Dreamer

가짜연구소 11기 러너
김민성

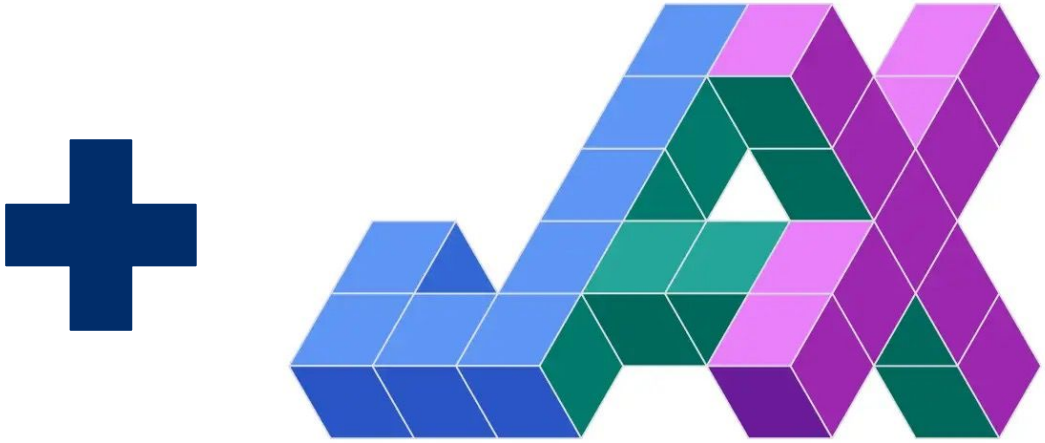
Contents

01

Introduction

- purpose of study
- research summary
- term definition

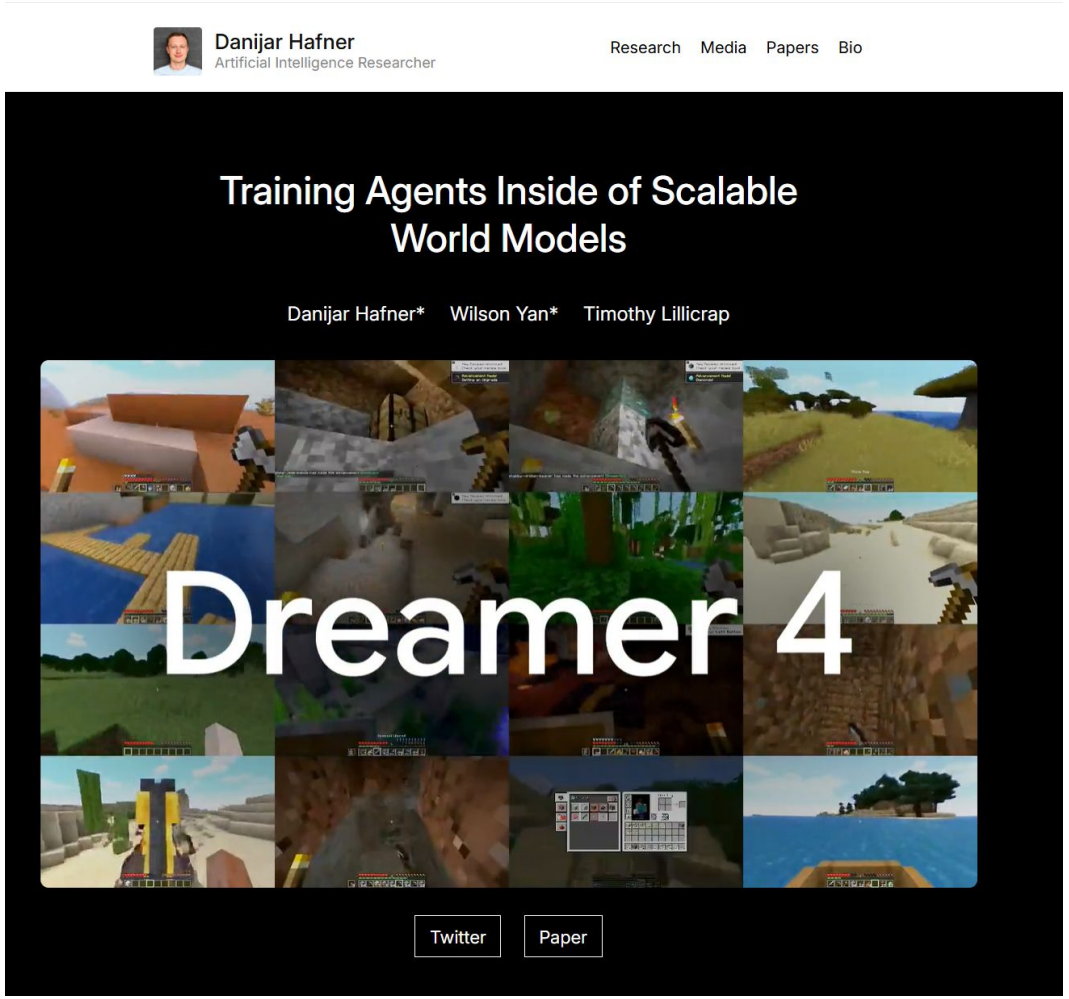
01. 개요- 목표(1차)



- **JAX 기반 RL 최적화 프로젝트 中 DreamerV3 기반 접근 제안**
  - **현 제안 방식 : DreamerV3 기반 Transformer -> NinjaX 통합**
  - **RSSM의 Robustness와 Numerical Method**
  - **히스토리 ?: PlaNet -> Dreamer V1 -> V2 -> V3 -> TransDreamerv3**
  - **최종 목표 : Transformer 기반의 TransDreamerV3(TSSM) + equinox||ninjax||evojax 활용 구현 및 성능 검증**

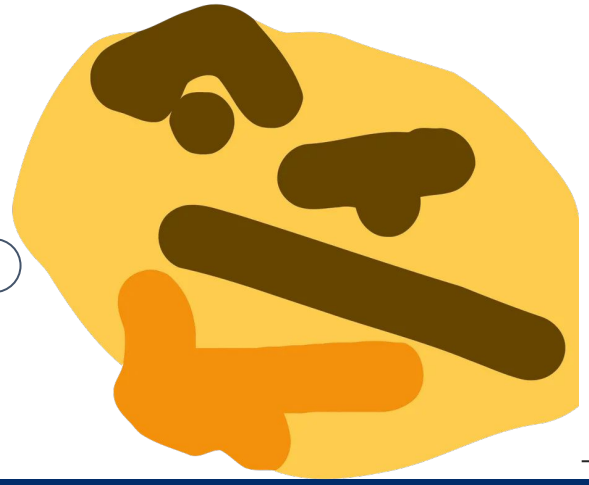
01. 개요- 목표


마참내!


Danijar Hafner
Artificial Intelligence Researcher
Research Media Papers Bio
Training Agents Inside of Scalable World Models
Danijar Hafner* Wilson Yan* Timothy Lillicrap
Dreamer 4
Twitter
Paper


즐겁다

01. 개요 - Dreamer?



02

Paper Review

- Dreamer 시리즈 : 논문 별 주요 발전 흐름
- research summary
- term definition

02. Dreamer 시리즈 : 논문 별 주요 발전 흐름

D. Hafner *et al.*, “Learning Latent Dynamics for Planning from Pixels,” June 04, 2019, *arXiv*: arXiv:1811.04551. doi: 10.48550/arXiv.1811.04551.

D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, “Dream to Control: Learning Behaviors by Latent Imagination,” Mar. 17, 2020, *arXiv*: arXiv:1912.01603. doi: 10.48550/arXiv.1912.01603.

D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba, “Mastering Atari with Discrete World Models,” Feb. 12, 2022, *arXiv*: arXiv:2010.02193. doi: 10.48550/arXiv.2010.02193.

D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, “Mastering Diverse Domains through World Models,” Apr. 17, 2024, *arXiv*: arXiv:2301.04104. doi: 10.48550/arXiv.2301.04104.

D. Hafner, W. Yan, and T. Lillicrap, “Training Agents Inside of Scalable World Models,” Sept. 29, 2025, *arXiv*: arXiv:2509.24527. doi: 10.48550/arXiv.2509.24527.

02. RSSM : PlaNet('2019)

Learning Latent Dynamics for Planning from Pixels

Danijar Hafner [1 2] Timothy Lillicrap [3] Ian Fischer [4] Ruben Villegas [1 5]
David Ha [1] Honglak Lee [1] James Davidson [1]

Abstract

Planning has been very successful for control tasks with known environment dynamics. To leverage planning in unknown environments, the agent needs to learn the dynamics from interactions with the world. However, learning dynamics models that are accurate enough for planning has been a long-standing challenge, especially in image-based domains. We propose the Deep Planning Network (PlaNet), a purely model-based agent that learns the environment dynamics from images and chooses actions through fast online planning in latent space. To achieve high performance, the dynamics model must accurately predict the rewards ahead for multiple time steps. We approach this using a latent dynamics model with both deterministic and stochastic transition components. Moreover, we propose a multi-step variational inference objective that we name latent overshooting. Using only pixel observations, our agent solves continuous control tasks with contact dynamics, partial observability, and sparse rewards, which exceed the difficulty of tasks that were previously solved by planning with learned models. PlaNet uses substantially fewer episodes and reaches final performance close to and sometimes higher than strong model-free algorithms.

