UiO : University of Oslo

institutt for informatikk

# IN3050/IN4050 - Introduction to Artificial Intelligence and Machine Learning

## Ethical Issues, Risks and Fundamental Limitations

Kai Olav Ellefsen

# The next weeks

- Next week: We’ll go through the suggested solution to the trial exam
- The week after: We’ll go through last year’s exam
- We encourage you to **try them yourselves** first

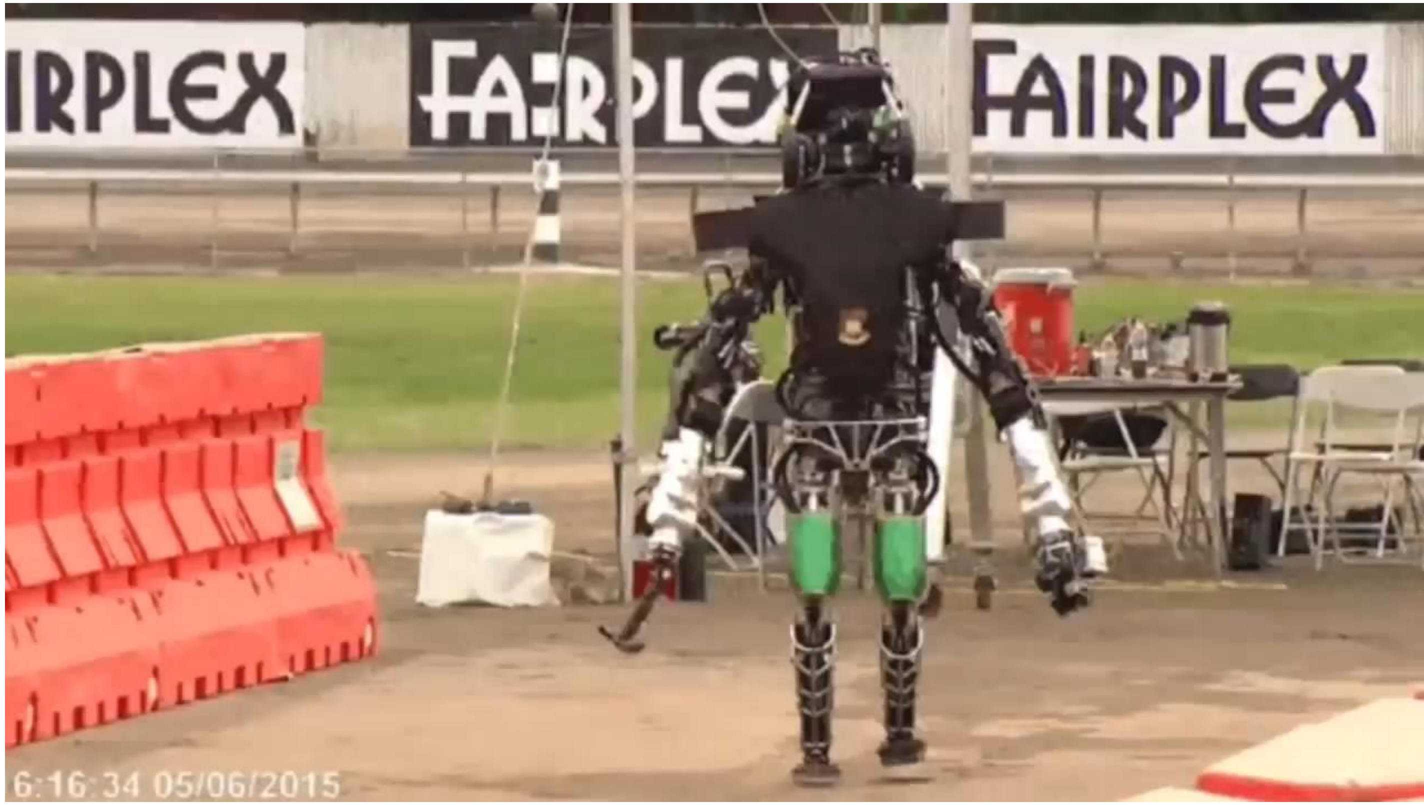
# Cooling down the Hype

**So far in this course:**

- What AI systems are good at
- Useful/beneficial things AI can do

**Next week:**

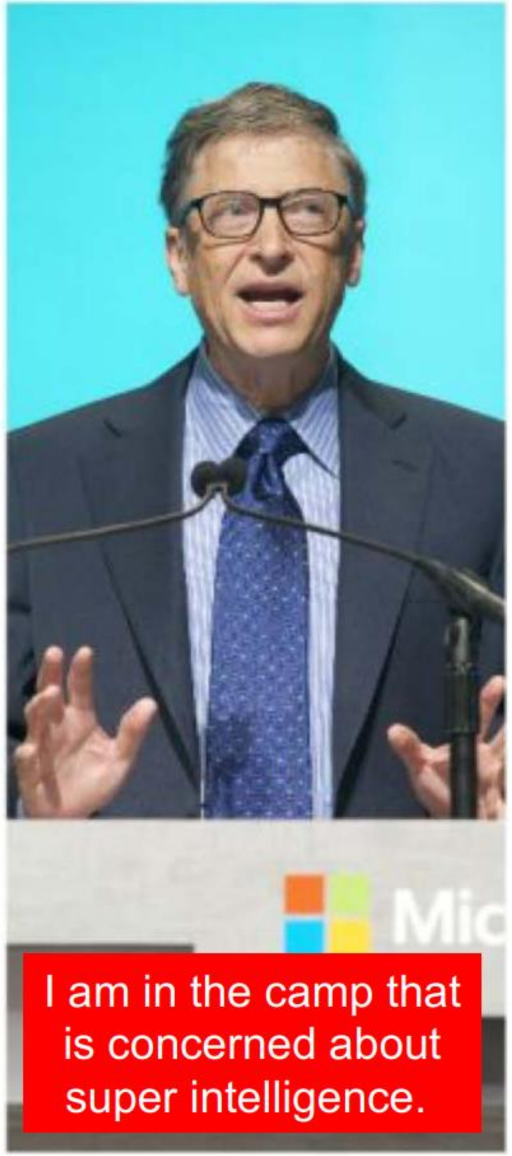
- Limitations of AI
- Potential dangers/negative effects of AI



