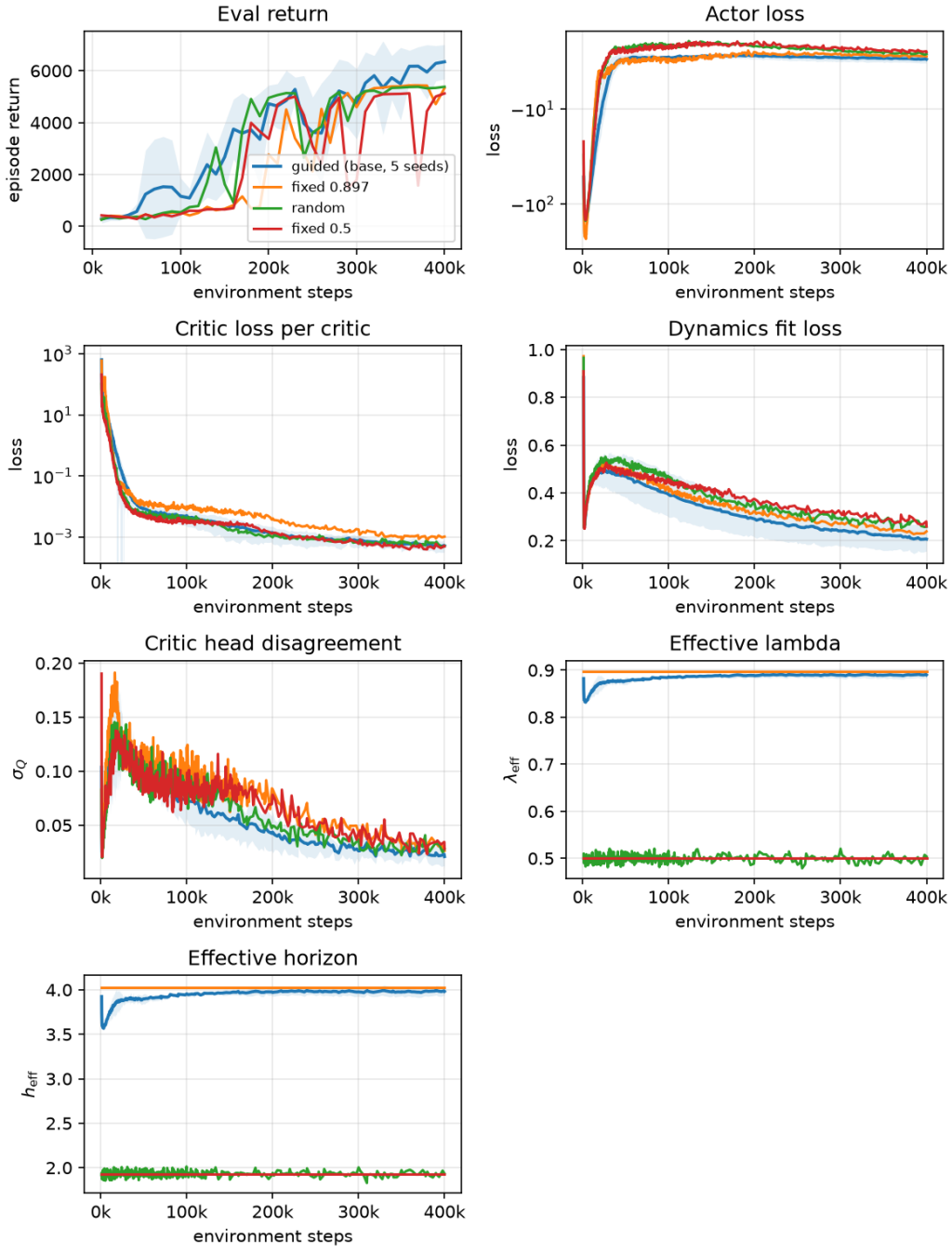

Figure 7: Guided λ_t against random $\lambda_t \sim \mathcal{U}[0, 1]$ and fixed λ, matched in effective horizon: fixed 0.897 to guided, fixed 0.5 to random. Return, training losses, and the effective λ and h_{eff} that the runs apply.

reaches the same average horizon, but it cuts the rollout short where the model is not trusted. This keeps the critic loss at the level of the short-horizon runs.