1. Introduction

Planning is a natural and powerful approach to decision making problems with known dynamics, such as game playing and simulated robot control (Tassa et al., 2012; Silver et al., 2017; Moravčík et al., 2017). To plan in unknown environments, the agent needs to learn the dynamics from experience. Learning dynamics models that are accurate enough for planning has been a long-standing challenge. Key difficulties include model inaccuracies, accumulating errors of multi-step predictions, failure to capture multiple possible futures, and overconfident predictions outside of the training distribution.

Planning using learned models offers several benefits over model-free reinforcement learning. First, model-based planning can be more data efficient because it leverages a richer training signal and does not require propagating rewards through Bellman backups. Moreover, planning carries the promise of increasing performance just by increasing the computational budget for searching for actions, as shown by Silver et al. (2017). Finally, learned dynamics can be independent of any specific task and thus have the potential to transfer well to other tasks in the environment.

Recent work has shown promise in learning the dynamics of simple low-dimensional environments (Deisenroth & Rasmussen, 2011; Gal et al., 2016; Amos et al., 2018; Chua et al., 2018; Henaff et al., 2018). However, these approaches typically assume access to the underlying state of the world and the reward function, which may not be available in practice. In high-dimensional environments, we would like to learn the dynamics in a compact latent space to enable fast planning. The success of such latent models has previously been limited to simple tasks such as balancing cartpoles and controlling 2-link arms from dense rewards (Watter et al., 2015; Banijamali et al., 2017).

In this paper, we propose the Deep Planning Network (PlaNet), a model-based agent that learns the environment dynamics from pixels and chooses actions through online planning in a compact latent space. To learn the dynamics, we use a transition model with both stochastic and deterministic components. Moreover, we experiment with a novel generalized variational objective that encourages multi-step predictions. PlaNet solves continuous control tasks from pixels that are more difficult than those previously solved by planning with learned models.

Key contributions of this work are summarized as follows:

- **Planning in latent spaces** We solve a variety of tasks from the DeepMind control suite, shown in Figure 1, by learning a dynamics model and efficiently planning in

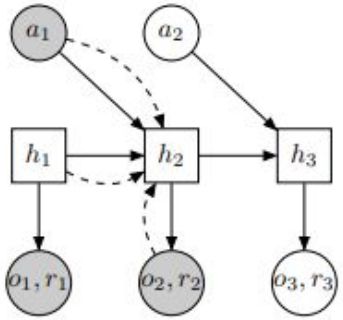
[1]Google Brain [2]University of Toronto [3]DeepMind [4]Google Research [5]University of Michigan. Correspondence to: Danijar Hafner <mail@danijar.com>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

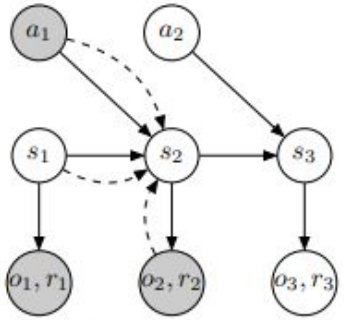
- RSSM = RNN + SSM(State Space Model) 결합
- 처음에는 PlaNet이라고 명명했으나, 본 구조를 RSSM으로 명명함
- 핵심: deterministic path (h_t) + stochastic path (s_t)
- 잠재 공간에서 계획(planning)하여 고차원 이미지 문제 해결

D. Hafner *et al.*, "Learning Latent Dynamics for Planning from Pixels," June 04, 2019, *arXiv*: arXiv:1811.04551. doi: 10.48550/arXiv.1811.04551.

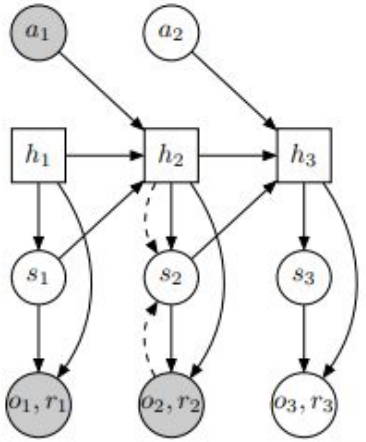
02. RSSM : PlaNet('2019)



(a) Deterministic model (RNN)



(b) Stochastic model (SSM)



(c) Recurrent state-space model (RSSM)

we name recurrent state-space model (RSSM),

$$
\begin{aligned}
&\text{Deterministic state model:} && h_t = f(h_{t-1}, s_{t-1}, a_{t-1}) \\
&\text{Stochastic state model:} && s_t \sim p(s_t \mid h_t) \\
&\text{Observation model:} && o_t \sim p(o_t \mid h_t, s_t) \\
&\text{Reward model:} && r_t \sim p(r_t \mid h_t, s_t),
\end{aligned}
\tag{4}
$$

Hafner, Danijar, Timothy Lillicrap, Ian Fischer, et al. "Learning Latent Dynamics for Planning from Pixels." arXiv:1811.04551. Preprint, arXiv, June 4, 2019. https://doi.org/10.48550/arXiv.1811.04551.

02. RSSM : PlaNet('2019)

$$
\begin{aligned}
\ln p(o_{1:T} \mid a_{1:T}) &\triangleq \ln \mathrm{E}_{p(s_{1:T}|a_{1:T})}\left[\prod_{t=1}^{T} p(o_t \mid s_t)\right] \\
&= \ln \mathrm{E}_{q(s_{1:T}|o_{1:T},a_{1:T})}\left[\prod_{t=1}^{T} p(o_t \mid s_t)p(s_t \mid s_{t-1}, a_{t-1})/q(s_t \mid o_{\leq t}, a_{<t})\right] \\
&\geq \mathrm{E}_{q(s_{1:T}|o_{1:T},a_{1:T})}\left[\sum_{t=1}^{T} \ln p(o_t \mid s_t) + \ln p(s_t \mid s_{t-1}, a_{t-1}) - \ln q(s_t \mid o_{\leq t}, a_{<t})\right] \\
&= \sum_{t=1}^{T} \Bigg(\underbrace{\mathrm{E}_{q(s_t|o_{\leq t},a_{<t})}[\ln p(o_t \mid s_t)]}_{\text{reconstruction}} - \underbrace{\mathrm{E}_{q(s_{t-1|o_{\leq t-1},a_{<a-1}})}\left[\mathrm{KL}[q(s_t \mid o_{\leq t}, a_{<t}) \parallel p(s_t \mid s_{t-1}, a_{t-1})]\right]}_{\text{complexity}} \Bigg).
\end{aligned}
$$

02. Dreamerv1(‘2020)

Published as a conference paper at ICLR 2020

DREAM TO CONTROL: LEARNING BEHAVIORS BY LATENT IMAGINATION

Danijar Hafner *
University of Toronto
Google Brain

Timothy Lillicrap
DeepMind

Jimmy Ba
University of Toronto

Mohammad Norouzi
Google Brain

Abstract

Learned world models summarize an agent’s experience to facilitate learning complex behaviors. 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 from images purely by latent imagination. We efficiently learn behaviors by propagating analytic gradients of learned state values back 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.

- PlaNet과의 차이점:
  1. PlaNet: CEM online planning (매 step 최적화)
  2. Dreamer V1: Policy network 학습 (latent imagination)
- 핵심 혁신: Imagined Trajectories로 Actor-Critic 학습
  1. World model에서 가상 궤적 생성
  2. 가상 경험으로 policy & value network 학습
- 주요 구성요소:
  1. World Model (RSSM - PlaNet과 동일)
  2. Actor Network (policy)
  3. Critic Network (value function)

D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, “Dream to Control: Learning Behaviors by Latent Imagination,” Mar. 17, 2020, *arXiv*: arXiv:1912.01603. doi: 10.48550/arXiv.1912.01603.