# Risks/Ethical Issues

- Job loss
- Superintelligence
- Biases/fairness
- Consciousness
- Ethical dilemmas
- Privacy





# Limitations/Challenges for Future AI Researchers

- Robustness
- Understanding «common sense»
  - Language
  - Images
  - Causality
- Explainability
- Continuous learning
- Extremely data-inefficient learning

Fig. 6. Author 3 predicted(90.2%) as famous Norwegian cross country skier Petter Northug.

# Today's plan

- Focus a key AI limitation
- Present state-of-the-art research into solving it
  - Using many techniques you learned in the course

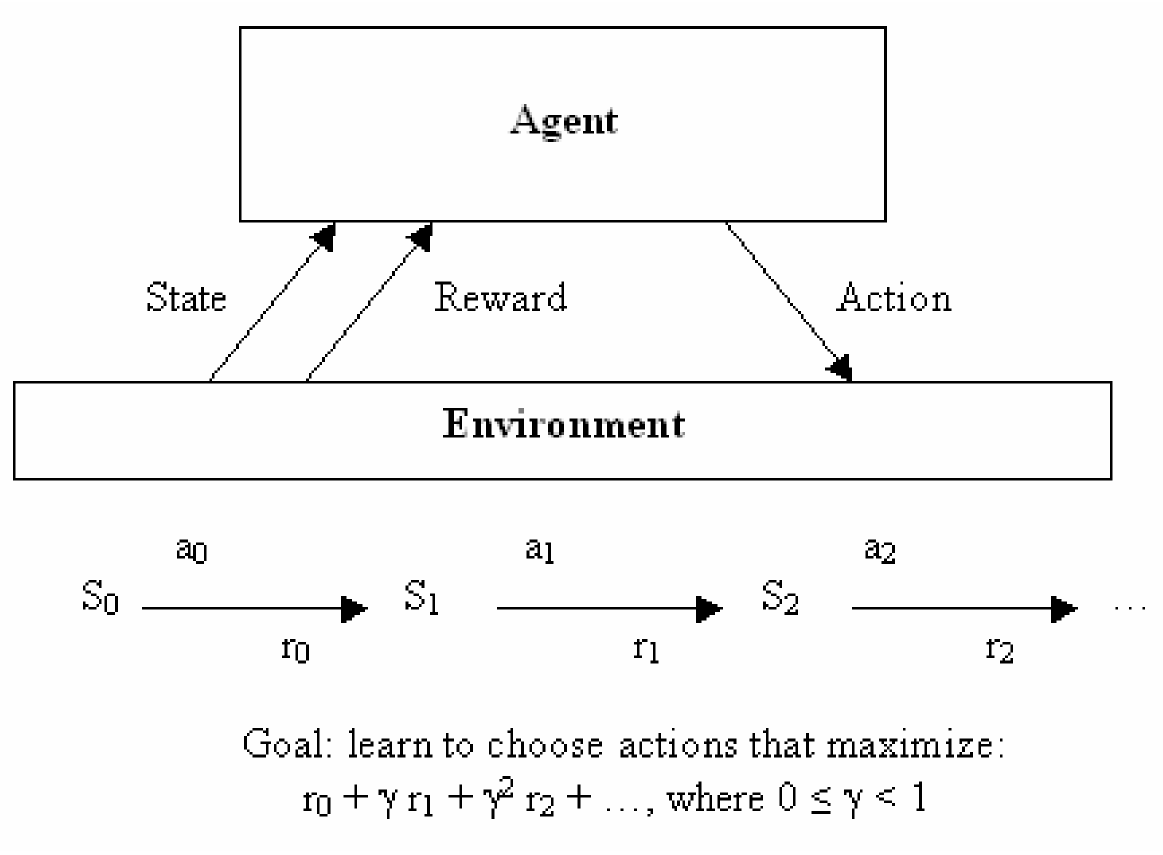
- 2 goals:
  - Another example of the relevance of the techniques you have seen
  - A «bridge» into more advanced AI courses

# Limitations/Challenges for Future AI Researchers

- **Robustness**
- **Understanding «common sense»**
  - Language
  - Images
  - Causality
- Explainability
- Continuous learning
- Extremely data-inefficient learning

Fig. 6. Author 3 predicted(90.2%) as famous Norwegian cross country skier Petter Northug.

