World Models

AUTHORS: DAVID HA, JÜRGEN SCHMIDHUBER

PRESENTED BY: OGNEN PENDAROVSKI

Motivation

- Key ideas:
  - Every action we take is based on a learned model of the world
  - Model-free RL is computationally expensive to simulate
- Can we train an agent with “observations” sampled from a learned world model, rather than a costly simulation?

Background

- Model-free methods were popular at the time: DQN for Atari games, A3C and PPO for robotics.
- Model-based methods were rare because of difficulty of making environments but had occasional successes.
- Synthesis: build imperfect but good enough model to sample instead of simulations.

Problem

- To train large neural network-based general agents, two main problems must be solved:
  - Credit assignment: difficult to update millions of weights with sparse reward signals - need smaller model to effectively iterate.
  - High dimensionality: the space of possible images is vast - need strong compression.
- Solution: use separate models.

V–M–C architecture

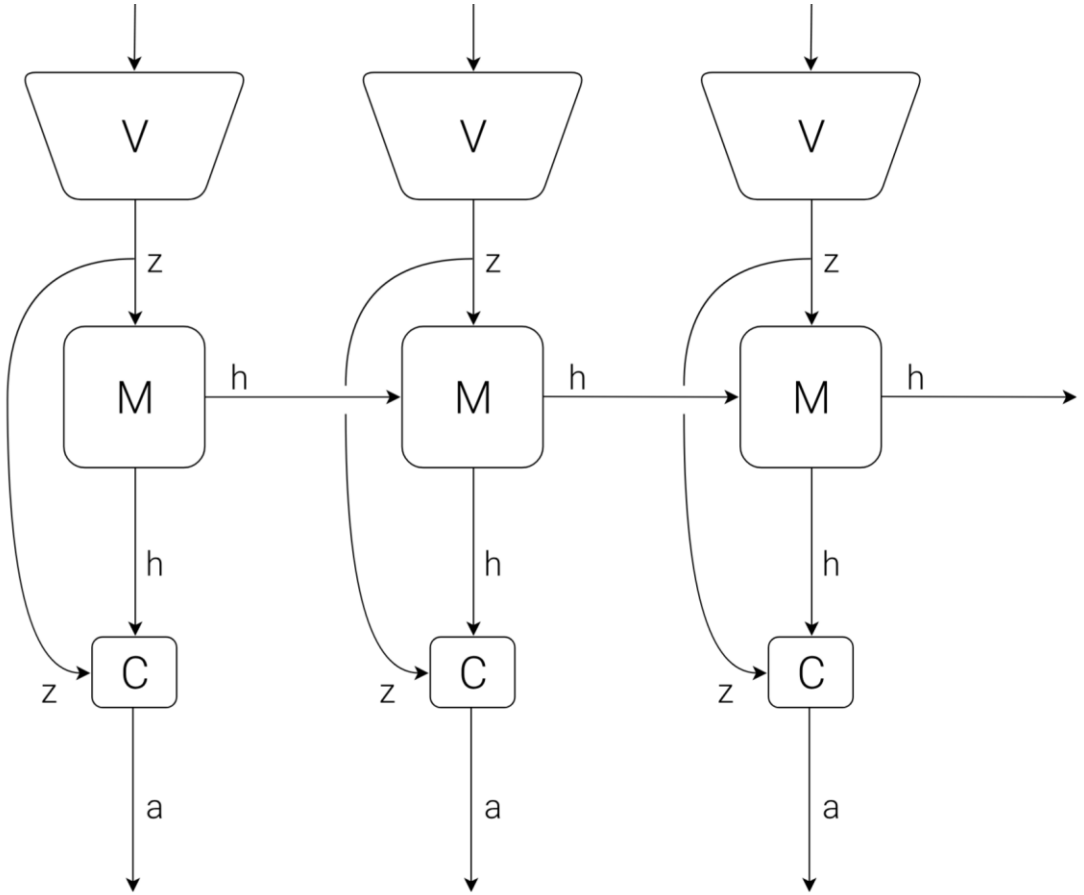
World Model

The **Vision Model (V)** encodes the high-dimensional observation into a low-dimensional latent vector.

The **Memory RNN (M)** integrates the historical codes to create a representation that can predict future states.

