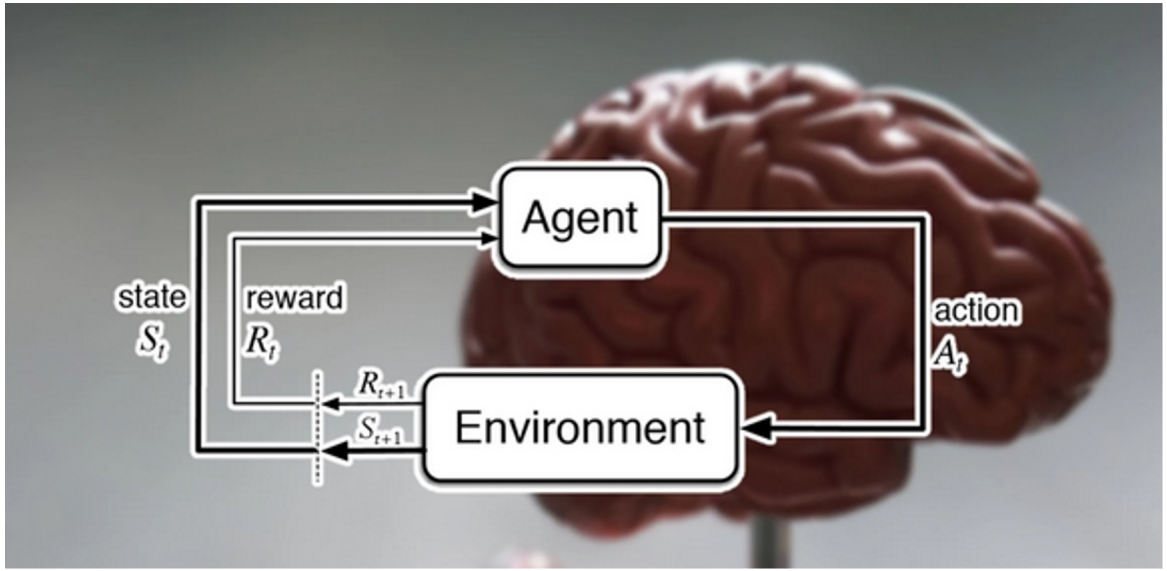
Dream to Control: Learning Behaviors by Latent Imagination

Presenter: Vrushabh Zinage

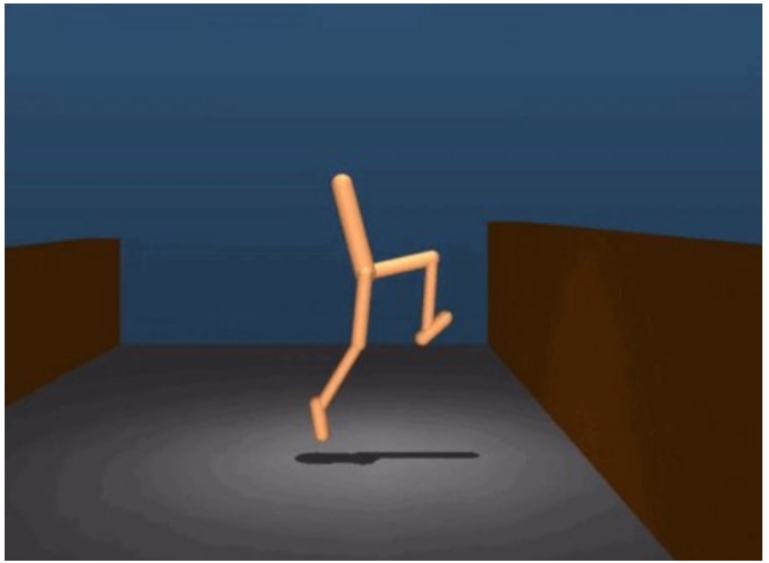
28 September 2023

Typical Reinforcement Learning paradigm



Some of the main limitations of this approach

- Exploration vs. Exploitation
- Sample Efficiency
- Sparse and Delayed Rewards:
- Generalization
- Scalability
- Exploration safety



Main Question:

- **Can we model the environment and train policies to be able to directly take actions in the actual environment and also make long time horizon predictions?**