# Reinforcement Learning



# No Understanding of the Environment -> Fragile Policy

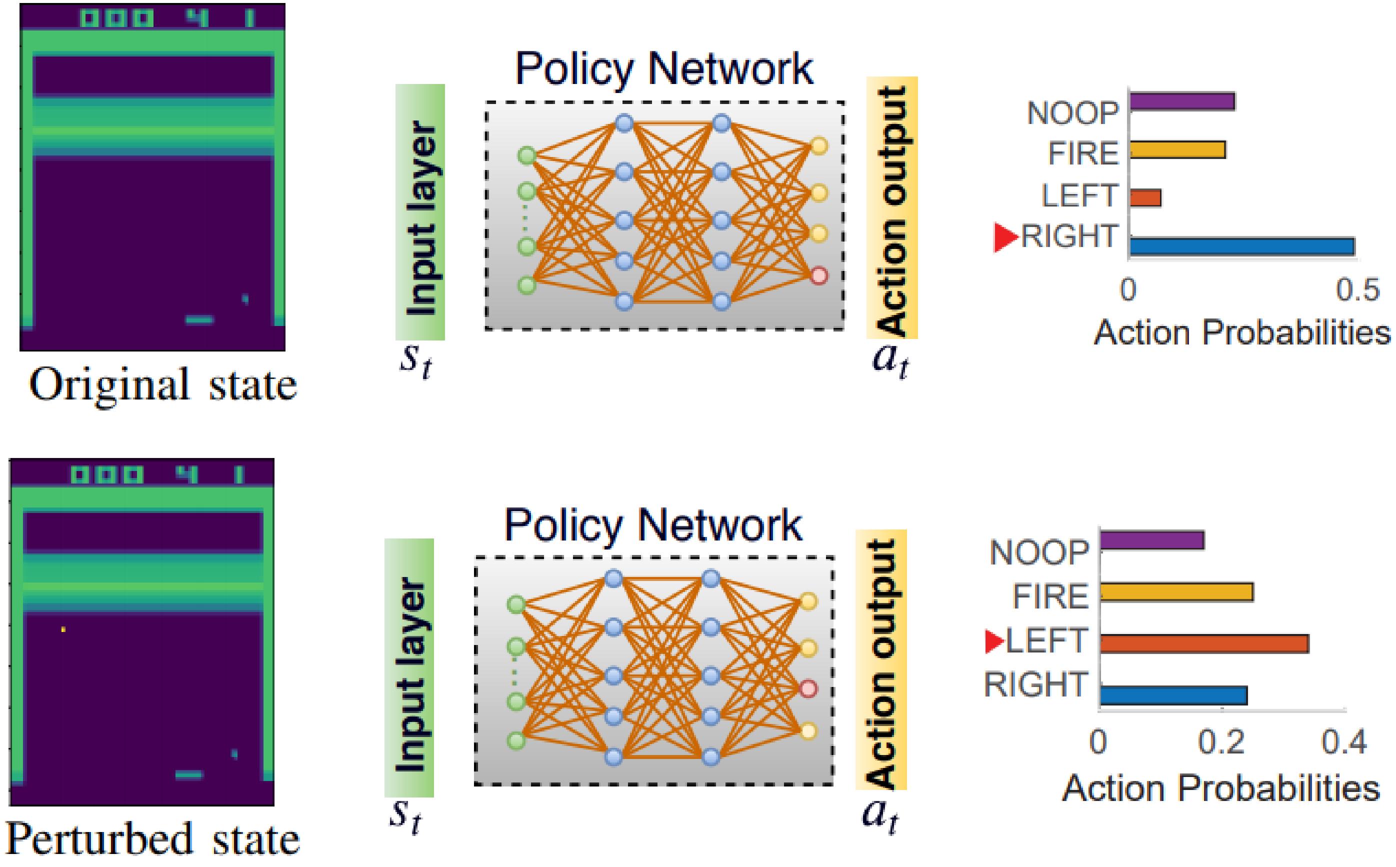
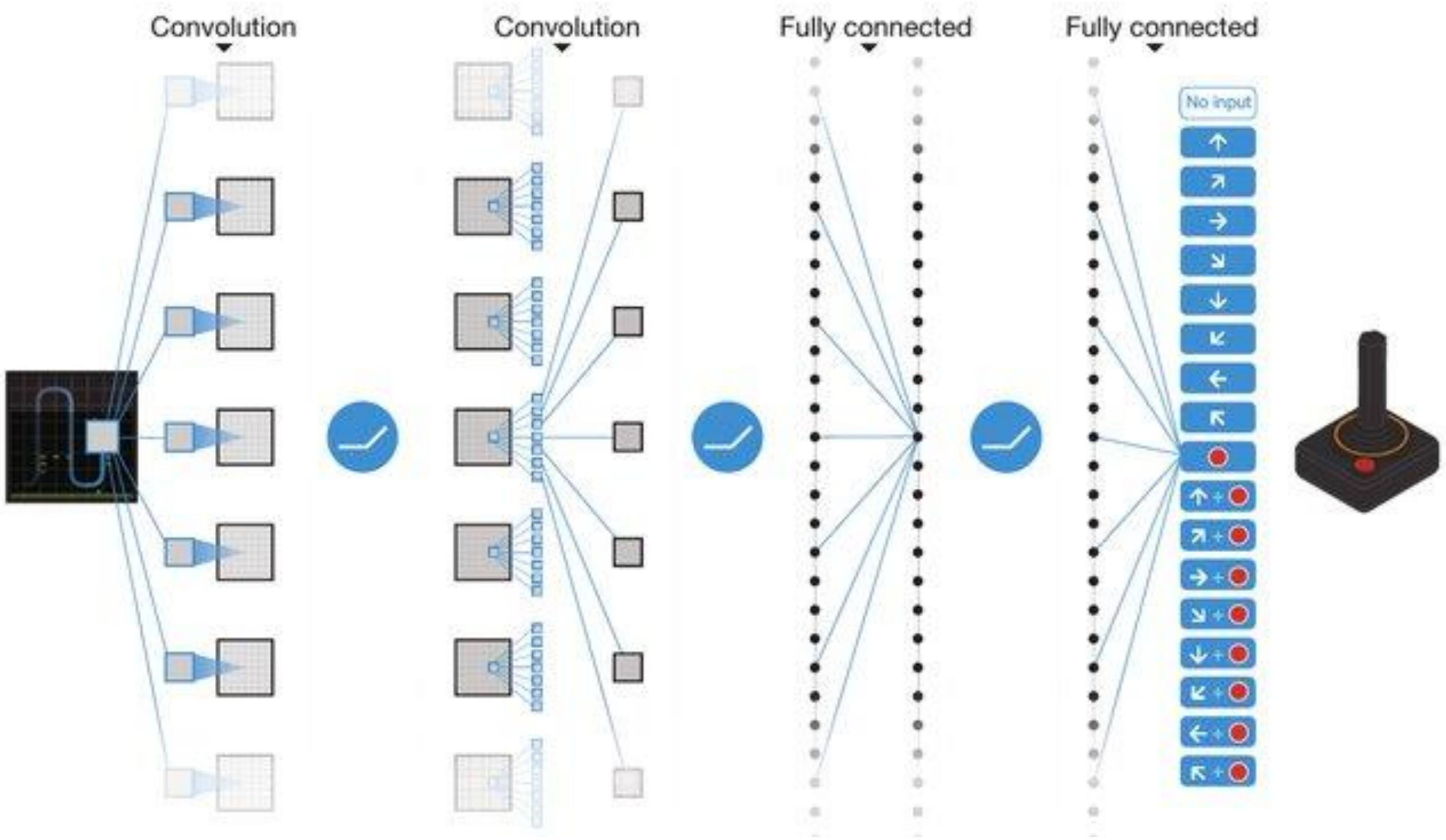