02. Dreamerv1(‘2020)

- Action & Value function

Action model: $a_\tau \sim q_\phi(a_\tau \mid s_\tau)$

Value model: $v_\psi(s_\tau) \approx \mathrm{E}_{q(\cdot|s_\tau)}\big(\sum_{\tau=t}^{t+H} \gamma^{\tau-t} r_\tau\big).$

- Reparameterization

$$a_\tau = \tanh\big(\mu_\phi(s_\tau) + \sigma_\phi(s_\tau)\,\epsilon\big), \quad \epsilon \sim \mathrm{Normal}(0, \mathbb{I}).$$

(a) Learn dynamics from experience (b) Learn behavior in imagination (c) Act in the environment

Figure 3: Components of Dreamer. (a) From the dataset of past experience, the agent learns to encode observations and actions into compact latent states (●), for example via reconstruction, and predicts environment rewards (●). (b) In the compact latent space, Dreamer predicts state values (🏆) and actions (🕹) that maximize future value predictions by propagating gradients back through imagined trajectories. (c) The agent encodes the history of the episode to compute the current model state and predict the next action to execute in the environment. See Algorithm 1 for pseudo code of the agent.

D. P. Kingma and M. Welling, “Auto-Encoding Variational Bayes,” Dec. 10, 2022, *arXiv*: arXiv:1312.6114. doi: 10.48550/arXiv.1312.6114.

D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, “Dream to Control: Learning Behaviors by Latent Imagination,” Mar. 17, 2020, *arXiv*: arXiv:1912.01603. doi: 10.48550/arXiv.1912.01603.

02. Dreamerv1('2020)

Algorithm 1: Dreamer

Initialize dataset $\mathcal{D}$ with S random seed episodes.
Initialize neural network parameters θ, ϕ, ψ randomly.
while *not converged* **do**
  for *update step* $c = 1..C$ **do**
    `// Dynamics learning`
    Draw B data sequences $\{(a_t, o_t, r_t)\}_{t=k}^{k+L} \sim \mathcal{D}$.
    Compute model states $s_t \sim p_\theta(s_t \mid s_{t-1}, a_{t-1}, o_t)$.
    Update θ using representation learning.
    `// Behavior learning`
    Imagine trajectories $\{(s_\tau, a_\tau)\}_{\tau=t}^{t+H}$ from each s_t.
    Predict rewards $\mathrm{E}\big(q_\theta(r_\tau \mid s_\tau)\big)$ and values $v_\psi(s_\tau)$.
    Compute value estimates $\mathrm{V}_\lambda(s_\tau)$ via Equation 6.
    Update $\phi \leftarrow \phi + \alpha \nabla_\phi \sum_{\tau=t}^{t+H} \mathrm{V}_\lambda(s_\tau)$.
    Update $\psi \leftarrow \psi - \alpha \nabla_\psi \sum_{\tau=t}^{t+H} \frac{1}{2} \big\| v_\psi(s_\tau) - \mathrm{V}_\lambda(s_\tau) \big\|^2$.
  `// Environment interaction`
  $o_1 \leftarrow$ `env.reset()`
  for *time step* $t = 1..T$ **do**
    Compute $s_t \sim p_\theta(s_t \mid s_{t-1}, a_{t-1}, o_t)$ from history.
    Compute $a_t \sim q_\phi(a_t \mid s_t)$ with the action model.
    Add exploration noise to action.
    $r_t, o_{t+1} \leftarrow$ `env.step(`a_t`)`.
  Add experience to dataset $\mathcal{D} \leftarrow \mathcal{D} \cup \{(o_t, a_t, r_t)_{t=1}^{T}\}$.

Model components

Representation	$p_\theta(s_t \mid s_{t-1}, a_{t-1}, o_t)$
Transition	$q_\theta(s_t \mid s_{t-1}, a_{t-1})$
Reward	$q_\theta(r_t \mid s_t)$
Action	$q_\phi(a_t \mid s_t)$
Value	$v_\psi(s_t)$

Hyper parameters

Seed episodes	S
Collect interval	C
Batch size	B
Sequence length	L
Imagination horizon	H
Learning rate	α

D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," Dec. 10, 2022, *arXiv*: arXiv:1312.6114. doi: 10.48550/arXiv.1312.6114.

D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to Control: Learning Behaviors by Latent Imagination," Mar. 17, 2020, *arXiv*: arXiv:1912.01603. doi: 10.48550/arXiv.1912.01603.

02. Dreamerv1('2020)

Value estimation To learn the action and value models, we need to estimate the state values of imagined trajectories $\{s_\tau, a_\tau, r_\tau\}_{\tau=t}^{t+H}$. 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),

$$\mathrm{V}_{\mathrm{R}}(s_\tau) \doteq \mathrm{E}_{q_\theta, q_\phi}\left(\sum_{n=\tau}^{t+H} r_n\right), \tag{4}$$

$$\mathrm{V}_{\mathrm{N}}^{k}(s_\tau) \doteq \mathrm{E}_{q_\theta, q_\phi}\left(\sum_{n=\tau}^{h-1} \gamma^{n-\tau} r_n + \gamma^{h-\tau} v_\psi(s_h)\right) \quad \text{with} \quad h = \min(\tau + k, t + H), \tag{5}$$

$$\mathrm{V}_\lambda(s_\tau) \doteq (1-\lambda) \sum_{n=1}^{H-1} \lambda^{n-1} \mathrm{V}_{\mathrm{N}}^{n}(s_\tau) + \lambda^{H-1} \mathrm{V}_{\mathrm{N}}^{H}(s_\tau), \tag{6}$$

Learning objective To update the action and value models, we first compute the value estimates $\mathrm{V}_\lambda(s_\tau)$ for all states s_τ along the imagined trajectories. The objective for the action model $q_\phi(a_\tau \mid s_\tau)$ is to predict actions that result in state trajectories with high value estimates. The objective for the value model $v_\psi(s_\tau)$, in turn, is to regress the value estimates,

$$\max_\phi \mathrm{E}_{q_\theta, q_\phi}\left(\sum_{\tau=t}^{t+H} \mathrm{V}_\lambda(s_\tau)\right), \tag{7}$$

$$\min_\psi \mathrm{E}_{q_\theta, q_\phi}\left(\sum_{\tau=t}^{t+H} \frac{1}{2}\left\| v_\psi(s_\tau) - \mathrm{V}_\lambda(s_\tau)\right\|^2\right). \tag{8}$$

D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to Control: Learning Behaviors by Latent Imagination," Mar. 17, 2020, *arXiv*: arXiv:1912.01603. doi: 10.48550/arXiv.1912.01603.

02. Dreamerv2('2022)

MASTERING ATARI WITH DISCRETE WORLD MODELS

Danijar Hafner * Google Research
Timothy Lillicrap DeepMind
Mohammad Norouzi Google Research
Jimmy Ba University of Toronto

ABSTRACT

Intelligent agents need to generalize from past experience to achieve goals in complex environments. World models facilitate such generalization and allow learning behaviors from imagined outcomes to increase sample-efficiency. While learning world models from image inputs has recently become feasible for some tasks, modeling Atari games accurately enough to derive successful behaviors has remained an open challenge for many years. We introduce DreamerV2, a reinforcement learning agent that learns behaviors purely from predictions in the compact latent space of a powerful world model. The world model uses discrete representations and is trained separately from the policy. DreamerV2 constitutes the first agent that achieves human-level performance on the Atari benchmark of 55 tasks by learning behaviors inside a separately trained world model. With the same computational budget and wall-clock time, Dreamer V2 reaches 200M frames and surpasses the final performance of the top single-GPU agents IQN and Rainbow. DreamerV2 is also applicable to tasks with continuous actions, where it learns an accurate world model of a complex humanoid robot and solves stand-up and walking from only pixel inputs.

D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba, "Mastering Atari with Discrete World Models," Feb. 12, 2022, *arXiv*: arXiv:2010.02193. doi: 10.48550/arXiv.2010.02193.

02. Dreamerv2(‘2022)

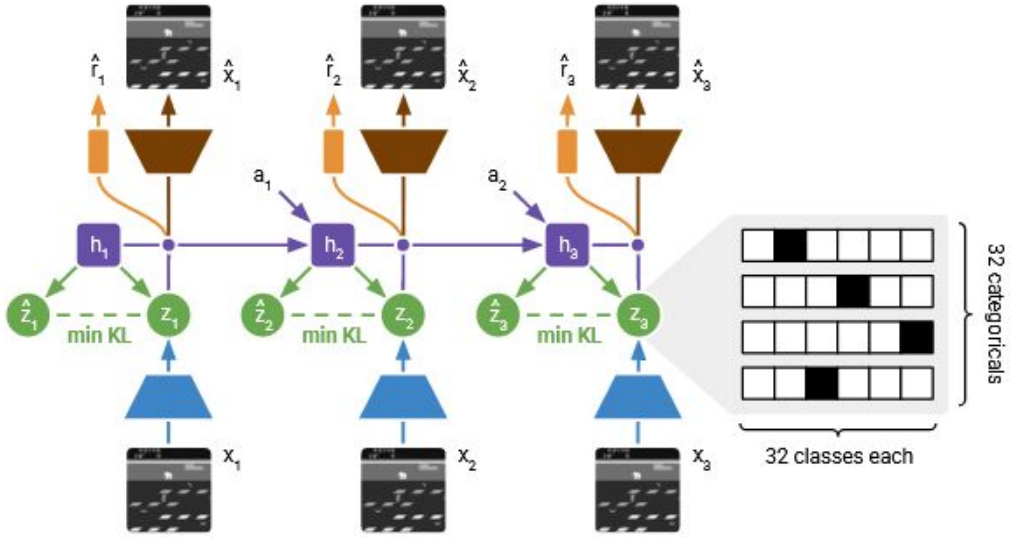


Figure 2: World Model Learning. The training sequence of images x_t is encoded using the CNN. The RSSM uses a sequence of deterministic recurrent states h_t. At each step, it computes a posterior stochastic state z_t that incorporates information about the current image x_t, as well as a prior stochastic state $\hat{z}_t$ that tries to predict the posterior without access to the current image. Unlike in PlaNet and DreamerV1, the stochastic state of DreamerV2 is a vector of multiple categorical variables. The learned prior is used for imagination, as shown in Figure 3. The KL loss both trains the prior and regularizes how much information the posterior incorporates from the image. The regularization increases robustness to novel inputs. It also encourages reusing existing information from past steps to predict rewards and reconstruct images, thus learning long-term dependencies.

