DREAM TO CONTROL: LEARNING BEHAVIORS BY LATENT IMAGINATION

Anonymous authors
Paper under double-blind review

Abstract
To select effective actions in complex environments, intelligent agents need to generalize from past experience. World models can represent knowledge about the environment to facilitate such generalization. While learning world models from high-dimensional sensory inputs is becoming feasible through deep learning, there are many potential ways for deriving behaviors from them. We present Dreamer, a reinforcement learning agent that solves long-horizon tasks purely by latent imagination. We efficiently learn behaviors by backpropagating analytic gradients of learned state values through trajectories imagined in the compact state space of a learned world model. On 20 challenging visual control tasks, Dreamer exceeds existing approaches in data-efficiency, computation time, and final performance.

1 Introduction
Intelligent agents can achieve goals in complex environments even though they never encounter the exact same situation twice. This ability requires building representations of the world from past experience that enable generalization to novel situations. World models offer an explicit way to represent an agent’s knowledge about the world in a parametric model learned from experience that can make predictions about the future.

When the sensory inputs are high-dimensional images, latent dynamics models can abstract observations to predict forward in compact state spaces (Watter et al., 2015; Oh et al., 2017; Gregor et al., 2019). Compared to predictions in image space, latent states have a small memory footprint and enable imagining thousands of trajectories in parallel. Learning effective latent dynamics models is becoming feasible through advances in deep learning and latent variable models (Krishnan et al., 2015; Karl et al., 2016; Doerr et al., 2018; Buesing et al., 2018).

Behaviors can be derived from learned dynamics models in many ways. Often, imagined rewards are maximized by learning a parametric policy (Sutton, 1991; Ha and Schmidhuber, 2018; Zhang et al., 2019) or by online planning (Chua et al., 2018; Haßner et al., 2019). However, considering only rewards within a fixed imagination horizon results in shortsighted behaviors. Moreover, prior work commonly resorts to derivative-free optimization for robustness to model errors (Ebert et al., 2017; Chua et al., 2018; Parmas et al., 2019), rather than leveraging the analytic gradients offered by neural network dynamics models (Henaff et al., 2018; Srinivas et al., 2018).

We present Dreamer, an agent that learns long-horizon behaviors from images purely by latent imagination. A novel actor critic algorithm accounts for rewards beyond the planning horizon while making efficient use of the neural network dynamics. For this, we predict state values and actions in the learned latent space as summarized in Figure 1. The values optimize Bellman consistency for imagined rewards and the policy maximizes the values by propagating their analytic gradients back through the dynamics.

In comparison to actor critic algorithms that learn online or by experience replay (Lillicrap et al., 2015; Mnih et al., 2016; Schulman et al., 2017; Haarnoja et al., 2018; Lee et al., 2019), world models enable interpolating between past experience and offer analytic gradients of multi-step returns for efficient policy optimization.
The key contributions of this paper are summarized as follows:

- **Learning long-horizon behaviors in imagination** Purely model-based agents can be short-sighted due to finite imagination horizons. We approach this limitation in latency predicting both actions and state values. Training purely by latent imagination lets us efficiently learn the policy by propagating analytic gradients of the value function back through latent state transitions.

- **Empirical performance for visual control** We pair Dreamer with three representation learning objectives to evaluate it on the DeepMind Control Suite with image inputs, shown in Figure 2. Using the same hyper parameters for all tasks, Dreamer exceeds existing model-based and model-free agents in terms of data-efficiency, computation time, and final performance.

2 **CONTROL WITH WORLD MODELS**

**Reinforcement learning** We formulate visual control as a partially observable Markov decision process (POMDP) with discrete time step $t \in [1; T]$, continuous vector-valued actions $a_t \sim p(a_t | o_{<t}, a_{<t})$ generated by the agent, and high-dimensional observations and scalar rewards $o_t, r_t \sim p(o_t, r_t | o_{<t}, a_{<t})$ generated by the unknown environment. The goal is to develop an agent that maximizes the expected sum of rewards $E_p(\sum_{t=1}^{T} r_t)$. Figure 2 shows a selection of our tasks.

