LATENT SUCCESSOR MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

An effective reinforcement learning (RL) agent must reason about both the immediate outcomes and long-term consequences of its actions. This challenge is especially pronounced when learning from pixel observations. Many prior RL methods learn visual representations for control by combining representation learning with reward prediction and value estimation. How can we acquire such representations without directly training them to predict rewards or values? In this paper, we propose *Latent Successor Models* (LSM), an approach that learns actionable visual representations through generative modeling of future outcomes at arbitrary horizons and behavioral cloning. This removes the need for direct reward and value prediction or pixel reconstruction when learning these representations. By sampling distant future outcomes under a policy, we construct targets for value estimation without predicting every intermediate state. Our analysis shows that directly predicting future outcomes yields more accurate value estimates at longer horizons. Experiments across a range of visual offline RL benchmarks (38 tasks) demonstrate substantial improvements over prior model-free and model-based approaches, with particularly large gains on challenging long-horizon tasks.

1 INTRODUCTION

An effective reinforcement learning (RL) agent must reason about both the immediate outcomes and long-term consequences of its actions. When learning from pixels, this challenge is particularly pronounced because image observations may contain information irrelevant to control (Islam et al., 2022; Sun et al., 2024). To address this problem, prior methods typically compress pixels into a compact representation space and learn policies using these representations. Such visual representations can be acquired through auxiliary losses (Hafner et al., 2020b; Laskin et al., 2020) or by combining representation learning with reward prediction and value estimation (Hansen et al., 2022; Zhang et al., 2021; Fujimoto et al., 2025). Building on these advances, we ask: *can we learn representations that support both long-horizon value estimation and control without directly training them to predict rewards or values?*

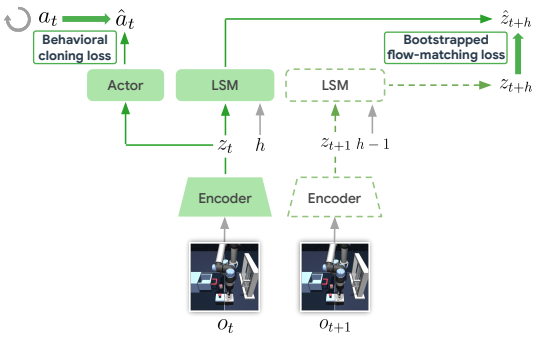


Figure 1: Can we learn visual representations that support RL without directly training them to predict rewards or values? We propose *Latent Successor Models* (LSM), an approach that learns visual representations for RL through generative modeling of future outcomes and behavioral cloning.

Predicting environmental transitions offers a natural starting point for answering this question. Prior methods typically learn a one-step transition model and roll out imagined trajectories step by step to estimate values (Maes et al., 2026; Rybkin et al., 2021; Zhu et al., 2025). However, prediction errors can accumulate over the horizon, resulting in inaccurate prediction of distant outcomes. Generative successor models (Janner et al., 2020; Thakoor et al., 2022; Farebrother et al., 2025; Zheng et al., 2026) provide a compelling alternative by directly modeling the distribution of distant future outcomes induced by a policy. Jointly learning these models with the visual representations provides a way to learn the encoder without direct reward and value supervision or pixel reconstruction. Our experiments demonstrate the practical significance of this design (Sec. 5.2): direct sampling in the

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.

learned latent space yields lower errors on both future prediction and value estimation when compared against alternative representation learning approaches.

In this paper, we propose *Latent Successor Models* (LSM), an approach that learns actionable visual representations through a generative successor model of future outcomes and behavioral cloning (BC) (Fig. 1). To train the successor model, we extend the temporal-difference flow-matching objective from prior work (Farebrother et al., 2025) and, instead, learn the successor model at *arbitrary horizons* by bootstrapping flow velocities from the next latent state. The resulting model predicts future outcomes more accurately than recursive one-step rollouts and yields more reliable value estimates at longer horizons (Sec. 5.1). We evaluate LSM on 38 visual offline RL tasks (Sec. 5.3) and show that improved long-horizon value estimation translates into better performance over prior model-free and model-based methods, with particularly large gains on challenging long-horizon tasks.

2 Related Work

Successor models for RL. Successor representations and successor features summarize long-term outcomes under a policy and have been widely used for various applications (Dayan, 1993; Barreto et al., 2017; Touati & Ollivier, 2021; Park et al., 2024; Pirotta et al., 2024; Eysenbach et al., 2022; Machado et al., 2020). Successor measures generalize this idea to distributions over future state visitation (Blier et al., 2021), which can be learned by a generative neural network (Janner et al., 2020; Thakoor et al., 2022; Zhu et al., 2024; Farebrother et al., 2025; Zheng et al., 2026). Such models support planning and value estimation by directly sampling future states (Farebrother et al., 2026; Chung et al., 2026). However, prior offline RL works with successor models predict the future observations in pixel space, which may capture information irrelevant for value estimation and control. In contrast, LSM learns a latent successor model, and uses samples from the successor model to construct long-horizon return targets for offline RL (Sec. 4).

Self-predictive representations. RL from pixels requires compact representations that retain information relevant for control while being predictable (Islam et al., 2022; Sun et al., 2024). In contrast to reconstruction-based methods (Wahlström et al., 2015; Zhang et al., 2019; Hafner et al., 2020b; 2025a), which may preserve visual details irrelevant to control, self-predictive representation learning instead utilizes contrastive learning or latent regularization objectives (van den Oord et al., 2018; Grill et al., 2020; Sobal et al., 2022; Assran et al., 2023). These approaches have been widely explored in RL, including future latent prediction (Schwarzer et al., 2021; Maes et al., 2026; Huang et al., 2026), or combined with task and policy objectives (Schrittwieser et al., 2020; Hafner et al., 2020b; Fujimoto et al., 2025; Zhang et al., 2021; Hansen et al., 2022; 2024). LSM builds on this family by learning visual representations for generative successor models via future prediction and behavioral cloning, without reward or value supervision of the encoder. LSM mitigates the compounding errors from recursive rollouts and improves long-horizon value estimation (Sec. 5.1 and Sec. 5.2).

Offline RL from pixels. Offline RL from pixel observations requires both representation learning and reliable value estimation from a fixed dataset (Levine et al., 2020; Lu et al., 2023). Model-free approaches (Kumar et al., 2022; Lu et al., 2023; Islam et al., 2022; Park et al., 2025c) learn value functions directly from visual representations, while model-based methods estimate values with imagined trajectories from a learned dynamics model (Maes et al., 2026; Rafailov et al., 2021; Lu et al., 2023; Wan et al., 2024; Hafner et al., 2025b). However, prior methods predominantly rely on short-horizon value estimation; model-free methods utilize one-step Bellman updates, and model-based methods imagine short trajectories (typically 5–10 steps (Wan et al., 2024; Rafailov et al., 2021; Lu et al., 2023)), which can lead to inaccurate value estimation (Park et al., 2025b). In contrast, LSM directly predicts future latent states at arbitrary horizons, enabling long-horizon value estimation.

3 Preliminaries

Problem setting. We consider an infinite-horizon *block* Markov decision process (Du et al., 2019) $(\mathcal{S}, \mathcal{A}, \mathcal{O}, p, q, r, \rho, \gamma)$, where $s \in \mathcal{S}$ denotes the unobservable state, $o \in \mathcal{O}$ denotes the observation, $a \in \mathcal{A}$ denotes the action, $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition distribution, $q : \mathcal{S} \to \mathcal{O}$ is the emission function that maps a state to its corresponding observation, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward

function, $\rho \in \Delta(\mathcal{S})$ is the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. Here, $\Delta(\cdot)$ denotes the set of all possible distributions over a space. We use t for RL time steps, h for prediction horizons, and τ for flow-matching noise levels. At each time step t, the agent receives an observation $o_t = q(s_t)$ and takes an action a_t sampled from a policy. Whenever possible, we omit the subscript t and use o and o' for current and next observations.

Value functions and successor representations. For a policy π, the value function and the Q function are defined as the discounted returns: $Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a\right]$, $V^\pi(s) = \mathbb{E}_\pi[Q^\pi(s, a)]$. We study offline RL, where the agent learns from a fixed dataset $D = \{(o, a, r, o')\}$ collected by a behavior policy, without further environment interaction. The goal is to learn a policy that maximizes the expected value function under the initial state distribution.

Successor measure (Dayan, 1993; Blier et al., 2021) provides a compact way to describe the value and the Q functions. This quantity summarizes the visitation of a future state s_f starting from (s, a) under the policy π. We can rewrite the Q function as an expectation over rewards under the successor measure (Thakoor et al., 2022):

$$\underbrace{p^\pi_\gamma(s_f \mid s, a)}_{\text{marginal successor measure}} = (1-\gamma) \sum_{h=1}^{\infty} \gamma^{h-1} \underbrace{p^\pi(s_f \mid s, a, h)}_{\text{horizon-conditioned successor measure}}, \tag{1}$$

$$Q^\pi(s, a) = r(s, a) + \frac{\gamma}{1-\gamma} \mathbb{E}_{\substack{p^\pi_\gamma(s_f \mid s, a), \\ \pi(a_f \mid s_f)}} [r(s_f, a_f)], \tag{2}$$

where $p^\pi(s_f \mid s, a, h)$ is the visitation of the future state s_f at horizon h. The horizon-conditioned successor measure satisfies a recursion:

$$p^\pi(s_f \mid s, a, h) = \begin{cases} p(s_f \mid s, a) & h = 1, \\ \mathbb{E}_{p(s' \mid s, a), \pi(a' \mid s')} \left[p^\pi(s_f \mid s', a', h-1)\right] & h = 2, 3, \cdots. \end{cases} \tag{3}$$

Since different policies induce different successor measures, our goal is to learn horizon-conditioned successor measures from pixels (Sec. 4.3) using *off-policy* objectives. We represent these distributions using flow-based generative models, or successor models, and review their learning objectives below.

Flow matching. Flow matching (Lipman et al., 2023; Albergo & Vanden-Eijnden, 2023; Liu et al., 2023) is a generative modeling framework that learns a velocity field $v : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$ to transport a noise distribution (e.g., standard Gaussian) to a target distribution $p(x)$. The corresponding flow map $\psi : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$ solves an ordinary differential equation (ODE)

$$\frac{\mathrm{d}}{\mathrm{d}\tau} \psi(x^0 \mid \tau) = v\left(\psi(x^0 \mid \tau) \mid \tau\right), \qquad \psi(x^0 \mid 0) = x^0. \tag{4}$$

A standard learning objective independently samples a noise $x^0 \sim \mathcal{N}(0, I_d)$, a target $x^1 \sim p(x)$, and a noise level $\tau \sim \textsc{Unif}([0, 1])$, then regresses onto the derivative of the linear interpolation $x^\tau = (1-\tau)x^0 + \tau x^1$:

$$\mathbb{E}\left[\left\|v(x^\tau \mid \tau) - (x^1 - x^0)\right\|_2^2\right]. \tag{5}$$

After training, flow samples are generated by integrating the ODE from $\tau = 0$ to $\tau = 1$.

TD-Flow. Prior work applies flow matching to learn successor measures. In particular, given a policy π, Farebrother et al. (2025) learn the marginal successor measure using the TD^2-CFM objective, which combines next state regression with bootstrapping the velocity at the next time step:

$$\begin{aligned} &(1-\gamma)\mathbb{E}_{s_f^\tau = (1-\tau)s_f^0 + \tau s'}\left[\left\|v(s_f^\tau \mid \tau, s, a) - (s' - s_f^0)\right\|_2^2\right] \\ &+ \gamma \mathbb{E}_{s_f^\tau \sim \bar{p}(s_f^\tau \mid \tau, s', a')}\left[\left\|v(s_f^\tau \mid \tau, s, a) - \bar{v}(s_f^\tau \mid \tau, s', a')\right\|_2^2\right], \end{aligned} \tag{6}$$

where (s, a, s') is a dataset transition, $a' \sim \pi(a' \mid s')$ is the next action, s_f^0 is Gaussian noise, and $s_f^\tau \sim \bar{p}(s_f^\tau \mid \tau, s', a')$ is a noisy future state that lies on the flow path of the target velocity $\bar{v}(s_f^\tau \mid \tau, s', a')$. Here, the target velocity $\bar{v}$ stabilizes bootstrapping and is updated using an exponential moving average of its online counterpart (Lillicrap et al., 2015).

Universal horizon models. However, learning the marginal successor measure loses the flexibility of predicting future states at arbitrary horizons. To overcome this issue, universal horizon models (Chung

et al., 2026) learn a successor measure that depends on the horizon by either regressing the next state or a clean future state starting from the next state-action pair at one shorter horizon:

$$\begin{aligned} &\mathbb{E}_{s_f^\tau=(1-\tau)s_f^0+\tau s'}\left[\|v(s_f^\tau \mid \tau, s, a, h) - (s' - s_f^0)\|_2^2\right], \quad h = 1, \\ &\mathbb{E}_{\substack{s_f^1=\bar{\psi}(s_f^0|1,s',a',h-1),\\ s_f^\tau=(1-\tau)s_f^0+\tau s_f^1}}\left[\|v(s_f^\tau \mid \tau, s, a, h) - (s_f^1 - s_f^0)\|_2^2\right], \quad h \geq 2, \end{aligned} \tag{7}$$

where $\bar{\psi}(s_f^0 \mid 1, s', a', h-1)$ is the clean future state obtained by integrating the target velocity with initial noise s_f^0. In Sec. 4.3, we will develop a flow-matching objective that is conditioned on the horizon, while bootstrapping the velocity field directly.