$$\text{Actor:} \quad \hat{a}_t \sim p_\psi(\hat{a}_t \mid \hat{z}_t)$$

$$\text{Critic:} \quad v_\xi(\hat{z}_t) \approx \mathrm{E}_{p_\phi, p_\psi}\left[\textstyle\sum_{\tau \geq t} \hat{\gamma}^{\tau - t} \hat{r}_\tau\right].$$

$$V_t^\lambda \doteq \hat{r}_t + \hat{\gamma}_t \begin{cases} (1-\lambda) v_\xi(\hat{z}_{t+1}) + \lambda V_{t+1}^\lambda & \text{if} \quad t < H, \\ v_\xi(\hat{z}_H) & \text{if} \quad t = H. \end{cases}$$

Algorithm 1: Straight-Through Gradients with Automatic Differentiation

```
sample = one_hot(draw(logits))      # sample has no gradient
probs  = softmax(logits)            # want gradient of this
sample = sample + probs - stop_grad(probs)  # has gradient of probs
```

D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba, “Mastering Atari with Discrete World Models,” Feb. 12, 2022, *arXiv*: arXiv:2010.02193. doi: 10.48550/arXiv.2010.02193.

02. Dreamerv2(‘2022)

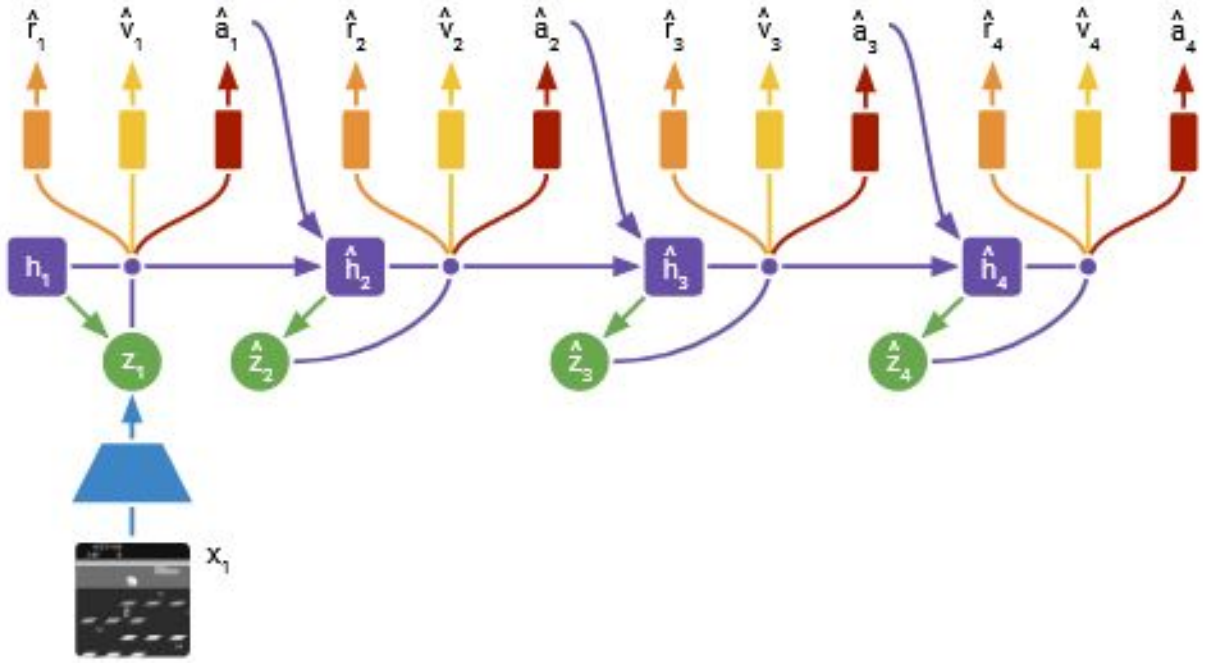


Figure 3: Actor Critic Learning. The world model learned in Figure 2 is used for learning a polic from trajectories imagined in the compact latent space. The trajectories start from posterior state computed during model training and predict forward by sampling actions from the actor networ The critic network predicts the expected sum of future rewards for each state. The critic uses tempora difference learning on the imagined rewards. The actor is trained to maximize the critic predictio via reinforce gradients, straight-through gradients of the world model, or a combination of them.

D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba, “Mastering Atari with Discrete World Models,” Feb. 12, 2022, *arXiv*: arXiv:2010.02193. doi: 10.48550/arXiv.2010.02193.

02. Dreamerv3('2024)

Mastering Diverse Domains through World Models

Danijar Hafner[1,2] Jurgis Pasukonis[1] Jimmy Ba[2] Timothy Lillicrap[1]

Abstract

Developing a general algorithm that learns to solve tasks across a wide range of applications has been a fundamental challenge in artificial intelligence. Although current reinforcement learning algorithms can be readily applied to tasks similar to what they have been developed for, configuring them for new application domains requires significant human expertise and experimentation. We present DreamerV3, a general algorithm that outperforms specialized methods across over 150 diverse tasks, with a single configuration. Dreamer learns a model of the environment and improves its behavior by imagining future scenarios. Robustness techniques based on normalization, balancing, and transformations enable stable learning across domains. Applied out of the box, Dreamer is the first algorithm to collect diamonds in Minecraft from scratch without human data or curricula. This achievement has been posed as a significant challenge in artificial intelligence that requires exploring farsighted strategies from pixels and sparse rewards in an open world. Our work allows solving challenging control problems without extensive experimentation, making reinforcement learning broadly applicable.

D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering Diverse Domains through World Models," Apr. 17, 2024, *arXiv*: arXiv:2301.04104. doi: 10.48550/arXiv.2301.04104.

02. Dreamerv3('2024)

$$\text{RSSM}\begin{cases} \text{Sequence model:} & h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1}) \\ \text{Encoder:} & z_t \sim q_\phi(z_t \mid h_t, x_t) \\ \text{Dynamics predictor:} & \hat{z}_t \sim p_\phi(\hat{z}_t \mid h_t) \\ \text{Reward predictor:} & \hat{r}_t \sim p_\phi(\hat{r}_t \mid h_t, z_t) \\ \text{Continue predictor:} & \hat{c}_t \sim p_\phi(\hat{c}_t \mid h_t, z_t) \\ \text{Decoder:} & \hat{x}_t \sim p_\phi(\hat{x}_t \mid h_t, z_t) \end{cases}$$

$$\mathcal{L}_{\text{pred}}(\phi) \doteq -\ln p_\phi(x_t \mid z_t, h_t) - \ln p_\phi(r_t \mid z_t, h_t) - \ln p_\phi(c_t \mid z_t, h_t)$$