**Agent components** The classical components of agents that learn in imagination are dynamics learning, behavior learning, and environment interaction (Sutton, 1991). In the case of Dreamer, the behavior is learned by predicting hypothetical trajectories in the compact latent space of the world model. As outlined in Figure 3 and detailed in Algorithm 1, Dreamer performs the following operations throughout the agent’s life time, either sequentially interleaved or in parallel:

- Learn the latent dynamics model from the dataset of past experience to predict future rewards from actions and past observations. Any learning objective for the world model can be incorporated with Dreamer. We review existing methods for learning latent dynamics in Section 4.

- Learn action and value models from predicted latent trajectories, as described in Section 3. The value model optimizes Bellman consistency for imagined rewards and the action model is updated by propagating gradients of value estimates back through the neural network dynamics.

- Execute the learned action model in the world to collect new experience for growing the dataset.

**Latent dynamics** Dreamer uses a latent dynamics model that consists of three components. The representation model encodes observations and actions to create continuous vector-valued model states $s_t$ with Markovian transitions (Watter et al., 2015; Zhang et al., 2019; Hafner et al., 2019). The transition model predicts future model states without seeing the corresponding observations that will cause them. The reward model predicts the rewards given the model states,

\[
\begin{align*}
\text{Representation model:} & \quad p(s_t | s_{t-1}, a_{t-1}, o_t) \\
\text{Transition model:} & \quad q(s_t | s_{t-1}, a_{t-1}) \\
\text{Reward model:} & \quad q(r_t | s_t). \quad (1)
\end{align*}
\]

The model mimics a non-linear Kalman filter (Kalman, 1960), latent state space model, or HMM with real-valued states. However, it is conditioned on actions and predicts rewards, allowing the agent to imagine the outcomes of potential action sequences without executing them in the environment.
Dreamer learns long-horizon behaviors in the compact latent space of a learned world model. For this, we propagate stochastic gradients of multi-step returns through neural network predictions of actions, states, rewards, and values using reparameterization. This section describes our core contribution.

**Imagination environment** The latent dynamics define a Markov decision process (MDP; Sutton, 1991) that is fully observed since the compact model states $s_t$ are Markovian. We denote imagined quantities with $\tau$ as the time index. Imagined trajectories start at the true model states $s_t$ of observation sequences drawn from the agent’s past experience. They follow predictions of the transition model $s_\tau \sim q(s_\tau \mid s_{t-1}, a_{t-1})$, reward model $r_\tau \sim q(r_\tau \mid s_\tau)$, and a policy $a_\tau \sim q(a_\tau \mid s_\tau)$. The objective is to maximize expected imagined rewards $E_q \left( \sum_{t=1}^{H} \gamma^{t-1} r_\tau \right)$ with respect to the policy.

**Action and value models** Consider imagined trajectories with a finite horizon $H$. Dreamer uses an actor critic approach to learn behaviors that consider rewards beyond the horizon. We learn an action model and a value model in the latent space of the world model. The action model implements the policy and aims to predict actions that solve the imagination environment. The value model estimates the state values $V(s_\tau) \equiv E_q(s_\tau) \left( \sum_{t=1}^{\infty} \gamma^{t-1} r_\tau \right)$ for the action model, the expected sum of imagined rewards that it achieves in each state $s_\tau$,

$$
\begin{align*}
\text{Action model:} & \quad a_\tau \sim q_\phi(a_\tau \mid s_\tau) \\
\text{Value model:} & \quad v_\xi(s_\tau) \approx V(s_\tau). 
\end{align*}
$$

The action and value models are trained cooperatively as typical in policy iteration: the action model aims to maximize an estimate of the value, while the value model aims to match an estimate of the value that changes as the action model changes.