4 Latent Successor Models

The central goal of LSM is to learn visual representations that support long-horizon prediction and control without directly training the image encoder to predict rewards or values. To achieve this goal, our method jointly learns the representation, a flow-based generative successor model, and a behavioral cloning policy. The successor model provides supervision about future outcomes, while behavioral cloning provides supervision about actions.

After learning the successor model, we use it to generate future outcomes for value estimation. Specifically, we follow the Dyna framework of constructing targets for critic learning (Sutton, 1991). Instead of generating imagined trajectories step by step (Janner et al., 2019; Hafner et al., 2020b), our learned successor models are conditioned on the prediction horizon (Eq. 1), enabling direct sampling of future outcomes without predicting every intermediate state (Chung et al., 2026). This mitigates the compounding errors associated with recursive rollouts and enables more accurate long-horizon value estimation (Sec. 5.1). Below, we first outline the method, then explain how its learning objectives shape the representation and support critic learning.

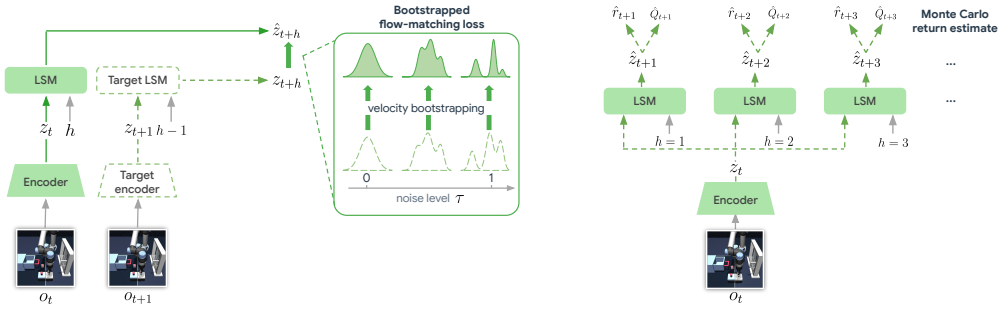
4.1 Overview

Achieving the aforementioned goals requires addressing three challenges:

- **C1.** Predicting future observations in pixel space requires modeling visual details that may be irrelevant to control (Islam et al., 2022; Sun et al., 2024). We therefore seek a compact latent space that preserves information useful for predicting future outcomes and selecting actions. The challenge is to learn this representation without pixel reconstruction or direct reward and value supervision, while avoiding degenerate solutions (Balestriero & LeCun, 2025; Grill et al., 2020; Chen & He, 2021).
- **C2.** Because different policies induce different successor models (Eq. 1), we need an *off-policy* objective that learns the successor model of a policy of interest from transitions collected by the behavioral policy. This objective must also provide a learning signal for the image encoder while fitting flow velocities across prediction horizons and noise levels, providing richer supervision (Chen et al., 2024).
- **C3.** Using the learned successor model, we want to aggregate predicted future outcomes across different horizons into an accurate return estimate without relying on autoregressive rollouts.

LSM addresses these challenges by learning successor measures in a compact latent space $\mathcal{Z} \subset \mathbb{R}^d$ (Fig. 2). An image encoder $f : \mathcal{O} \to \mathcal{Z}$ maps observations to latent states shared by the predictive and control components, while a policy $\pi : \mathcal{Z} \to \Delta(\mathcal{A})$ produces actions. A flow velocity field $v : \mathcal{Z} \times [0, 1] \times \mathcal{Z} \times \mathcal{A} \times \mathbb{N} \to \mathcal{Z}$ represents the successor measure induced by this policy at different horizons. A reward predictor $r : \mathcal{Z} \times \mathcal{A} \to \mathbb{R}$ and critic $Q : \mathcal{Z} \times \mathcal{A} \to \mathbb{R}$ estimate immediate rewards and Q-values from the latent.

We next explain how successor model learning and behavioral cloning train the representation, and how asymmetric targets stabilize this joint learning process (C1). We then derive the bootstrapped flow-matching objective for learning the encoder and successor model across horizons (C2). Finally, we discuss how direct samples from the learned successor model produce Monte Carlo return estimates for critic learning (C3).



(a) Bootstrapped flow matching (b) Monte Carlo value estimation

Figure 2: **Key components of LSM.** *(a)* From an offline dataset of transitions, the agent jointly learns visual representations and a latent successor model by bootstrapping flow velocities at all noise levels. Behavioral cloning is omitted for clarity. *(b)* Using the learned successor model, we directly sample future states at different horizons and construct a Monte Carlo return estimate for learning the critic, without predicting every intermediate state. See Alg. 1 for the pseudocode.

4.2 Learning representations without reward and value predictions

The representation learning objective is a central design choice in LSM (C1). Our approach builds on the latent prediction principle from prior work (Maes et al., 2026; Laskin et al., 2020; Bagatella et al., 2026; Fujimoto et al., 2025; Hafner et al., 2020b; Hansen et al., 2022), but does not rely on auxiliary objectives, reconstruction, or reward and value prediction. Instead, our successor model predicts future outcomes in the latent space, and its objective also determines which information the encoder should retain. We use consistency between future outcome predictions at different horizons to train the representation, together with behavioral cloning. Both objectives connect representation learning to prediction and control without requiring additional supervision for the encoder (see Sec. 5.2 for empirical comparisons to alternative representation learning methods).

However, jointly learning the representation and successor model may admit degenerate solutions (Balestriero & LeCun, 2025; Grill et al., 2020; Chen & He, 2021): if the same trainable encoder produces both the input and the target representations, it can make the prediction task trivial by mapping distinct observations to the same latent state. Following BYOL (Grill et al., 2020), we mitigate this issue using an asymmetric target encoder $\bar{f}$ that is updated gradually using the exponential moving averages of its online counterpart f. This provides stable learning targets at future time steps while allowing the target encoder to improve throughout training. In the next section, we discuss the bootstrapped flow matching objective for learning the image encoder and the successor model across different horizons, using the target image encoder.

4.3 Bootstrapping flow velocities at all noise levels

With the asymmetric target encoder in place, we next address how to learn the successor model from transitions collected by a different policy (C2). Our key idea is to use predictions from the next time step to supervise predictions at the current time step, following the recursion in Eq. 3. We implement this recursion by directly bootstrapping flow velocities at different noise levels, extending Eq. 6 to horizon-conditioned latent prediction (Fig. 2a). This enables more accurate future prediction than alternative representation learning objectives, while improving value estimation at longer horizons (see Sec. 5.1 for empirical studies).

To jointly learn the image encoder with our successor model for a policy π, we use the online encoder for the current observation and use the target encoder for the next observation. Specifically, given a transition $(o, a, o') \sim D$ sampled from the offline dataset, we use the online encoder to encode the current observation and use the target encoder to encode the next observation: $z = f(o), z' = \bar{f}(o')$. Here, the online latent z serves as the input to our successor model, while the target latent z' is used to construct the learning targets for the flow velocity. Our bootstrapped flow matching loss uses different targets for horizon $h = 1$ and horizons $h \geq 2$. For horizon $h = 1$, we use the target latent state z' to construct a linear interpolation path: $z^\tau = (1-\tau)z^0 + \tau z', z^0 \sim \mathcal{N}(0, I_d)$, and then apply the standard flow-matching loss (Eq. 5) with the regression targets $z' - z^0$. For horizons $h \geq 2$, we first sample the next action $a' \sim \pi(a' \mid z')$ and a *noisy* future state along the flow path at one shorter

horizon:

$$z^\tau \sim \bar{p}(z^\tau \mid \tau, z', a', h-1) \iff z^\tau = z^0 + \int_0^\tau \bar{v}(z^u \mid u, z', a', h-1)du, \quad z^0 \sim \mathcal{N}(0, I_d),$$

and then bootstrap the target velocity $\bar{v}(z^\tau \mid \tau, z', a', h-1)$ directly. Unlike the clean future state in Eq. 7, we obtain the noisy future state by partially integrating the flow ODE (Eq. 4) until noise level $\tau \sim \text{UNIF}([0, 1])$. Taken together, our flow-matching objective at prediction horizon h is

$$\mathcal{L}(f, v; h) = \begin{cases} \mathbb{E}_{z^\tau = (1-\tau)z^0 + \tau z'} \left[\|v(z^\tau \mid \tau, z, a, h) - (z' - z^0)\|_2^2 \right], & h = 1, \\ \mathbb{E}_{z^\tau \sim \bar{p}(z^\tau \mid \tau, z', a', h-1)} \left[\|v(z^\tau \mid \tau, z, a, h) - \bar{v}(z^\tau \mid \tau, z', a', h-1)\|_2^2 \right], & h \geq 2. \end{cases} \tag{8}$$

Because gradients of the encoder flow only through $z = f(o)$, but not through $z' = \bar{f}(o')$, we prevent the online encoder from reducing the flow matching objective by changing both the inputs and the predictive targets of the successor model. In Appendix A.1, we show that minimizing $\mathcal{L}(f, v; h)$ actually recovers the horizon-conditioned successor measure of the policy π. This loss extends the TD^2-CFM loss (Eq. 6) to learning successor models that depend on horizons. During training, we sample h from a discrete uniform distribution $\text{UNIF}(\{1, \ldots, H\})$ with maximum horizon H, resulting in the complete objective

$$\mathcal{L}(f, v) = \mathbb{E}_{h \sim \text{UNIF}(\{1, \cdots, H\})} \left[\mathcal{L}(f, v; h) \right]. \tag{9}$$

For critic learning, we instead sample horizons according to the weighting required by the value estimator. In particular, as we show in the next section, sampling horizons from a truncated geometric distribution allows us to construct an MC estimate of the λ-return to fit the Q-value.

4.4 Monte Carlo Value Estimation

Next, we address how to turn the predictions from the flow-based successor models into return estimates for critic learning (C3). Our idea is to sample future latent states directly and use their predicted rewards and values to estimate returns (Fig. 2b). This allows us to use long-horizon predictions while avoiding the compounding errors associated with repeated one-step model rollouts (see Sec. 5.1). The resulting estimates serve as targets for fitting the critic. Below, we describe how to construct these estimates using truncated λ-returns and defer the full derivation to Appendix A.2.

We first express the return estimate along a trajectory and then describe how to estimate its expectation by directly sampling the successor model. Given a trajectory sampled from $\pi(a \mid z)$, a reward predictor $r(z, a)$, and a critic $Q(z, a)$, we consider the truncated λ-return (Sutton et al., 1998; Van Seijen et al., 2016; Hafner et al., 2020b) with maximum horizons H:

$$G^{\lambda, H}(z, a) = r(z, a) + \gamma \sum_{h=1}^{H} (\lambda\gamma)^{h-1} \left[(1 - \alpha_h) r(z_h, a_h) + \alpha_h Q(z_h, a_h) \right], \tag{10}$$

where $\alpha_h = 1 - \lambda$ for $h = 1, \cdots, H-1$ and $\alpha_H = 1$. By linearity of expectation, we can estimate the expected truncated λ-return using latent states sampled directly from the successor model at selected horizons. Sampling horizons according to the geometric weights and applying normalization yields a Monte Carlo estimate $\widehat{G}^{\lambda, H}(z, a)$ (see Appendix A.2 for details) without generating every intermediate state. We use this estimate as the target for critic learning and fit the reward predictor to dataset rewards:

$$\mathcal{L}(Q) = \mathbb{E}\left[\left(Q(z, a) - \widehat{G}^{\lambda, H}(z, a) \right)^2 \right], \quad \mathcal{L}(r) = \mathbb{E}\left[(r(z, a) - r)^2 \right], \tag{11}$$

where $z = f(o)$ is the latent state from the online image encoder, and the expectations are over dataset transitions and Monte Carlo future state predictions. Importantly, we stop gradients from propagating into the image encoder by excluding it from both losses. We explicitly learn a critic because our λ-return estimate requires bootstrapping at the maximum horizon H. In the next section, we use this critic to instantiate the latent policy $\pi(a \mid z)$ by performing rejection sampling.

