Mastering Diverse Domains through World Models

February 17, 2025

Reference

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2024. URL https://arxiv.org/abs/2301.04104.

Edan Meyer, Adam White, and Marlos C. Machado. Harnessing discrete representations for continual reinforcement learning, 2024. URL https://arxiv.org/abs/2312.01203.

Chuning Zhu, Max Simchowitz, Siri Gadipudi, and Abhishek Gupta. Repo: Resilient model-based reinforcement learning by regularizing posterior predictability, 2023. URL https://arxiv.org/abs/2309.00082.

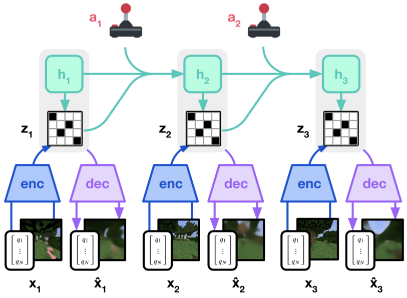
Why Dreamer V3?

Dreamer V3 has several key advantages over other reinforcement learning algorithms:

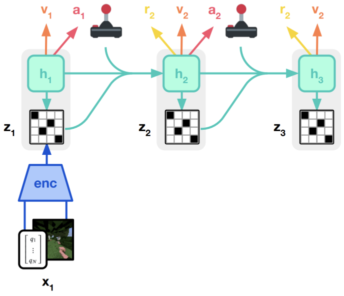
1. Works effectively on over 100 diverse environments:
   - Atari games
   - DeepMind Control (DMC) vision tasks
   - Minecraft challenges
   - And many more...
2. No hyperparameter tuning required:
   - Uses the same hyperparameters across all environments
   - Reduces the need for extensive experimentation
3. Efficient learning without search algorithms
   - Unlike MuZero, doesn't require Monte Carlo Tree Search
   - More computationally efficient
   - Simpler implementation

Learning Process of Dreamer V3

There are two phases of training in Dreamer V3 (Hafner et al. [2024]), the world model learning phase, and the actor critic learning phrase. We will constantly switch between these two phases in training.



(a) World Model Learning



(b) Actor Critic Learning

Why Latent Imagination?

Latent imagination is a crucial component of Dreamer V3's success:

1. Data Efficiency:
   - Deep Reinforcement Learning (DRL) collects trajectories very slowly
   - Even in offline settings, real-world data collection is expensive
   - Latent imagination allows learning from imagined trajectories
   - Can generate thousands of trajectories in parallel
2. Handling Complex Inputs:
   - Real-world inputs are high-dimensional (e.g., images, sensor data)
   - Raw inputs contain many irrelevant details and distractors
   - Direct learning from such inputs is challenging
   - Latent space provides a compact, relevant representation

World Model Learning

The world model:

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

$$\begin{array}{ll}\text{Reward predictor:} & \hat{r}_t \sim p_\phi(\hat{r}_t \mid h_t, z_t)\\ \text{Continue predictor:} & \hat{c}_t \sim p_\phi(\hat{c}_t \mid h_t, z_t)\\ \text{Decoder:} & \hat{x}_t \sim p_\phi(\hat{x}_t \mid h_t, z_t)\end{array}$$

The Dynamics predictor is also called "prior", and the Encoder is also called "posterior".

We use h_t and z_t to mimic a state s_t in the world model. Since h_t gets passed down into each iteration, it is like the deterministic part of a state, and z_t is a somewhat like a random variable.

When z_t is discrete (implemented using a set of onehot encoded vectors), the performance is a lot better, but the reason is still unclear. (Meyer et al. [2024])

World Model Learning

The loss function is given by:

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

where:

$$\begin{aligned}
\mathcal{L}_{\text{pred}}(\phi) &\doteq -\ln p_\phi(x_t \mid z_t, h_t) - \ln p_\phi(r_t \mid z_t, h_t) - \ln p_\phi(c_t \mid z_t, h_t) \\
\mathcal{L}_{\text{dyn}}(\phi) &\doteq \max\big(1, \mathrm{KL}\big[\,\mathrm{sg}(q_\phi(z_t \mid h_t, x_t)) \,\big\|\, \quad p_\phi(z_t \mid h_t)\,\big]\big) \\
\mathcal{L}_{\text{rep}}(\phi) &\doteq \max\big(1, \mathrm{KL}\big[\quad q_\phi(z_t \mid h_t, x_t) \,\big\|\, \mathrm{sg}(p_\phi(z_t \mid h_t))\big]\big)
\end{aligned}$$

with corresponding loss weights $\beta_{\text{pred}} = 1$, $\beta_{\text{dyn}} = 1$, and $\beta_{\text{rep}} = 0.1$.