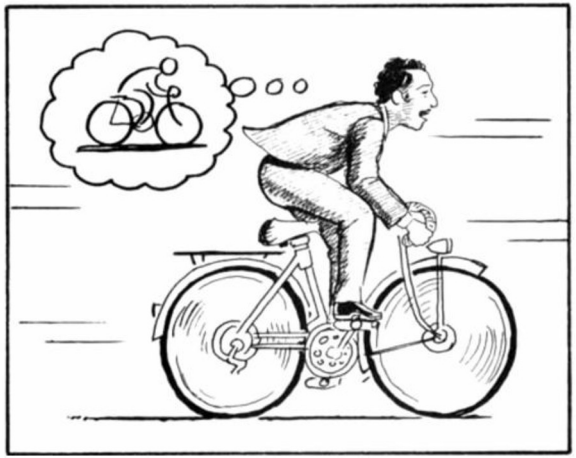


Figure 1. A World Model, from Scott McCloud's *Understanding Comics*. (McCloud, 1993; E, 2012)

Why chose Model-based RL vs Model-free for control?

- Model-Based benefits mainly come from learning a simplified representation of the world, called Latent Dynamics.

1) **Easy Changes**: Instead of actively interacting with the real environment or using tools like cameras to understand it, you can just tweak the latent state that governs the latent model.

1) **Needs Fewer Examples**: With a good representation, you can create new training situations by just changing the numbers in this vector.

1) **Safety**: Using this representation helps design behaviors that are likely safe in real-life situations.

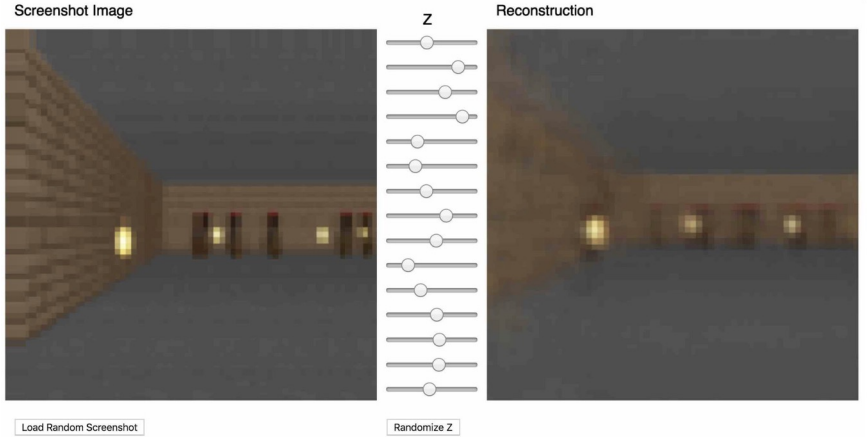
1) **"Imagining" the Environment**: When we use this representation during training, it's like the model is visualizing or daydreaming about the environment.

Prior Work

World Models (David Ha, Jurgen Schmidhuber (2018))

- Acquire a dynamics model in latent space for an RL setting.
- The model's hidden space encompasses essential characteristics, simplifying the process of learning the best policy.
- They train the model entirely in the latent space which saves time and resources compared

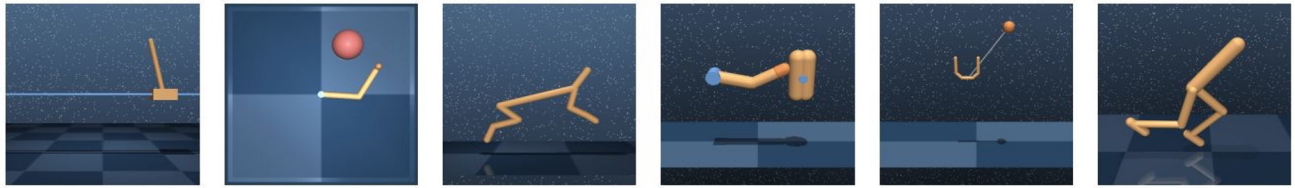
To running it in actual environments



Prior Work

Learning Latent Dynamics for Planning with Pixels (Hafner et. al (2019))

- Learn a dynamics model for various activities within the DeepMind Control suite.
- Plan only using the latent space of the dynamics model (PlaNet)
- Generalize to include multi-step predictions in latent space
- Performance on par with current state-of-the-art model-free approaches, with ~200x less environment interactions
- They used gradient-free planning