Fig. 1: The visualization of the single pixel attack on Breakout.



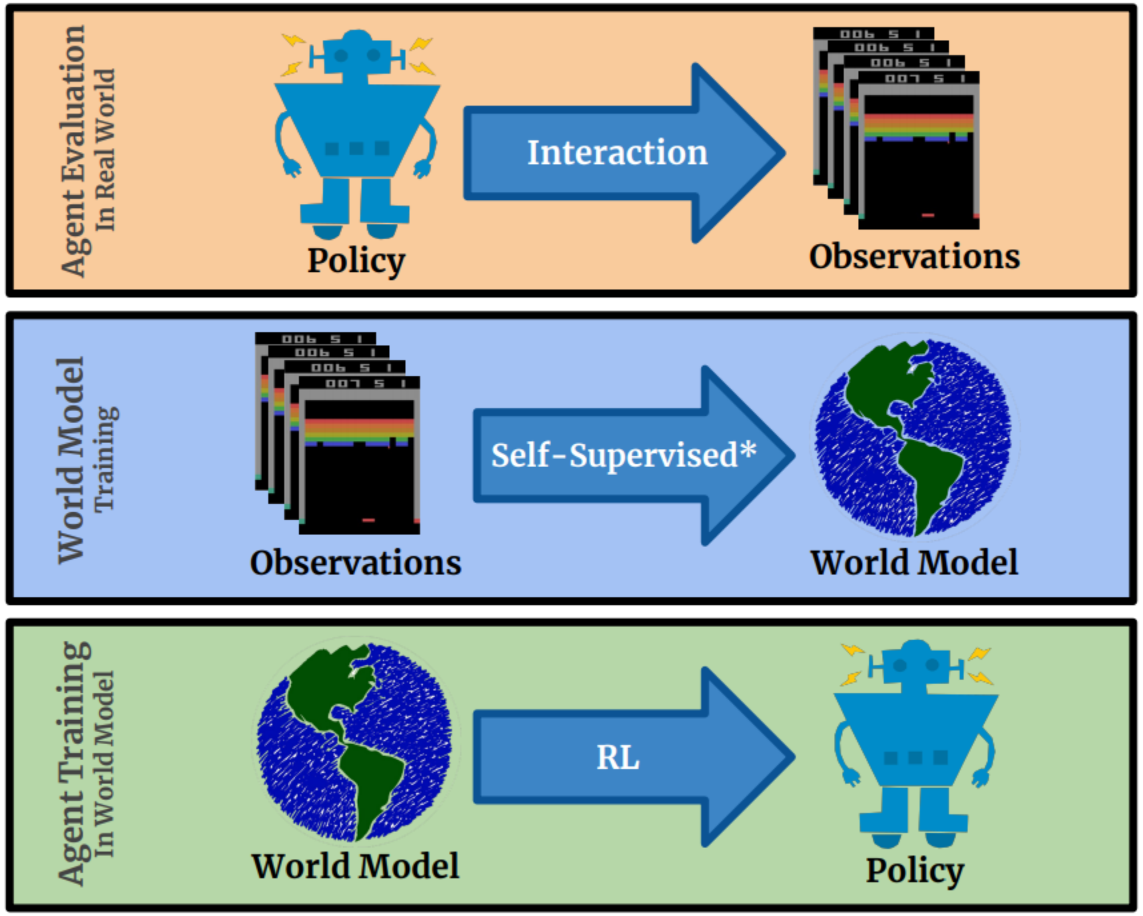
Qu, Xinghua, et al. "Minimalistic Attacks: How Little it Takes to Fool Deep Reinforcement Learning Policies." *IEEE Transactions on Cognitive and Developmental Systems* (2020).

# Model-free Reinforcement Learning: Learn without an explicit predictive model




Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *nature* 518.7540 (2015): 529-533.