We will discuss more in the next slide.

KL Divergence Losses in Dreamer V3

If we want to make the loss function of prior (p_ϕ) and posterior (q_ϕ), one elementrary construct method may be:

$$\mathcal{L}(\phi) \doteq \mathbb{E}_{q_\phi}\left[\sum_{t=1}^{T} \mathrm{KL}\big[q_\phi(z_t \mid h_t, x_t) \,\big\|\, p_\phi(z_t \mid h_t)\big]\right]$$

But in Dreamer V3, the loss function is given by:

$$\mathcal{L}(\phi) \doteq \mathbb{E}_{q_\phi}\left[\sum_{t=1}^{T} \overbrace{\beta_{\text{dyn}}}^{1} \mathcal{L}_{\text{dyn}}(\phi) + \overbrace{\beta_{\text{rep}}}^{0.1} \mathcal{L}_{\text{rep}}(\phi)\right]$$

where

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

This technique is called "KL balancing".

Critic Learning with Bootstrapped λ-Returns

The critic is given by:

$$v_\psi(R_t \mid s_t)$$

We would train the critic using the steps:

1. Using the world model to imagine states $s_{1:T}$, actions $a_{1:T}$, rewards $r_{1:T}$.
2. Sample predicted values from critic $v_t \doteq \mathbb{E}\left[v_\psi(\cdot \mid s_t)\right]$, for $t = 1, \ldots, T$.
3. Calculate return estimates R_t^λ, for $t = T, \ldots, 1$, using:

$$R_t^\lambda \doteq r_t + \gamma c_t \Big((1-\lambda) v_t + \lambda R_{t+1}^\lambda\Big) \qquad R_T^\lambda \doteq v_T$$

4. Calculate the loss by:

$$\mathcal{L}(\psi) \doteq -\textstyle\sum_{t=1}^{T} \ln p_\psi(R_t^\lambda | s_t)$$

Actor Learning with Return Regularization

The actor is given by:

$$a_t \sim \pi_\theta(a_t \mid s_t)$$

The actor has the following loss function:

$$\begin{aligned}\mathcal{L}(\theta) \doteq -\textstyle\sum_{t=1}^{T} \mathrm{sg}\Big(\big(R_t^\lambda - v_\psi(s_t)\big) / \max(1,\ S)\Big) \log \pi_\theta(a_t \mid s_t) \\ + \ \eta\, \mathrm{H}\left[\pi_\theta(a_t \mid s_t)\right]\end{aligned}$$

where $S \doteq \mathrm{EMA}\big(\,\mathrm{Per}(R_t^\lambda, 95) - \mathrm{Per}(R_t^\lambda, 5), 0.99\big)$.

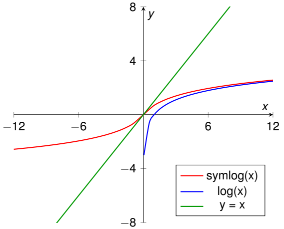
symlog Predictions

Dreamer V3 uses symlog predictions to handle diverse scales in inputs. For reward and decoder predicting, authors opt for this loss function for a network $f(x, \theta)$ with inputs x and parameters θ, learns to predict a transformed version of its targets y:

$$\mathcal{L}(\theta) = \frac{1}{2} \left(f(x, \theta) - \text{symlog}(y)\right)^2$$

where $\text{symlog}(x) \doteq \text{sign}(x) \ln(|x| + 1)$.

And it is worth noticing predicted values are reversible using the inverse function: $\text{symexp} \doteq \text{sign}(x)(\exp(|x|) - 1)$.



Distributions - Uniform Mix

To prevent spikes, zero probabilities and inifite log probabilities in the KL loss, the categorical distributions (encoder, dynamics predictor, and actor distributions) are parametrized as mixtures 1% of uniform and 99% neural network output.

$$\mathbf{v}_{1:n} \leftarrow 0.99\mathbf{v}_{1:n} + \frac{0.01}{n}$$

Distributions - Train Twoards Twohot Encoded Targets

In Dreamer V3, critics learn from symlog-transformed, twohot-encoded returns, improving the prediction accuracy for a wide range of return distributions. This enhancement aids in better value estimation and policy performance.

$$\operatorname{twohot}(x)_i \doteq \begin{cases} |b_{k+1} - x| \,/\, |b_{k+1} - b_k| & \text{if } i = k \\ |b_k \quad - x| \,/\, |b_{k+1} - b_k| & \text{if } i = k+1 \\ 0 & \text{else} \end{cases} \qquad k \doteq \sum_{j=1}^{|B|} \delta(b_j < x)$$

For example, if $b = [0, 2.5, 5, 7.5, 10]$, then $\operatorname{twohot}(5.5) = [0, 0, 0.8, 0.2, 0]$.