The guidance also pays off through the data. λ_t does not enter the model loss, since the model trains only on real transitions from the replay buffer. The guided policy, however, collects better data. These data improve the dynamics fit, and the better fit lets the guidance lengthen the rollout.

An uninformed λ collects worse data from the start. Its model therefore does not become accurate enough within the budget for the long rollout to pay off.

E.5 Confidence Scales κ_f and κ_Q

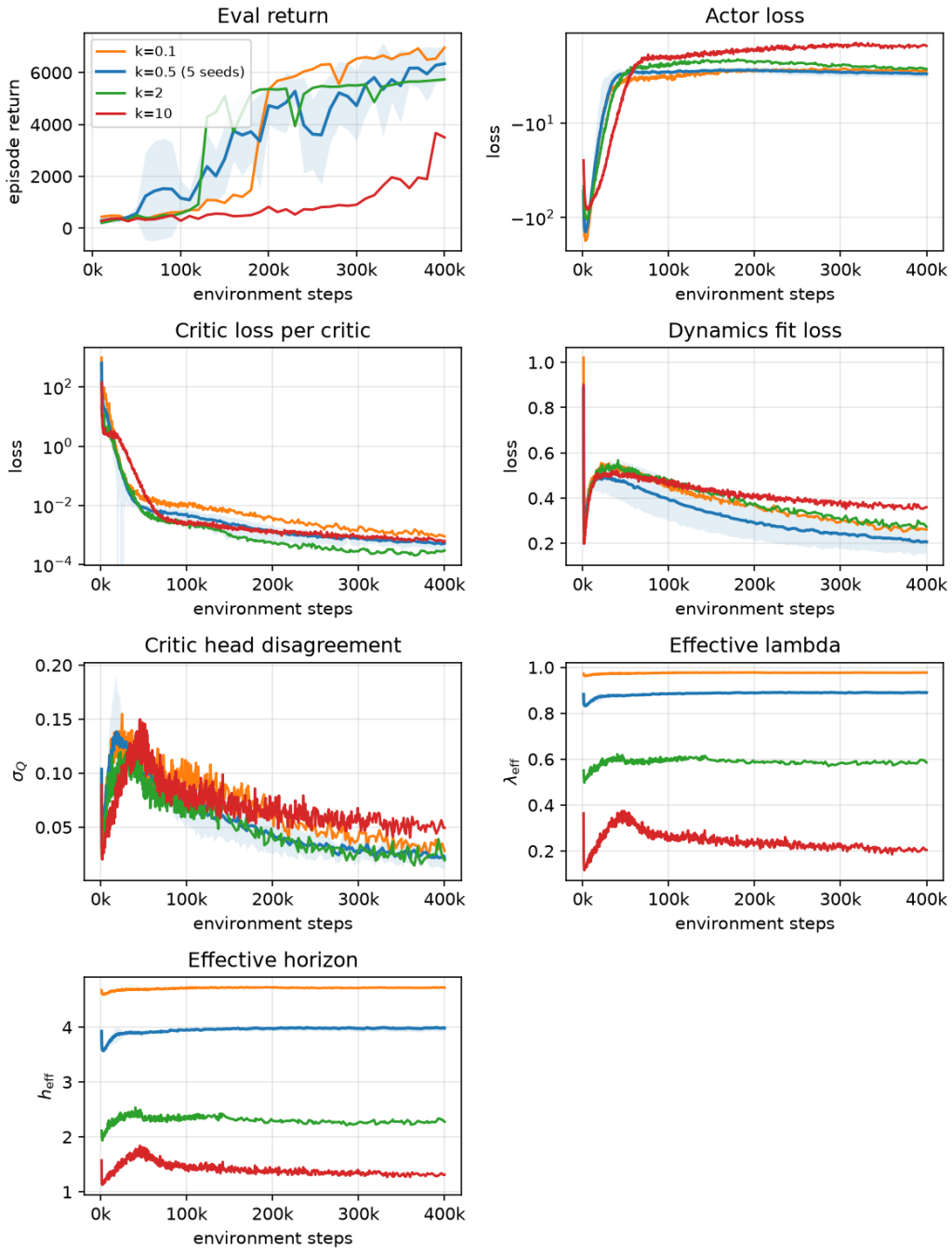

Figure 8: Model confidence scale $\kappa_f \in \{0.1, 0.5, 2, 10\}$ at $\kappa_Q = 5$: return, training losses, and the effective λ and h_{eff}.