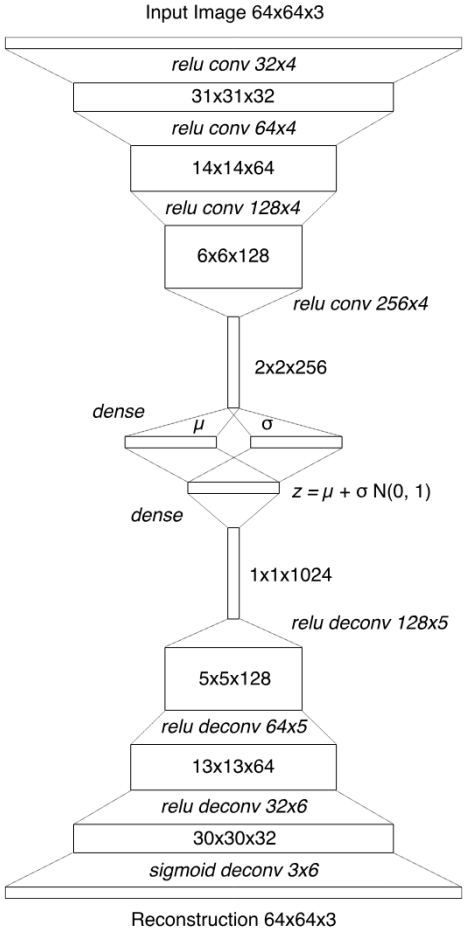
A small **Controller (C)** uses the representations from both **V** and **M** to select good actions.

The agent performs **actions** that go back and affect the environment.



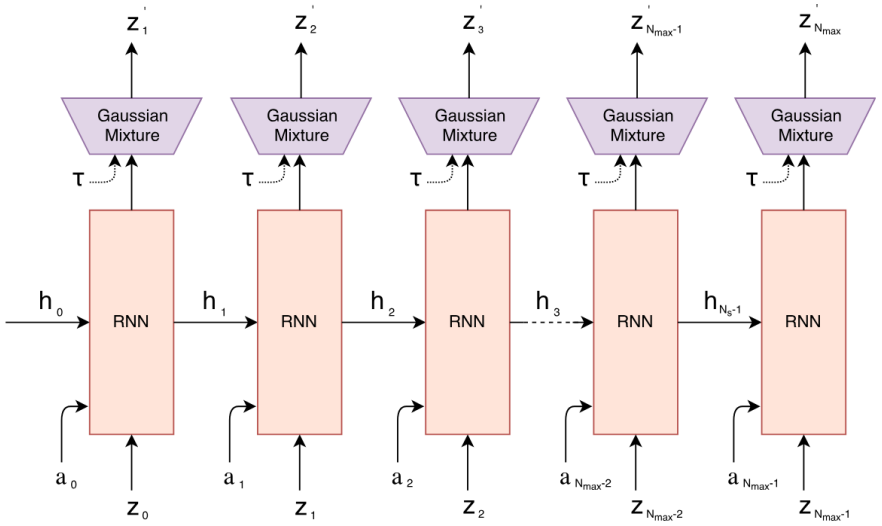
Vision component

- The vision component (V) of the architecture is a variational autoencoder (VAE).
- The encoder consists of four convolutional layers which compress the input image into two low-dimensional latent vectors: μ and σ.
- The encoder output z is sampled from the Gaussian distribution $N(\mu, \sigma I)$.
- For training purposes, z is passed through a symmetrical set of deconvolution layers to reconstruct the image.



Memory component

- The memory module (M) is represented by an RNN (specifically LSTM) with a Mixture Density Network (MDN) at the output layer.
- The memory component takes in latent vector z_t, action a_t, hidden state h_t.
- At the output, the memory component gives a probability density function of the next latent vector z_{t+1}, modified by τ.