Now the loss function for the critic is inroduced by:

$$\mathcal{L}(\theta) \doteq -\operatorname{twohot}(y)^{\intercal} \log \operatorname{softmax}(f(x, \theta))$$

RePo Model: the Insight

Authors of RePo (Zhu et al. [2023]) purposes that a state s can consist of useful information $s^{(1)}$ and distractors $s^{(2)}$, where only $s^{(1)}$ is related to the rewards.

Examples of $s^{(2)}$ can be lighting conditions, moving backgrounds, these are espically critical in real-life situations.

So the authors would want find a way to extract out the the informations that are not useful, and therefore increase the performance of the model.

Mutal Infomation

The mutal inforamtion of two random variables X and Y is definied as:

$$\mathbf{I}(X;Y) = \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$

The range of mutal information is $[0,\infty)$. Intuitively speaking, this can be regard as "how helpful is it given X to predict Y".

Notations

Let us define the following notations:

- o_t denote the high-dimensional image observation at time t,
- z_t be the latent representation obtained via an encoder,
- a_t be the action taken at time t,
- r_t be the reward at time t.

The Learning Problem

The authors of RePo suggests that to conduct:

$$\max_{\phi} \mathbf{I}_{\phi}(z_{1:T}; r_{1:T} \mid a_{1:T}) \text{ s.t. } \mathbf{I}_{\phi}(z_{1:T}; o_{1:T} \mid a_{1:T}) < \epsilon.$$

As maximizing $\mathbf{I}(z_{1:T}; r_{1:T} \mid a_{1:T})$ being a way to make the latent state information s_t useful by making it a information that is able to predict rewards, while $\mathbf{I}(z_{1:T}; o_{1:T} \mid a_{1:T}) < \epsilon$ being a way to "filter out" some information related to the input image o_t that is not helpful for predicting rewards.

The Learning Problem (Cont.)

It can be shown that:

$$\mathbf{I}_\phi(z_{1:T}; r_{1:T} \mid a_{1:T}) \geq \mathbb{E}_\phi \left[\sum_{t=1}^{T} \log q_\phi(r_t|z_t) \right]$$

$$\mathbf{I}_\phi(z_{1:T}; o_{1:T} \mid a_{1:T}) \leq \mathbb{E}_\phi \left[\sum_{t=0}^{T-1} \mathrm{KL}\big[(q_\phi(\cdot \mid z_t, x_t, a_t)) \,\|\, p_\phi(\cdot \mid z_t, a_t)\big] \right]$$

And the original problem:

$$\max_\phi \mathbf{I}_\phi(z_{1:T}; r_{1:T} \mid a_{1:T}) \text{ s.t. } \mathbf{I}_\phi(z_{1:T}; o_{1:T} \mid a_{1:T}) < \epsilon.$$

can be transformed into:

$$\max_\phi \mathbb{E}_\phi \left[\sum_{t=1}^{T} \log q_\phi(r_t|z_t) \right] \text{ s.t. } \mathbb{E}_\phi \left[\sum_{t=0}^{T-1} \mathrm{KL}\big[(q_\phi(\cdot \mid z_t, x_t, a_t)) \,\|\, p_\phi(\cdot \mid z_t, a_t)\big] \right] < \epsilon.$$

The Learning Problem (Cont.)

Then we can transform the problem into a Lagrangian optimization problem:

$$\max_{\phi}\min_{\beta}\left\{\mathbb{E}_{\phi}\left[\sum_{t=1}^{T}\log q_{\phi}(r_t|z_t)\right]\right.$$
$$\left.+\,\beta\left(\mathbb{E}_{\phi}\left[\sum_{t=0}^{T-1}\mathrm{KL}\big[(p_{\phi}(\cdot\mid z_t,x_t,a_t))\,\big\|\,q_{\phi}(\cdot\mid z_t,a_t)\big]\right]-\epsilon\right)\right\}$$

Then we can conduct dual gradient descent for this problem.

