Dream to Control: Learning Behaviors by Latent Imagination

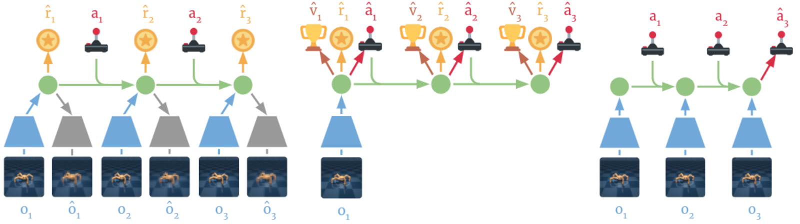
Francesco Spena 1911090

Outline

- Idea Paper
- World Models
- Environment
- Dimension State Space
- Transition Model
- Reward Model
- Dreamer
- Results

Idea of the Paper

- **World Model** Architecture.
- **Dreamer:** An agent that learns behaviors by propagating value gradients through imagined trajectories in the latent space of the world model.



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

World Models

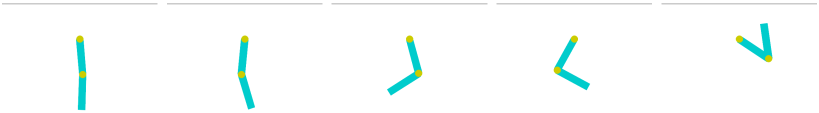
- Type of architecture with an internal representation of the world to simulate experiences

- Three **main** components:
  - **Representation Model:** Transform high-dimensional observations into a latent space.
  - **Transition Model:** Predict the state s_{t+1} given (s_t, a_t).
  - **Reward Model:** Predict the reward r_t given the state s_t.

Are they **all necessary** in our case? **NO!**

Environment: Acrobot

- **Description:** Classic Dynamic Control Problem
- **Agent:** Manipulator with 2 revolute joints
- **Goal:** Reach a specific height
- Dimension **state** space: 6
- Dimension **action** space: 3 (1, -1, 0 torque)



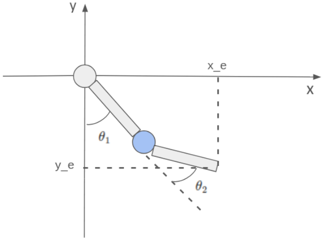
Dimension of the state space

- We can represent the state **without** ambiguity with:

$$\cos(\theta_1) \quad \sin(\theta_1) \quad \cos(\theta_2) \quad \sin(\theta_2) \quad \dot{\theta}_1 \quad \dot{\theta}_2$$

Direct Kinematics:

$$\begin{cases} x_e = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) \\ y_e = -l_1 \cos(\theta_1) - l_2 \cos(\theta_1 + \theta_2) \end{cases}$$



Dimension of the state space (2)

- Compute the time derivative of the direct kinematics:

$$\begin{cases} \dot{x}_e = l_1 \dot{\theta}_1 \cos(\theta_1) + l_2(\dot{\theta}_1 + \dot{\theta}_2)\cos(\theta_1 + \theta_2) \\ \dot{y}_e = l_1 \dot{\theta}_1 \sin(\theta_1) + l_2(\dot{\theta}_1 + \dot{\theta}_2)\sin(\theta_1 + \theta_2) \end{cases}$$

- Dynamic Model:

$$M(\theta)\begin{pmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{pmatrix} + C(\theta, \dot{\theta})\begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{pmatrix} + G(\theta) = \begin{pmatrix} 0 \\ \tau_2 \end{pmatrix}$$

- We update the angles and the angular velocities as:

