Dream to Control: Learning Behaviors by Latent Imagination

Published in 2020

by Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi

Fabian Wieczorek

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

09. December 2021

Outline

Imagine throwing a basketball

Source: https://blog.playo.co/how-to-improve-free-throw-shooting/

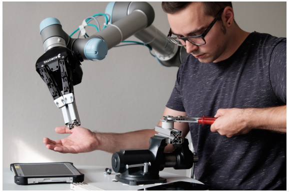
Imagine throwing a ~~basketball~~ bowlingball

Source: https://blog.playo.co/how-to-improve-free-throw-shooting/

Motivation: Latent Imagination

Continuous control

- In complex environments
  - Uncertainties
  - Dynamic environments
  - Unpredictable situations
- With contact forces
  - Peg-insertion, Assembly
  - Locomotion (bipedal robots)



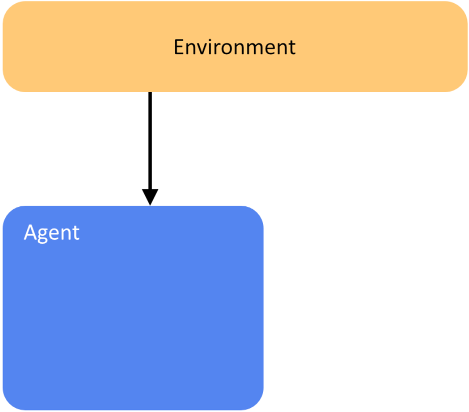
Left: Human-Robot collaboration (https://interactive-robotics.engineering.asu.edu/autonomous-robots-special-issue/),
Right: Locomotion in uncertain environment (https://www.youtube.com/watch?v=k7s1sr4JdlI)

Reinforcement Learning - Overview

Reinforcement Learning - Overview

State
$s_t \in S$

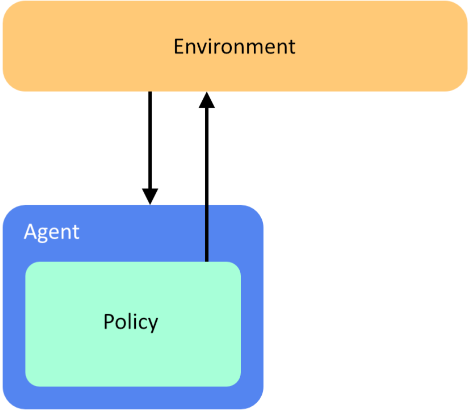
Reinforcement Learning - Overview



State
$s_t \in S$

Action
$a \in A$

Reinforcement Learning - Overview

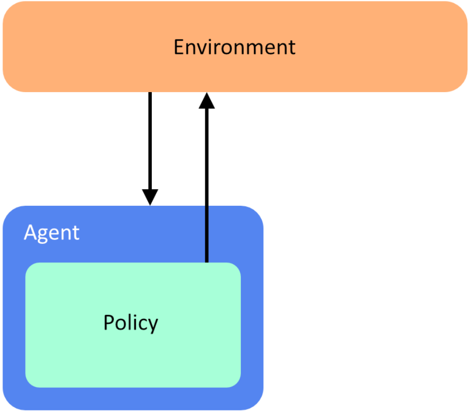


State
$s_t \in S$

Action
$a \in A$

Policy
π

Reinforcement Learning - Overview



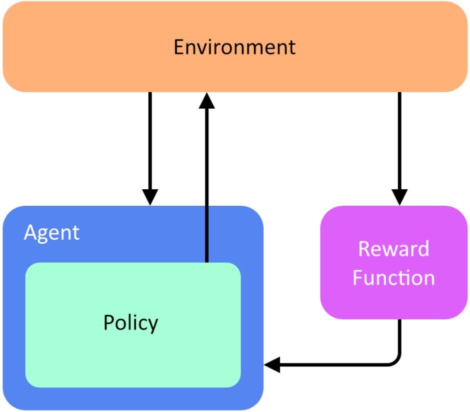
State
$s_t \in S$

Action
$a \in A$

Policy
π

Environment changes
$P_a(s, s')$

Reinforcement Learning - Overview



State
$s_t \in S$

Action
$a \in A$

Policy
π

Environment changes
$P_a(s, s')$

Reward
$R_a(s, s')$

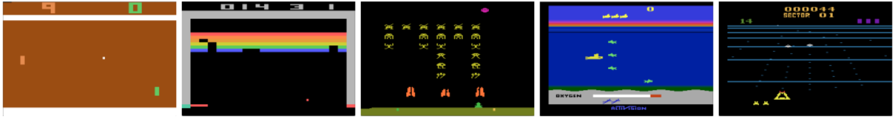
Related Work - Model-free

Playing Atari with Deep Reinforcement Learning [MKS+13]