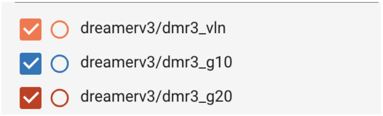
Experiments

Environment: Atari100k Asterix. Trying to test:

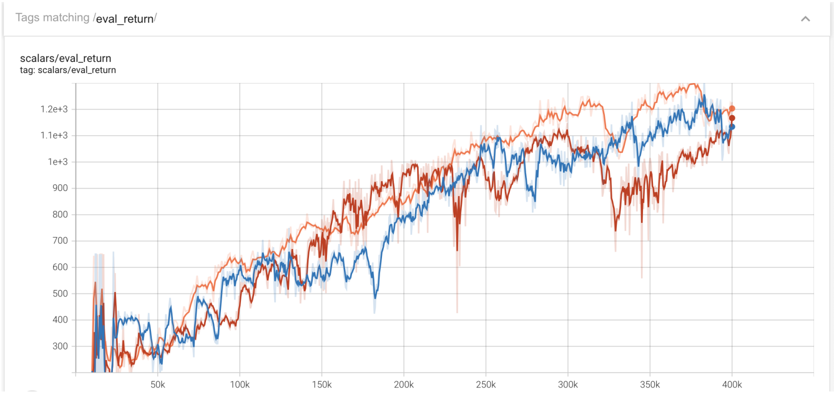
1. The effect of adding noise on the input image while training Dreamer V3.
2. If RePo outperforms Dreamer V3 in noisy environments.

Experiments (Cont.)

Trained 3 Dreamer V3 models, for the input images, each of them received models that is added Gaussian noise of $\sigma = 0, 10, 20$. The scores are nearly identical when evaluating using clean images.



✓	○	dreamerv3/dmr3_vln
✓	○	dreamerv3/dmr3_g10
✓	○	dreamerv3/dmr3_g20



Experiments (Cont.)

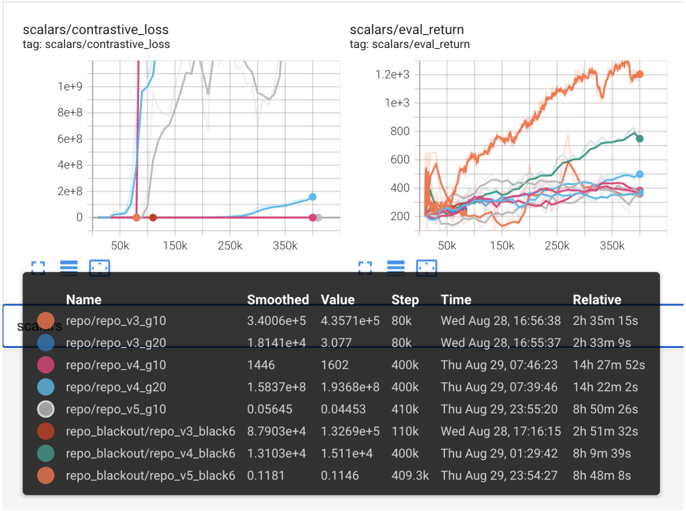
When inferencing these three models, with different kinds for input noise, these are the results.

Model trained with	Inference		
	$\sigma = 0$	$\sigma = 10$	$\sigma = 20$
$\sigma = 0$	1091.67 ± 50.94	996.67 ± 141.36	1083.33 ± 77.55
$\sigma = 10$	950.00 ± 94.21	875.00 ± 151.38	916.66 ± 107.44
$\sigma = 20$	916.66 ± 92.91	1008.33 ± 97.87	958.33 ± 83.42

The experiments are inferenced with $n = 12$ and reported their 95% confidence interval.

Experiments (Cont.)

When trying to modify the code to the structure of RePo, the model isn't successful. Contrastive Loss should be around 10^{-2}, but lots of models have the value around 10^{8}. There still need debugging.



	Name	Smoothed	Value	Step	Time	Relative
●	repo/repo_v3_g10	3.4006e+5	4.3571e+5	80k	Wed Aug 28, 16:56:38	2h 35m 15s
●	repo/repo_v3_g20	1.8141e+4	3.077	80k	Wed Aug 28, 16:55:37	2h 33m 9s
●	repo/repo_v4_g10	1446	1602	400k	Thu Aug 29, 07:46:23	14h 27m 52s
●	repo/repo_v4_g20	1.5837e+8	1.9368e+8	400k	Thu Aug 29, 07:39:46	14h 22m 2s
●	repo/repo_v5_g10	0.05645	0.04453	410k	Thu Aug 29, 23:55:20	8h 50m 26s
●	repo_blackout/repo_v3_black6	8.7903e+4	1.3269e+5	110k	Wed Aug 28, 17:16:15	2h 51m 32s
●	repo_blackout/repo_v4_black6	1.3103e+4	1.511e+4	400k	Thu Aug 29, 01:29:42	8h 9m 39s
●	repo_blackout/repo_v5_black6	0.1181	0.1146	409.3k	Thu Aug 29, 23:54:27	8h 48m 8s