This ablation varies the scales of the confidence kernel in equation 15: κ_f between 0.1, 0.5, 2, and 10, and κ_Q between 1, 5, 15, and 50. The effective λ falls with κ_f across almost all of $[0, 1]$

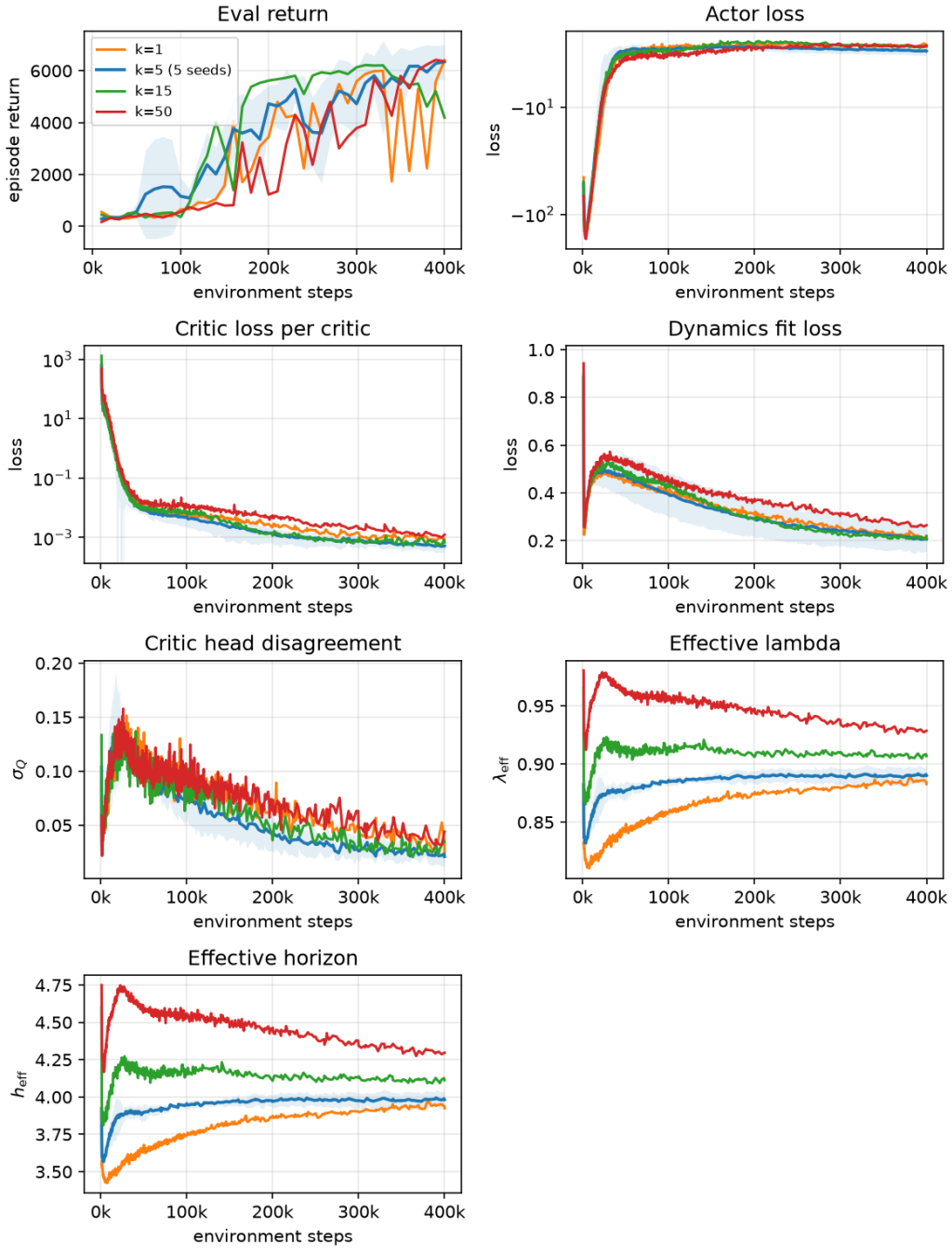

Figure 9: Critic confidence scale $\kappa_Q \in \{1, 5, 15, 50\}$ at $\kappa_f = 0.5$: return, training losses, and the effective λ and h_{eff}.