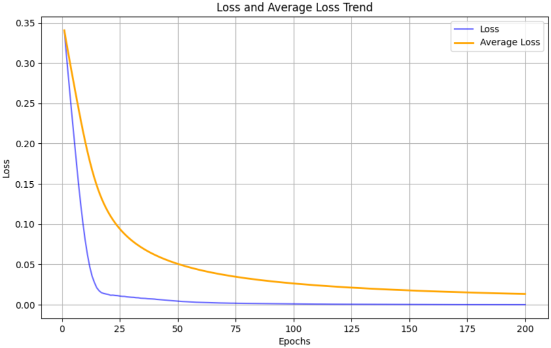
$$\begin{cases} \dot{\theta}_1' = \dot{\theta}_1 + \ddot{\theta}_1 \triangle t \\ \dot{\theta}_2' = \dot{\theta}_2 + \ddot{\theta}_2 \triangle t \\ \theta_1' = \theta_1 + \dot{\theta}_1' \triangle t \\ \theta_2' = \theta_2 + \dot{\theta}_2' \triangle t \end{cases} \Rightarrow \begin{cases} \cos(\theta_1') = \cos(\theta_1 + \dot{\theta}_1' \triangle t) \\ \sin(\theta_1') = \sin(\theta_1 + \dot{\theta}_1' \triangle t) \\ \cos(\theta_2') = \cos(\theta_2 + \dot{\theta}_2' \triangle t) \\ \sin(\theta_2') = \sin(\theta_2 + \dot{\theta}_2' \triangle t) \end{cases}$$

Transition Model

- **Neural Network** capable of predicting s_{t+1} given (s_t, a_t):

$$s_{t+1} = f_\theta(s_t, a_t)$$

- **Small State: MLP** with 2 hidden layers with 128 units + ReLU activation function.

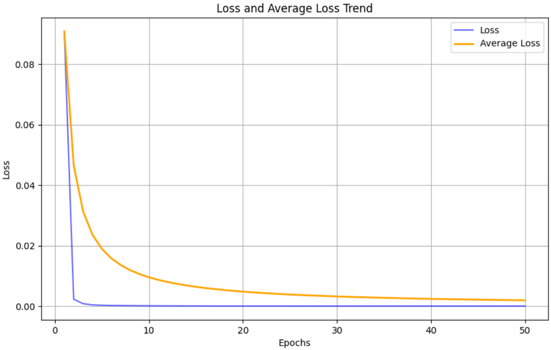


Reward Model

- **Neural Network** capable to predicting r_t given s_t:

$$r_t = f_\theta(s_t)$$

- **Small Action Space:** **MLP** with 2 hidden layers with 32 units + ReLU activation function.



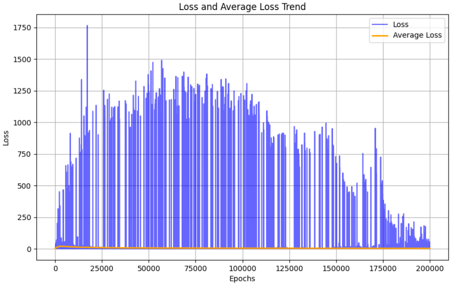
Dreamer

- Based on **actor-critic** schema.
- Main steps:
  - Collect Experiences
  - **Imagine** trajectories (implicit in V_λ)
  - Update **Critic** and **Discount** Model
  - Update **Actor**
- **Actor:** Policy Model **MLP** with 1 hidden layer with 128 units $+$ ELU activation function.
- **Critic:** Value Model **MLP** with 3 hidden layers with (128,64,32)-units $+$ ReLU activation function.

Results: Mean Square Error

Original Loss Function:

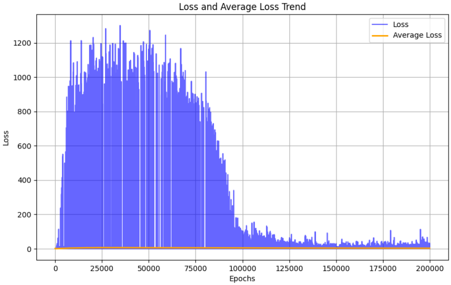
$$MSE = \frac{1}{N}\sum_{i=1}^{N}(V(s_i) - y_i)^2$$



