COMP4434 Group Project

Group 5 Project Presentation

WANG Yuqi - [redacted]
YANG Xikun - [redacted]
CUI Mingyue - [redacted]

Contents

1 Selected Paper

Mastering Diverse Domains through World Models

Journal: Nature 2025
Author: Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, Timothy Lillicrap
Scope: Sample-efficient RL via self-supervised learning (World Models)

Challenge

- **Reinforcement Learning is “BAD”**
  - Lacked generality and robustness
  - Sample Inefficiency - tremendous trial & error
  - Sparse Reward (Credit Assignment Problem)

- **Research Problem**
  - How can we design a sample efficient RL algorithm?
  - How can we overcome sparse reward + high dimension problem

Solution: World Model

Learning Abstract World Models

Learning a description of how the world works

- Predict future outcomes to enable **planning**
- Learn **compact representation** of images
- Form **Markov states** in POMDPs

Application

- **Exploration** by planning from expected surprise
- **improved generalization** from fixed datasets
- Solve many **new tasks** zero-shot

Main Challenge

- Learning **accurate world models** is challenging!!!
- **DreamerV3**: one of the SOTA world model based RL techniques

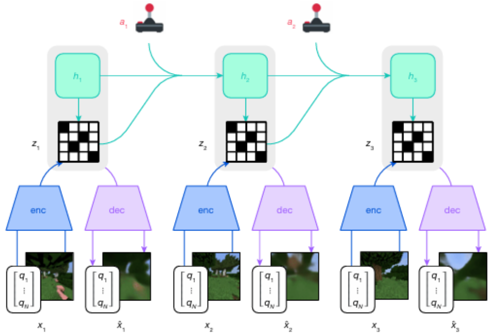


Figure 1: World Model learning

1. **World Model**
   - Encoder: $x_t \rightarrow z_t$ (**categorical discrete latent**)
   - RSSM: $(h_{t-1}, z_{t-1}, a_{t-1}) \rightarrow h_t$
   - Decoder: $z_t \rightarrow \hat{x}_t$ (reconstruction)
   - **Purpose**: Generate massive number of imagined rollouts in latent space

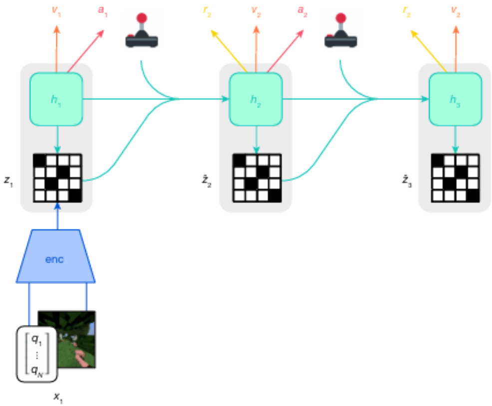


Figure 2: Actor-Critic

2. **Reinforcement Learning**
   - Actor-Critic Learning based on the imagined rollouts generated by the world model.
3. **Various other tricks**
   - Symlog, Discrete World Models, KL Balancing etc.

Datasets (Environments)

- **DeepMind Control Suite**
  - 18 Control tasks based on proprioceptive vector input
  - 20 Continous tasks based on RGB image input
- **Atari** - 26 Atari Games
- **ProcGen** - 16 procedually generated games
- **DMLab** - 30 3D environment tasks
- **Minecraft** - Open-world 3D sandbox game

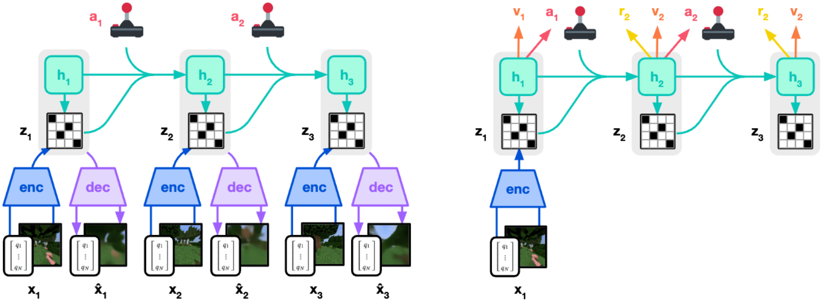
(a) Control Suite **(b)** Atari **(c)** ProcGen **(d)** DMLab **(e)** Minecraft