- DQN
- Input direct from images
  - Converted to greyscale
  - Downscaled/Cropped to 84x84
- Uses non-continuous actions

	B. Rider	Breakout	Enduro	Pong	Q*bert	Seaquest	S. Invaders
Random	354	1.2	0	−20.4	157	110	179
Sarsa [3]	996	5.2	129	−19	614	665	271
Contingency [4]	1743	6	159	−17	960	723	268
DQN	**4092**	**168**	**470**	**20**	**1952**	**1705**	**581**
Human	7456	31	368	−3	18900	28010	3690
HNeat Best [8]	3616	52	106	19	1800	920	**1720**
HNeat Pixel [8]	1332	4	91	−16	1325	800	1145
DQN Best	**5184**	**225**	**661**	**21**	**4500**	**1740**	1075

Performance comparison of DQN and other approaches in different Atari games. [MKS+13]

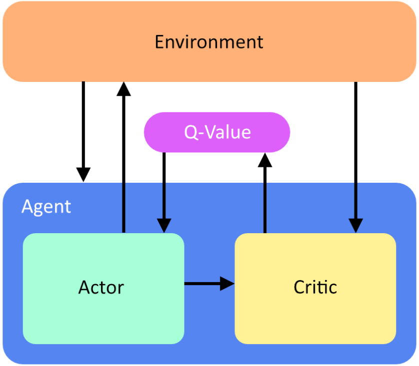


Different Atari games learned by DQN. [MKS+13]

Related Work - Model-free

Continuous Control with Deep Reinforcement Learning [LHP+19]

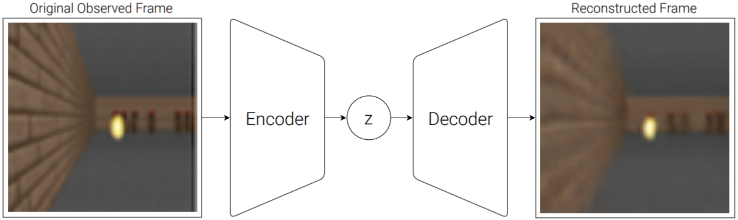
- Uses continuous actions
- Actor-critic
- Q-Learning



Overview of the actor critic approach in reinforcement learning.

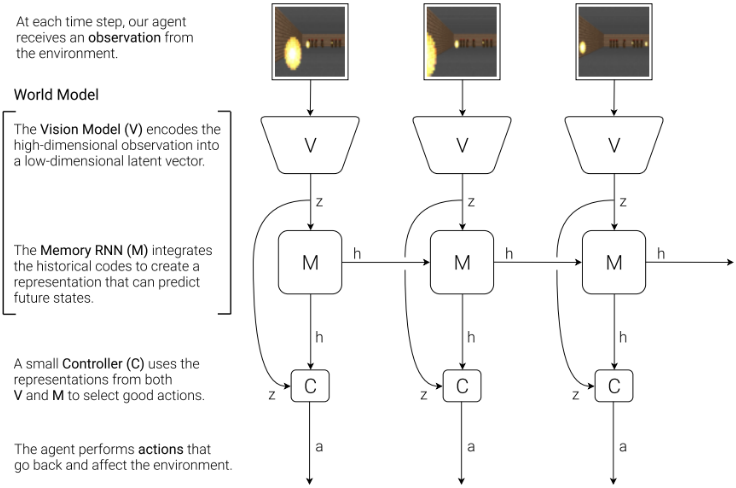
World Models [HS18]

- Learns world model using Variational Auto Encoder (VAE)
- Two training phases
  - 1. Encoding World
  - 2. Predict future states



VAE encodes an image to a small latent vector representing the world. [HS18]

Related Work - Model-based

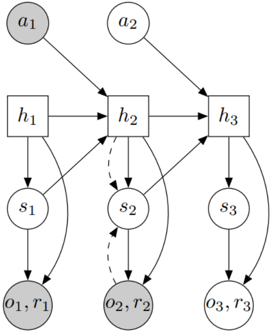


Process overview of PlaNet. [HS18]