(Figure 8) and rises with κ_Q only within a narrow band, which is widest early in training (Figure 9). With $\kappa_f = 10$, h_{eff} stays close to one, the return rises far more slowly than in all other runs, and the dynamics loss and σ_Q end highest. $\kappa_f = 2$ rises earliest and $\kappa_f = 0.1$ ends highest, both within the seed spread of $\kappa_f = 0.5$. All κ_Q variants reach the plateau of $\kappa_Q = 5$, and $\kappa_Q = 50$ reaches it last. Among the runs that reach the plateau, the critic loss grows with h_{eff}.

κ_f thus sets the effective horizon, and the return degrades only when a large κ_f cuts the rollout at most nodes and brings the estimate close to the model-free one-step estimate. κ_Q can only lift λ_t above the floor c_f (Section 5.3). At $\kappa_f = 0.5$ this floor is already high, so κ_Q leaves little room to change the effective horizon. That the short horizon of $\kappa_f = 2$ rises earliest and the long horizon of $\kappa_f = 0.1$ ends highest is consistent with the horizon ablation (Appendix E.3): a short rollout carries less model error while the model is poor, and a long one pays off once the model is accurate. A longer rollout also carries more model error into the critic target, which raises the critic loss without lowering the return.

E.6 Ensemble Sizes

Dynamics ensemble. This part varies the number of dynamics members E between 2, 4, and 8. The runs with $E = 2$ and $E = 8$ were exited preemptively. With $E = 2$, λ-SAC learns slowest, has the highest critic and dynamics losses, and from the middle of training on the highest σ_Q (Figure 10). $E = 8$ learns as fast as $E = 4$ but not faster, so the four additional members bring no gain. The effective λ and h_{eff} decrease with E, and they grow over training as the dynamics loss drops.

Two members are too few to estimate the disagreement reliably. The standard deviation over two members underestimates their spread, which lowers σ_f and raises confidence c_f, although the dynamics fit of $E = 2$ is the worst of the three. Its still quite high σ_Q lowers the confidence c_Q, which by equation 15 raises λ_t further. Thus $E = 2$ runs a slightly longer effective horizon than $E = 4$ and $E = 8$, although its dynamics fit is the worst of the three.

Critic ensemble. This part varies the number of critics N between 2, 4, and 8. The runs with $N = 2$ and $N = 8$ end before 400k environment steps. $N = 4$ learns fastest (Figure 11). $N = 2$ learns much more slowly and has the highest critic and dynamics losses, and $N = 8$ learns slightly more slowly than $N = 4$ at a similar critic and dynamics losses. Early in training, σ_Q and the effective λ grow with N. Later, both are similar across N, and $N = 2$ keeps the lowest effective λ.

The minimum over more heads counters overestimation, consistent with the lower critic loss for $N \geq 4$. The disagreement estimate shows the same small-sample effect as for $E = 2$: two heads underestimate their spread, so σ_Q is low and c_Q high, and by equation 15 the critic grants most of the cut that the model requests. $N = 2$ therefore bootstraps earlier than $N = 4$ and $N = 8$.