(a) Cartpole (b) Reacher (c) Cheetah (d) Finger (e) Cup (f) Walker

Main Contributions of Dreamer

- **Learning long-horizon behaviors by latent imagination**
  - The value functions optimize for Bellman consistency for imagined rewards whereas the policy maximizes these value functions by taking analytic gradients backward.
  - Actor critic algorithm accounts for rewards beyond the imagination horizon
  - Instead, explore the environment and train the new policy according to our current policy
- **Simultaneously predict the state and action values**
  - Allows for faster convergence to an optimal policy by learning long-horizon behaviors
  - Back propagation can be leveraged for the value function through dynamics model's latent space

Latent dynamics

1) **Latent dynamics Dreamer**: Dreamer utilizes a latent dynamics model.

1) **Three Components of the Latent Dynamics Model:**
   - Representation Model
   - Transition Model
   - Reward Model

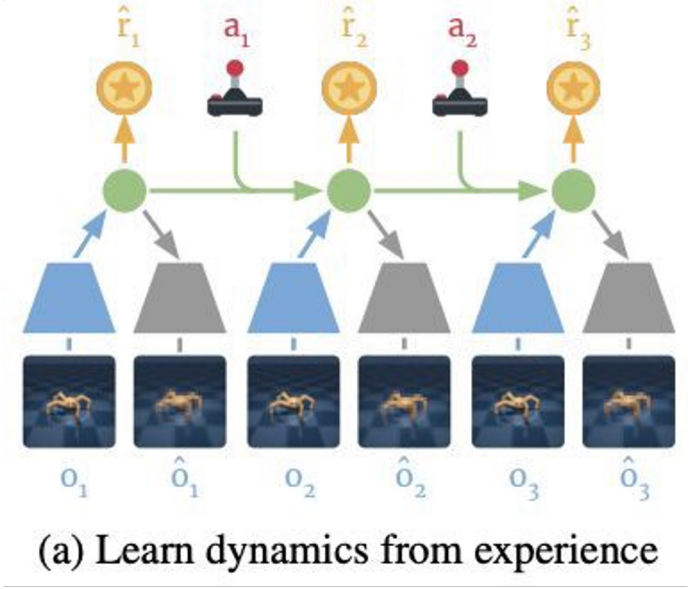
1) **Representation Model:**
   - It encodes observations and actions.
   - The aim is to produce continuous vector-valued model states.
   - The transitions of these model states are Markovian.

Latent dynamics

4) **Transition Model:**
- This model predicts the future model states.
- It does this without needing to see the corresponding observations that will influence these future states.

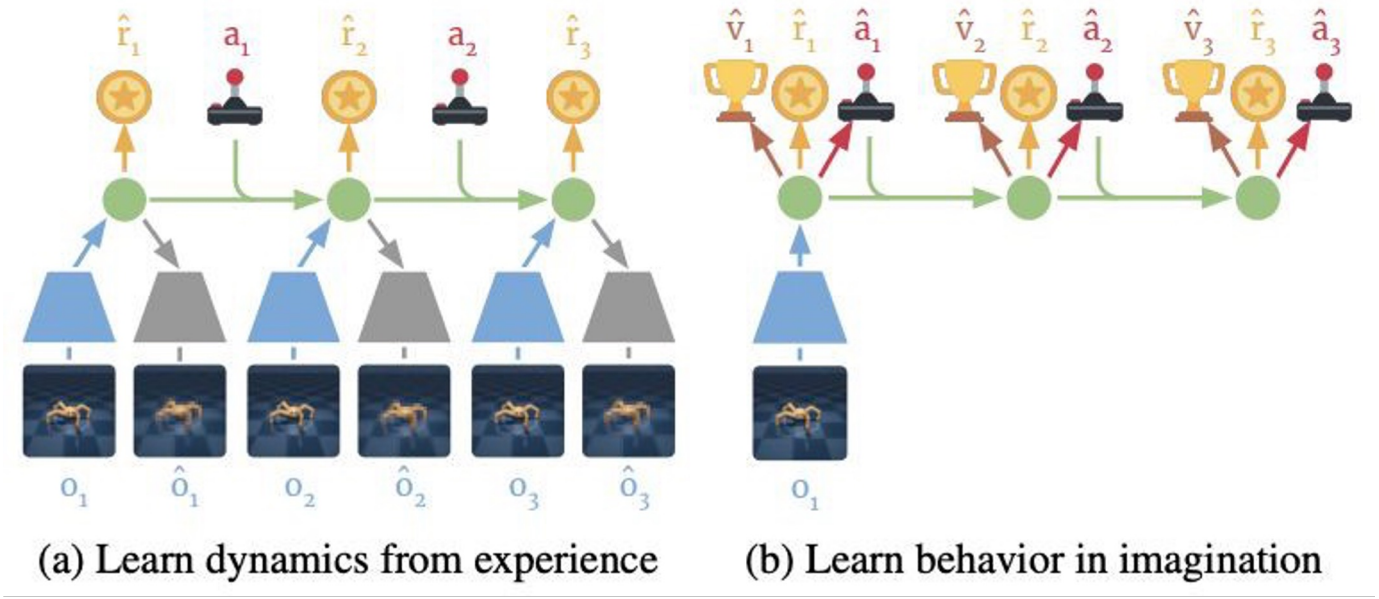
5) **Reward Model:**
- Predicts rewards based on the model states.