Related Work - Model-based

Learning Latent Dynamics for Planning from Pixels [HLF+19]

- Deep Planning Network (PlaNet)
- Recurrent State Space Model (RSSM)
- Same Encoder/Decoder from *World models* [HS18]
- Predicts multiple (few 1000) solutions and pick the best at each time step
  - No policy required



Overview of the Recurrent State Space Model. [HLF+19]

Concept: Train directly in latent space

- Saves computational resources skipping the image encoding

The three stages

- 1. Learn to encode world from past experience
- 2. Learn to pick best actions in latent space
- 3. Perform in new scenarios and collect new data

Difference to previous approaches

Training iterates through all stages multiple times

- Own performance influences experience

Dreamer Overview

Concept: Train directly in latent space

- Saves computational resources skipping the image encoding

The three stages

- 1. Learn to encode world from past experience
- 2. Learn to pick best actions in latent space
- 3. Perform in new scenarios and collect new data

Difference to previous approaches

Training iterates through all stages multiple times

- Own performance influences experience

Dreamer Overview

Concept: Train directly in latent space

- Saves computational resources skipping the image encoding

The three stages

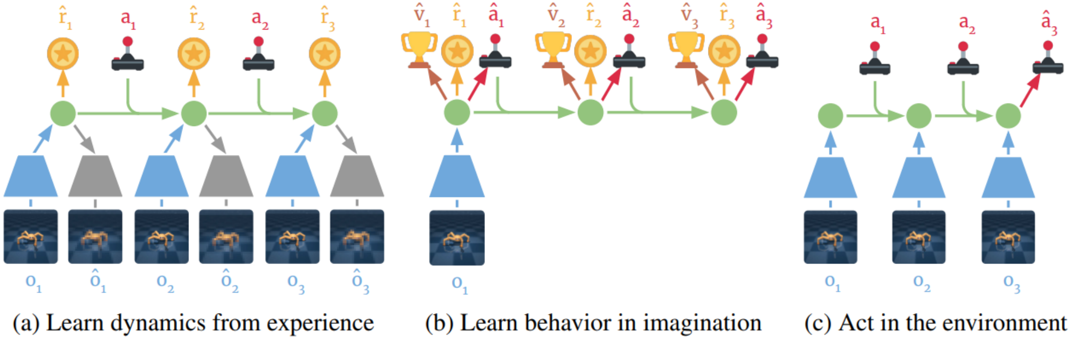
- 1. Learn to encode world from past experience
- 2. Learn to pick best actions in latent space
- 3. Perform in new scenarios and collect new data

Difference to previous approaches

Training iterates through all stages multiple times

- Own performance influences experience

Dreamer Overview



The three training stages of the Dreamer architecture. [HLBN20]

Dreamer Architecture

Models for World Model

- Representation
  - Convolutional Neural Network (Encoder/Decoder)
  - Encode Image to latent state
- Transition
  - Recurrent State Space Model
  - Predict next latent state given latent state + action
- Reward
  - Fully Connected Neural Network
  - Predict the reward for given latent state

Models for Behavior Learning

- Actor Network
  - Fully Connected Neural Network
  - Pick action given latent state
- Value Network
  - Fully Connected Neural Network
  - Estimate best value given latent state

Dreamer Architecture

Models for World Model

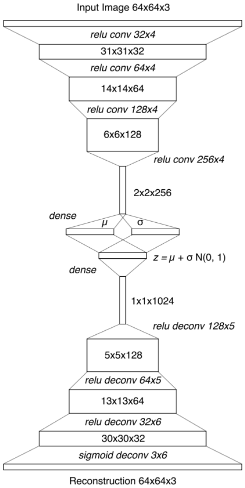
- Representation
  - Convolutional Neural Network (Encoder/Decoder)
  - Encode Image to latent state
- Transition
  - Recurrent State Space Model
  - Predict next latent state given latent state + action
- Reward
  - Fully Connected Neural Network
  - Predict the reward for given latent state

Models for Behavior Learning

- Actor Network
  - Fully Connected Neural Network
  - Pick action given latent state
- Value Network
  - Fully Connected Neural Network
  - Estimate best value given latent state