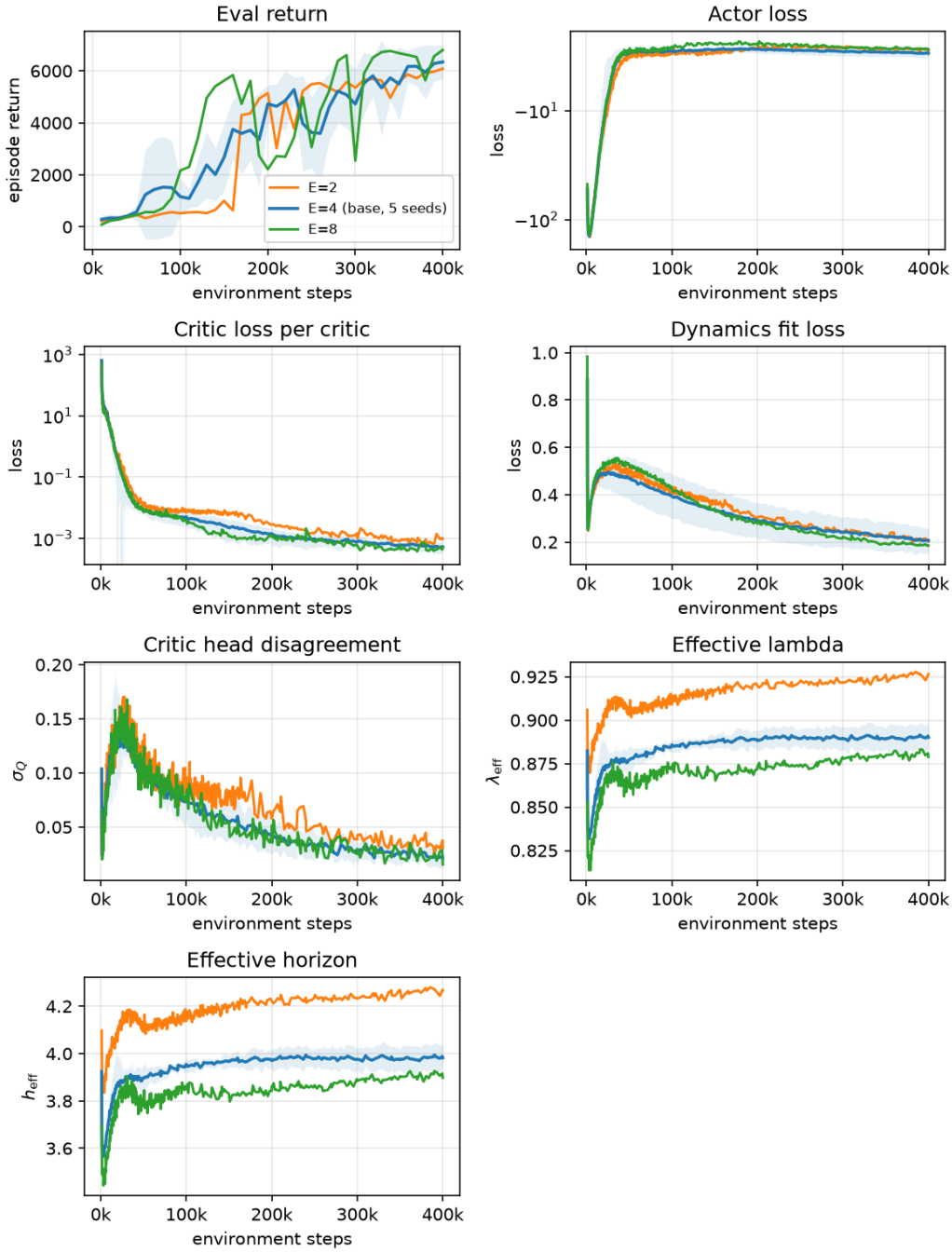

Figure 10: Dynamics ensemble size $E \in \{2, 4, 8\}$: return, training losses, and the effective λ and h_{eff}.