Dream to control approach



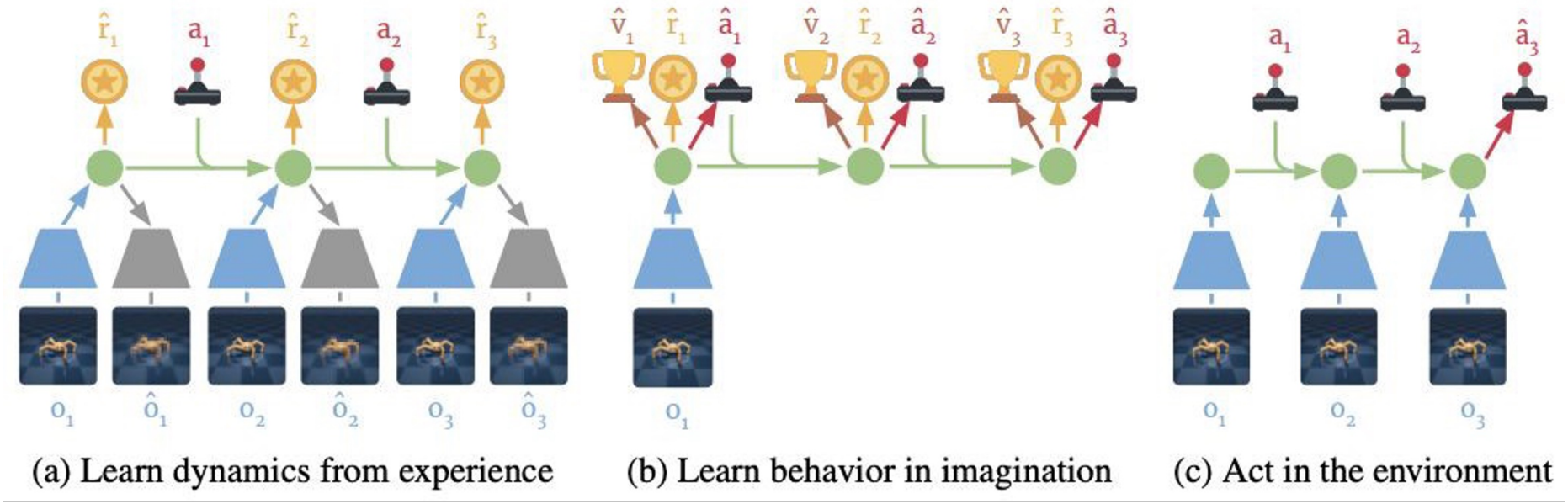
(a) Learn dynamics from experience

Dream to control approach



(a) Learn dynamics from experience (b) Learn behavior in imagination

Dream to control approach



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

Action and Value model

1) **Action model**: It's designed to predict actions that effectively address the imagination environment.

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

1) **Value model:**

a) Estimates the expected imagined rewards that the action model can achieve from each state

b) Both the action and value models are trained in tandem, akin to the policy iteration method. The action model focuses on maximizing the estimated value, while the value model strives to align with the estimated value that adjusts with changes in the action model

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

Value estimation

State values can be estimated in multiple ways that trade off bias and variance. One of the method leveraged in this paper is as follows:

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

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

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

Learning objective

1) The objective for the action model is to predict actions that result in state trajectories with high value estimates.