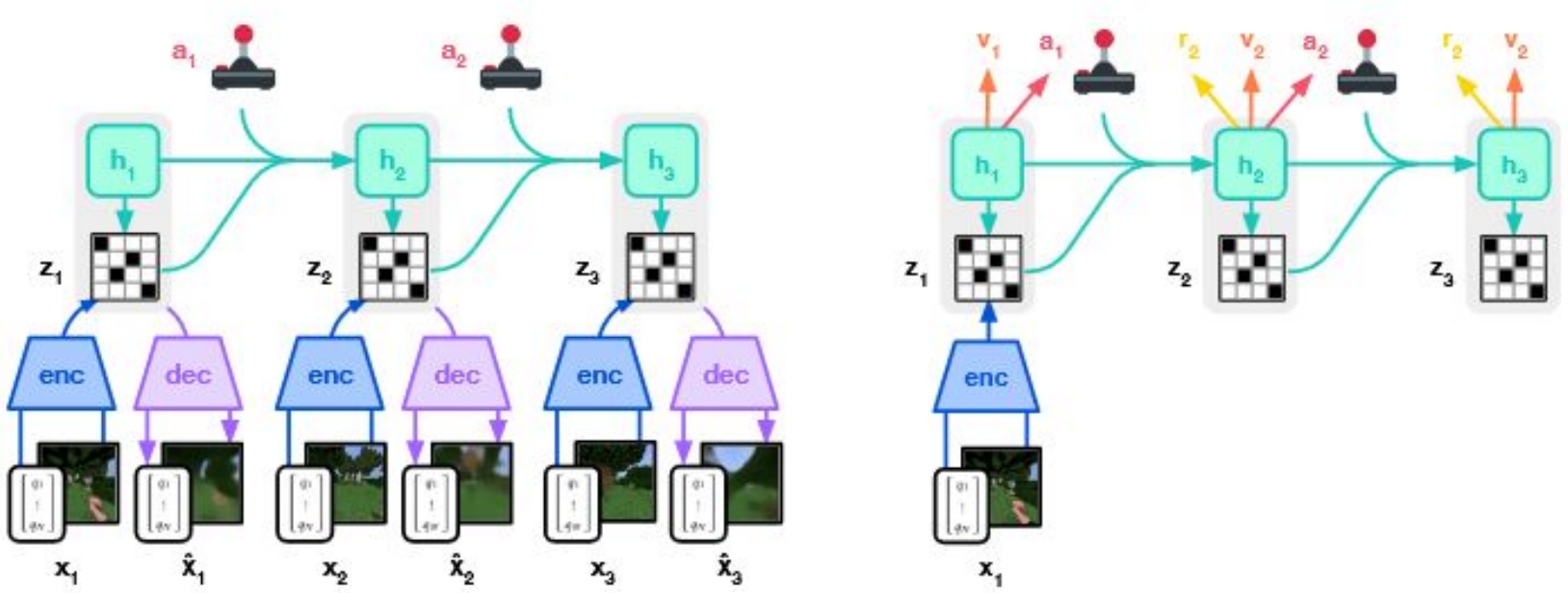
$$\mathcal{L}_{\text{dyn}}(\phi) \doteq \max\big(1, \mathrm{KL}\big[\,\mathrm{sg}(q_\phi(z_t \mid h_t, x_t)) \,\big\|\, p_\phi(z_t \mid h_t)\,\big]\big)$$

$$\mathcal{L}_{\text{rep}}(\phi) \doteq \max\big(1, \mathrm{KL}\big[\,q_\phi(z_t \mid h_t, x_t) \,\big\|\, \mathrm{sg}(p_\phi(z_t \mid h_t))\big]\big)$$

$$\mathcal{L}(\phi) \doteq \mathrm{E}_{q_\phi}\Big[\textstyle\sum_{t=1}^{T}\big(\beta_{\text{pred}}\mathcal{L}_{\text{pred}}(\phi) + \beta_{\text{dyn}}\mathcal{L}_{\text{dyn}}(\phi) + \beta_{\text{rep}}\mathcal{L}_{\text{rep}}(\phi)\big)\Big].$$

D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering Diverse Domains through World Models," Apr. 17, 2024, *arXiv*: arXiv:2301.04104. doi: 10.48550/arXiv.2301.04104.

02. Dreamerv3('2024)



(a) World Model Learning (b) Actor Critic Learning

Figure 3: Training process of Dreamer. The world model encodes sensory inputs into discrete representations z_t that are predicted by a sequence model with recurrent state h_t given actions a_t. The inputs are reconstructed to shape the representations. The actor and critic predict actions a_t and values v_t and learn from trajectories of abstract representations predicted by the world model.

D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering Diverse Domains through World Models," Apr. 17, 2024, *arXiv*: arXiv:2301.04104. doi: 10.48550/arXiv.2301.04104.

02. Dreamerv4('2025)

Training Agents Inside of Scalable World Models

Danijar Hafner* Wilson Yan* Timothy Lillicrap

World models learn general knowledge from videos and simulate experience for training behaviors in imagination, offering a path towards intelligent agents. However, previous world models have been unable to accurately predict object interactions in complex environments. We introduce Dreamer 4, a scalable agent that learns to solve control tasks by reinforcement learning inside of a fast and accurate world model. In the complex video game Minecraft, the world model accurately predicts object interactions and game mechanics, outperforming previous world models by a large margin. The world model achieves real-time interactive inference on a single GPU through a shortcut forcing objective and an efficient transformer architecture. Moreover, the world model learns general action conditioning from only a small amount of data, allowing it to extract the majority of its knowledge from diverse unlabeled videos. We propose the challenge of obtaining diamonds in Minecraft from only offline data, aligning with practical applications such as robotics where learning from environment interaction can be unsafe and slow. This task requires choosing sequences of over 20,000 mouse and keyboard actions from raw pixels. By learning behaviors in imagination, Dreamer 4 is the first agent to obtain diamonds in Minecraft purely from offline data, without environment interaction. Our work provides a scalable recipe for imagination training, marking a step towards intelligent agents.

- We introduce Dreamer 4, a scalable agent that learns to solve challenging control tasks by imagination training inside of a world model.
- Dreamer 4 is the first agent to collect diamonds in Minecraft from only offline data, substantially improving over OpenAI's VPT offline agent[15] despite using 100× less data.
- We introduce a high-capacity world model that achieves real-time inference on a single GPU through a shortcut forcing objective and an efficient transformer architecture.
- We show that the world model accurately predicts a wide range of object interactions and game mechanics in Minecraft, substantially outperforming previous world models.
- We show that the world model can learn from unlabeled videos and requires only a small amount of aligned data to learn action conditioning with strong generalization.
- An extensive ablation study measures the improvements of the objective and architecture.

D. Hafner, W. Yan, and T. Lillicrap, "Training Agents Inside of Scalable World Models," Sept. 29, 2025, *arXiv*: arXiv:2509.24527. doi: 10.48550/arXiv.2509.24527.

02. Dreamerv4(‘2025)

D. Hafner, W. Yan, and T. Lillicrap, “Training Agents Inside of Scalable World Models,” Sept. 29, 2025, *arXiv*: arXiv:2509.24527. doi: 10.48550/arXiv.2509.24527.

02. Dreamerv4('2025)

Model	Parameters	Resolution	Context	FPS	Success
MineWorld	1.2B	384×224	0.8s	2	—
Lucid-v1	1.1B	640×360	1.0s	44	0/16
Oasis (small)	500M	640×360	1.6s	20	0/16
Oasis (large)	—	360×360	1.6s	~5	5/16
Dreamer 4	2B	640×360	9.6s	21	**14/16**

Table 1: Comparison of Minecraft world models. Dreamer 4 is the first world model to accurately simulate a wide range of object interactions and game mechanics in Minecraft. Moreover, Dreamer 4 pushes the limits of context length compared to previous models by 6×, while maintaining real-time interactive inference. We measure the inference speed of each model on a single H100 GPU, and translate the inference speed for the proprietary large Oasis model based on public information. The Minecraft dataset is recorded at 20 FPS, matching the update rate of the game.

D. Hafner, W. Yan, and T. Lillicrap, "Training Agents Inside of Scalable World Models," Sept. 29, 2025, *arXiv*: arXiv:2509.24527. doi: 10.48550/arXiv.2509.24527.

02. Dreamerv4('2025)

Algorithm 1 Dreamer 4

Phase 1: World Model Pretraining

- Train tokenizer on videos using (5).
- Train world model on tokenized videos and optionally actions using (7).

Phase 2: Agent Finetuning

- Finetune world model with task inputs for policy and reward heads using (7) and (9).

Phase 3: Imagination Training

- Optimize policy head using (11) and value head using (10) on trajectories generated by the world model and the policy head.

Masked autoencoding We train the tokenizer using a straightforward reconstruction objective, consisting of mean squared error and LPIPS[22] loss. To simplify weighing the two loss terms, we employ loss normalization as explained later.