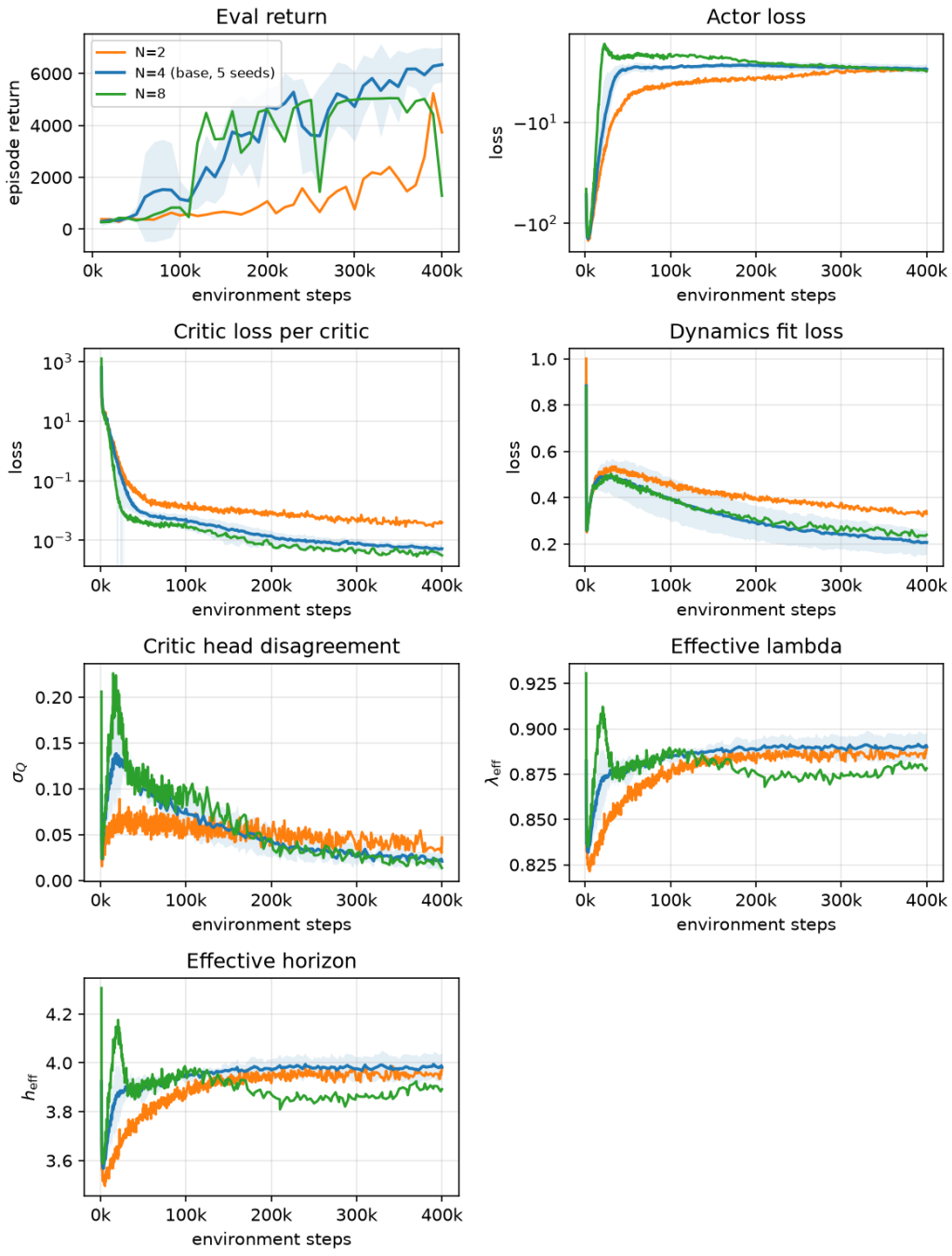

Figure 11: Critic ensemble size $N \in \{2, 4, 8\}$: return, training losses, and the effective λ and h_{eff}.