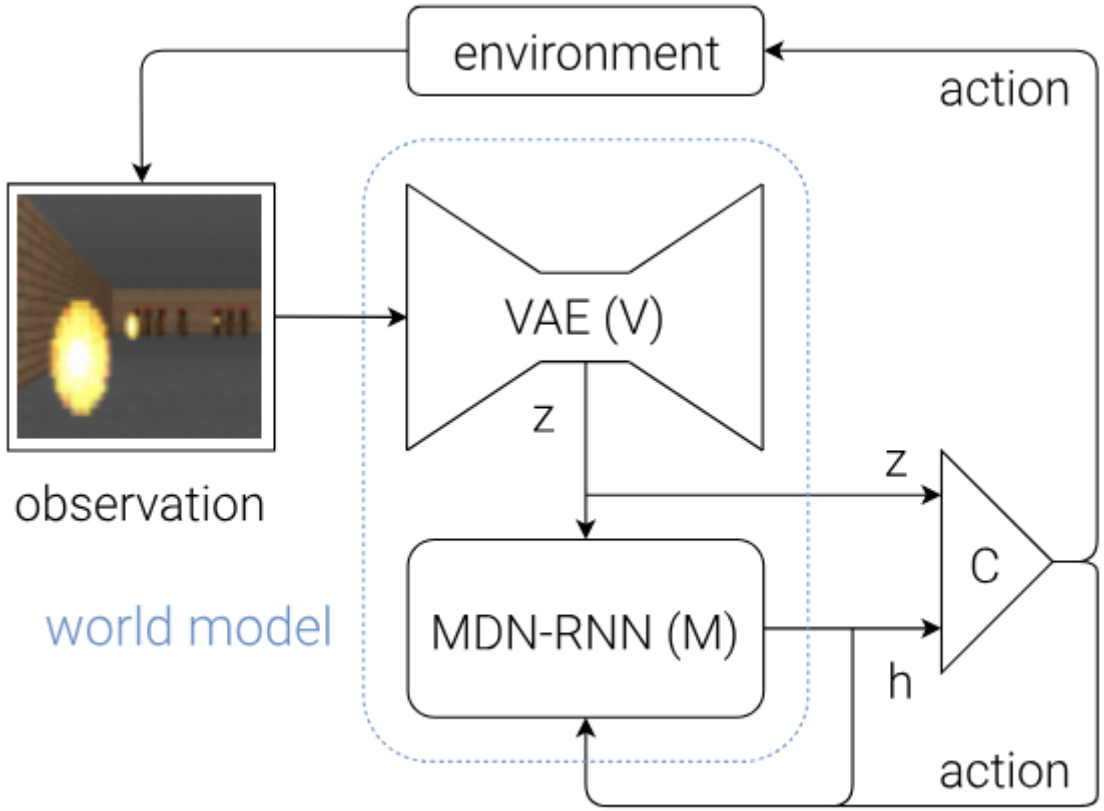
Controller component

- The controller component (C) is a very simple single layer perceptron.
- Because the vision and memory components contain all of the complexity, the controller can be tiny, allowing for easy credit assignment and policy switching.

$$a_t = W_c[z_t h_t] + b_c$$

- Because the model is so small, backpropagation is not needed. The authors used Covariance Matrix Adaptation Evolution Strategy (CMA-ES).

Architecture flow diagram



Evaluation – car racing

- Randomly generated tracks, reward given for visiting most in shortest time.
- Actions: steering, acceleration, braking.

Model	Parameter Count
VAE	4,348,547
MDN-RNN	422,368
Controller	867

Evaluation – car racing

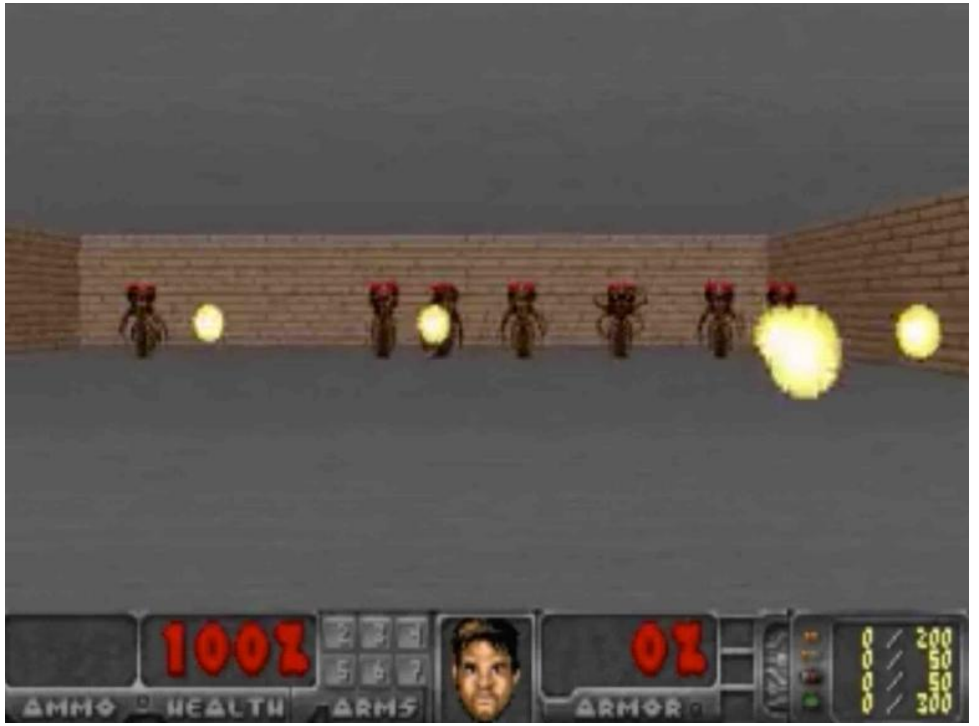
- Procedure:
  1. Collect 10,000 rollouts from random policy.
  2. Train VAE to encode frames in $z \in \mathbb{R}^{64}$.
  3. Train MDN-RNN to model $P(z_{t+1}|a_t, z_t, h_t)$.
  4. Define controller as $a_t = W_c[z_t h_t] + b_c$.
  5. Use CMA-ES to solve for W_c and b_c that maximize the cumulative expected reward.
- The policy trained in the real environment works in the "dream" environment.