4.5 The Complete Algorithm

We instantiate the policy $\pi(a \mid z)$ by selecting the action among N candidates sampled from a flow-based behavioral cloning (BC) policy π_β (Chen et al., 2023):

$$\pi(a \mid z) \overset{d}{=} \underset{a_1, \cdots, a_N :\, a_i \sim \pi_\beta(a \mid z)}{\arg\max} Q(z, a_i), \tag{12}$$

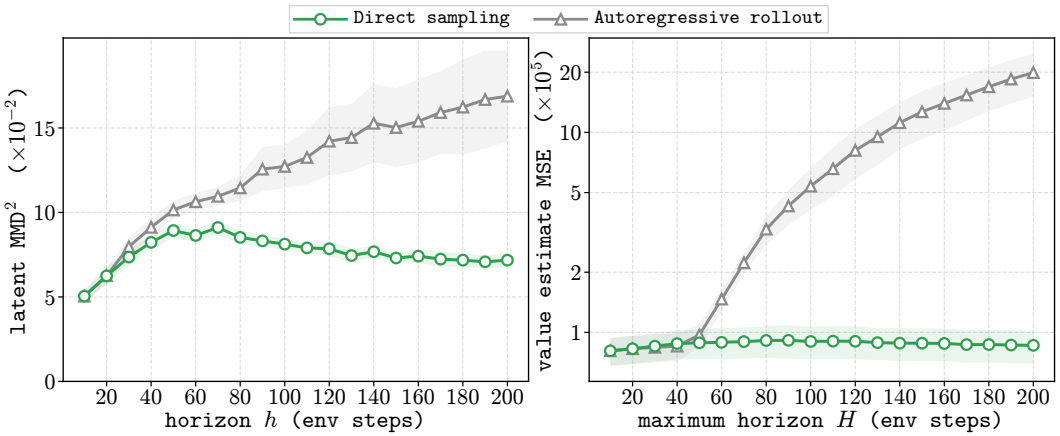


Figure 3: **LSM improves long-horizon future prediction and value estimation.** On `visual-cube-double` *(Left)*, we compare LSM using autoregressive rollout or direct sampling. We measure future prediction error using the squared maximum mean discrepancy (↓) against ground-truth distributions at different horizons *(Right)*, and measure value estimation error using mean squared error (↓) against the ground-truth value with different maximum horizon H in the λ-return estimate. In both comparisons, direct sampling from LSM achieves the lowest errors. See Sec. 5.1 for details.

where $\stackrel{d}{=}$ denotes equality in distribution. We learn the BC policy along with the image encoder using the standard flow-matching loss (Eq. 5) and denote the loss as $\mathcal{L}(f, \pi_\beta)$. We optimize the image encoder jointly with the successor model and flow BC policy while stopping gradients from the reward and critic losses. Alg. 1 summarizes the training procedure of LSM.

5 Experiments

We conduct experiments to study how well LSM can perform long-horizon future prediction and value estimation, and how good LSM is at solving a range of challenging visual offline RL tasks. Concretely, we study the following research questions:

- **Q1.** How accurately can LSM perform long-horizon future prediction and value estimation compared to autoregressive rollouts?
- **Q2.** Does LSM learn good visual representations for MC value estimation compared to other representation learning methods?
- **Q3.** How does LSM compare to prior offline model-free and model-based methods on visual RL benchmarks?

Experimental setup. We describe the experimental benchmarks and defer details to Appendix E. We evaluate LSM as a visual offline RL method on 38 tasks across three benchmarks (Fig. 4). On Visual OGBench (Park et al., 2025a;c), we consider all 25 single-task variants (5 per domain) of `visual-cube-{single,double}`, `visual-scene`, and `visual-puzzle-{3x3,4x4}`. On MM-Bench2 (Hansen & Wang, 2026), we use the 10 tasks held out from its world model pretraining set ("unseen" tasks), and train on each task's own offline data. On Visual Robomimic (Mandlekar et al., 2021), we use the multi-human (MH) datasets of three tasks, `visual-lift`, `visual-can`, and `visual-square`, from the agent-view and wrist cameras. Unless otherwise mentioned, we report the mean and standard deviation across 4 random seeds.

5.1 LSM improves long-horizon future prediction and value estimation

We first study whether LSM supports accurate long-horizon future prediction and value estimation. On `visual-cube-double-task3`, we compare latent successor models under two prediction strategies: direct sampling at a specified horizon (`direct sampling`) or step-by-step prediction (`autoregressive rollout`). We collect 100 real trajectories under this policy as ground truth. Future-prediction error is measured using squared maximum mean discrepancy (MMD^2) (Gretton et al., 2012) between predicted and real future-state distributions after h steps, while value-estimation error is measured by the MSE between the λ-return estimates truncated at horizon H and the ground-truth value. Full details are provided in Appendix E.4.



Figure 4: **Benchmarks for evaluation.** Example tasks from *(Left)* Visual OGBench (25 tasks), *(Center)* MMBench2 (10 tasks), and *(Right)* Visual Robomimic (3 tasks).

Results in Fig. 3 *(Left)* show that LSM achieves substantially lower prediction errors than latent-space autoregressive rollout at long horizons. At $h = 200$, direct sampling achieves an MMD^2 of approximately 0.07, compared with 0.17 for autoregressive rollout, corresponding to a reduction of about 60%. While the error of autoregressive rollout increases with the prediction horizon, the error of direct sampling remains relatively stable beyond $h = 50$. We next examine value estimation by comparing the truncated λ-return estimates against ground-truth value using different maximum horizons H. Results in Fig. 3 *(Right)* show that direct sampling maintains low value estimation errors throughout the trajectory, whereas autoregressive rollout exhibits increasingly large errors at later time steps. Taken together, these results suggest that directly sampling future outcomes in a learned latent space supports more accurate long-horizon prediction and value estimation than autoregressive rollout.

5.2 LSM LEARNS GOOD VISUAL REPRESENTATIONS FOR MONTE CARLO VALUE ESTIMATION

We next examine whether the visual representations learned by LSM support more accurate Monte Carlo value estimation (Sec. 4.4) than those learned with alternative objectives. Specifically, we compare four alternative objectives for training the image encoder: reconstruction (Wahlström et al., 2015), contrastive learning (`SimCLR` (Chen et al., 2020)), one-step transition prediction (`One-step BYOL`) (Grill et al., 2020), and bisimulation (Zhang et al., 2021). One-step BYOL (Grill et al., 2020) learns to predict the latent state at the next time step encoded by a target image encoder. Bisimulation is specifically designed for RL and uses reward supervision to learn representations, whereas LSM learns its representations without reward supervision. For each alternative objective, we also jointly learn latent successor models and policies on top of the learned representation, without propagating their gradients into the image encoder. We conduct these ablations using the same task and dataset as in Sec. 5.1. For evaluation, we compute the λ-return estimates and measure their MSE against the ground-truth value. See Appendix E.5 for the full experimental details.

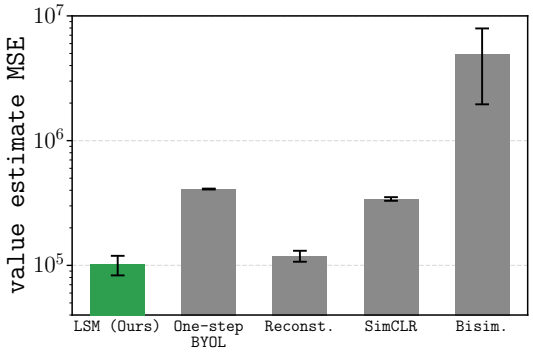


Figure 5: We compare the value estimate MSEs of latent successor models based on different representation learning objectives: LSM, reconstruction, SimCLR, One-step BYOL, and bisimulation. LSM achieves lower errors than other representation learning methods.

Results in Fig. 5 show that LSM achieves lower value estimation error than all four alternative objectives. Reconstruction, SimCLR, and one-step BYOL all perform worse, while bisimulation exhibits substantially larger errors due to representation collapse. In the sparse-reward `visual-cube-double-task3` setting, independently sampled transitions often share the same reward, allowing the encoder to map observations to nearly constant latents. Overall, these results suggest that combining multi-step future prediction with behavioral cloning yields representations that better support Monte Carlo value estimation.

5.3 LSM OUTPERFORMS PRIOR OFFLINE RL METHODS ON VISUAL BENCHMARKS

Having shown that LSM improves future prediction and value estimation, we next evaluate its performance on 38 visual offline RL tasks from Visual OGBench, MMBench2, and Visual Robomimic. We compare against representative model-free methods (BC, IQL, ReBRAC, IFQL, FQL, QC, and QC-IFQL) and model-based methods (LOMPO, OffDV2, PTD-Flow, and LUHM); details are provided in Appendix E.2.

Table 1: **Offline evaluation on Visual OGBench and MMBench2.** We report success rates $(0-100)$ on OGBench tasks and normalized scores $(0-100)$ on MMBench2 tasks. LSM achieves the highest mean performance in 4 of 5 OGBench domains and 2 of 4 MMBench2 domains. We highlight mean performance at or above 95% of the best in each row in green. See Table 3 and 4 for full results.

	Model-free					Model-based				
						One-step model		Successor model		
Domain	IQL	ReBRAC	IFQL	FQL	QC	LOMPO	OffDV2	PTD-Flow	LUHM	LSM
OGBench (25 tasks)										
visual-cube-single (5 tasks)	65 ±4	74 ±1	36 ±5	62 ±4	78 ±2	0 ±0	0 ±0	9 ±5	61 ±5	82 ±1
visual-cube-double (5 tasks)	13 ±1	4 ±3	2 ±1	5 ±3	3 ±1	0 ±0	0 ±0	0 ±0	13 ±1	54 ±5
visual-scene (5 tasks)	18 ±8	33 ±4	21 ±2	41 ±2	57 ±2	2 ±1	4 ±2	4 ±3	22 ±1	88 ±6
visual-puzzle-3x3 (5 tasks)	2 ±3	19 ±1	21 ±1	22 ±1	20 ±0	3 ±2	2 ±1	0 ±0	1 ±1	30 ±11
visual-puzzle-4x4 (5 tasks)	0 ±0	13 ±1	4 ±5	10 ±1	20 ±2	0 ±0	1 ±0	0 ±0	0 ±0	0 ±0
MMBench2 (10 tasks)										
dmcontrol (2 tasks)	19 ±2	39 ±2	17 ±2	53 ±1	22 ±2	14 ±4	44 ±12	14 ±1	24 ±3	36 ±3
maniskill (1 task)	5 ±2	8 ±3	11 ±4	3 ±2	3 ±2	23 ±3	0 ±0	5 ±1	6 ±0	4 ±3
og-point (2 tasks)	7 ±4	15 ±2	10 ±3	30 ±4	6 ±2	0 ±0	1 ±0	7 ±8	0 ±1	37 ±6
pygame (5 tasks)	41 ±2	50 ±5	47 ±1	58 ±1	50 ±1	31 ±3	43 ±1	35 ±5	58 ±1	64 ±1

Table 2: **Offline evaluation on Visual Robomimic.** We report success rates for each task. LSM matches the best mean success rates on `visual-lift` and `visual-can` and achieves the highest mean success rate on `visual-square`. We highlight mean success rates at or above 95% of the best in each row in green.

	Model-free			Model-based				
				One-step model		Successor model		
Task	BC	QC	QC-IFQL	LOMPO	OffDV2	PTD-Flow	LUHM	LSM
visual-lift	100±0	100±0	100±0	27 ±42	51 ±19	100 ±0	99 ±1	100±0
visual-can	97 ±3	97 ±3	92 ±3	0 ±0	0 ±0	89 ±1	86 ±6	97 ±1
visual-square	68 ±21	55 ±13	55 ±10	0 ±0	0 ±0	52 ±12	44 ±1	72 ±8

We report results on visual OGBench and MMBench2 in Table 1, and on Robomimic in Table 2. On Visual OGBench, LSM performs best in four of five domains, with particularly large gains on long-horizon manipulation tasks. However, on `visual-puzzle-4x4`, all model-based methods including LSM fail to make progress. We conjecture that the lack of transitions reaching the maximum-reward state in the offline dataset makes accurate value estimation difficult, especially for model-based methods. On MMBench2, LSM achieves the highest mean scores on `og-point` and `pygame`, while trailing the strongest baselines on `dmcontrol` and `maniskill`. On Robomimic, LSM matches the best performance on `lift` and `can` and achieves the highest mean success rate on most challenging `square`. These results demonstrate the effectiveness of LSM across a range of visual offline RL benchmarks, with strong performance on robotic manipulation tasks.

Additional experiments. In Appendix D.1, we include qualitative visualizations showing that LSM preserves visual structure in future latent predictions, even though the encoder is trained without pixel reconstruction. Our experiments in Appendix D.2 ablate three different sources of gradients for the image encoder, showing that including all of them is important for LSM. In Appendix D.3, our architecture ablation shows that using a residual block with adaptive layer normalization for the successor model in LSM yields substantially better policy performance than an MLP backbone.

6 Conclusion

We present LSM, an offline RL approach that learns visual representations through future prediction and behavioral cloning, without pixel reconstruction or direct reward and value supervision. Theoretically, we show that the successor model learned with our bootstrapped flow-matching objective converges to the horizon-conditioned successor measure in latent space. Our method enables direct sampling of future latent states at specified horizons, improving long-horizon value estimation. Experiments on 38 visual offline RL tasks demonstrate strong performance against prior model-free and model-based methods.

Limitations. Our convergence analysis assumes a fixed policy and a fixed encoder that induces a Markovian latent space. These results do not directly establish convergence when the encoder, successor model, and policy are learned jointly. Our formulation also assumes that individual observations identify the underlying state; extending it to partially observable environments remains a potential future direction.

AI USE STATEMENT

In this work, we used generative AI tools for polishing the writing and implementing the methods. We have not used generative AI tools for designing or providing feedback on research methodology or experiments, proposing or refining hypotheses, assisting with translation. Supporting qualitative and thematic data analysis and interpreting the results, generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, cleaning and reformatting datasets do not apply to this work. The authors have manually reviewed the paper to prevent plagiarism, and read the LLM-generated code and verified and tested its correctness. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We include implementation details needed for reproduction, such as hyperparameters (including the baselines), datasets, and evaluation protocols in Appendix E. Clear explanations and the proofs of the theoretical results are provided in Appendix A.1.