We use dense neural networks for the action and the value model with parameters $\phi$ and $\xi$, respectively. The action model outputs a tanh-transformed Gaussian (Haarnoja et al., 2018) with sufficient statistics predicted by the neural network. This allows for reparameterized sampling (Kingma and Welling, 2013; Rezende et al., 2014) that lets sampled actions depend deterministically on the neural network output, allowing to backpropagate analytic gradients through the sampling operation,

$$
a_\tau = \tanh(\mu_\phi(s_\tau) + \sigma_\phi(s_\tau) \epsilon), \quad \epsilon \sim \text{Normal}(0, 1). $$

**Value estimation** To learn the action and value models, we need to estimate the state values of imagined trajectories $\{s_\tau, a_\tau, r_\tau\}_{\tau=1}^{H}$. These trajectories branch off of the model states $s_t$ of sequence batches drawn from the agent’s dataset of experience and predict forward for the imagination horizon $H$ using actions sampled from the action model. State values can be estimated in multiple
ways that trade off bias and variance (Sutton and Barto, 2018),

$$\mathcal{V}_R(s_t) \triangleq \mathbb{E}_{q(s_t)} \left( \sum_{h=1}^{t+H} r_n \right),$$  

(4)

$$\mathcal{V}^k_N(s_t) \triangleq \mathbb{E}_{q(s_t)} \left( \sum_{n=\tau}^{t+H} \gamma^{n-\tau} r_n + \gamma^{h-\tau} v_\lambda(s_h) \right) \quad \text{with} \quad h = \min(\tau + k, t + H),$$  

(5)

$$\mathcal{V}_\lambda(s_t) \triangleq (1-\lambda) \sum_{n=1}^{H-1} \lambda^n \mathcal{V}^n_N(s_t) + \lambda^H \mathcal{V}_N^H(s_t),$$  

(6)

where the expectations are estimated with the imagined trajectories. $\mathcal{V}_R$ simply sums the rewards from $\tau$ until the horizon and ignores rewards beyond it. This allows learning the action model without value model, an ablation we compare to in our experiments. $\mathcal{V}^k_N$ estimates rewards beyond $k$ steps with the learned value model. Dreamer uses $\mathcal{V}_\lambda$, which computes an exponentially-weighted average of the estimates for different $k$ to balance bias and variance. Figure 4 shows that learning a value function in imagination enables Dreamer to solve long-horizon tasks while being robust to the imagination horizon. The experimental details and results on all tasks are described in Section 5.

**Learning objective** To update the action and value models, we first compute the value estimates $\mathcal{V}_\lambda(s_t)$ for states $s_t$ along the imagined trajectories. The objective for the action model $q_\phi(a_t | s_t)$ is to output actions that result in state trajectories with high value estimates. The objective for the value model $v_\xi(s_t)$, in turn, is to regress the value estimates,

$$\max_\phi \mathbb{E}_{q_\phi, q_\theta} \left( \sum_{\tau=t}^{t+H} \mathcal{V}_\lambda(s_\tau) \right),$$  

(7)

$$\min_\xi \mathbb{E}_{q_\phi, q_\theta} \left( \sum_{\tau=t}^{t+H} \frac{1}{2} \left\| v_\xi(s_\tau) - \mathcal{V}_\lambda(s_\tau) \right\|^2 \right).$$  

(8)

The value model is simply updated to regress the targets, around which we stop the gradient as typical in temporal difference learning (Sutton and Barto, 2018). The action model uses analytic gradients through the learned dynamics to maximize the value estimates. To understand this, we note that the value estimates depend on the reward and value predictions, which depend on the imagined states, which in turn depend on the imagined actions. Since these steps are all implemented as neural networks with reparameterized sampling, we analytically compute $\nabla_\phi \mathbb{E}_{q_\phi, q_\theta} \left( \sum_{\tau=t}^{t+H} \mathcal{V}_\lambda(s_\tau) \right)$ by stochastic backpropagation (Kingma and Welling, 2013; Rezende et al., 2014). The world model is fixed while learning the action and value models.