Dreamer Architecture - Encoder/Decoder

- Uses VAE from *World models* [HS18]
- Latent space
  - 2 vectors of length 30
  - Represent 30 mean and variance pairs
  - *True* state: just mean vector
  - Stochastic state: sample from gaussian

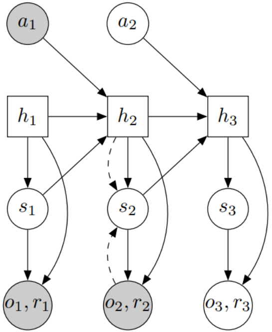


VAE used to encode/decode latent vectors. [HS18]

Dreamer Architecture - RSSM

Recurrent State Space Model [HLF+19]

- Recurrent Neural Network
- Deterministic part h_t
  - Forwards the actual information present
- Stochastic part s_t
  - Helps predicting multiple futures
  - Useful for partial observability



Overview of RSSM. [HLF+19]

All other models (Reward, Action, Value)

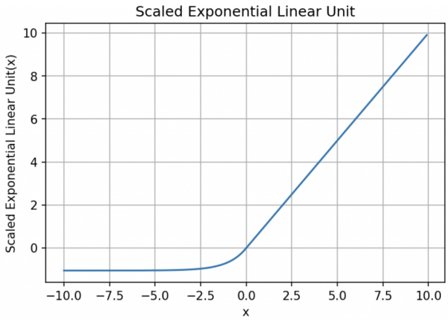
- 3 Dense Layers with 300 neurons
- Exponential Linear Unit activation

Reward/Value Model

- Scalar output (1 neuron)

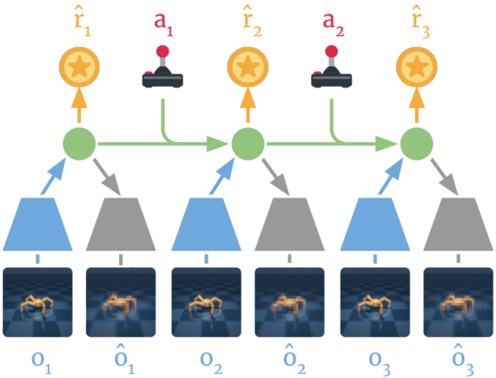
Actor Model

- High dimensional (depends on task)
- Continuous (real numbers)



ELU activation. Source: https://blog.robofied.com/scaled-elu-activation-function/

Training Process - Learn World Model



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.

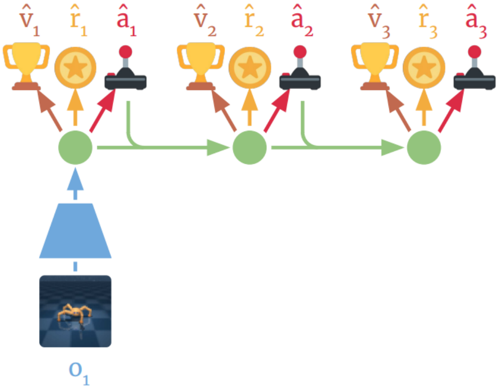
    Add exploration noise to action.

    $r_t, o_{t+1} \leftarrow$ `env.step(`a_t`)`.

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

Overview of world model learning step. [HLBN20]

Training Process - Learn Behaviors



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 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(o_t, a_t, r_t)_{t=1}^{T}\}$.

Overview of behavior learning step. [HLBN20]

Training Process - Value Estimator

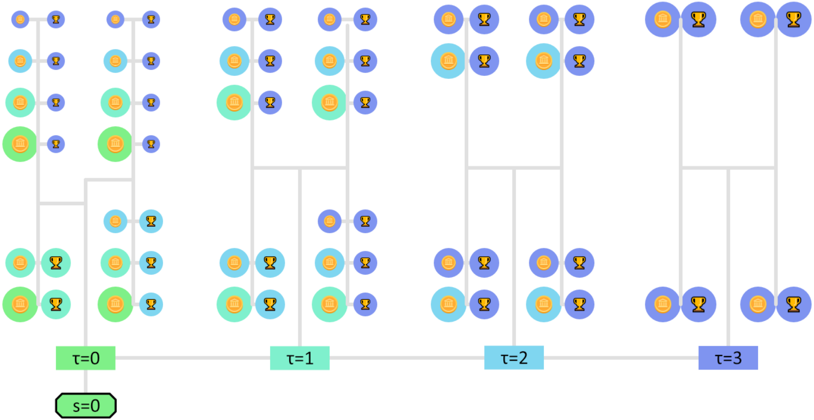
- Trains the actor and value Network
- exponentially weighted average of value estimates