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

1) The objective for the value model, in turn, is to regress the value estimates

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

Algorithm, formally

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

Algorithm, formally

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

Algorithm, formally

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

Algorithm, formally

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

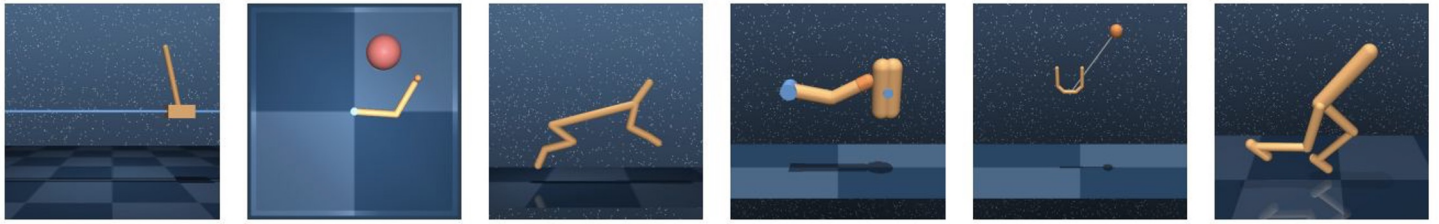
Representation	$p_\theta(s_t \mid s_{t-1}, a_{t-1}, o_t)$
Transition	$q_\theta(s_t \mid s_{t-1}, a_{t-1})$
Reward	$q_\theta(r_t \mid s_t)$
Action	$q_\phi(a_t \mid s_t)$
Value	$v_\psi(s_t)$

Hyper parameters

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

Experimental Setup

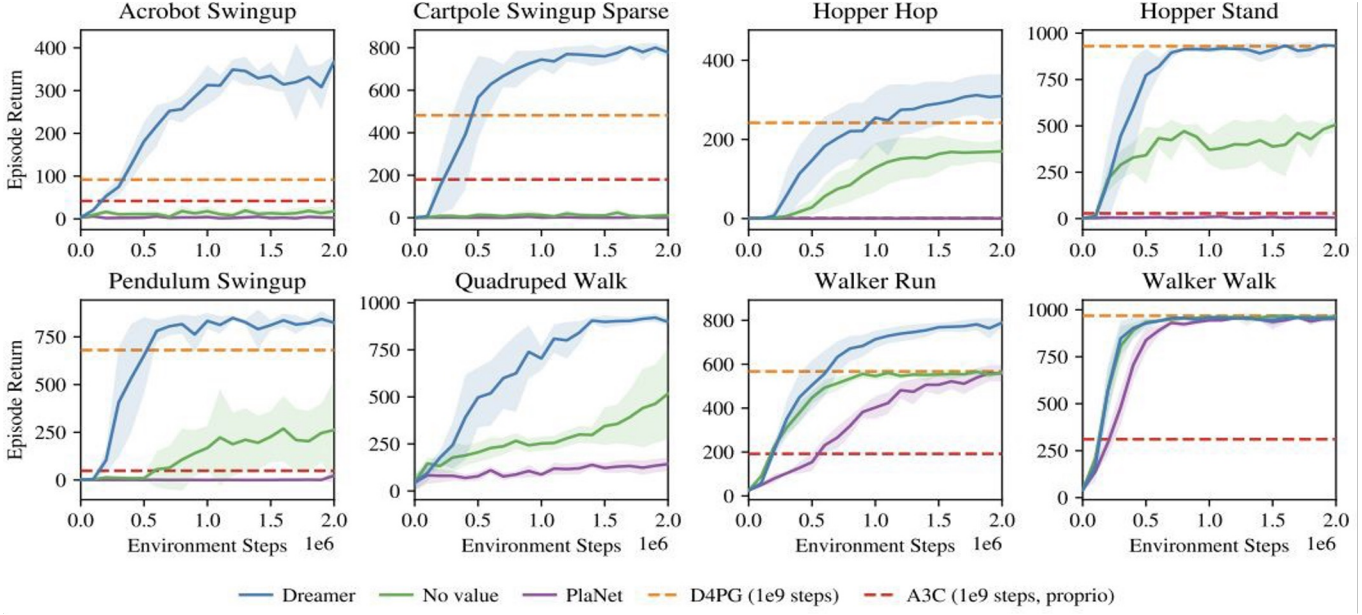
- Performance evaluated on visual control tasks which are present in the DeepMind Control Suite
- Comparisons are made against:
  - PlaNet, previous latent imagination state-of-the-art
  - D4PG, top model-free agent
  - A3C, state-of-the-art actor-critic method