**Comparison to actor critic methods** Agents using Reinforce gradients (Williams, 1992) employ a value baseline to reduce gradient variance, such as A3C (Mnih et al., 2016) and PPO (Schulman et al., 2017), while Dreamer backpropagates through the value model. This is similar to analytic actor critics (Silver et al., 2014), such as DDPG (Lillicrap et al., 2015) and SAC (Haarnoja et al., 2018). However, these do not leverage gradients through the state transitions and only maximize immediate Q-values. MVE and STEVE (Feinberg et al., 2018; Buckman et al., 2018) extend these to multi-step Q-learning using learned dynamics to help rewards propagate faster into the value estimates. We simply predict state values, which is sufficient for policy optimization since we backpropagate through the dynamics.

We empirically compare learning action and value models from $\mathcal{V}_R$ which does not require a value model, and online planning in our experiments in Figure 7.
Learning behaviors in imagination requires a world model that generalizes well. We focus on latent dynamics models that predict forward in a compact latent space, facilitating long-term predictions and allowing to imagine thousands of trajectories in parallel. Several objectives for learning representations for control have been proposed (Watter et al., 2015; Jaderberg et al., 2016; Oord et al., 2018; Eslami et al., 2018). We review three approaches for learning representations to use with Dreamer: image reconstruction, contrastive estimation, and reward prediction.

**Reward prediction** Latent imagination requires a representation model \( p(s_t \mid s_{t-1}, a_{t-1}, o_t) \), transition model \( q(s_t \mid s_{t-1}, a_{t-1}) \), and reward model \( q(r_t \mid s_t) \), as described in Section 2. In principle, this could be achieved by simply learning to predict future rewards given actions and past observations (Oh et al., 2017; Gelada et al., 2019). Given a large and diverse dataset, such representations should be sufficient for solving a given control problem. However, while the agent is still exploring and especially when the reward signal is limited, additionally learning about observations is likely to improve the world model (Jaderberg et al., 2016; Gregor et al., 2019).

**Representation learning** The world model is learned from sequences \( \{(o_t, a_t, r_t)\}_{t=1}^T \) drawn from the agent’s dataset of experience. To learn representations that generalize, the model states \( s_{1:T} \) should be predictive of observations \( o_{1:T} \) and rewards \( r_{1:T} \) while not overfitting to individual examples in the dataset. At a high level, this is formalized by an information bottleneck (Tishby et al., 2000),

\[
\max_\theta J_{\text{INFO}}, \quad J_{\text{INFO}} \triangleq I(s_{1:T}; (o_{1:T}, r_{1:T}) \mid a_{1:T}) - \beta I(s_{1:T}; i_{1:T} \mid a_{1:T}). \tag{9}
\]

The first terms encourages mutual information between the model states and the observations and rewards. The second term penalizes information between model states and dataset indices \( i_{1:T} \) by an amount \( 0 \leq \beta \leq 1 \). The dataset indices relate to the images by a Dirac delta \( p(o_t \mid i_t) \) as in Alemi et al. (2016). The information bottleneck poses the representation learning problem in a generic way and provides a common view on pixel reconstruction and contrastive estimation. While the two information terms are difficult to estimate, they are easy to bound an optimize (Poole et al., 2019).

**Reconstruction** We first describe the world model used by PlaNet (Hafner et al., 2019), shown in Figure 3a. It bounds the objective by predicting observations and rewards from the model states,

\[
J_{\text{INFO}} \geq E_p \left( \sum_t \left( J_{\text{REC}}^t + J_{\text{R}}^t + J_{\text{KL}}^t \right) \right) + \text{const} \quad J_{\text{REC}}^t \triangleq \ln q(o_t \mid s_t) \tag{10}
\]

\[
J_{\text{R}}^t \triangleq \ln q(r_t \mid s_t) \quad J_{\text{KL}}^t \triangleq -\beta \text{KL} \left( p(s_t \mid s_{t-1}, a_{t-1}, o_t) \parallel q(s_t \mid s_{t-1}, a_{t-1}) \right)
\]

where the expectation samples sequences from the dataset and states from the representation model. The bound includes a reconstruction term, a reward prediction term, and a KL regularizer. We refer to Appendix C for the derivation. The bound uses four distributions that we implement as neural networks and optimize jointly to increase the bound,