# Model-based Reinforcement Learning:



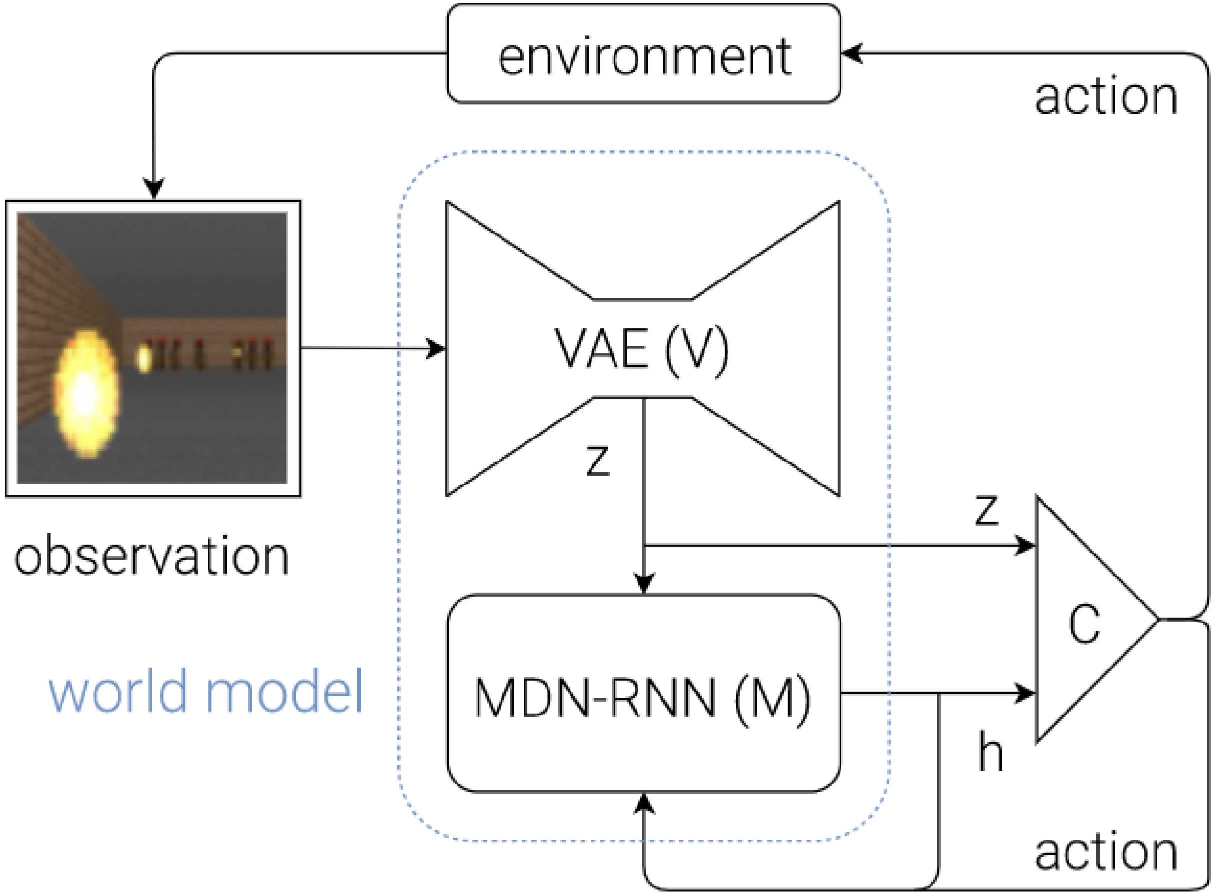

Kaiser, Lukasz, et al. "Model-based reinforcement learning for atari." *arXiv preprint arXiv:1903.00374* (2019).

# Model-Based RL

- Aims to solve some key problems with model-free RL:
  1) It requires very large amounts of training data
  2) It can allow efficient *transfer learning*: A single predictive model could be used to learn new tasks in the same environment.
  3) It can make agents more robust as they *understand* their effect on the environment
- Even so, model-free approaches have so far been most successful.