E.7 Dynamics UTD Ratio

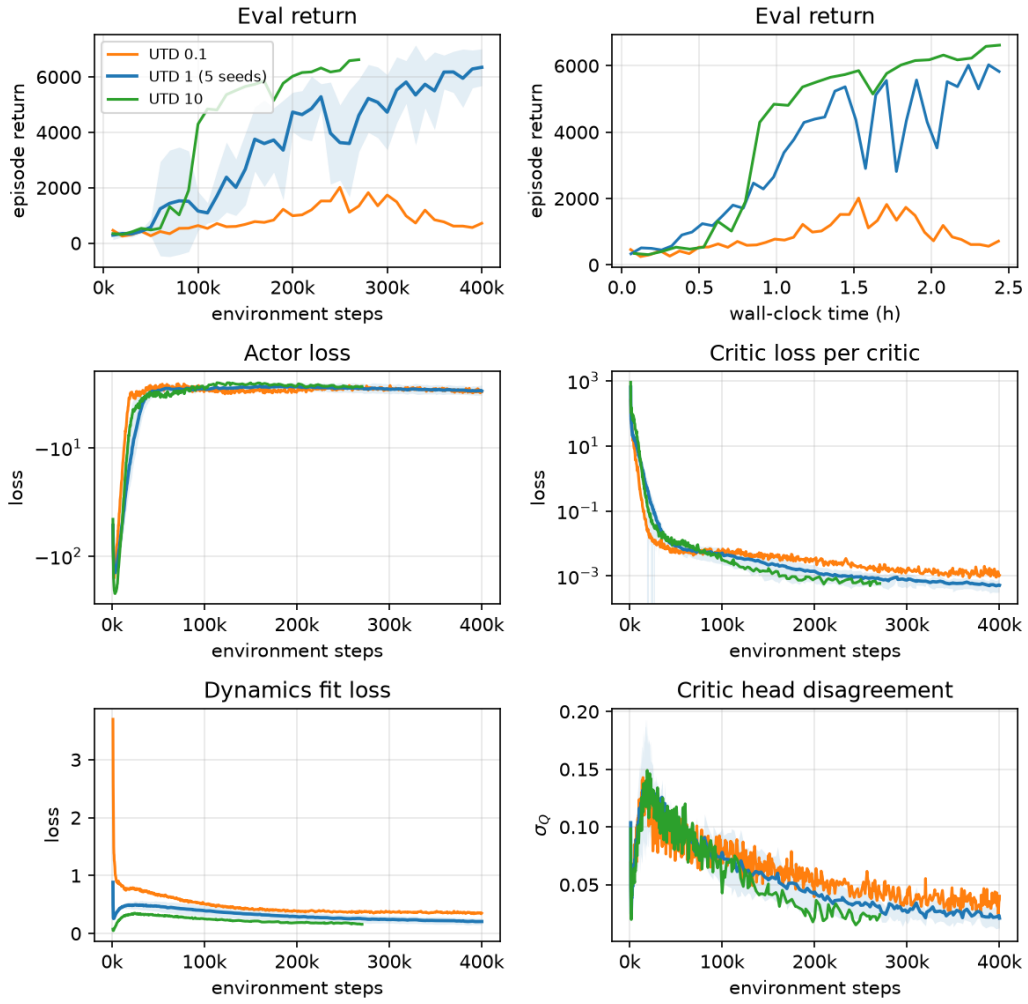

Figure 12: UTD ratio of the dynamics model, $\{0.1, 1, 10\}$, at an unchanged SAC update. Top: return over environment steps and wall-clock time. Middle and bottom: training losses.

This ablation varies the UTD ratio of the dynamics model between 0.1, 1, and 10. Each refit runs 10, 100, or 1000 gradient steps every 100 environment steps, while the SAC update stays unchanged. All three runs stop after the same wall-clock time. The return grows with the dynamics UTD ratio, over environment steps and over wall-clock time (Figure 12). With UTD 0.1, the return stays low. Over wall-clock time, UTD 10 starts slower than UTD 1 and then overtakes it. The dynamics loss decreases with the UTD ratio throughout. The critic loss and σ_Q of UTD 1 and UTD 10, in contrast, are similar at equal steps.

More gradient steps on the dynamics model lower the model error. A more accurate model makes exploration more directed and speeds up policy convergence. This is most visible in the phase in which the agent learns to stand and walk. UTD 1 and UTD 10 differ in return but not in critic loss and σ_Q, so these two quantities are only a coarse proxy for the return.

E.8 SAC UTD Ratio

This ablation varies the UTD ratio of the SAC update between 0.5, 1, 2, and 4. The SAC update runs 50, 100, 200, or 400 gradient steps every 100 environment steps, while the dynamics refit stays unchanged. UTD 1 learns fastest, and both lower and higher ratios learn more slowly (Figure 13).