- **Representation model**: \( s_t \sim p_\theta(s_t \mid s_{t-1}, a_{t-1}, o_t) \)
- **Observation model**: \( q_\theta(o_t \mid s_t) \)
- **Reward model**: \( q_\theta(r_t \mid s_t) \)
- **Transition model**: \( q_\theta(s_t \mid s_{t-1}, a_{t-1}) \).
Figure 6: Performance comparison to existing methods. Dreamer exhibits the data-efficiency of PlaNet while exceeding the asymptotic performance of the best model-free agents. After $2 \times 10^6$ environment steps, Dreamer reaches an average performance of 802 across tasks, compared to PlaNet at 312 and the top model-free D4PG agent at 786 after $10^9$ steps. Results are averages over 3 seeds.

We implement the transition model as recurrent state space model (RSSM; Hafner et al., 2019), the representation model by combining the RSSM with a convolutional neural network (CNN; LeCun et al., 1989) applied to the image observation, the observation model as a transposed CNN, and the reward model as dense network. The combined parameter vector $\theta$ is updated by reparameterization gradients (Kingma and Welling, 2013; Rezende et al., 2014). Figure 5 shows video predictions of this model. We refer to Appendix B and Hafner et al. (2019) model details.

**Contrastive estimation** Accurately predicting pixels in visually complex environments can be a challenging task. We can avoid reconstruction by instead predicting model states (Guo et al., 2018). While the observation marginal above was a constant, we now face the state marginal. Using the InfoNCE bound (Gutmann and Hyvärinen, 2010; Oord et al., 2018) as described in Appendix C,

$$J_{\text{INFO}} \geq E \left( \sum_t \left( J_{\text{NCE}}^t + J_{\text{R}}^t + J_{\text{KL}}^t \right) \right),$$

where $\sum_{o'} q(s_t | o')$ estimates the marginal by summing over observations $o'$ of the current sequence batch. Intuitively, $q(s_t | o_t)$ makes the state predictable from the current image and $\ln \sum_{o'} q(s_t | o')$ keeps it diverse to prevent collapse. Instead of the observation model, the bound uses a state model,

$$q_\theta(s_t | o_t).$$

We implement the state model as a CNN and again optimize the bound with respect to the combined parameter vector $\theta$ using reparameterization gradients. While avoiding pixel prediction, the amount of information this bound can extract efficiently is limited (McAllester and Statos, 2018). We empirically compare reconstruction, contrastive, and reward objectives in our experiments in Figure 8.

## 5 Experiments

**Visual control tasks** We evaluate Dreamer on 20 continuous control tasks with image observations of the DeepMind Control Suite (Tassa et al., 2018), illustrated in Figure 2. These tasks pose a variety of challenges, including partial observability, sparse rewards, contact dynamics, and 3D environments. We selected the tasks on which Tassa et al. (2018) report non-zero performance from image inputs. Agent observations are images of shape $64 \times 64 \times 3$, actions range from 1 to 12 dimensions, rewards are between 0 and 1, episodes contain 1000 steps, and initial states are randomized. Visualizations of our agent are available at https://dreamrl.github.io.

**Implementation** All experiments used a single Nvidia V100 GPU and 10 CPU cores per training run. Our implementation uses TensorFlow Probability (Dillon et al., 2017) and will be open sourced. The training time for our implementation of Dreamer is 10 hours per $10^6$ environment steps without parallelization, compared to 17 hours for online planning using PlaNet, and 24 hours for D4PG. We use the same hyper parameters across all tasks including a fixed action repeat of $R = 2$, as detailed in Appendix B. The world models are learned by reconstruction unless noted otherwise.