(a) Cartpole (b) Reacher (c) Cheetah (d) Finger (e) Cup (f) Walker

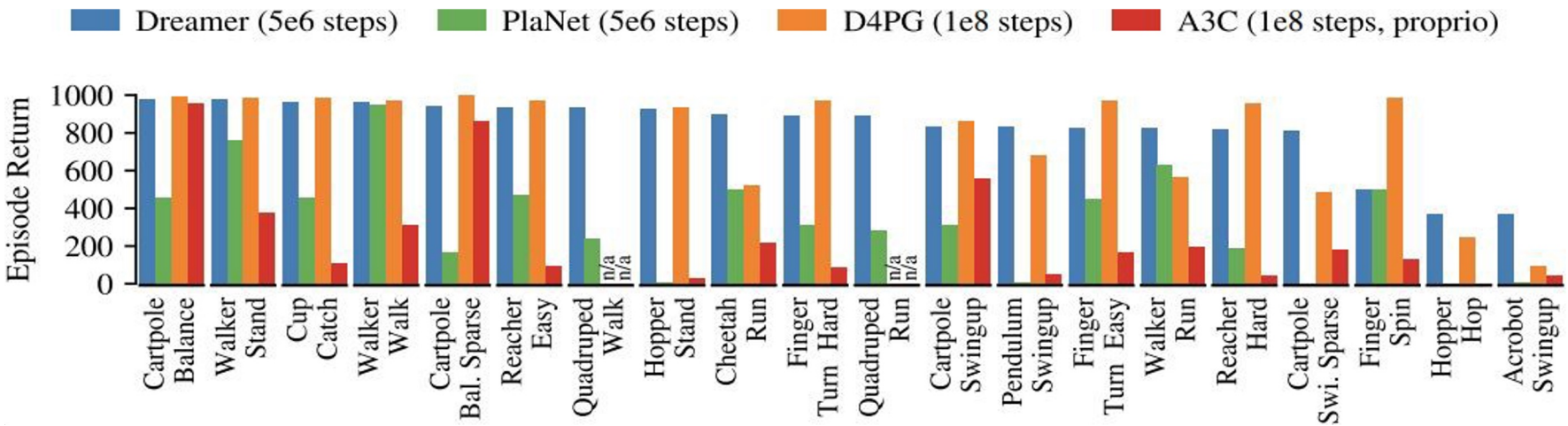
Experimental Results

1) Dreamer succeeds at visual control tasks that require long-horizon credit assignment, such as the acrobot and hopper tasks
2) Optimizing only imagined rewards within the horizon via an action model