REFERENCES

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *International Conference on Learning Representations (ICLR)*, 2023.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629. IEEE, 2023.

Marco Bagatella, Matteo Pirotta, Ahmed Touati, Alessandro Lazaric, and Andrea Tirinzoni. Td-jepa: Latent-predictive representations for zero-shot reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 36211–36244, 2026.

Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics, 2025. URL https://arxiv.org/abs/2511.08544.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado Van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. *arXiv preprint arXiv:2101.07123*, 2021.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=42zs3qa2kpy.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*, 2020.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *2021 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 15745–15753. IEEE, 2021.

Hojun Chung, Junseo Lee, and Songhwai Oh. Offline reinforcement learning with universal horizon models. *arXiv preprint arXiv:2605.15603*, 2026.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624, 1993.

Perry Dong, Chongyi Zheng, Chelsea Finn, Dorsa Sadigh, and Benjamin Eysenbach. Value flows. In *International Conference on Learning Representations*, volume 2026, pp. 128614–128646, 2026.

Simon Du, Akshay Krishnamurthy, Nan Jiang, Alekh Agarwal, Miroslav Dudik, and John Langford. Provably efficient rl with rich observations via latent state decoding. In *International Conference on Machine Learning*, pp. 1665–1674. PMLR, 2019.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

Jesse Farebrother, Matteo Pirotta, Andrea Tirinzoni, Remi Munos, Alessandro Lazaric, and Ahmed Touati. Temporal difference flows. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=j6H7c3aQyb.

Jesse Farebrother, Matteo Pirotta, Andrea Tirinzoni, Marc G Bellemare, Alessandro Lazaric, and Ahmed Touati. Compositional planning with jumpy world models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=cwztmjMQ77.

Scott Fujimoto, Pierluca D'Oro, Amy Zhang, Yuandong Tian, and Michael Rabbat. Towards general-purpose model-free reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=R1hIXdST22.

Izrail Moiseevitch Gelfand, Richard A Silverman, et al. *Calculus of variations*. Courier Corporation, 2000.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The journal of machine learning research*, 13:723–773, 2012.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020a.

Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations (ICLR)*, 2020b.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025a.

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025b.

Nicklas Hansen and Xiaolong Wang. Hallucination in world models is predictable and preventable, 2026.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning (ICML)*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *ArXiv*, abs/2304.10573, 2023.

Hsiang-Wei Huang, Jianxu Shangguan, Junbin Lu, and Jenq-Neng Hwang. Leflow: Generative latent flow planning for world models. *arXiv preprint arXiv:2608.24855*, 2026.

Riashat Islam, Manan Tomar, Alex Lamb, Yonathan Efroni, Hongyu Zang, Aniket Didolkar, Dipendra Misra, Xin Li, Harm Van Seijen, Remi Tachet des Combes, et al. Agent-controller representations: Principled offline rl with rich exogenous information. *arXiv preprint arXiv:2211.00164*, 2022.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Michael Janner, Igor Mordatch, and Sergey Levine. Gamma-models: Generative temporal difference learning for infinite-horizon prediction. *Advances in neural information processing systems*, 33: 1724–1735, 2020.

Vadim Kaplunovsky. Gradient, divergence, curl and related formulae. https://web2.ph.utexas.edu/~vadim/Classes/2016s/diffop.pdf, 2016. URL https://web2.ph.utexas.edu/~vadim/Classes/2016s/diffop.pdf.

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations (ICLR)*, 2021.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Aviral Kumar, Anikait Singh, Stephen Tian, Chelsea Finn, and Sergey Levine. A workflow for offline model-free robotic reinforcement learning. In Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 417–428. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/kumar22a.html.

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.

Sergey Levine, Aviral Kumar, G. Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv*, abs/2005.01643, 2020.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. *AI&M*, 1(2):3, 2006.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. *arXiv preprint arXiv:2507.07969*, 2025.

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. *arXiv preprint arXiv:1509.02971*, 2015.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023.

Cong Lu, Philip J Ball, Tim GJ Rudner, Jack Parker-Holder, Michael A Osborne, and Yee Whye Teh. Challenges and opportunities in offline reinforcement learning from visual observations. *Transactions on Machine Learning Research (TMLR)*, 2023.

Marlos C Machado, Marc G Bellemare, and Michael Bowling. Count-based exploration with the successor representation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 5125–5133, 2020.

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

Ajay Mandlekar, Danfei Xu, J. Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Mart'in-Mart'in. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

Tianwei Ni, Benjamin Eysenbach, Erfan Seyedsalehi, Michel Ma, Clement Gehring, Aditya Mahajan, and Pierre-Luc Bacon. Bridging state and history representations: Understanding self-predictive rl. In *International Conference on Learning Representations*, volume 2024, pp. 23555–23569, 2024.

Kwanyoung Park and Youngwoon Lee. Model-based offline reinforcement learning with lower expectile q-learning. *International Conference on Learning Representations (ICLR)*, 2025.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=LhNsSaAKub.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations (ICLR)*, 2025a.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable. *arXiv*, abs/2506.04168, 2025b.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *International Conference on Machine Learning (ICML)*, 2025c.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Matteo Pirotta, Andrea Tirinzoni, Ahmed Touati, Alessandro Lazaric, and Yann Ollivier. Fast imitation via behavior foundation models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=qnWtw3l0jb.

Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *Learning for dynamics and control*, pp. 1154–1168. PMLR, 2021.

Oleh Rybkin, Chuning Zhu, Anusha Nagabandi, Kostas Daniilidis, Igor Mordatch, and Sergey Levine. Model-based reinforcement learning via latent-space collocation. In *International Conference on Machine Learning*, pp. 9190–9201. PMLR, 2021.

Julian Schrittwieser, Ioannis Antonoglou, T. Hubert, K. Simonyan, L. Sifre, Simon Schmitt, A. Guez, Edward Lockhart, Demis Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering Atari, Go, Chess and Shogi by planning with a learned model. *Nature*, 588 7839:604–609, 2020.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021.

Vlad Sobal, Jyothir SV, Siddhartha Jalagam, Nicolas Carion, Kyunghyun Cho, and Yann LeCun. Joint embedding predictive architectures focus on slow features. *arXiv preprint arXiv:2211.10831*, 2022.

Ruixiang Sun, Hongyu Zang, Xin Li, and Riashat Islam. Learning latent dynamic robust representations for world models. *arXiv preprint arXiv:2405.06263*, 2024.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3 (1):9–44, 1988.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Shantanu Thakoor, Mark Rowland, Diana Borsa, Will Dabney, Rémi Munos, and André Barreto. Generalised policy improvement with geometric policy composition. In *International Conference on Machine Learning*, pp. 21272–21307. PMLR, 2022.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. *Advances in Neural Information Processing Systems*, 34:13–23, 2021.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

Harm Van Seijen, A Rupam Mahmood, Patrick M Pilarski, Marlos C Machado, and Richard S Sutton. True online temporal-difference learning. *Journal of Machine Learning Research*, 17(145):1–40, 2016.

Niklas Wahlström, Thomas B Schön, and Marc Peter Deisenroth. From pixels to torques: Policy learning with deep dynamical models. *arXiv preprint arXiv:1502.02251*, 2015.

Shenghua Wan, Ziyuan Chen, Le Gan, Shuai Feng, and De-Chuan Zhan. Semopo: learning high-quality model and policy from low-quality offline visual datasets. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 49867–49889, 2024.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=-2FCwDKRREu.

Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew Johnson, and Sergey Levine. Solar: Deep structured representations for model-based reinforcement learning. In *International conference on machine learning*, pp. 7444–7453. PMLR, 2019.

Chongyi Zheng, Seohong Park, Sergey Levine, and Benjamin Eysenbach. Intention-conditioned flow occupancy models. In *International Conference on Learning Representations*, volume 2026, pp. 116781–116817, 2026.

Chuning Zhu, Xinqi Wang, Tyler Han, Simon S Du, and Abhishek Gupta. Distributional successor features enable zero-shot policy optimization. *Advances in Neural Information Processing Systems*, 37:124612–124639, 2024.

Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv preprint arXiv:2504.02792*, 2025.

A Proofs and derivations

A.1 Convergence of the bootstrapped flow-matching objective

We show that minimizing the bootstrapped flow-matching objective in Eq. 8 results in a flow-based successor model that converges to the horizon-conditioned successor measure (Eq. 1) under the assumptions stated below. Specifically, we consider a fixed latent policy π and a fixed image encoder f, and iteratively optimize the velocity field of the successor model v_k using the target velocity field $\bar{v}$. Here, $k = 0, 1, 2, \cdots$ indexes the iterations, with $k = 0$ denoting random initialization, and the target velocity field $\bar{v}$ is an exponential moving average over all historical velocity fields up to iteration $k - 1$. Below, we first state the assumption for the image encoder and then show the convergence of the bootstrapped flow-matching objective by induction. The same argument applies directly in the state space $\mathcal{S}$.

Assumption: the encoder f induces a Markovian latent space. We assume that the fixed encoder f preserves Markovian transitions in the latent space: for any observations o such that $z = f(o)$, any action $a \in \mathcal{A}$, and any next latent state $z' \in \mathcal{Z}$, there exists a latent transition distribution $p_{\mathcal{Z}} : \mathcal{Z} \times \mathcal{A} \to \Delta(\mathcal{Z})$ such that

$$p(z' \mid o, a) = p_{\mathcal{Z}}(z' \mid f(o), a),$$

which is the transition-preserving property of a model-irrelevance abstraction (Li et al., 2006; Ni et al., 2024). This assumption suggests that the latent transition does not depend on the observations given the image encoder f and the action a, i.e., the latent space is Markovian. We note that the unobservable state space $\mathcal{S}$ also satisfies this property under the block MDP assumption (Sec. 3). Under this assumption, the horizon-conditioned successor measure in latent space satisfies the following recursion:

$$p^{\pi}_{\mathcal{Z}}(z_f \mid z, a, h) = \begin{cases} p_{\mathcal{Z}}(z_f \mid z, a) & h = 1, \\ \mathbb{E}_{p_{\mathcal{Z}}(z' \mid z, a),\, \pi(a' \mid z')} \left[p^{\pi}_{\mathcal{Z}}(z_f \mid z', a', h-1) \right] & h = 2, 3, \cdots. \end{cases} \tag{13}$$

In particular, $p^{\pi}_{\mathcal{Z}}(z_f \mid z, a, h)$ is uniquely determined by this recursion at every finite horizon. We next use this recursion to prove the convergence of the bootstrapped flow-matching objective.

Fitted iteration. Let $v(z^\tau \mid \tau, z, a, h)$ and $p(z^\tau \mid \tau, z, a, h)$ denote the velocity field and the probability density path of the current successor model. Let $\bar{v}(z^\tau \mid \tau, z, a, h)$ and $\bar{p}(z^\tau \mid \tau, z, a, h)$ denote the velocity field and the probability density path of the target successor model. Since the velocity field v is trained by minimizing a standard conditional flow-matching loss (Eq. 5) to fit the latent transition $p_{\mathcal{Z}}(z' \mid z, a)$ at horizon $h = 1$, we can show that the current successor model at horizon $h = 2, 3, \cdots$ converges to the horizon-conditioned successor measure in latent space whenever the successor model converges at horizon $h = 1$ first.

Proposition 1. *Given a policy π and an encoder f that induces a Markovian latent space, as the number of iterations k tends to infinity, the successor model v and p that optimize the bootstrapped flow-matching objective (Eq. 8) converge to the horizon-conditioned successor measure $p^{\pi}_{\mathcal{Z}}(z_f \mid z, a, h)$ at horizon $h = 1, 2, 3, \cdots$.*

Proof. We show the convergence of the successor model by first proving the desired results for horizon $h = 1$ and then performing induction for horizons $h = 2, 3, \cdots$.