**Baseline methods** We compare Dreamer to several baselines: The current best reported performance on the considered tasks is by D4PG (Barth-Maron et al., 2018), an improved variant of DDPG (Lillicrap et al., 2015) that uses distributed experience collection, distributional Q-learning, multi-step returns, and prioritized experience replay. PlaNet (Hafner et al., 2019) learns the same world model as Dreamer and selects actions using online planning instead of learning an action model. We include the numbers for D4PG from Tassa et al. (2018) and re-run PlaNet with $R = 2$ for a fair comparison.
Performance comparison To evaluate the performance of Dreamer, we compare with state-of-the-art reinforcement learning agents. The results are summarized in Figure 6. With an average score of 802 across tasks after $2 \times 10^6$ environment steps, Dreamer exceeds the performance of the strong model-free D4PG agent that achieves an average of 786 within $10^6$ environment steps. At the same time, Dreamer inherits the data-efficiency of PlaNet, confirming that the learned world model can help to generalize from small amounts of experience. The empirical success of Dreamer shows that learning behaviors by latent imagination can outperform top methods based on experience replay.

Long-horizon behavior To investigate its ability to learn long-horizon behaviors, we compare Dreamer to alternatives for deriving behaviors from the world model at various horizon lengths. For this, we learn an action model to maximize imagined rewards without value model and compare to online planning using PlaNet. Figure 4 shows the final performance for different imagination horizons, confirming that the value model makes Dreamer more robust to the horizon and results in high performance even for short horizons. Performance curves for all 20 tasks with horizon of 20 are shown in Appendix D, where Dreamer outperforms the alternatives on 15 of 20 tasks and 3 ties.

Representation learning Dreamer can be used with any dynamics model that predicts future rewards given actions and past observations. Since the representation learning objective is orthogonal to our algorithm, we compare three natural choices described in Section 4: pixel reconstruction, contrastive estimation, and pure reward prediction. Figure 8 shows clear differences in task performance for different representation learning approaches, with pixel reconstruction outperforming contrastive estimation on most tasks. This suggests that future improvements in representation learning are likely to transfer over to task performance with Dreamer. Reward prediction along was not sufficient to solve any of the tasks in our experiments. Further ablations are included in the appendix of the paper.

6 RELATED WORK

Prior work learns latent dynamics for visual control by derivative-free policy learning or online planning, augments model-free agents with multi-step predictions, or uses analytic gradients of Q-values or multi-step rewards, often for low-dimensional tasks. In comparison, Dreamer uses analytic gradients to efficiently learn long-horizon behaviors for visual control purely by latent imagination.

Control with latent dynamics E2C (Watter et al., 2015) and RCE (Banijamali et al., 2017) embed images to predict forward in a compact space to solve simple tasks. World Models (Ha and Schmidhuber, 2018) learn latent dynamics in a two-stage process to evolve linear controllers in imagination. PlaNet (Hafner et al., 2019) learns them jointly and solves visual locomotion tasks by latent online planning. Similarly, SOLAR (Zhang et al., 2019) solves robotic tasks via guided policy search in
Figure 8: Comparison of representation learning objectives to be used with Dreamer. Pixel reconstruction performs best for the majority of tasks. The contrastive objective solves about half of the tasks, while predicting rewards alone was not sufficient in our experiments. The results suggest that future developments in learning representations are likely to translate into improved task performance when using Dreamer. The performance curves for all tasks are included in Appendix E.

We present Dreamer, an agent that learns long-horizon behaviors purely by latent imagination. For this, we propose a novel actor critic method that optimizes a parametric policy by propagating analytic gradients of multi-step values back through latent neural network dynamics. Dreamer outperforms previous approaches in data-efficiency, computation time, and final performance on a variety of challenging continuous control tasks from image inputs. While our approach compares favourably on these tasks, future research on learning representations is likely needed to scale latent imagination to visually more complex environments.
REFERENCES


A Detailed Algorithm

Algorithm 1: Dreamer

Hyper parameters: Neural network models:

<table>
<thead>
<tr>
<th>Hyper parameter</th>
<th>Neural network model</th>
</tr>
</thead>
<tbody>
<tr>
<td>S Seed episodes</td>
<td>( p_0(s_t</td>
</tr>
<tr>
<td>C Collect interval</td>
<td>( q_0(s_t</td>
</tr>
<tr>
<td>B Batch size</td>
<td>( q_0(r_t</td>
</tr>
<tr>
<td>L Sequence length</td>
<td>( q_\phi(a_t</td>
</tr>
<tr>
<td>H Imagination horizon</td>
<td>( v_\xi(s_t) ) Value model</td>
</tr>
</tbody>
</table>