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{MSE}}(\theta) + 0.2\,\mathcal{L}_{\text{LPIPS}}(\theta) \tag{5}$$

$$b' = (f_\theta(\tilde{z}, \tau, \tfrac{d}{2}, a) - z_\tau)/(1-\tau) \qquad z' = \tilde{z} + b'\,\tfrac{d}{2}$$
$$b'' = (f_\theta(z', \tau + \tfrac{d}{2}, \tfrac{d}{2}, a) - z')/(1-(\tau + \tfrac{d}{2}))$$
$$\mathcal{L}(\theta) = \begin{cases} \|\hat{z}_1 - z_1\|_2^2 & \text{if } d = d_{\min} \\ (1-\tau)^2 \|(\hat{z}_1 - \tilde{z})/(1-\tau) - \text{sg}(b_1 + b_2)/2\|_2^2 & \text{else} \end{cases} \tag{7}$$

$$\mathcal{L}(\theta) = -\sum_{n=0}^{L} \ln p_\theta(a_{t+n} \mid h_t) - \sum_{n=0}^{L} \ln p_\theta(r_{t+n} \mid h_t) \tag{9}$$

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \ln p_\theta(R_t^\lambda \mid s_t) \qquad R_t^\lambda = r_t + \gamma c_t\big((1-\lambda)v_t + \lambda R_{t+1}^\lambda\big) \qquad R_T^\lambda = v_T \tag{10}$$

$$\mathcal{L}(\theta) = \frac{1-\alpha}{|\mathcal{D}^-|} \sum_{i \in \mathcal{D}^-} \ln \pi_\theta(a_i \mid s_i) - \frac{\alpha}{|\mathcal{D}^+|} \sum_{i \in \mathcal{D}^+} \ln \pi_\theta(a_i \mid s_i) + \frac{\beta}{N} \sum_{i=1}^{N} \text{KL}[\pi_\theta(a_i \mid s_i) \,\|\, \pi_{\text{prior}}] \tag{11}$$

D. Hafner, W. Yan, and T. Lillicrap, "Training Agents Inside of Scalable World Models," Sept. 29, 2025, *arXiv*: arXiv:2509.24527. doi: 10.48550/arXiv.2509.24527.

02. Dreamerv4('2025)

Masked autoencoding We train the tokenizer using a straightforward reconstruction objective, consisting of mean squared error and LPIPS[22] loss. To simplify weighing the two loss terms, we employ loss normalization as explained later.

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{MSE}}(\theta) + 0.2\,\mathcal{L}_{\text{LPIPS}}(\theta) \tag{5}$$

R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," Apr. 10, 2018, *arXiv*: arXiv:1801.03924. doi: 10.48550/arXiv.1801.03924.

D. Hafner, W. Yan, and T. Lillicrap, "Training Agents Inside of Scalable World Models," Sept. 29, 2025, *arXiv*: arXiv:2509.24527. doi: 10.48550/arXiv.2509.24527.

actions $a = \{a_t\}$, discrete signal levels $\tau = \{\tau_t\}$ and step sizes $d = \{d_t\}$, and corrupted representations $\tilde{z} = \{z_t^{(\tau_t)}\}$ as input and predicts the clean representations $z_1 = \{z_t^1\}$. Note that $t \in [1, T]$ is the sequence timestep while $\tau_t \in [0, 1]$ is the signal level at that step.

$$z_0 \sim \mathrm{N}(0, 1) \qquad z_1 \sim \mathcal{D} \qquad \tau, d \sim p(\tau, d) \qquad \tau, d \in [0, 1]^T$$
$$\hat{z}_1 = f_\theta(\tilde{z}, \tau, d, a) \qquad \tilde{z} = (1 - \tau)\, z_0 + \tau\, z_1 \tag{6}$$

Loss function

$$\mathcal{L}(\theta) = \begin{cases} \|\hat{z}_1 - z_1\|_2^2 & \text{if } d = d_{\min} \\ (1-\tau)^2 \|(\hat{z}_1 - \tilde{z})/(1-\tau) - \mathrm{sg}(b_1 + b_2)/2\|_2^2 & \text{else} \end{cases}$$

이 때,

$$b' = (f_\theta(\tilde{z}, \tau, \tfrac{d}{2}, a) - z_\tau)/(1 - \tau) \qquad z' = \tilde{z} + b' \tfrac{d}{2}$$
$$b'' = (f_\theta(z', \tau + \tfrac{d}{2}, \tfrac{d}{2}, a) - z')/(1 - (\tau + \tfrac{d}{2}))$$

D. Hafner, W. Yan, and T. Lillicrap, "Training Agents Inside of Scalable World Models," Sept. 29, 2025, *arXiv*: arXiv:2509.24527. doi: 10.48550/arXiv.2509.24527.

02. Dreamerv4('2025)

ramp loss

Low signal levels contain less learning signal, because the flow matching term degenerates to predicting the dataset mean, while the bootstrap term is generally easier to optimize because it has deterministic targets compared to the noisy flow matching term. To focus the model capacity on signal levels with the most learning signal, we propose a `ramp` loss weight that linearly increases with the signal level τ, where $\tau = 0$ corresponds to full noise and $\tau = 1$ to clean data:

$$w(\tau) = 0.9\tau + 0.1 \tag{8}$$

At inference time, the dynamics model supports different noise patterns. We sample autoregressively in time and generate the representations of each frame using the shortcut model with $K = 4$ sample steps with corresponding step size $d = 1/4$. We slightly corrupt the past inputs to the dynamics model to signal level $\tau_{\text{ctx}} = 0.1$ to make the model robust to small imperfections in its generations.

D. Hafner, W. Yan, and T. Lillicrap, "Training Agents Inside of Scalable World Models," Sept. 29, 2025, *arXiv*: arXiv:2509.24527. doi: 10.48550/arXiv.2509.24527.

02. Dreamerv4('2025)

The value head is trained to predict the discounted sum of future rewards, allowing the policy to maximize rewards beyond the imagination horizon. It uses a symexp twohot output to robustly learn across different scales of values[1]. We train the value head using temporal difference learning (TD-learning)[28] to predict λ-returns computed from the predicted rewards and values along the sequence, where $\gamma = 0.997$ is a discount factor and c_t indicates non-terminal states:

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \ln p_\theta(R_t^\lambda \mid s_t) \qquad R_t^\lambda = r_t + \gamma c_t\big((1-\lambda)v_t + \lambda R_{t+1}^\lambda\big) \qquad R_T^\lambda = v_T \tag{10}$$

Unlike previous generations of Dreamer, the policy head learns using PMPO[29], a robust reinforcement learning objective that uses the sign of the advantages $A_t = R_t^\lambda - v_t$ and ignores their magnitude. This property alleviates the need for normalizing returns or advantages and ensures equal focus on all tasks despite potentially differing return scales. PMPO balances the focus on positive and negative feedback by separately averaging a simple maximum likelihood loss over the states with positive and negative advantages, respectively. We assign all imagined states s_i across batch and time dimensions

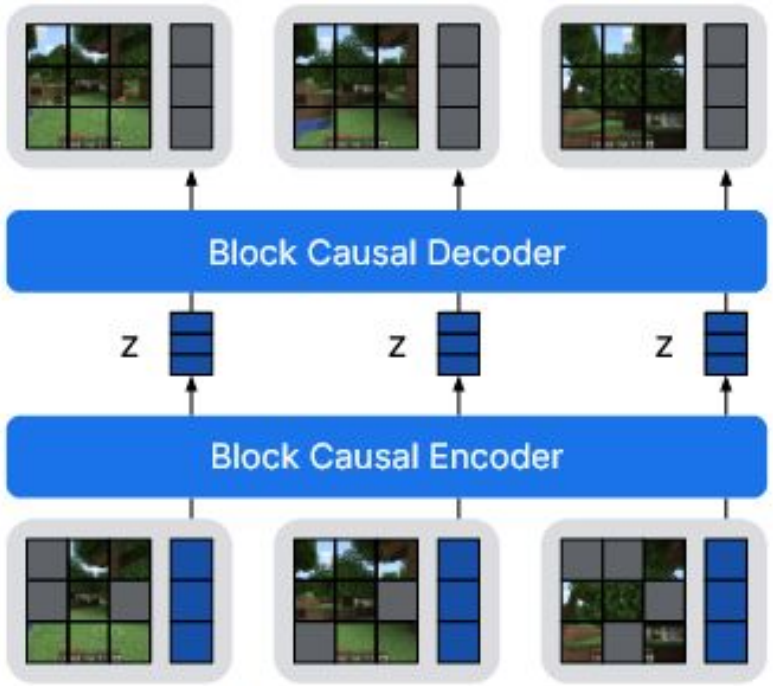
to the positive set $\mathcal{D}^+ = \{s_i \mid A_t \geq 0\}$ or the negative set $\mathcal{D}^- = \{s_i \mid A_t < 0\}$ and apply the following policy loss:

$$\mathcal{L}(\theta) = \frac{1-\alpha}{|\mathcal{D}^-|} \sum_{i \in \mathcal{D}^-} \ln \pi_\theta(a_i \mid s_i) - \frac{\alpha}{|\mathcal{D}^+|} \sum_{i \in \mathcal{D}^+} \ln \pi_\theta(a_i \mid s_i) + \frac{\beta}{N} \sum_{i=1}^{N} \mathrm{KL}[\pi_\theta(a_i \mid s_i) \,\|\, \pi_{\text{prior}}] \tag{11}$$