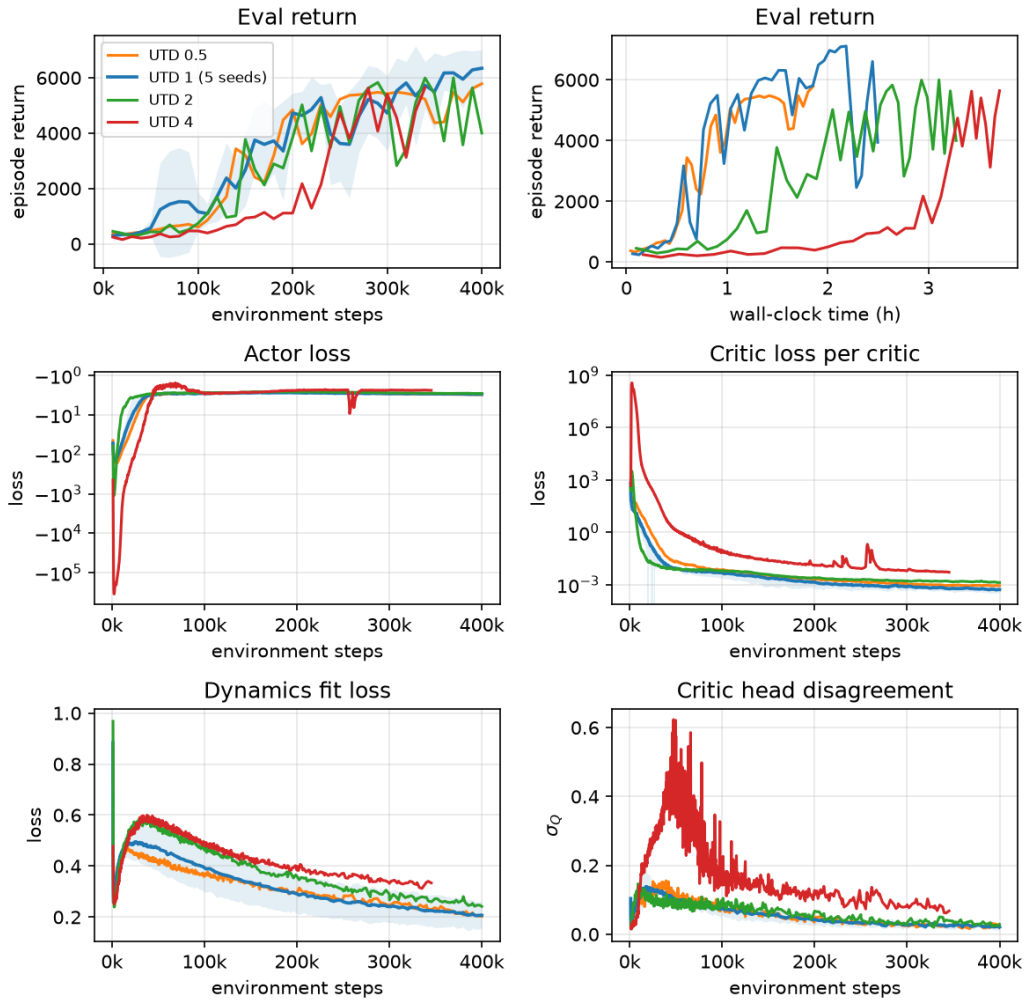

Figure 13: UTD ratio of the SAC update, $\{0.5, 1, 2, 4\}$, at an unchanged dynamics refit. Top: return over environment steps and wall-clock time. Middle and bottom: training losses.

Over wall-clock time, UTD 0.5 and UTD 1 are similar, and higher ratios are slower. UTD 4 shows a large early spike in critic loss and σ_Q. The dynamics loss follows the same order as the return.

λ-SAC thus gets the most out of the model when the dynamics loss is low. More SAC updates do not lower the dynamics loss by themselves. We therefore expect a higher SAC UTD ratio to pay off only together with a higher dynamics UTD ratio.

The SAC UTD ratio also sets a large part of the wall-clock time. Each gradient step of λ-SAC rolls out the model for the policy evaluation target and backpropagates through a second rollout for policy improvement, so it costs more than a SAC update, and its cost grows with H. The SAC UTD ratio is therefore a direct lever for tuning the wall-clock time.