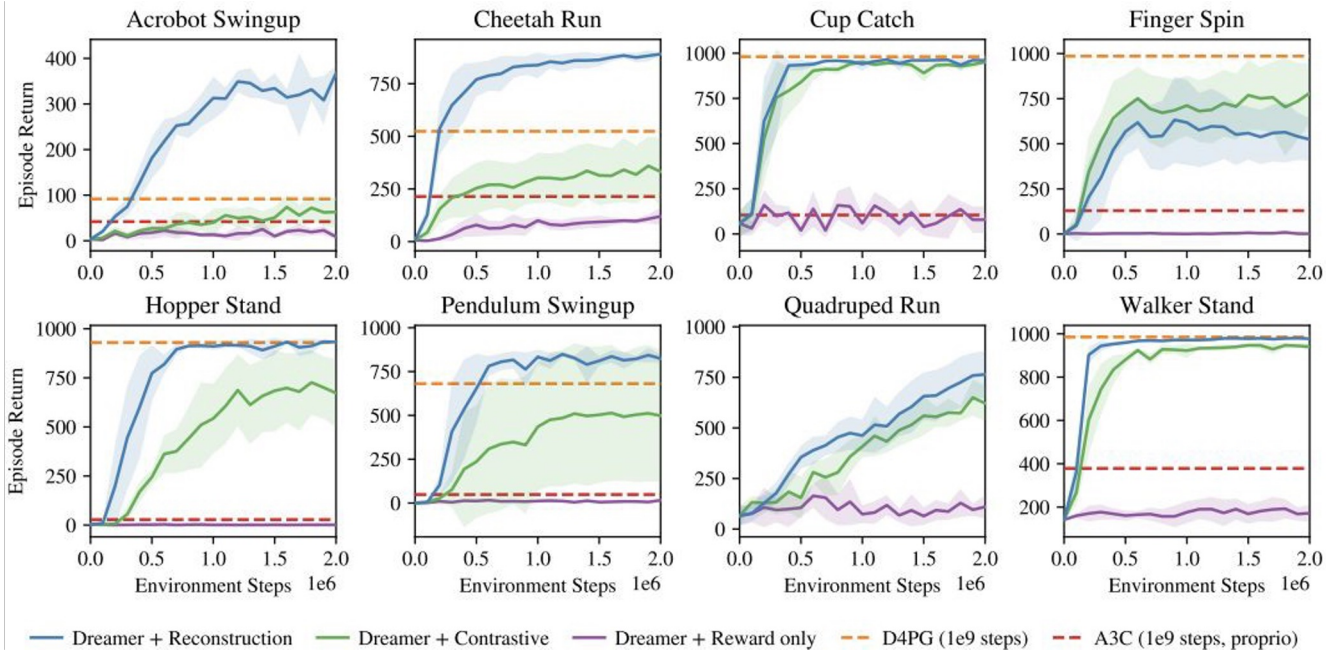
Experimental Results

1) Dreamer demonstrates data-efficiency similar to PlaNet.
2) Dreamer surpasses the asymptotic performance of top model-free agents.
3) After 5e6 environment steps, Dreamer achieves an average performance of 823 across tasks whereas PlaNet's average performance is 332.
4) The top model-free D4PG agent reaches a performance of 786 after 1e8 steps.
5) All results are based on averages taken from 5 different seeds.



Experimental Results

- **Dreamer does not require as long to train as model-free agents**
- **Dreamer performs better in environments that require long term prediction**



Discussion of Results

- They showed that Dreamer is able to learn long-horizon behaviors from beyond the horizon, which outperforms more short-sighted approaches
- Demonstrate that Dreamer is able to be as efficient as PlaNet while matching or even outperforming state-of-the-art model-free agents
- Performance of Dreamer is affected by the method of representation learning used
  - Better representation learning performance implies Better Dreamer performance

Limitations and open problems

- **More dependance on a better representation learner**
  - Limits the breadth of tasks that this can be applied to rather than traditional reinforcement learning
  - Adding inductive bias for more faster and accurate learning
- **Different Value estimation functions are not evaluated**
  - Can we do better value estimation leading to faster learning?

Future Work

- Apply inductive bias for faster learning
- Learn more complex visual tasks with sparse rewards
- Apply latent imagination to more input modalities, potentially getting us closer to real-world uses
- Leverage more recent specialized representation learning approaches to perform more task-specific learning in the latent space
- Pixel wise reconstruction is expensive

Summary

- Reinforcement learning traditionally involves many interactions with the environment
- Environment interactions can be time consuming and computationally expensive
- Therefore we can chose the model-RL based paradigm to train in a latent space, limiting need for more interactions with the environment
  - Prior works used a fixed imagination horizon (short-sighted behaviors) and had to use derivative-free optimization
  - Back propagation can be performed using the value function estimated
  - Better data efficiency and computational time

Follow up on Dreamer

- **DreamerV2:**
  - DreamerV2, a novel reinforcement learning agent, achieves behaviors solely through predictions in a world model's compact latent space, using discrete representations and training the policy separately from the model.
  - DreamerV2 is the first agent to reach human-level performance across 55 Atari tasks by learning within a separately trained world model
- **DreamerV3:**
  - DreamerV3 is a general and scalable reinforcement learning algorithm based on world models that consistently outperforms previous methods across diverse domains using fixed hyperparameters.
  - Notably, without relying on human data or curricula, DreamerV3 becomes the first algorithm to autonomously collect diamonds in Minecraft

Extended Readings

- **World Models**
- **Learning Latent Dynamics for Planning With Pixels (PlaNet)**
- **Dream to Explore: Adaptive Simulations for Autonomous Systems**
- **Mastering Atari with Discrete World Models (DreamerV2)**
- **Latent Skill Planning for Exploration and Transfer**
  - How can Dreamers learned knowledge of the environment transfer to new tasks