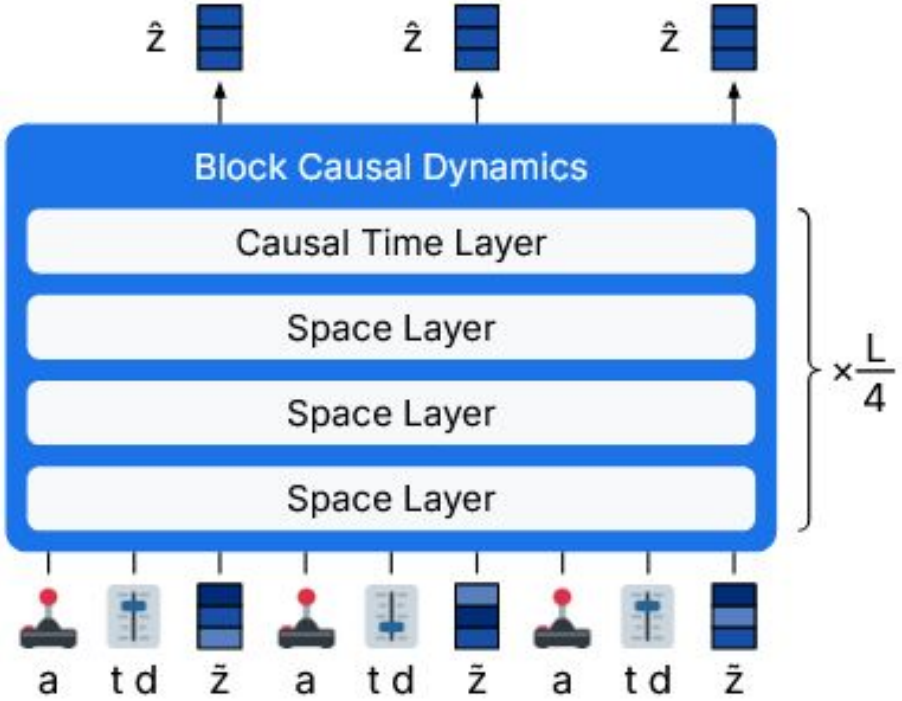
We set $\alpha = 0.5$ to balance the positive and negative sets equally and use a weaker scale of $\beta = 0.3$

D. Hafner, W. Yan, and T. Lillicrap, "Training Agents Inside of Scalable World Models," Sept. 29, 2025, *arXiv*: arXiv:2509.24527. doi: 10.48550/arXiv.2509.24527.

02. Dreamerv4('2025)



(a) Causal Tokenizer



(b) Interactive Dynamics

D. Hafner, W. Yan, and T. Lillicrap, "Training Agents Inside of Scalable World Models," Sept. 29, 2025, *arXiv*: arXiv:2509.24527. doi: 10.48550/arXiv.2509.24527.

02. Dreamerv4('2025)

Algorithm 1 Dreamer 4

Phase 1: World Model Pretraining

- Train tokenizer on videos using (5).
- Train world model on tokenized videos and optionally actions using (7).

Phase 2: Agent Finetuning

- Finetune world model with task inputs for policy and reward heads using (7) and (9).

Phase 3: Imagination Training

- Optimize policy head using (11) and value head using (10) on trajectories generated by the world model and the policy head.

Masked autoencoding We train the tokenizer using a straightforward reconstruction objective, consisting of mean squared error and LPIPS[22] loss. To simplify weighing the two loss terms, we employ loss normalization as explained later.

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{MSE}}(\theta) + 0.2\,\mathcal{L}_{\text{LPIPS}}(\theta) \tag{5}$$

$$b' = (f_\theta(\tilde{z}, \tau, \tfrac{d}{2}, a) - z_\tau)/(1-\tau) \qquad z' = \tilde{z} + b'\,\tfrac{d}{2}$$
$$b'' = (f_\theta(z', \tau + \tfrac{d}{2}, \tfrac{d}{2}, a) - z')/(1-(\tau + \tfrac{d}{2}))$$
$$\mathcal{L}(\theta) = \begin{cases} \|\hat{z}_1 - z_1\|_2^2 & \text{if } d = d_{\min} \\ (1-\tau)^2 \|(\hat{z}_1 - \tilde{z})/(1-\tau) - \text{sg}(b_1 + b_2)/2\|_2^2 & \text{else} \end{cases} \tag{7}$$

$$\mathcal{L}(\theta) = -\sum_{n=0}^{L} \ln p_\theta(a_{t+n} \mid h_t) - \sum_{n=0}^{L} \ln p_\theta(r_{t+n} \mid h_t) \tag{9}$$

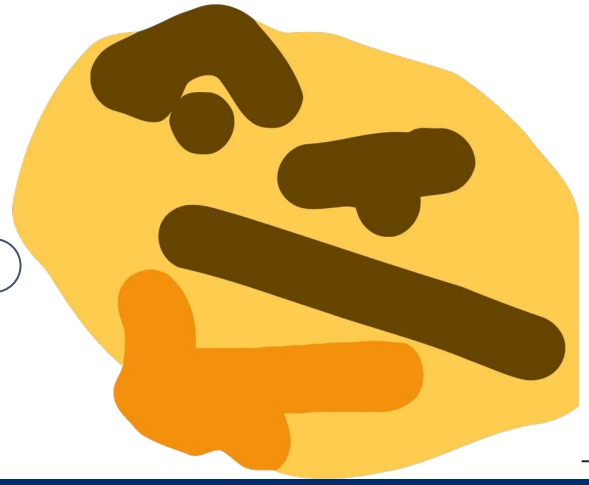
$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \ln p_\theta(R_t^\lambda \mid s_t) \qquad R_t^\lambda = r_t + \gamma c_t\big((1-\lambda)v_t + \lambda R_{t+1}^\lambda\big) \qquad R_T^\lambda = v_T \tag{10}$$

$$\mathcal{L}(\theta) = \frac{1-\alpha}{|\mathcal{D}^-|} \sum_{i \in \mathcal{D}^-} \ln \pi_\theta(a_i \mid s_i) - \frac{\alpha}{|\mathcal{D}^+|} \sum_{i \in \mathcal{D}^+} \ln \pi_\theta(a_i \mid s_i) + \frac{\beta}{N} \sum_{i=1}^{N} \text{KL}[\pi_\theta(a_i \mid s_i) \,\|\, \pi_{\text{prior}}] \tag{11}$$

D. Hafner, W. Yan, and T. Lillicrap, "Training Agents Inside of Scalable World Models," Sept. 29, 2025, *arXiv*: arXiv:2509.24527. doi: 10.48550/arXiv.2509.24527.

02. Dreamerv4(‘2025)

이게 **RL…?**



03

Challenges and Solutions

- purpose of study
- research summary
- term definition

03. 무엇을 해야하는가

1. Shortcut Forcing

Flow matching + Bootstrap objective

2. Efficient Transformer

Block-causal architecture
Space layers (every) + Time layers (every 4th)
GQA (Grouped Query Attention)

3. X-prediction

Clean representation 직접 예측
V-prediction 대신 사용
긴 rollout error accumulation 방지

4. Action Generalization

label data와 unlabeled data how to?

D. Hafner, W. Yan, and T. Lillicrap, “Training Agents Inside of Scalable World Models,” Sept. 29, 2025, *arXiv*: arXiv:2509.24527. doi: 10.48550/arXiv.2509.24527.

03. Phase 1 - Tokenizer 구현

Encoder: Patch-based with learned latent tokens

Input: Image patches
MAE dropout: p~U(0, 0.9) random per image
Output: Latent tokens → tanh projection

Decoder: Patch reconstruction

Input: Latent projection + learned tokens
Output: Image patches

Loss: MSE + 0.2×LPIPS

RMS normalization across losses

Causal Time Attention:

Each frame attends to itself and past
Enables temporal compression
Frame-by-frame decoding support

D. Hafner, W. Yan, and T. Lillicrap, “Training Agents Inside of Scalable World Models,” Sept. 29, 2025, *arXiv*: arXiv:2509.24527. doi: 10.48550/arXiv.2509.24527.