# World Models

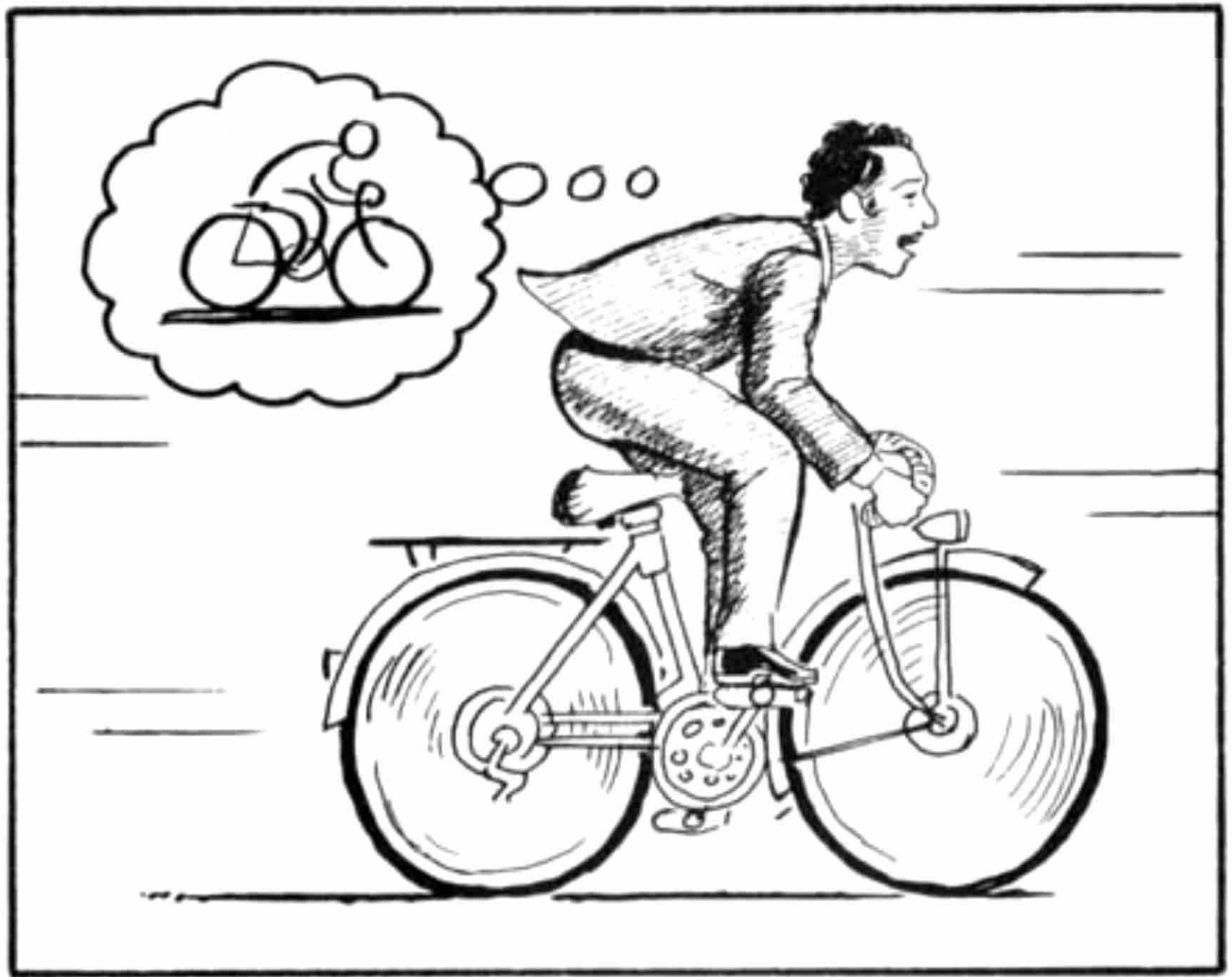
- We'll look at a SOA Model-based RL algorithm
- It illustrates several topics from this course:
  - Autoencoders
  - RNNs
  - (Neuro)evolution
  - Reinforcement Learning
  - Unsupervised Learning




David Ha & Jürgen Schmidhuber: «World Models», NeurIPS 2018
https://worldmodels.github.io/

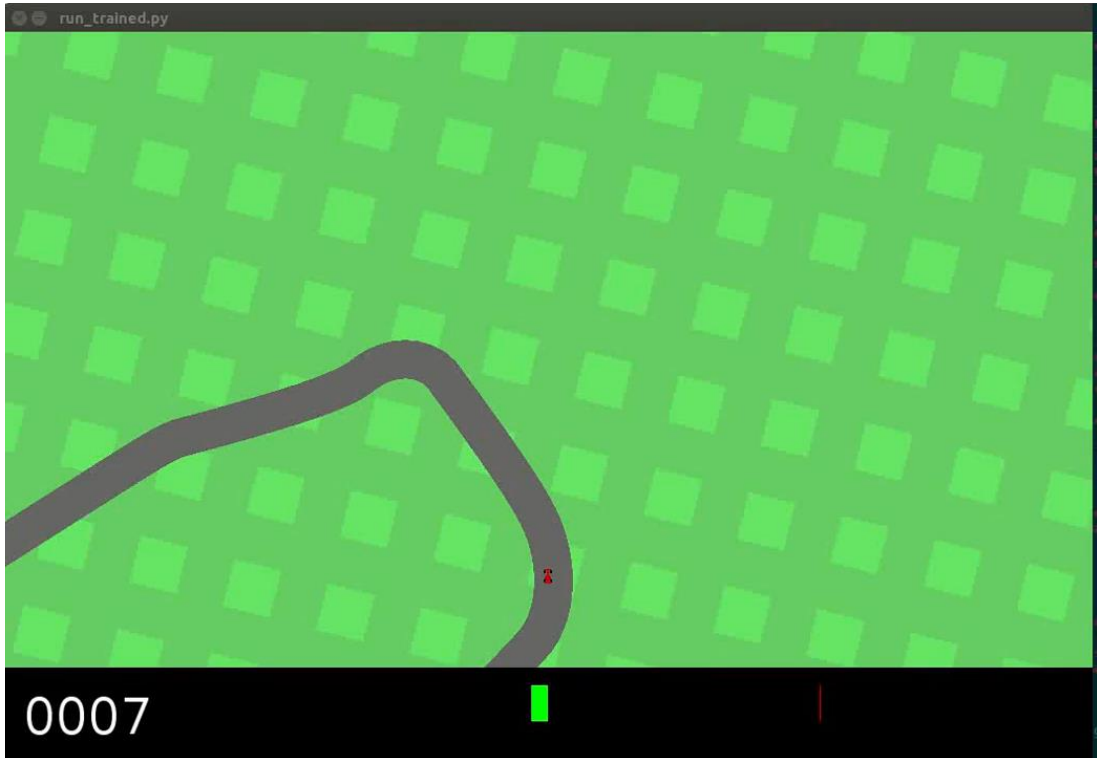
# The idea

- With a predictive model of our world, we can make better decisions
- We can learn a large predictive model of the world in an *unsupervised fashion*
- Then, we can learn a simpler controller using RL and input from the predictive model