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

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

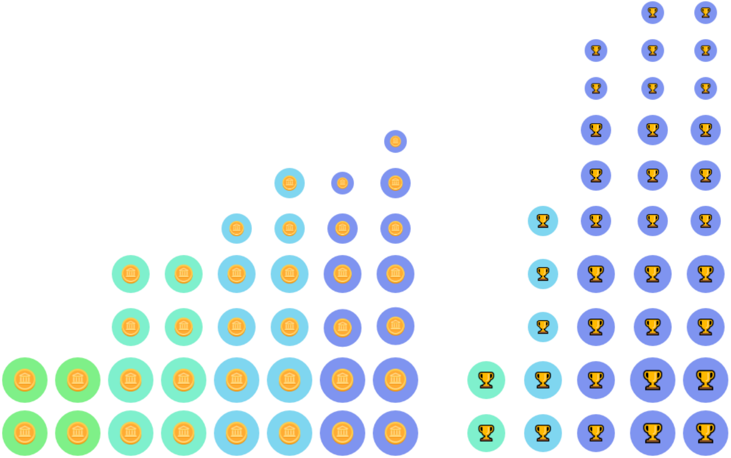
Equation for the value estimator. [HLBN20]

Training Process - Value Estimator



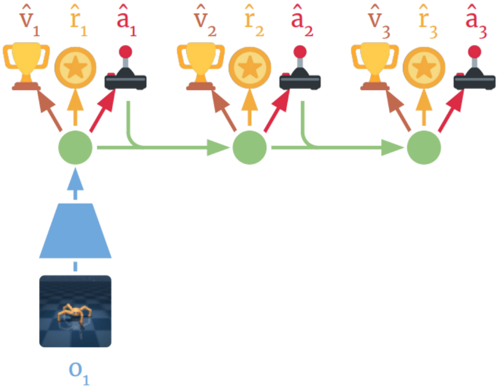
Visualisation of the value estimation $V_\lambda(s_\tau)$ for $t = 0$ (here $s = 0$) with $H = 3$.

Training Process - Value Estimator



Visualisation of the distribution of the value estimation $V_\lambda(s_\tau)$ for $t = 0$ with $H = 3$.

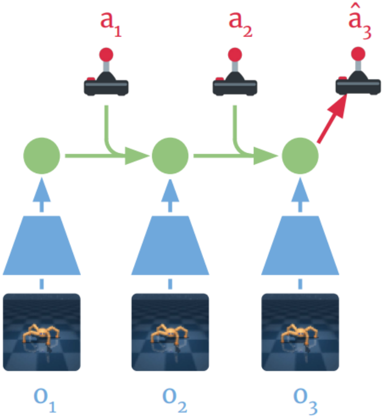
Training Process - Learn Behaviors



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 $\psi \leftarrow \psi - \alpha \nabla_\psi \sum_{\tau=t}^{t+H} \frac{1}{2} \left\| v_\psi(s_\tau) - \mathrm{V}_\lambda(s_\tau) \right\|^2$.
  `// 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 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(o_t, a_t, r_t)_{t=1}^{T}\}$.

Overview of behavior learning step. [HLBN20]

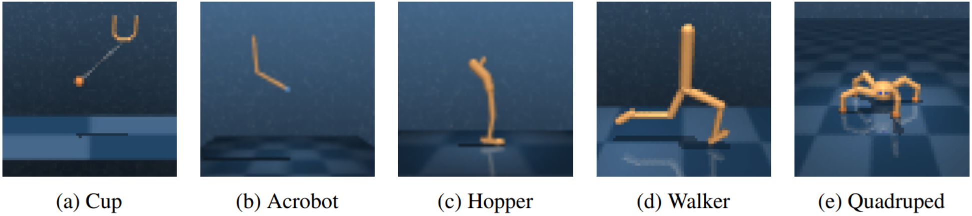
Training Process - Interact in Environment



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 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(o_t, a_t, r_t)_{t=1}^{T}\}$.