03. Phase 1 - Shortcut Forcing 구현

Objective Formulation

Given:

- $z_0 \sim N(0,1)$ (noise)
- $z_1 \sim$ Data (clean)
- $\tau \in [0,1]$ (signal level)
- d (step size)

Noisy input

Flow Loss (d=d_min)

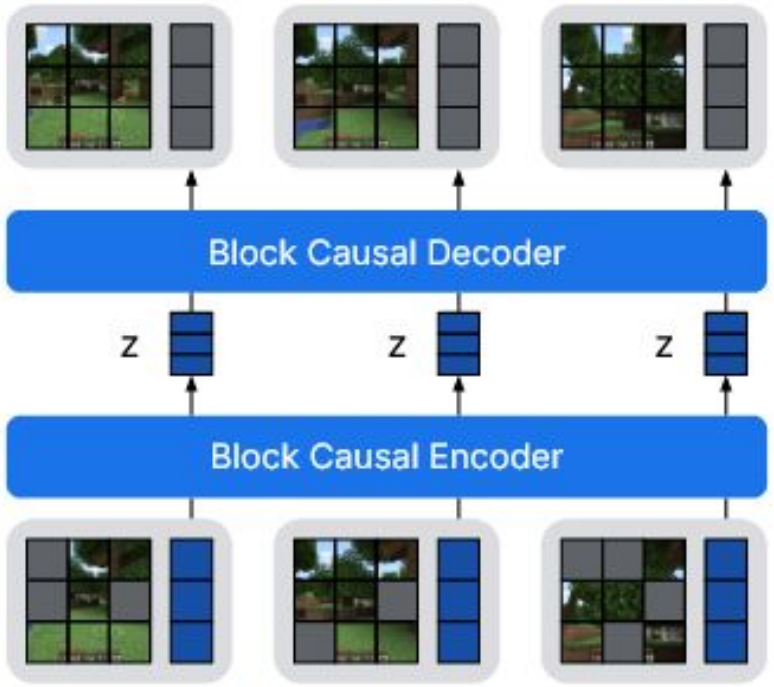
Bootstrap Loss (d>d_min)

- Take two d/2 steps: $z' = \tilde{z} + b' \cdot d/2$
- Target: $v_{target} = (b' + b'')/2$
- Loss: $(1-\tau)^2 \, ||(\hat{z}_1-\tilde{z})/(1-\tau) - sg(v_{target})||^2$

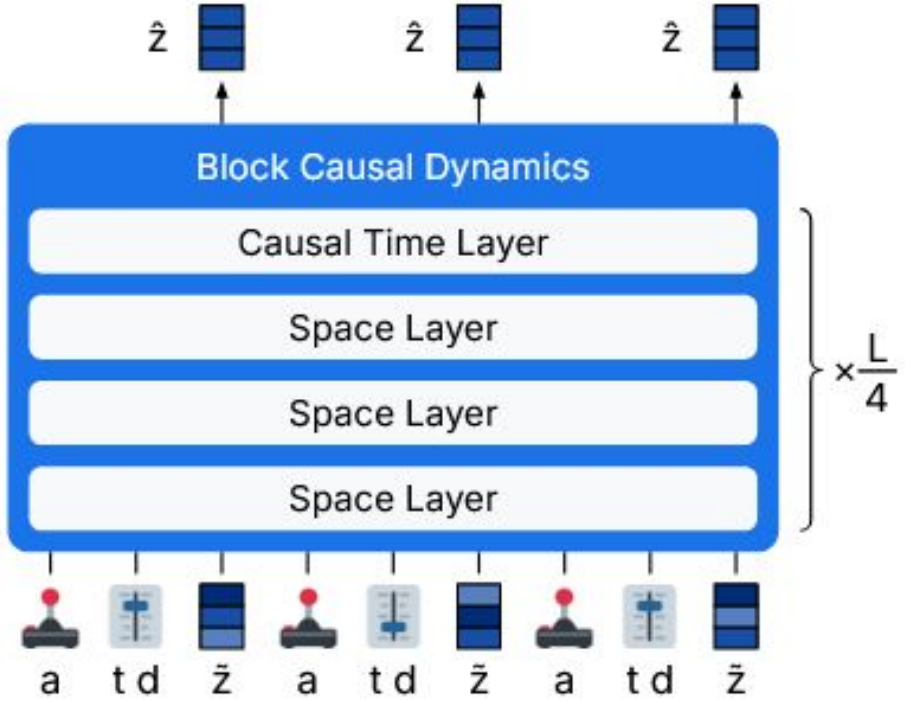
Ramp Weight

D. Hafner, W. Yan, and T. Lillicrap, “Training Agents Inside of Scalable World Models,” Sept. 29, 2025, *arXiv*: arXiv:2509.24527. doi: 10.48550/arXiv.2509.24527.

03. Phase 2 - Efficient Transformer 구현



(a) Causal Tokenizer



(b) Interactive Dynamics

03. Framework

Modeling:

- Tokenizer: Flax (CNN, Patch ops)
- Dynamics: Flax (Large transformer, GQA)
- Policy/Value: Flax or Equinox

Why Flax?

- Transformer 구현 예제 풍부
- GQA, RoPE 레퍼런스 많음
- Attention masking 유연
- Google 공식 지원

Training Utilities

- Optimizer: Optax (Adam, gradient clip, warmup)
- Checkpointing: Orbax
- Logging: WandB

Key Components to Implement

1. RoPE (Rotary embeddings)
2. SwiGLU (activation)
3. QKNorm (Q,K normalization)
4. GQA (grouped attention)
5. Block-causal masking
6. Shortcut forcing loss

03. benchmark

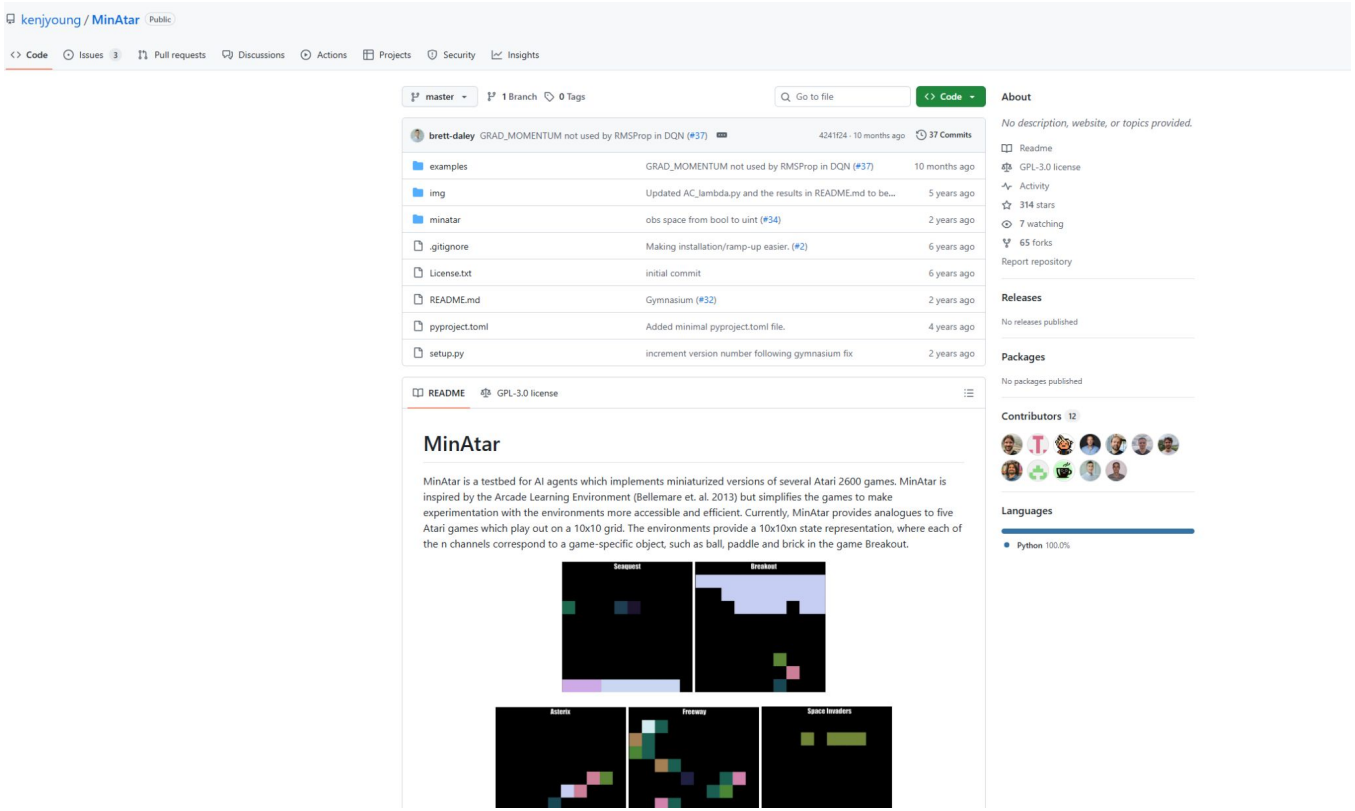
Stage 1: MinAtar (빠른 검증)

Environment: Breakout, Asterix, etc.
Frames: 5-10M
Time: 1-3 hours per game (Ada6000)

Purpose:

- World model 기본 동작 확인
- Tokenizer reconstruction quality
- Policy learning 검증
- Hyperparameter tuning

https://github.com/kenjyoung/MinAtar



04

Performance