One-step prediction. For horizon $h = 1$, the objective reduces to a standard conditional flow-matching objective that fits samples from the latent transition $z' \sim p_{\mathcal{Z}}(z' \mid z, a)$:

$$\min_v \mathbb{E}_{\substack{(z,a)\sim D,\, p_{\mathcal{Z}}(z'|z,a),\\ \tau\sim \text{UNIF}([0,1]),\\ z^0\sim\mathcal{N}(0,I_d),\, z^\tau=\tau z'+(1-\tau)z^0}} \left[\| v(z^\tau \mid \tau, z, a, 1) - (z' - z^0) \|_2^2 \right].$$

By construction (Lipman et al., 2023), the flow generated by the minimizer $v^\star$ transports the Gaussian distribution to the target distribution, i.e., the latent transition $p_{\mathcal{Z}}(z' \mid z, a)$, using the linear interpolation path $p^\star$:

$$v^\star(z^\tau \mid \tau, z, a, 1) = \mathbb{E}_{p_{\mathcal{Z}}(z'|z,a),\, z^0\sim\mathcal{N}(0,I_d)} \left[z' - z^0 \mid z^\tau, \tau \right]$$
$$p^\star(z^\tau \mid \tau, z, a, 1) = \mathbb{E}_{p_{\mathcal{Z}}(z'|z,a)} \left[p_{\mathcal{N}(\tau z', (1-\tau)^2 I_d)}(z^\tau) \right],$$

where $p_{\mathcal{N}(\tau z', (1-\tau)^2 I_d)}(\cdot)$ is the probability density of a Gaussian distribution with mean $\tau z'$ and covariance $(1-\tau)^2 I_d$. Plugging in the noise level $\tau = 1$ into $p^\star$, the Gaussian reduces to a Dirac delta measure at the next latent state z', recovering the latent transition distribution:

$$p^\star(z^1 \mid 1, z, a, 1) = \mathbb{E}_{p_{\mathcal{Z}}(z'|z,a)} \left[\delta_{z^1}(z') \right] = p_{\mathcal{Z}}(z^1 \mid z, a).$$

Bootstrapped prediction. For horizon $h \geq 2$, we can use the optimal probability density path for one-step prediction to perform induction. Specifically, our induction hypothesis is that the target velocity field $\bar{v}$ and its probability density path $\bar{p}$ are optimal at a shorter horizon $h - 1$[1] and they satisfy the continuity equation (Lipman et al., 2023):

$$\frac{\partial}{\partial \tau} \bar{p}(z^\tau \mid \tau, z, a, h) = -\text{div} \left(\bar{p}(z^\tau \mid \tau, z, a, h) \bar{v}(z^\tau \mid \tau, z, a, h) \right), \tag{14}$$

[1] Because the target velocity field is an exponential moving average of historical velocity fields, when $k \to \infty$, the target velocity field will also converge to $v^\star$ and $p^\star$ at horizon $h = 1$.

where $\mathrm{div}(\cdot)$ denotes the divergence operator with respect to the noisy latent state z^τ. We will first find the minimizer $v^\star$ and its corresponding probability density path $p^\star$ at horizon h and then invoke the induction hypothesis.

The minimizer $v^\star$ of the bootstrapped flow-matching loss can be computed by setting the functional derivative (Gelfand et al., 2000) $\delta/\delta v$ to zero:

$$\min_v \mathbb{E}_{\substack{(z,a)\sim D,\, p_{\mathcal{Z}}(z'|z,a) \\ \tau\sim\textsc{Unif}([0,1]), \\ \pi(a'|z'),\, \bar{p}(z^\tau|\tau,z',a',h-1)}} \left[\|v(z^\tau \mid \tau, z, a, h) - \bar{v}(z^\tau \mid \tau, z', a', h-1)\|_2^2\right],$$

$$\frac{\delta}{\delta v} = 2\mathbb{E}_{p_{\mathcal{Z}}(z'|z,a),\, \pi(a'|z')}\left[\bar{p}(z^\tau \mid \tau, z', a', h-1)(v(z^\tau \mid \tau, z, a, h) - \bar{v}(z^\tau \mid \tau, z', a', h-1))\right] = 0$$

$$\implies v^\star(z^\tau \mid \tau, z, a, h) = \frac{\mathbb{E}_{p_{\mathcal{Z}}(z'|z,a),\, \pi(a'|z')}\left[\bar{p}(z^\tau \mid \tau, z', a', h-1)\bar{v}(z^\tau \mid \tau, z', a', h-1)\right]}{\mathbb{E}_{p_{\mathcal{Z}}(z'|z,a),\, \pi(a'|z')}\left[\bar{p}(z^\tau \mid \tau, z', a', h-1)\right]}.$$

Next, we find the probability density path $p^\star(z^\tau \mid \tau, z, a, h)$ of $v^\star(z^\tau \mid \tau, z, a, h)$ by verifying the continuity equation, as done in Farebrother et al. (2025) and Dong et al. (2026). Specifically, we show that

$$p^\star(z^\tau \mid \tau, z, a, h) \triangleq \mathbb{E}_{p_{\mathcal{Z}}(z'|z,a),\, \pi(a'|z')}\left[\bar{p}(z^\tau \mid \tau, z', a', h-1)\right] \tag{15}$$

is the probability density path that corresponds to $v^\star$ by direct computation:

$$\begin{aligned}
\frac{\partial}{\partial\tau}p^\star(z^\tau \mid \tau, z, a, h) &\overset{(a)}{=} \mathbb{E}_{p_{\mathcal{Z}}(z'|z,a),\, \pi(a'|z')}\left[\frac{\partial}{\partial\tau}\bar{p}(z^\tau \mid \tau, z', a', h-1)\right] \\
&\overset{(b)}{=} -\mathbb{E}_{p_{\mathcal{Z}}(z'|z,a),\, \pi(a'|z')}\left[\mathrm{div}\left(\bar{p}(z^\tau \mid \tau, z', a', h-1)\bar{v}(z^\tau \mid \tau, z', a', h-1)\right)\right] \\
&\overset{(c)}{=} -\mathrm{div}\left(\mathbb{E}_{p_{\mathcal{Z}}(z'|z,a),\, \pi(a'|z')}\left[\bar{p}(z^\tau \mid \tau, z', a', h-1)\bar{v}(z^\tau \mid \tau, z', a', h-1)\right]\right) \\
&\overset{(d)}{=} -\mathrm{div}\left(p^\star(z^\tau \mid \tau, z, a, h)v^\star(z^\tau \mid \tau, z, a, h)\right),
\end{aligned}$$

where, in *(a)*, we swap the partial differentiation with the expectation because the sampling distribution is independent of the noise level τ, in *(b)*, we apply the continuity equation for $\bar{p}$ (Eq. 14); and in *(c)*, we swap the divergence $\mathrm{div}(\cdot)$ with the expectation since the divergence is a linear operator (Kaplunovsky, 2016), and, in *(d)*, we plug in the definition of the optimal velocity field $v^\star$ and the optimal probability density path $p^\star$. Moreover, we have $p^\star(z^0 \mid 0, z, a, h) = \mathcal{N}(0, I_d)$, since every target path starts from the same Gaussian distribution. Thus, the velocity field $v^\star$ generates the probability density path $p^\star$.

Finally, we identify the endpoint distribution at horizon h. Specifically, since the target probability density path $\bar{p}$ is optimal, at noise level $\tau = 1$, we have

$$\bar{p}(z^1 \mid 1, z, a, h-1) = p^\pi_{\mathcal{Z}}(z^1 \mid z, a, h-1).$$

Substituting this identity into Eq. 15 above shows that $p^\star$ satisfies the same recursion at horizon h as the horizon-conditioned successor measure (Eq. 13):

$$p^\star(z^1 \mid 1, z, a, h) = \mathbb{E}_{p_{\mathcal{Z}}(z'|z,a),\pi(a'|z')}\left[p^\pi_{\mathcal{Z}}(z^1 \mid z', a', h-1)\right].$$

Therefore, we have $p^\star(z^1 \mid 1, z, a, h)$ converges to $p^\pi_{\mathcal{Z}}(z^1 \mid z, a, h)$. □

A.2 Monte Carlo return estimate using successor models

Prior work Chung et al. (2026) defines a ν-Bellman operator and uses it to construct a target for fitting the critic. For simplicity, we use the standard λ-return estimate in TD(λ) (Sutton, 1988) to construct the target.

Given a trajectory sampled from the policy $\pi(a \mid z)$, a reward predictor $r(z, a)$, and a critic estimate $Q(z, a)$, the n-step return estimate (Sutton, 1988) is defined as

$$G^{(n)}(z, a) = \sum_{t=0}^{n-1} \gamma^t r(z_t, a_t) + \gamma^n Q(z_n, a_n), \quad z_0 = z, \quad a_0 = a,$$

where (z, a) is the starting state-action pair. We choose to use the truncated λ-return estimate ($\lambda \in [0, 1]$) with maximum horizon H as our return estimate (Sutton et al., 1998; Van Seijen et al., 2016; Hafner et al., 2020b):

$$G^{\lambda,H}(z,a) = (1-\lambda)\sum_{n=1}^{H-1}\lambda^{n-1}G^{(n)}(z,a) + \lambda^{H-1}G^{(H)}(z,a). \tag{16}$$

Note that the truncated λ-return estimate uses the H-step return estimate $G^{(H)}(z, a)$ to replace the tail. The coefficient λ interpolates between the one-step return estimate $G^{\lambda=0,H}(z, a) = G^{(1)}(z, a)$ and the H-step return estimate $G^{\lambda=1,H}(z, a) = G^{(H)}(z, a)$, balancing the bias-variance tradeoff (Schulman et al., 2016). Expanding Eq. 16 by using the definition of n-step return estimates gives a particularly useful form:

$$\begin{aligned}
G^{\lambda,H}(z,a) &= (1-\lambda)\sum_{n=1}^{H-1}\lambda^{n-1}G^{(n)}(z,a) + \lambda^{H-1}G^{(H)}(z,a) \\
&= \sum_{k=0}^{H-1}\gamma^k \underbrace{\left[(1-\lambda)\sum_{n=k+1}^{H-1}\lambda^{n-1} + \lambda^{H-1}\right]}_{=\lambda^k} r(z_k,a_k) \\
&\quad + \gamma(1-\lambda)\sum_{h=1}^{H-1}(\lambda\gamma)^{h-1}Q(z_h,a_h) + \gamma(\lambda\gamma)^{H-1}Q(z_H,a_H) \\
&= r(z,a) + \gamma\sum_{h=1}^{H}(\lambda\gamma)^{h-1}\left[(1-\alpha_h)r(z_h,a_h) + \alpha_h Q(z_h,a_h)\right], \\
\alpha_h &= \begin{cases} 1-\lambda, & h = 1,\dots,H-1, \\ 1, & h = H. \end{cases}
\end{aligned}$$

This expansion allows us to estimate the expected truncated λ-return using Monte Carlo samples from the successor models.

We first define the truncated geometric distribution $\texttt{TruncGeom}(1-\lambda\gamma, H)$ with probability mass function

$$p_{\texttt{TruncGeom}(1-\lambda\gamma,H)}(h) = \frac{1-\lambda\gamma}{1-(\lambda\gamma)^H}(\lambda\gamma)^{h-1}, \quad h = 1,\dots,H,$$

where $(1-\lambda\gamma)/(1-(\lambda\gamma)^H)$ is the normalizing constant. Then, replacing the geometric sum in the truncated λ-return with an expectation over this distribution and sampling future states from the learned successor model, we get

$$\mathbb{E}_\pi\left[G^{\lambda,H}(z,a)\right] \approx r(z,a) + \gamma\frac{1-(\lambda\gamma)^H}{1-\lambda\gamma}\mathbb{E}_{\substack{h\sim\texttt{TruncGeom}(1-\lambda\gamma,H),\\ z_h\sim\hat{p}(\cdot|1,z,a,h),\\ a_h\sim\pi(\cdot|z_h)}}\left[(1-\alpha_h)r(z_h,a_h) + \alpha_h Q(z_h,a_h)\right], \tag{17}$$

where $z_h \sim \hat{p}(\cdot \mid 1, z, a, h)$ is a clean future latent state at horizon h sampled from the learned successor model. The approximation becomes exact when the learned successor distributions match the true horizon-conditioned successor measure of the policy π. Thus, we can use Monte Carlo sampling to construct the truncated λ-return estimate for fitting the critic. Below, we show the expression for the sampled return target with the target critic estimate $\bar{Q}$ (again, to stabilize learning).

$$\widehat{G}^{\lambda,H}(z,a) = r(z,a) + \gamma\frac{1-(\lambda\gamma)^H}{1-\lambda\gamma}\left[(1-\alpha_h)r(z_h,a_h) + \alpha_h\bar{Q}(z_h,a_h)\right].$$

Taken together, we learn the reward predictor and the critic estimate without propagating their gradients into the image encoder through z:

$$\begin{aligned}
\mathcal{L}(Q) &= \mathbb{E}_{\substack{(o,a,r,o')\sim D,\ h\sim\texttt{TruncGeom}(1-\lambda\gamma,H),\\ z_h\sim\hat{p}(z_h|1,z,a,h),\ a_h\sim\pi(a_h|z_h)}}\left[\left(Q(z,a) - \widehat{G}^{\lambda,H}(z,a)\right)^2\right], \\
\mathcal{L}(r) &= \mathbb{E}_{(o,a,r,o')\sim D}\left[(r(z,a)-r)^2\right].
\end{aligned}$$

Algorithm 1 Latent Successor Models (LSM)

Input: Dataset D, maximum horizon H, discount γ, return coefficient λ.

Initialize image encoder f_θ, successor velocity v_ϕ, reward predictor r_ν, critic Q_μ, and flow BC velocity v_ξ^β.
Initialize target parameters $\bar{\theta} \leftarrow \theta$, $\bar{\phi} \leftarrow \phi$, and $\bar{\mu} \leftarrow \mu$.
for each training iteration **do**
    Sample a batch $\{(o, a, r, o')\} \sim D$.
    Encode latent $z = f_\theta(o)$ and target latent $z' = f_{\bar{\theta}}(o')$.

    $\triangledown$ Learn the image encoder and the successor model
    Sample prediction horizons $h \sim \textsc{Unif}(\{1, \dots, H\})$.
    Update θ and ϕ using bootstrapped loss $\mathcal{L}(f_\theta, v_\phi; h)$ with target networks $f_{\bar{\theta}}$ and $v_{\bar{\phi}}$ (Eq. 8).