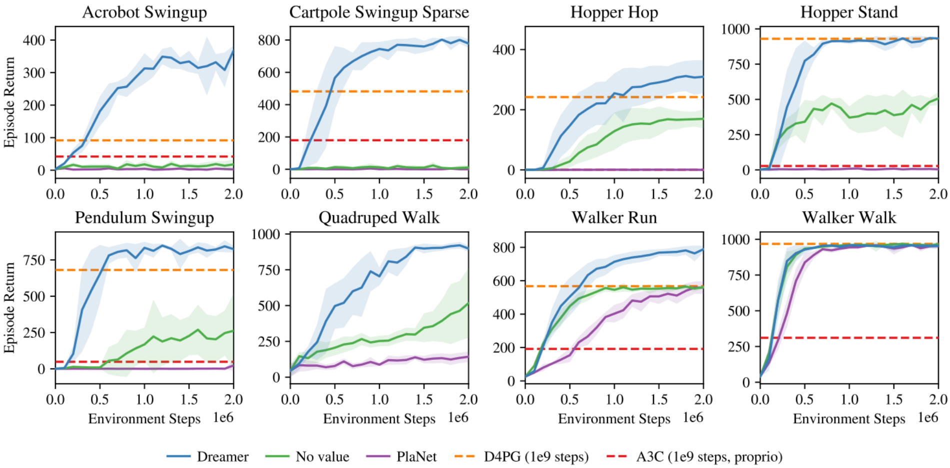
Overview of environment interaction step. [HLBN20]

Results - Tasks



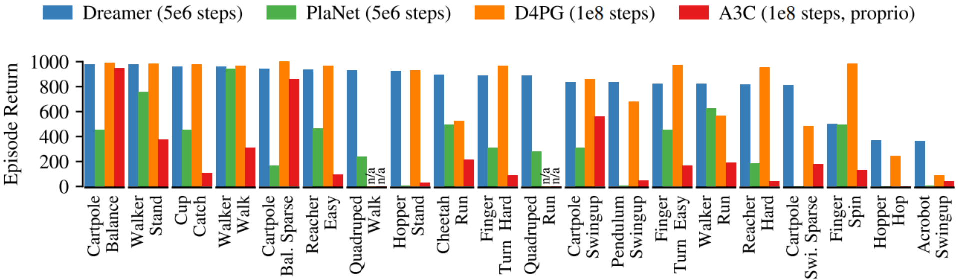
Selection of different tasks requiring continuous control. [HLBN20]

Results - Scores in Tasks



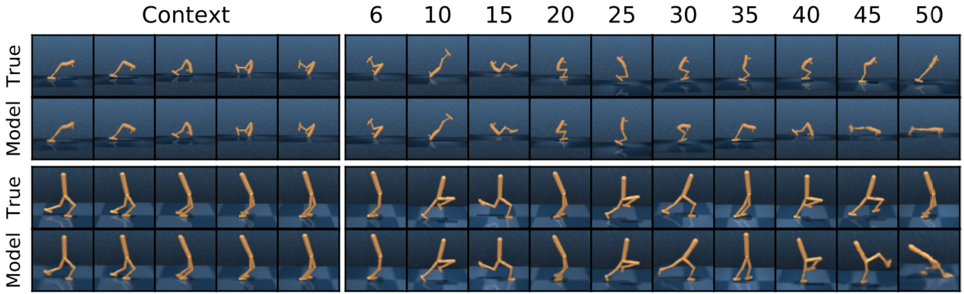
Comparison of overall performance between different algorithms. [HLBN20]

Results - Efficiency



Comparison of efficiency between different algorithms. [HLBN20]

Results - Reconstructed predictions



Comparison between the reconstructed predictions of dynamics (given five images) and the actual outcome. [HLBN20]

Conclusion

- Learns accurate world model using images
- Efficiently trains directly on latent states
- Estimates values beyond time horizon
- Exceeds state-of-the-art algorithms in performance/efficiency

Future Work

- Mastering Atari with Discrete World Models [HLNB21]

References

[HLBN20] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi, *Dream to control: Learning behaviors by latent imagination*, 2020.

[HLF+19] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson, *Learning latent dynamics for planning from pixels*, 2019.

[HLNB21] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba, *Mastering atari with discrete world models*, 2021.

[HS18] David Ha and Jürgen Schmidhuber, *World models*, CoRR **abs/1803.10122** (2018).

[LHP+19] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra, *Continuous control with deep reinforcement learning*, 2019.

References (cont.)

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, *Playing atari with deep reinforcement learning*, 2013.