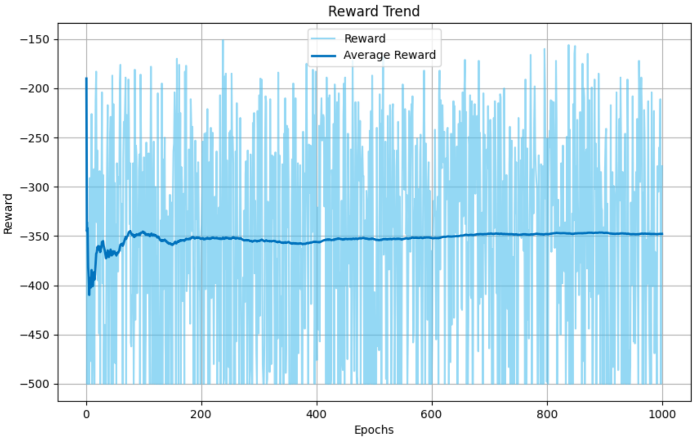
Results: Regularization Based Loss

Stabilization of the training phase:

$$L = MSE + \lambda ||\theta||_2$$

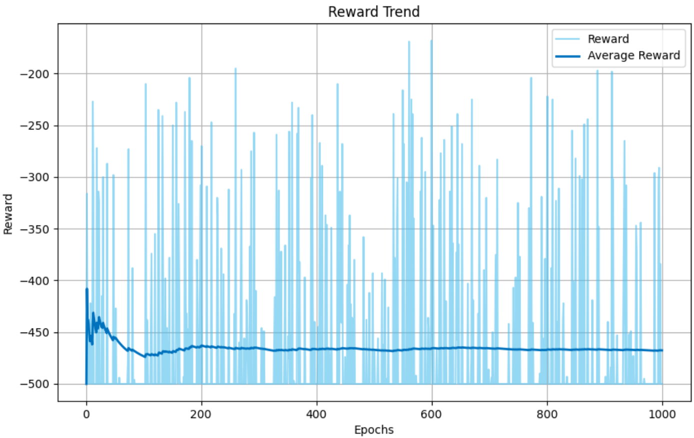


Reward: MSE VS Regularization



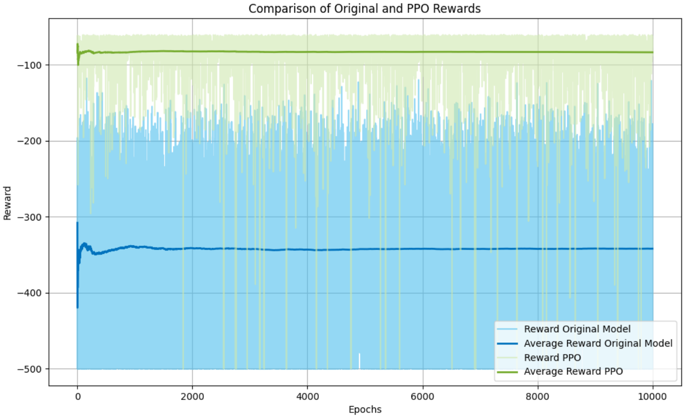
MSE Loss

Reward: MSE VS Regularization



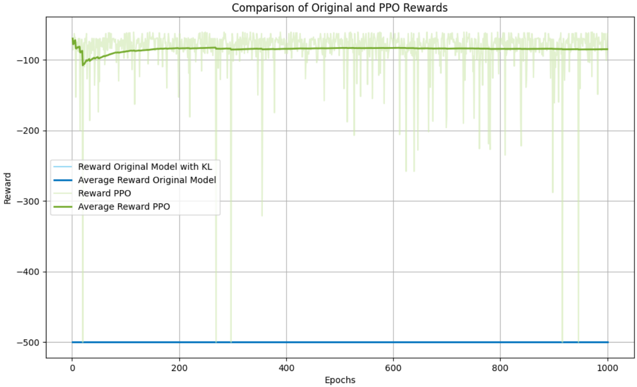
Regularization Loss

Dreamer VS PPO



Dreamer with KL VS PPO

- KL divergence measure the difference between the distribution of the actor and some target.
- **Limited exploration!**



Video Demonstration

10 seconds

5 seconds