    $\triangledown$ Learn the image encoder and the flow BC policy
    Update θ and ξ using the standard flow-matching loss $\mathcal{L}\big(f_\theta, v_\xi^\beta\big)$ (Eq. 5).

    $\triangledown$ Learn the critic using MC return estimates by sampling the successor model
    Construct MC return targets $\widehat{G}^{\lambda,H}(z, a)$ using v_ϕ, r_ν, $Q_{\bar{\mu}}$, and π (Sec. 4.4).
    Update μ using regression loss $\mathcal{L}(Q_\mu)$ (Eq. 11), treating z and the return targets as constants.

    $\triangledown$ Learn the reward predictor
    Update ν using regression loss $\mathcal{L}(r_\nu)$ (Eq. 11), treating z as constant.

    Update $\bar{\theta}$, $\bar{\phi}$, and $\bar{\mu}$ using exponential moving averages of θ, ϕ, and μ.
end for

B Algorithm

We use neural networks to parameterize the image encoder f_θ, successor velocity field v_ϕ, reward predictor r_ν, critic Q_μ, and the flow BC velocity v_ξ^β. At each iteration, we learn the successor models v_ϕ using the bootstrapped flow-matching loss $\mathcal{L}(f_\theta, v_\phi)$, the reward predictor r_ν by regressing onto dataset rewards, the critic Q_μ using MC return targets, and the flow BC policy parameterized by v_ξ^β using the standard flow-matching loss $\mathcal{L}(f_\theta, v_\xi^\beta)$. We optimize the image encoder jointly with the successor model and flow BC policy while stopping gradients from the reward and critic losses. The target networks $f_{\bar{\theta}}$, $v_{\bar{\phi}}$, and $Q_{\bar{\mu}}$ are updated using exponential moving averages, and we use $Q_{\bar{\mu}}$ to construct the MC return targets. We show the pseudocode of LSM in Alg. 1.

C Complete numerical results

For completeness, we provide the full results for every task from Visual OGBench and MMBench2 in Table 3 and Table 4 (corresponding to Table 1). The results are averaged over 4 seeds, and we report the mean and the standard deviations for each task. We highlight the results within 95% of the best performance using .

D Additional experiments

D.1 LSM preserves visual structure in future latent predictions

We examine whether the future latent states predicted by LSM retain meaningful visual information by reconstructing them with a learned decoder. Fig. 6 compares these reconstructions with pixel-space predictions from TD-Flow (Farebrother et al., 2025) and UHM (Chung et al., 2026). TD-Flow produces noisy samples that can contain spurious or missing objects, while UHM's predictions become increasingly noisy at longer horizons. The decoded predictions from LSM remain visually coherent through $h = 80$ and better preserve task-relevant information. These qualitative results

Table 3: **Complete results on Visual OGBench tasks.**

	Model-free					Model-based				
Environment	IQL	ReBRAC	IFQL	FQL	QC	LOMPO	OffDV2	PTD-Flow	LUHM	LSM
visual-cube-single-task1	73 ±7	84 ±4	44 ±6	67 ±3	88 ±5	0 ±0	0 ±0	0 ±0	78 ±2	88 ±3
visual-cube-single-task2	57 ±6	71 ±6	29 ±9	70 ±4	80 ±5	0 ±0	0 ±0	1 ±1	55 ±9	87 ±4
visual-cube-single-task3	79 ±4	85 ±1	59 ±10	68 ±13	90 ±4	0 ±0	0 ±0	44 ±28	90 ±2	95 ±2
visual-cube-single-task4	62 ±18	69 ±4	27 ±7	51 ±11	68 ±5	0 ±0	0 ±0	0 ±0	46 ±2	75 ±5
visual-cube-single-task5	54 ±4	62 ±4	22 ±4	57 ±8	62 ±6	0 ±0	0 ±0	0 ±0	35 ±17	65 ±2
visual-cube-double-task1	25 ±7	5 ±3	6 ±2	15 ±12	6 ±2	0 ±0	1 ±1	0 ±0	51 ±1	87 ±7
visual-cube-double-task2	4 ±1	1 ±0	0 ±0	0 ±0	2 ±2	0 ±0	0 ±0	0 ±0	0 ±0	34 ±16
visual-cube-double-task3	13 ±4	4 ±3	1 ±1	2 ±2	2 ±1	0 ±0	0 ±0	0 ±0	1 ±1	53 ±19
visual-cube-double-task4	1 ±1	0 ±0	0 ±1	0 ±0	0 ±1	0 ±0	0 ±0	0 ±0	1 ±1	22 ±13
visual-cube-double-task5	21 ±3	10 ±14	2 ±1	4 ±2	4 ±1	0 ±0	0 ±0	0 ±0	10 ±4	76 ±7
visual-scene-task1	69 ±40	97 ±2	88 ±4	99 ±1	100 ±0	7 ±5	9 ±6	19 ±17	56 ±2	100 ±0
visual-scene-task2	0 ±0	65 ±16	0 ±1	88 ±5	96 ±3	5 ±6	9 ±5	0 ±0	44 ±4	100 ±1
visual-scene-task3	17 ±2	4 ±3	15 ±6	19 ±5	89 ±9	0 ±0	0 ±0	0 ±0	9 ±3	76 ±14
visual-scene-task4	1 ±0	0 ±1	0 ±0	1 ±1	1 ±1	0 ±0	0 ±0	0 ±0	1 ±1	91 ±8
visual-scene-task5	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	72 ±16
visual-puzzle-3x3-task1	7 ±12	90 ±4	100 ±1	97 ±1	100 ±0	13 ±9	9 ±5	0 ±0	4 ±4	56 ±31
visual-puzzle-3x3-task2	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	84 ±25
visual-puzzle-3x3-task3	0 ±1	2 ±1	1 ±1	1 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	11 ±17
visual-puzzle-3x3-task4	0 ±0	1 ±1	1 ±0	6 ±2	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±1
visual-puzzle-3x3-task5	1 ±1	0 ±1	1 ±2	7 ±3	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	1 ±2
visual-puzzle-4x4-task1	0 ±0	22 ±5	11 ±19	28 ±5	57 ±7	0 ±0	2 ±1	0 ±0	0 ±0	0 ±0
visual-puzzle-4x4-task2	0 ±0	9 ±2	0 ±1	1 ±0	2 ±1	0 ±0	1 ±0	0 ±0	0 ±0	0 ±0
visual-puzzle-4x4-task3	0 ±0	15 ±2	2 ±3	14 ±3	31 ±11	0 ±0	1 ±1	0 ±0	0 ±0	0 ±0
visual-puzzle-4x4-task4	0 ±0	10 ±1	4 ±4	7 ±2	8 ±8	0 ±0	0 ±1	0 ±0	0 ±0	0 ±1
visual-puzzle-4x4-task5	0 ±0	10 ±1	0 ±0	1 ±0	1 ±1	0 ±0	1 ±1	0 ±0	0 ±0	0 ±1

Table 4: **Complete results on MMBench2 tasks.**

	Model-free					Model-based				
Environment	IQL	ReBRAC	IFQL	FQL	QC	LOMPO	OffDV2	PTD-Flow	LUHM	LSM
cup-catch-var1	13 ±2	47 ±6	11 ±1	71 ±1	22 ±4	4 ±3	45 ±26	9 ±1	21 ±4	40 ±12
finger-turn-easy-var1	24 ±5	32 ±4	23 ±4	35 ±3	23 ±2	24 ±9	43 ±1	19 ±1	27 ±1	31 ±5
ms-push-banana	5 ±2	8 ±3	11 ±4	3 ±2	3 ±2	23 ±3	0 ±0	5 ±1	6 ±0	4 ±3
og-point-var1	8 ±5	17 ±2	11 ±3	32 ±6	7 ±2	0 ±0	1 ±0	10 ±14	1 ±1	36 ±7
og-point-var2	7 ±4	14 ±3	9 ±3	28 ±5	6 ±2	0 ±0	1 ±0	7 ±10	0 ±0	38 ±7
pygame-point-maze-var4	10 ±6	25 ±14	26 ±7	45 ±4	16 ±1	2 ±3	5 ±1	17 ±24	45 ±1	56 ±7
pygame-reacher-easy	11 ±2	11 ±1	10 ±1	12 ±2	9 ±1	5 ±1	28 ±4	12 ±10	31 ±1	26 ±3
pygame-dungeon-explorer1	43 ±2	57 ±4	46 ±3	65 ±5	64 ±1	27 ±8	28 ±7	68 ±3	67 ±7	75 ±6
pygame-foraging	65 ±0	65 ±2	65 ±1	76 ±1	68 ±1	44 ±8	52 ±3	69 ±2	69 ±1	77 ±0
pygame-whirlpool	74 ±7	89 ±8	86 ±4	91 ±1	92 ±3	79 ±18	99 ±1	13 ±5	76 ±1	84 ±3

suggest that LSM learns to predict future latent states that retain reconstructible visual information, even though pixel reconstruction is not used as a representation learning objective.

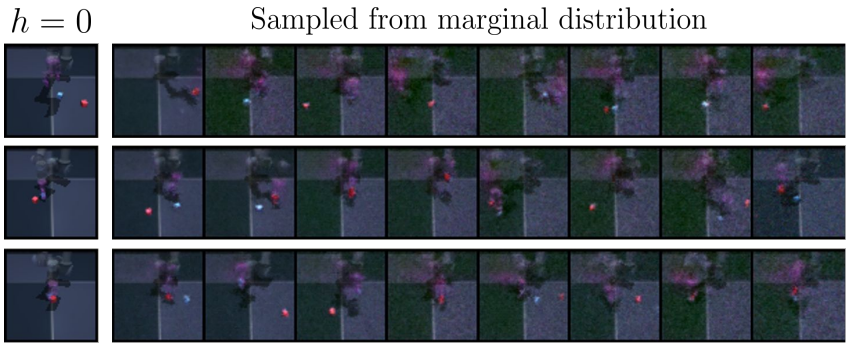
D.2 Removing gradients for the image encoder degrades the performance

Our ablation experiments study three design choices for learning the image encoder in LSM: gradients from the BC objective, gradients from the successor modeling objective, and the use of a target encoder. Since LSM incorporates all three ingredients, we ablate each choice separately by removing them one at a time. We again choose `visual-cube-double-task3` from visual OGBench to conduct our experiments.

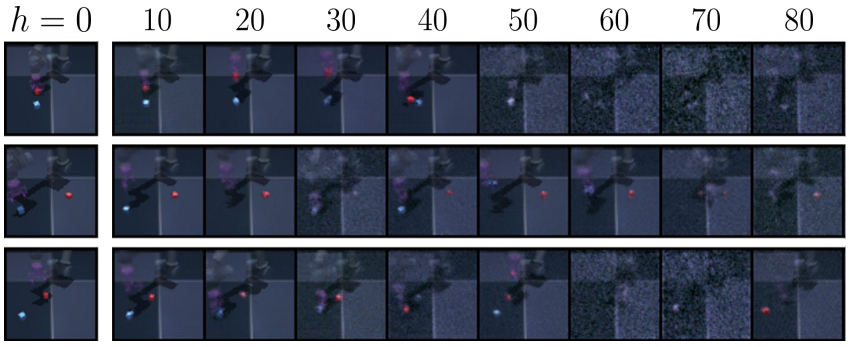
First, results in Fig. 7 show that stopping gradients from either the BC or successor modeling objective to the image encoder substantially reduces performance, suggesting that both BC and future prediction are important for shaping the latent representation. Second, removing the target encoder also leads to a large performance drop, supporting the use of a separate target encoder for self-predictive representation learning. Together, these results highlight the importance of combining both sources of supervision with a target encoder to learn actionable visual representations.

D.3 Using a residual block with adaptive layer normalization is important for LSM

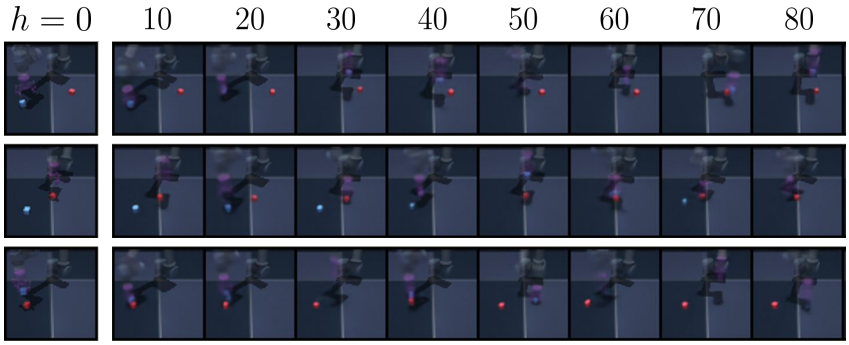
How does the architecture of the successor model affect the downstream policy performance? To answer this question, we compare LSM with a variant of it that uses an MLP as the backbone for the successor model (`MLP`). As discussed in Appendix E.2, our method uses one residual block with width



(a) TD-Flow trained in pixel space.



(b) UHM trained in pixel space.



(c) LSM trained in latent space with reconstructions generated by a learned decoder.