For our project: 18 proprio-control tasks from Deepmind Control Suite

- Reason: Least computationally demanding

2 Designed Model

DreamerV3 Approach:



(a) World Model Learning

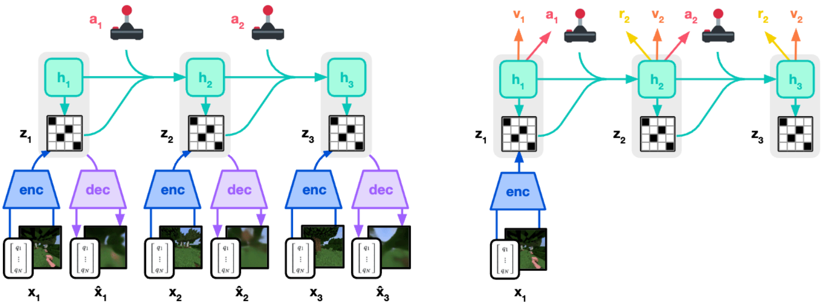
(b) Actor Critic Learning

RECALL: World Model Definition

- Learning a description of how the world works
- Learn **compact representation** of images
- Form **Markov states** in POMDPs

Hypothesized Limitation (ii)

DreamerV3 Approach:



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

Hypothesized Issues:

- Autoencoder inevitably encodes irrelevant details to RL learning
- Pixel uncertainty → necessitate **engineering tricks** for uncertainty modeling

Hypothesized Limitation (iii)

Our Assumptions

- If predict at a sufficiently abstract and high-level representation space
- **No need for uncertainty modeling!**

An Extreme Example:

- You can deterministically predict planetary orbit with 6 variables
  - 3 variables for 3D position
  - 3 variables for velocity vector
- But if predicting planetary orbit at **atomic level**:
  - Tremendous uncertainty!
  - Error accumulation after each particle simulation

Proposed Method - Pipeline Overview

Producer:

```
while is_running:
  episode = []
  while not done:
    latent = world_model.encode(obs)
    action = agent.act(latent)
    obs, reward, done = env.step(action)
    episode.append((latent, action, reward))

  replay_buffer.add(episode)
```

Consumer:

```
while is_running:
  episode = replay_buffer.sample()
  world_model.train(episode)
  agent.train(episode)
```

Proposed Method - World Model

Components

- Encoder: $x_t \to z_t$
- Predictor: $z_t, a_t \to z_{t+1}, r_{t+1}$

Given Variables

- Observed trajectory: $[x_0, x_1, ..., x_t]$
- Action sequences: $[a_0, a_1, ..., a_t]$

Contrastive Training Procedure (`world_model.train()`)

- **Encoder**: encodes trajectory into latents: $[x_0, x_1, ..., x_t] \to [z_0, z_1, ..., z_t]$
- **Predictor**: predicts $[\hat{z}_1, \hat{z}_2, ..., \hat{z}_t]$ from $[z_0] + [a_0, a_1, ..., a_{t-1}]$
- Compute $\cos(z, \hat{z})$ cosine similarity matrix:

$$\begin{pmatrix} \cos(z_1, \hat{z}_1) & \dots & \cos(z_1, \hat{z}_t) \\ \vdots & \ddots & \vdots \\ \cos(z_t, \hat{z}_1) & \dots & \cos(z_t, \hat{z}_t) \end{pmatrix}$$

- `F.cross_entropy` maximize value on the diagonal, and minimize everything else

Intuition

- This **avoids reconstruction** in autoencoders
- Speeds up training and prevent encoding irrelevant visual details

Proposed Method - RL Agent

Algorithm:
- Proximal Policy Optimization (PPO)

During Training
- Data: z_t, a_t, z_{t+1}, r_t (state, action, next state, reward)
- Input: z_t (current observation/state)
- Output: $a_t \sim \pi(\cdot \mid z_t), V(z_t)$ (sampled action & state value estimate)
- Training objective:

$$L_{\text{total}} = \text{clamp}(L_{\text{CLIP}}, 1 - \varepsilon, 1 + \varepsilon) + c_1 L_{\text{VF}} - c_2 S[\pi(\cdot \mid s_t)]$$

Where:
  - L_{CLIP}: Clipped policy gradient
  - L_{VF}: Value function MSE loss
  - S: Entropy regularization

During Inference $z_t \to a_t \sim \pi(\cdot \mid z_t)$ (single-step action selection)