METHOD	AVG. SCORE
DQN (PRIEUR, 2017)	343 ± 18
A3C (CONTINUOUS) (JANG ET AL., 2017)	591 ± 45
A3C (DISCRETE) (KHAN & ELIBOL, 2016)	652 ± 10
CEOBILLIONAIRE (GYM LEADERBOARD)	838 ± 11
V MODEL	632 ± 251
V MODEL WITH HIDDEN LAYER	788 ± 141
FULL WORLD MODEL	$\mathbf{906 \pm 21}$

Evaluation – VizDoom

- Why not do the inverse – train an agent in a low fidelity dream and transfer it to the real world?
- Agent must avoid fireballs shot by monsters – no explicit rewards, only number of survival timesteps (max 2100, win criterion > 750).

Model	Parameter Count
VAE	4,446,915
MDN-RNN	1,678,785
Controller	1,088



Evaluation – VizDoom

- With sufficiently accurate world model, it can act as the environment.
- The agents trains only on the latent representations of the images.
- The real environment and the "dream" have the same interface – agents trained for the latter can directly operate in the former.
- This works, but the model is imperfect, and the agent can utilize these glitches for reward hacking – mitigate with τ.

TEMPERATURE τ	VIRTUAL SCORE	ACTUAL SCORE
0.10	2086 ± 140	193 ± 58
0.50	2060 ± 277	196 ± 50
1.00	1145 ± 690	868 ± 511
1.15	918 ± 546	1092 ± 556
1.30	732 ± 269	753 ± 139
RANDOM POLICY	N/A	210 ± 108
GYM LEADER	N/A	820 ± 58

Impact

- PlaNet (2019) – used RSSM instead of VAE + RNN and used planning rather than perceptron, solved control tasks in learned latent space, reducing data vs model free methods.
- MuZero (2020) – combines tree-search with learned models, superhuman performance in various games without prior knowledge.
- Dreamer model series (2020, 2022, 2024) – learned actor-critic policy in latent space, learns long-horizon behaviours and strong continuous control, first human-level Atari performance across 55 tasks, and first algorithm to collect diamonds in Minecraft.

Opinion

- Strengths:
  - Highly abstractable
  - SOTA performance for many tasks
  - Potential for lower computational overhead vs model-free methods
- Weaknesses:
  - More detailed comparisons needed for sample efficiency - makes sense in theory but empirical backing could be provided
  - Potential for vastly incorrect/biased world models to be created - could exacerbate reward hacking and induce errors

References

- Ha, D., Schmidhuber, J. (2018). World Models. arXiv preprint, arXiv:1803.10122.
- Hafner, D. et al. (2019). Learning Latent Dynamics for Planning from Pixels. arXiv preprint, arXiv:1811.04551.
- Schrittwieser, J., Antonoglou, I., Hubert, T., Silver, D. et al. (2020). Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. arXiv preprint, arXiv:1911.08265.
- Hafner, D. et al. (2020). Dream to Control: Learning Behaviors by Latent Imagination. arXiv preprint, arXiv:1912.01603.
- Hafner, D. et al. (2022). Mastering Atari with Discrete World Models. arXiv preprint, arXiv:2010.02193.
- Hafner, D. et al. (2024). Mastering Diverse Domains through World Models. arXiv preprint, arXiv:2301.04104.