Figure 6: **Visualization of future predictions from different successor models.** The first column shows the initial observation at horizon $h = 0$. *(a)* TD-Flow generates 8 independent samples from its marginal successor distribution without conditioning on a specific horizon. Its pixel-space predictions are noisy and can contain spurious objects. *(b)* UHM generates pixel-space predictions conditioned on horizons $h = 10, 20, \ldots, 80$ (env steps). Visual quality degrades as the horizon increases, with predictions becoming increasingly noisy. *(c)* LSM predicts future latent states at the same horizons, where we visualize future latent predictions using a decoder trained to reconstruct observations. The decoded predictions remain visually coherent through $h = 80$, better preserving task-relevant information.

384 and adaptive layer normalization for its successor model. We conduct ablation experiments again using the `visual-cube-double-task3` from Visual OGBench. As shown in Fig. 8, LSM improves substantially over training, reaching approximately 53% success after 500K gradient steps, whereas `MLP` remains near 0% throughout training. These results suggest that the choice of architecture for the successor model plays an important role in effective policy learning.

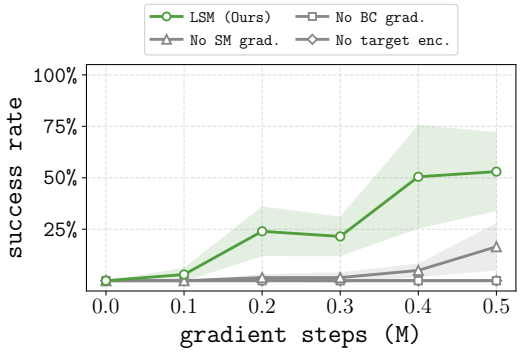


Figure 7: Stopping gradients from either the behavioral cloning or successor modeling objective to the image encoder substantially reduces performance, as does removing the target image encoder.

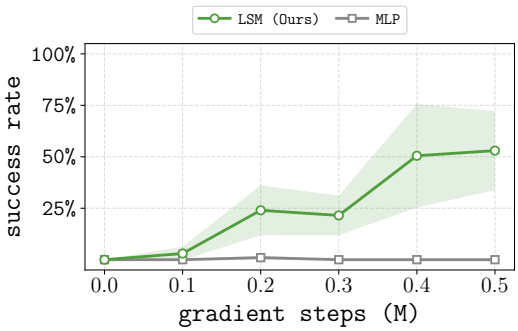


Figure 8: Using a residual block with adaptive layer normalization is important for LSM.

E Experimental Details

E.1 Datasets and Evaluation Protocols

We describe the datasets and evaluation protocols on three benchmarks: Visual OGBench, MMBench2, and Visual Robomimic.

Visual OGBench. We evaluate on 25 visual single-task environments from OGBench (Park et al., 2025a), comprising 5 tasks in each of 5 domains: `visual-cube-single`, `visual-cube-double`, `visual-scene`, `visual-puzzle-3x3`, and `visual-puzzle-4x4`. We use the 1M-transition `play` datasets and semi-sparse rewards for each task provided by the benchmark. For observations, we use 64×64 RGB images. Evaluation episodes terminate upon success or after 200 environment steps for `visual-cube-single`, 500 steps for `visual-cube-double`, `visual-puzzle-3x3`, and `visual-puzzle-4x4`, and 750 steps for `visual-scene`. We report the mean and one standard deviation of success rates at the end of training, evaluated over 50 episodes and averaged across 4 random seeds.

MMBench2. We evaluate on the 10 "unseen" tasks from MMBench2 (Hansen & Wang, 2026), which are held out from the pretraining task set. We train different methods on each task's own offline dataset, combining all 7 available `active_*` data partitions. We resize RGB observations from 224×224 to 64×64. Following prior work (Hansen & Wang, 2026), we report the normalized score on each task on a $0 - 100$ scale. Evaluation episodes are limited to 25 steps for `ms-push-banana`, 100 steps for `og-point-var1` and `og-point-var2`, 200 steps for `pygame-point-maze-var4` and `pygame-reacher-easy`, 250 steps for `pygame-dungeon-explorer1`, and 500 steps for the remaining four tasks. Each algorithm is evaluated over 50 episodes at the end of training, and results are the mean and one standard deviation across 4 random seeds.

Visual Robomimic. We evaluate on 3 visual manipulation tasks from Robomimic (Mandlekar et al., 2021): `visual-lift`, `visual-can`, and `visual-square`. We use the multi-human (MH) datasets for training different methods. Observations consist of 84×84 RGB images from the agent-view and wrist cameras, concatenated along the channel dimension, without proprioceptive inputs. Evaluation episodes terminate upon success or after 300 environment steps for `visual-lift` and `visual-can`, and 400 steps for `visual-square`. We report task success rates over 50 evaluation episodes per seed, averaged across 4 random seeds.

All experiments use visual observations without temporal frame stacking, satisfying the block MDP assumption in Sec. 3.

E.2 Methods for Comparison

We compare LSM with model-free and model-based visual offline RL methods across three benchmarks to examine the benefits of latent successor modeling. For the model-free baselines, BC provides

a reference for pure imitation learning. IQL (Kostrikov et al., 2022) and ReBRAC (Tarasov et al., 2023) cover implicit value learning and behavior-regularized policy improvement, while IFQL and FQL (Park et al., 2025c) assess the performance of their counterparts that use expressive flow policies. QC and QC-IFQL (Li et al., 2025) additionally examine whether action chunking and multi-step value learning suffice without learning a generative successor model. For model-based comparisons, LOMPO (Rafailov et al., 2021) and Offline DreamerV2 (Lu et al., 2023) represent approaches that learn latent representations through reconstruction and estimate values using autoregressive rollouts of one-step models. PTD-Flow (Farebrother et al., 2025; Zheng et al., 2026) and LUHM (Chung et al., 2026) provide closer comparisons with direct prediction of future outcomes: PTD-Flow predicts a marginal successor measure in pixel space, whereas LUHM predicts horizon-conditioned latent outcomes using a different training objective, return estimate, and policy parametrization. We describe each method and the sweep for their hyperparameters below.

Flow behavioral cloning (BC). Flow BC learns to predict actions in the dataset using flow matching (Lipman et al., 2023), without reward or value learning. Since LSM uses a chunk of actions as input, to make a fair comparison, we also use the flow policy to predict a chunk of consecutive actions using 10 integration steps. We sweep the action chunk size n_{ac} within $\{5, 10\}$.

Implicit Q-learning (IQL). IQL (Kostrikov et al., 2022) learns values through expectile regression on dataset actions and extracts a policy using advantage-weighted regression. We include it as a model-free approach that avoids querying out-of-distribution actions during value learning. For hyperparameters, we use the expectile coefficient $\eta = 0.9$ and the inverse temperature for advantage-weighting $\alpha \in \{0.3, 1, 3, 10\}$.

ReBRAC. ReBRAC (Tarasov et al., 2023) performs policy improvement by learning a parametric policy to maximize the critic while regularizing both the actor objective and the critic target toward dataset actions. It provides a comparison with a behavior-regularized actor-critic method that does not require a generative model for transitions. We denote its actor and critic regularization coefficients by α_{actor} and α_{critic}, respectively. We choose to sweep these hyperparameters with $\alpha_{\text{actor}} \in \{0.01, 0.1, 0.3, 1.0\}$ and $\alpha_{\text{critic}} \in \{0, 0.01\}$.

Implicit flow Q-learning (IFQL). IFQL is the flow-policy variant of IDQL (Hansen-Estruch et al., 2023; Park et al., 2025c). It combines the implicit value learning in IQL with a flow BC policy and selects the action via rejection sampling as in Eq. 12. This comparison examines whether an expressive flow policy and the same policy-extraction mechanism as LSM suffice when paired with model-free value learning. We use the expectile coefficient $\eta = 0.9$ and use $K = 32$ candidate actions with 10 integration steps.

Flow Q-learning (FQL). FQL (Park et al., 2025c) trains a one-step actor to maximize its critic estimate while distilling the output of a flow BC policy. We include it as a comparison against model-free value learning with an expressive policy and direct gradient-based policy improvement. The coefficient α weights the squared distillation penalty, which regularizes the one-step actor toward the behavior policy. We sweep the coefficient $\alpha \in \{0.1, 0.3, 1.0\}$ on MMBench2 and, for Visual OGBench, we reuse the α from Park et al. (2025c).

QC and QC-IFQL. QC (Li et al., 2025) learns a critic estimate over action chunks and selects among K chunks proposed by a flow BC policy using rejection sampling (Eq. 12). For the TD target for fitting the critic, it uses the sample rejection sampling strategy to select the next action chunk. QC-IFQL (Li et al., 2025) instead uses implicit value learning in IQL for the critic estimation while retaining rejection sampling for policy extraction. These comparisons assess the benefits of temporal difference value learning with chunked action space without predicting future observations. We sweep action chunk size n_{ac} within $\{5, 10\}$ for these methods and use $K = 32$ for rejection sampling with 10 integration steps. For QC-IFQL, we use the expectile coefficient $\eta = 0.9$.

Latent offline model-based policy optimization (LOMPO). LOMPO (Rafailov et al., 2021) learns a one-step latent transition model via reconstruction and performs uncertainty-aware policy optimization using imagined trajectories. We include it as an approach to offline RL from pixels that estimates values through autoregressive latent rollouts. We train a Gaussian policy to maximize the

critic for LOMPO. Its uncertainty penalty discourages the policy from exploiting unreliable model predictions. For the hyperparameters, we sweep the penalty coefficient λ for the uncertainty reward within $\lambda \in \{1, 3, 5, 10, 30\}$.

Offline DreamerV2 (OffDV2). Offline DreamerV2 (Lu et al., 2023) adapts DreamerV2 (Hafner et al., 2020a) to offline RL by penalizing model uncertainty when learning the value using imagined trajectories. Similar to LOMPO, it compares LSM against prior model-based RL methods that learn a one-step latent transition model using reconstruction and use autoregressive latent rollouts to estimate the value. We train a Gaussian policy to maximize the critic for OffDV2. For OffDV2, we sweep the penalty coefficient λ for the uncertainty reward within $\lambda \in \{1, 3, 5, 10, 30\}$.

Pixel-based TD Flows (PTD-Flow). TD Flows (Farebrother et al., 2025) learns a marginal successor measure using a bootstrapped flow-matching objective (TD^2-CFM), making it closely related to LSM. We include a pixel-space variant (PTD-Flow) adapted from the implementation of Zheng et al. (2026). Unlike LSM, PTD-Flow predicts observations from a discounted mixture of future horizons without explicitly conditioning on the horizon. We construct critic targets by averaging predicted rewards over $M = 1$ MC future observation sample (Zheng et al., 2026). PTD-Flow learns a Gaussian policy by maximizing the critic estimate with behavioral cloning regularization. We sweep the regularization coefficient over $\alpha \in \{10, 30, 100, 300\}$. We use an action chunk size n_{ac} of 10 on Visual OGBench and Visual Robomimic, and use an action chunk size n_{ac} of 5 on MMBench2.

Latent universal horizon models (LUHM). We adapt UHM (Chung et al., 2026) to visual observations using the same encoder and successor model architecture as LSM. Like LSM, LUHM learns a horizon-conditioned successor measure to directly predict future outcomes in a learned latent space. However, LUHM uses the coupled conditional flow matching objective from Farebrother et al. (2025) and constructs a ν-return estimate from $M = 1$ MC sample of a future latent state. Following UHM, we jointly predict terminal indicators with future latent states and gradually increase the return coefficient λ and maximum horizon H during training. We train a deterministic actor by critic maximization and a behavioral cloning penalty weighted by α. We sweep $\alpha \in \{0.003, 0.01, 0.03, 0.1, 0.3, 1.0, 3.0\}$, use final values $\lambda_{\text{final}} = 0.95$ and $H_{\text{final}} = 40$ at the end of training. We also use an action chunk size n_{ac} of 10 on Visual OGBench and Visual Robomimic, and use an action chunk size n_{ac} of 5 on MMBench2.

Latent successor models (LSM). Our method jointly learns a visual encoder and a horizon-conditioned latent successor model using a bootstrapped flow-matching objective (Eq. 8). The encoder is updated by both future prediction and behavioral cloning, while reward and critic losses do not update the encoder. We fit the critic using Monte Carlo λ-return targets, with $M = 1$ MC future latent state at a geometric horizon. Following prior work (Park & Lee, 2025), we use a chunked action space. For policy extraction, we learn a flow BC policy and select a chunk of actions among $K_{\text{train}} = 8$ candidates during training and $K_{\text{eval}} = 32$ at evaluation. We use a latent dimension of 256 and a successor model with one residual block of width 384 with adaptive layer normalization as in DiT (Peebles & Xie, 2023), jointly predicting future latent states and terminal indicators as in LUHM. We use 10 integration steps for successor sampling and 5 integration steps for the flow BC teacher in one-step distillation. During training, we gradually increase the return coefficient λ and the maximum horizon H as in LUHM, with a final coefficient $\lambda_{\text{final}} = 0.95$ and $H_{\text{final}} = 40$ (100 for MMBench2) at the end of training. We again use an action chunk size n_{ac} of 10 on Visual OGBench and Visual Robomimic, and use an action chunk size n_{ac} of 5 on MMBench2.