1. Initialize dataset \( \mathcal{D} \) with \( S \) random seed episodes.
2. Initialize neural network parameters \( \theta, \phi, \xi \) randomly.

while not converged do

for update step \( s = 1..C \) do

// Dynamics learning

5. Draw \( B \) sequences \( \{(a_t, o_t, r_t)\}_{t=k}^{k+L} \sim \mathcal{D} \) from the dataset.
6. Compute model states \( s_t \sim p_0(s_t | s_{t-1}, a_{t-1}, o_t) \).
7. Update \( \theta \) to predict rewards using representation learning.

// Behavior learning

8. Imagine trajectories \( \{(s_t, a_t)\}_{t=k+L}^{t+H} \) from each \( s_t \) using action model.
9. Predict rewards \( \mathbb{E}(q_0(r_t | s_t)) \) and values \( v_\xi(s_t) \).
10. Compute value estimates \( \mathcal{V}_\lambda(s_t) \) via Equation 6.
11. Update \( \phi \) according to maximize \( \sum_{t=k}^{t+H} \mathcal{V}_\lambda(s_t) \) by gradient ascent.
12. Update \( \xi \) to minimize \( \frac{1}{2} \mathbb{E}[v_\xi(s_t) - \mathcal{V}_\lambda(s_t)]^2 \) by gradient descent.

// Environment interaction

13. \( o_1 \leftarrow \text{env.reset()} \)

for time step \( t = 1..T \) do

15. Compute model state \( s_t \sim p_0(s_t | s_{t-1}, a_{t-1}, o_t) \) from history.
16. Compute action \( a_t \sim q_\phi(a_t | s_t) \) with the action model.
17. Add exploration noise to action.
18. \( r_t, o_{t+1} \leftarrow \text{env.step}(a_t). \)

20. Add experience to dataset \( \mathcal{D} \leftarrow \mathcal{D} \cup \{(o_t, a_t, r_t)_{t=1}^T\}. \)

B Hyper Parameters

Model components We use the convolutional encoder and decoder networks from Ha and Schmidhuber (2018), the RSSM of Hafner et al. (2019), and implement all other functions as three dense layers of size 300 with ELU activations (Clevert et al., 2015). Distributions in latent space are 30-dimensional diagonal Gaussians. The action model outputs an unconstrained mean and softplus standard deviation for the Normal distribution that is then transformed using tanh.

Learning updates We draw batches of 50 sequences of length 50 to train the world model, value model, and action model models using Adam (Kingma and Ba, 2014) with learning rates \( 10^{-3}, 3 \times 10^{-4}, 3 \times 10^{-4}, \) respectively. We do not scale the KL regularizers (\( \beta = 1 \)) but clip them below 3 free nats as in PlaNet. The imagination horizon is \( H = 20 \) and the same trajectories are used to update both action and value models. We use a slow moving value network that is updated every 100 gradient steps to compute the \( \mathcal{V}_\lambda \) value estimates with \( \gamma = 0.99 \) and \( \lambda = 0.95 \).

Environment interaction The dataset is initialized with \( C = 5 \) episodes collected using random actions. We iterate between 100 training steps and collecting 1 episode by executing the predicted mode action with Normal(0, 0.3) exploration noise. Instead of manually selecting the action repeat for each environment as in Hafner et al. (2019) and Lee et al. (2019), we fix it to 2 for all environments. See Figure 11 for an assessment of the robustness to different action repeat values.
\section{Derivations}

The information bottleneck objective defined in Equation 9 for latent dynamics models is,

\begin{equation}
J_{\text{INFO}} \triangleq \text{I}(s_{1:T} \mid (o_{1:T}, r_{1:T}) \mid a_{1:T}) - \beta \text{I}(s_{1:T}, i_{1:T} \mid a_{1:T})
\end{equation}