E.3 Hyperparameters

We report the hyperparameters used by LSM (Table 5) and the baselines on Visual OGBench (Table 6), MMBench2 (Table 7), and Visual Robomimic (Table 8). Whenever possible, we reuse the hyperparameters of LSM for prior methods. Both LSM and LUHM use a convolutional encoder adapted from DreamerV3 (Hafner et al., 2025a), denoted as `CNN` in the tables. For other baselines, we also use the `impala small` encoder adapted from Park et al. (2025c) and the RSSM encoder adapted from Hafner et al. (2020b;a). For image augmentation, we apply random shifts from DrQ (Kostrikov et al., 2021).

Table 5: Hyperparameters for LSM.

	Visual OGBench	MMBench2	Visual Robomimic
Optimization			
Optimizer		Adam	
Learning rate		3×10^{-4}	
Batch size		256	
Gradient steps		500K	
Target update rate (successor model, critic)		5×10^{-3}	
Target update rate (image encoder)		2×10^{-2}	
Discount factor γ	0.999	0.99	0.999
Action chunk size n_{ac}	10	5	10
Image encoder			
Image resolution	64×64	64×64	84×84, 2 cameras
Frame stack		1	
Architecture		CNN	
Channels		$[32, 48, 64, 64]$	
Kernel size		5×5	
Latent dimension		256	
Image augmentation		DrQ shift, $p_{\text{aug}} = 1.0$	
Successor model			
Architecture		residual MLP with adaLN	
Number of residual blocks		1	
Hidden width		384	
Integration steps		10	
Terminal prediction		Yes	
Final maximum horizon H_{final}	40	100	40
Return estimate and critic			
MLP size		$[512, 512, 512, 512]$	
MLP activation function		GELU	
Layer normalization		Yes	
Reward predictor MLP size		$[384]$	
Critic output		two-hot, 255 bins	
Number of Q ensembles		1	
Number of MC future latent samples M		1	
Final return coefficient λ_{final}		0.95	
Policy			
Integration steps		5	
Number of candidates K_{train} / K_{eval}		8 / 32	

Table 6: Hyperparameters for prior methods on Visual OGBench tasks.

	Model-free			Model-based			
				One-step model		Successor model	
Shared hyperparameters							
Learning rate	3×10^{-4}						
Optimizer	Adam						
MLP size	$[512, 512, 512, 512]$						
Image resolution	64×64						
Gradient steps	500K			100K		500K	
Batch size	256			64		256	
Batch length	1			50		1	
Encoder	impala small			RSSM		impala small	CNN
Frame stack	3			1		1	
Image augmentation	DrQ shift, $p_{\text{aug}} = 0.5$			–		DrQ shift, $p_{\text{aug}} = 0.5$	
Discount factor γ	0.99			0.99		0.99	0.999
Domain-specific hyperparameters	FQL (α)	ReBRAC ($\alpha_{\text{actor}}, \alpha_{\text{critic}}$)	IQL (α)	LOMPO (λ)	OffDV2 (λ)	PTD-Flow (α)	LUHM (α)
visual-cube-single-play	300	$(1.0, 0)$	1	5	5	10	0.3
visual-cube-double-play	100	$(0.1, 0)$	0.3	5	10	10	0.1
visual-scene-play	100	$(0.1, 0.01)$	10	3	10	10	0.03
visual-puzzle-3x3-play	300	$(0.3, 0.01)$	10	3	10	10	0.1
visual-puzzle-4x4-play	300	$(0.3, 0.01)$	3	5	3	10	0.1

E.4 LSM IMPROVES LONG-HORIZON FUTURE PREDICTION AND VALUE ESTIMATION

We conduct ablation experiments using `visual-cube-double-task3` from the Visual OGBench benchmark with the 1M play datasets. Since different policies induce different successor measures (Eq. 1), to make apples-to-apples comparisons, we want to learn pixel successor models to predict the successor measure of the same policy as our LSM. To achieve this goal, we first train LSM as in Alg. 1 for 500K gradient steps and then use the learned policy as a reference to learn pixel successor models. These pixel successor models learn to predict the raw future observations directly, using a decoder with a symmetric architecture of the image encoder (CNN with 482K parameters). We reuse

Table 7: Hyperparameters for prior methods on the MMBench2 tasks.

	Model-free	Model-based					
		One-step model		Successor model			
Shared hyperparameters							
Learning rate	3×10^{-4}						
Optimizer	Adam						
MLP size	$[512, 512, 512, 512]$						
Image resolution	64×64						
Gradient steps	500K	100K		500K			
Batch size	256	64		256			
Batch length	1	50		1			
Encoder	impala small	RSSM		impala small		CNN	
Frame stack	3	1		3			
Image augmentation	DrQ shift, $p_{\text{aug}} = 0.5$	–		DrQ shift, $p_{\text{aug}} = 0.5$			
Discount factor γ	0.99	0.99		0.99			
Task-specific hyperparameters	FQL (α)	ReBRAC ($\alpha_{\text{actor}}, \alpha_{\text{critic}}$)	IQL (α)	LOMPO (λ)	OffDV2 (λ)	PTD-Flow (α)	LUHM (α)
cup-catch-var1	0.3	$(0.01, 0)$	1	1	10	30	0.03
finger-turn-easy-var1	0.3	$(0.01, 0.01)$	1	30	1	30	0.3
ms-push-banana	1.0	$(1.0, 0)$	1	30	30	30	0.03
og-point-var1	0.3	$(0.01, 0.01)$	1	10	10	30	0.03
og-point-var2	0.3	$(0.01, 0.01)$	1	1	10	30	0.03
pygame-point-maze-var4	0.1	$(0.01, 0)$	1	3	1	30	0.3
pygame-reacher-easy	1.0	$(0.1, 0.01)$	1	30	1	30	0.03
pygame-dungeon-explorer1	0.3	$(0.1, 0)$	1	10	3	30	0.3
pygame-foraging	0.1	$(1.0, 0)$	1	30	10	30	0.3
pygame-whirlpool	0.1	$(0.1, 0)$	1	3	10	30	0.3

Table 8: Hyperparameters for prior methods on the Visual Robomimic tasks.

	Model-free	Model-based					
		One-step model		Successor model			
Shared hyperparameters							
Learning rate	3×10^{-4}						
Optimizer	Adam						
MLP size	$[512, 512, 512, 512]$						
Image resolution	84×84						
Gradient steps	500K	100K		500K			
Batch size	256	64		256			
Batch length	1	50		1			
Encoder	impala small	RSSM		impala small		CNN	
Frame stack	3	1		1			
Image augmentation	DrQ shift, $p_{\text{aug}} = 0.5$	–		DrQ shift, $p_{\text{aug}} = 0.5$			
Discount factor γ	0.99	0.99		0.99		0.99	0.999
Task-specific hyperparameters	BC (n_{ac})	QC (n_{ac})	QC-IFQL (n_{ac})	LOMPO (λ)	OffDV2 (λ)	PTD-Flow (α)	LUHM (α)
lift	10	5	5	3	10	100	3.0
can	10	5	10	10	10	100	3.0
square	10	5	5	10	10	100	3.0

the same bootstrapped flow-matching objective as in Eq. 8, but sample the next action a' using the reference policy, effectively performing SARSA (Sutton et al., 1998). To make a fair comparison, we also learn a pixel-space BC policy to distill the reference policy for pixel successor models. We also train pixel successor models for 500K gradient steps and evaluate both latent successor models and pixel successor models at the end of training.

For evaluation, we first sample 16 initial observations from the initial state distribution and then roll out 100 trajectories in the real environment using the reference policy for each initial observation. We then collect observations at different time steps and compute discounted return-to-go as the ground truth for measuring future prediction and value estimation error. For both latent and pixel successor models, we also compare two different strategies for sampling the future outcomes and value estimates: direct sampling at a specified horizon (`direct sampling`) or step-by-step prediction (`autoregressive rollout`).

Future prediction. For LSM, starting from the initial state, we either query the successor model with horizon input as 1 to autoregressively roll out imagined trajectories until a specific horizon h or query the successor model at horizon h to generate a future latent state directly. We repeat this process multiple times to generate 100 imagined trajectories or 100 MC future latent states and then compute the squared maximum mean discrepancy against the ground-truth observations with a kernel bandwidth of 1. For pixel successor models, we use the pixel BC policy and employ a similar

sampling procedure in the pixel space. Because the pixel space is different from the latent space, computing the MMD in pixel space will not give us an apples-to-apples comparison. Instead, we encode the predictions from the pixel successor models using the image encoder learned by LSM and measure the MMD in the latent space. Therefore, Fig. 3 *(Center)* shows the MMD of LSM and pixel successor models in the same latent space.

Value estimation. For value estimation, we measure the MSE between the value estimate and ground-truth return-to-go at different time steps t. Specifically, for LSM, at each time step t we either autoregressively roll out imagined trajectories until the end of the episode and then construct a truncated λ-return estimate or directly query the successor model at different horizons h to construct the truncated λ-return estimate as in Eq. 17. Again, we repeat this process multiple times to generate 100 imagined trajectories or 100 MC future latent states and then compute the mean squared error against the ground-truth discounted return-to-go. For pixel successor models, we use the pixel BC policy and employ a similar sampling procedure in the pixel space.

E.5 LSM LEARNS GOOD VISUAL REPRESENTATIONS FOR MONTE CARLO VALUE ESTIMATION

We conduct ablation experiments using the same setup as in Appendix E.4 and follow the same value estimation protocol. For alternative representation learning objectives, we use them to train the image encoder and then jointly learn the latent successor models and policies on top of each representation. We train each method for 500K gradient steps and evaluate the value estimation error at the end of training. Below, we describe the alternative representation learning objectives.

Reconstruction. Reconstruction (Hafner et al., 2020b) utilizes an autoencoder objective that reconstructs the observation from its representation:

$$\mathcal{L}_{\text{recon}}(f, g) = \mathbb{E}\left[\|g(f(o)) - o\|_2^2\right], \tag{18}$$

where $g : \mathcal{Z} \to \mathcal{O}$ is the decoder that is jointly trained with the encoder. This objective encourages the latent representation to preserve information for reconstructing the full visual observation.

Contrastive learning (SimCLR). SimCLR (Chen et al., 2020) learns a visual representation using the standard InfoNCE objective (van den Oord et al., 2018) with data augmentation. Specifically, given two independent augmentations (random cropping) of the same observation o denoted as o_{aug1} and o_{aug2}, SimCLR (Chen et al., 2020) uses them as a positive pair and uses other augmented observations in the same batch o^-_{aug} as negative pairs. With a normalized representation space, we use $\tilde{z}_{\text{aug1}} = f(o_{\text{aug1}})/\|f(o_{\text{aug1}})\|_2$, $\tilde{z}_{\text{aug2}} = f(o_{\text{aug2}})/\|f(o_{\text{aug2}})\|_2$, and $\tilde{z}^-_{\text{aug}} = f(o^-_{\text{aug}})/\|f(o^-_{\text{aug}})\|_2$ to denote the normalized representations of observations o_{aug1} and o_{aug2}. Then, the SimCLR loss is defined as

$$\mathcal{L}_{\text{SimCLR}}(f) = -\mathbb{E}\left[\text{sim}\left(\tilde{z}_{\text{aug1}}, \tilde{z}_{\text{aug2}}\right) - \log \sum_{\tilde{z}^-_{\text{aug}}} \exp\left(\text{sim}\left(\tilde{z}_{\text{aug1}}, \tilde{z}^-_{\text{aug}}\right)\right)\right],$$

where $\text{sim}(\cdot, \cdot)$ denotes cosine similarity with a fixed temperature. This objective encourages the representation to be invariant to visual augmentations, but does not learn to separate different observations.

Bisimulation. Bisimulation (Li et al., 2006; Zhang et al., 2021) learns a compact representation that encourages equivalent states (states with similar rewards and one-step transitions) to be mapped close together. Specifically, given two independent transitions (o_1, a_1, r_1, o_1') and (o_2, a_2, r_2, o_2'), we minimize

$$\mathcal{L}_{\text{Bisim}}(f) = \mathbb{E}\left[\left(\|f(o_1) - f(o_2)\|_1 - |r_1 - r_2| - \gamma \left\|\bar{f}(o_1') - \bar{f}(o_2')\right\|_2\right)^2\right],$$

where $\bar{f}(\cdot)$ denotes the target image encoder. This objective encourages distances in the latent space to reflect differences in immediate rewards and future transitions, preserving actionable information.

One-step BYOL. We learn representations using a one-step transition model in a compact latent space. This method adapts the idea of BYOL (Grill et al., 2020). Given a transition (o, a, o'), we learn representations by predicting the next latent state $\bar{z} = \bar{f}(o')$ using the current latent state $z = f(o)$

and the current action a as input. Here, the next latent state is encoded using a target image encoder $\bar{f}$. We learn the image encoder f by minimizing

$$\mathcal{L}_{\text{BYOL}}(f, g) = \mathbb{E}\left[\|g(f(o), a) - \bar{f}(o')\|_2^2\right],$$

where $g : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$ is an auxiliary latent predictor. We note that the target encoder $\bar{f}$ does not propagate gradients. This objective encourages the representations to be predictable, without pixel reconstruction or negative pairs.