For the generative objective, we lower bound the first term using the non-negativity of the KL divergence and drop the marginal data probability as it does not depend on the representation model,

\begin{align}
\text{I}(s_{1:T}; (o_{1:T}, r_{1:T}) \mid a_{1:T}) & = \mathbb{E}_{p(o_{1:T}, r_{1:T}, s_{1:T}, a_{1:T})} \left( \sum_t \ln p(o_{1:T}, r_{1:T} \mid s_{1:T}, a_{1:T}) - \ln p(o_{1:T}, r_{1:T} \mid a_{1:T}) \right) \\
& \leq \mathbb{E} \left( \sum_t \ln p(o_{1:T}, r_{1:T} \mid s_{1:T}, a_{1:T}) \right) \\
& \geq \mathbb{E} \left( \sum_t \ln q(o_t \mid s_t) - \sum_{t'} q(s_t \mid s_{t'}) + \ln q(r_t \mid s_t) \right).
\end{align}

For the contrastive objective, we subtract the constant marginal probability of the data under the variational encoder, apply Bayes rule, and use the InfoNCE mini-batch bound (Poole et al., 2019),

\begin{align}
\text{E} \left( \ln q(o_t \mid s_t) + \ln q(r_t \mid s_t) \right) & \leq \mathbb{E} \left( \ln q(o_t \mid s_t) - \ln q(o_t) + \ln q(r_t \mid s_t) \right) \\
& = \mathbb{E} \left( \ln q(s_t \mid o_t) - \ln q(s_t) + \ln q(r_t \mid s_t) \right) \\
& \geq \mathbb{E} \left( \ln q(s_t \mid o_t) - \sum_{t'} q(s_t \mid s_{t'}) + \ln q(r_t \mid s_t) \right).
\end{align}

For the second term, we use the non-negativity of the KL divergence to obtain an upper bound,

\begin{align}
\text{I}(s_{1:T}; i_{1:T} \mid a_{1:T}) & = \mathbb{E}_{p(o_{1:T}, r_{1:T}, s_{1:T}, a_{1:T}, i_{1:T})} \left( \sum_t \ln p(s_t \mid s_{t-1}, a_{t-1}, i_t) - \ln p(s_t \mid s_{t-1}, a_{t-1}) \right) \\
& = \mathbb{E} \left( \sum_t \ln p(s_t \mid s_{t-1}, a_{t-1}, o_t) - \ln p(s_t \mid s_{t-1}, a_{t-1}) \right) \\
& \leq \mathbb{E} \left( \sum_t \ln p(s_t \mid s_{t-1}, a_{t-1}, o_t) \right) - \ln q(s_t \mid s_{t-1}, a_{t-1}) \\
& = \mathbb{E} \left( \sum_t \text{KL} \left( p(s_t \mid s_{t-1}, a_{t-1}, o_t) \parallel q(s_t \mid s_{t-1}, a_{t-1}) \right) \right).
\end{align}

This lower bounds the objective.
D Behavior Learning

Figure 9: Comparison of action selection schemes on the continuous control tasks of the DeepMind Control Suite from pixel inputs. The lines show mean scores over environment steps and the shaded areas show the standard deviation across 3 seeds. We compare Dreamer that learns both actions and values in imagination, to only learning actions in imagination, and to online planning using CEM without policy learning. The baselines include the top model-free algorithm D4PG, the common A3C agent, and the hybrid SLAC agent.


E REPRESENTATION LEARNING

Figure 10: Comparison of representation learning methods for Dreamer. The lines show mean scores and the shaded areas show the standard deviation across 3 seeds. We compare generating both images and rewards, generating rewards and using a contrastive loss to learn about the images, and only predicting rewards. Image reconstruction provides the best learning signal across most of the tasks.
Figure 11: Robustness of Dreamer to different control frequencies. Reinforcement learning methods can be sensitive to this hyper parameter, which could be amplified when learning dynamics models at the control frequency of the environment. For this experiment, we train Dreamer with different amounts of action repeat. The areas show one standard deviation across 2 seeds. We find that a value of $R = 2$ works well across the majority of tasks.