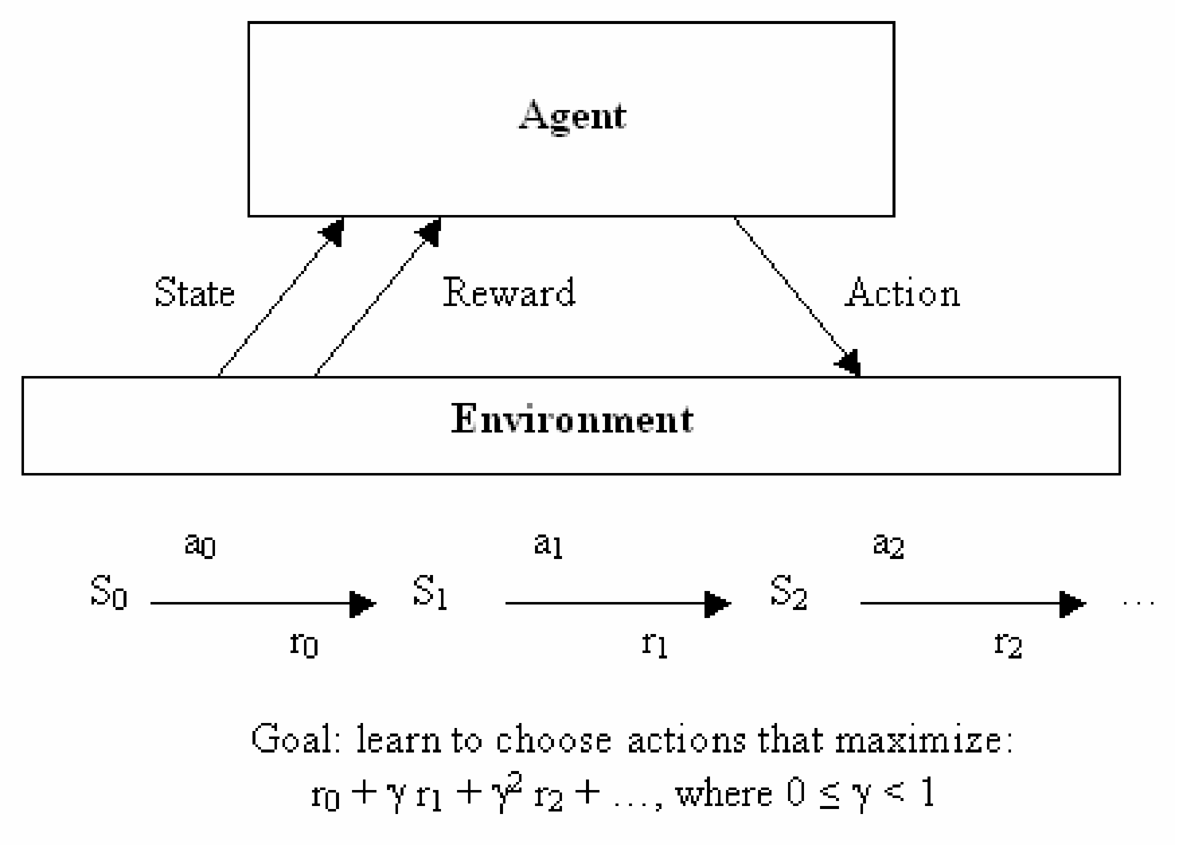


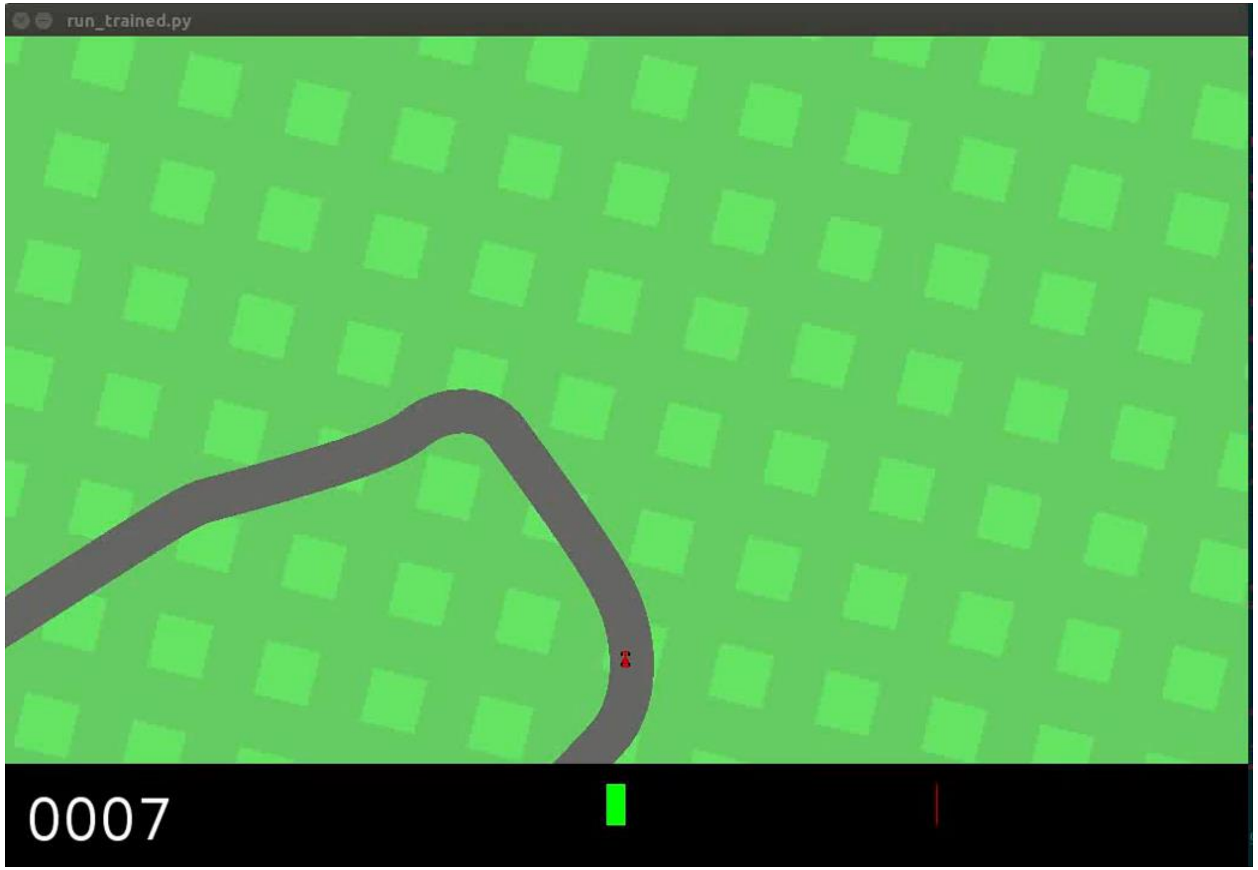
Source: https://worldmodels.github.io/

# The RL environments

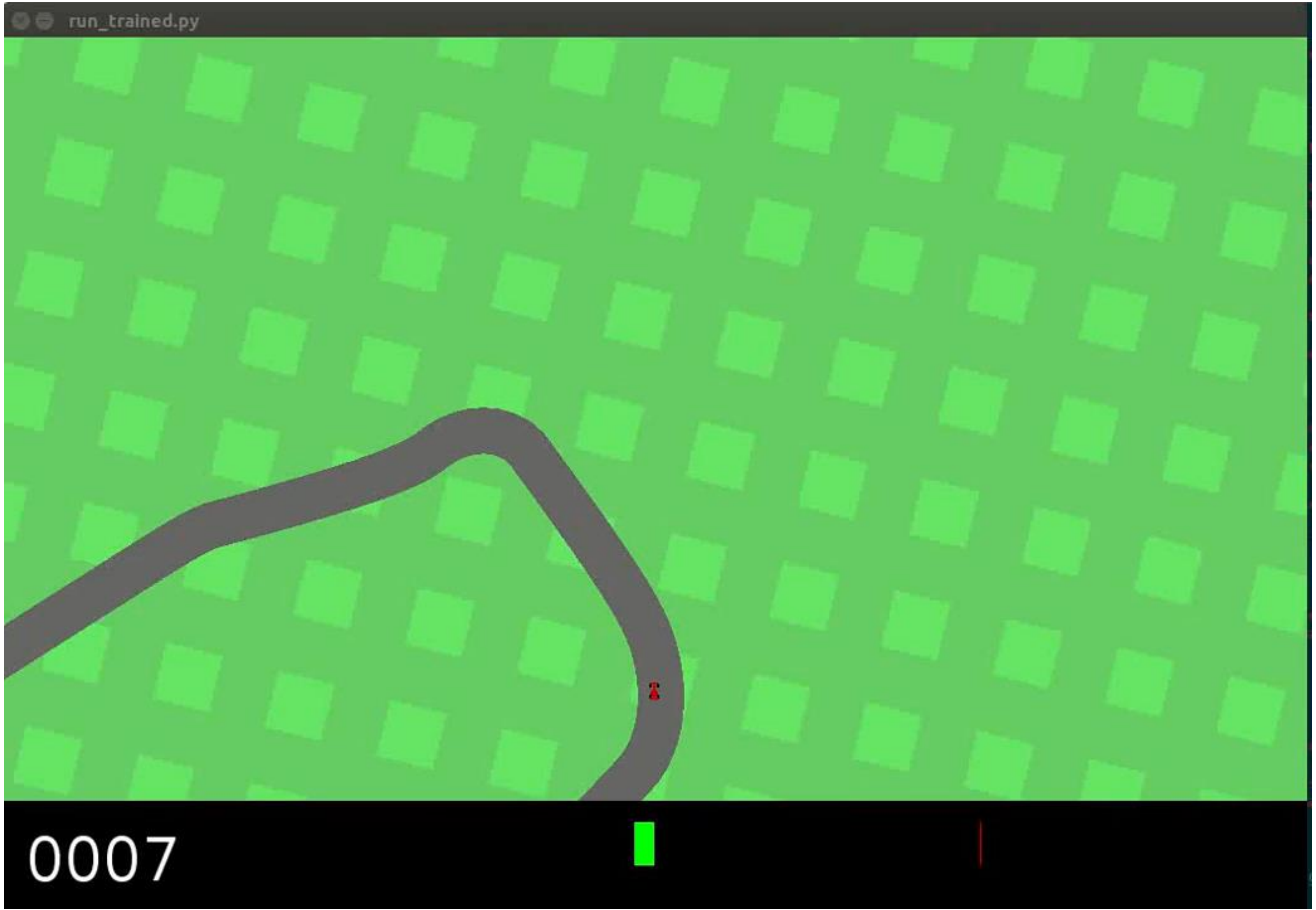



# Reinforcement Learning - Reminder



# Quiz: What are states, actions, rewards?



At each time step, our agent receives an **observation** from the environment.

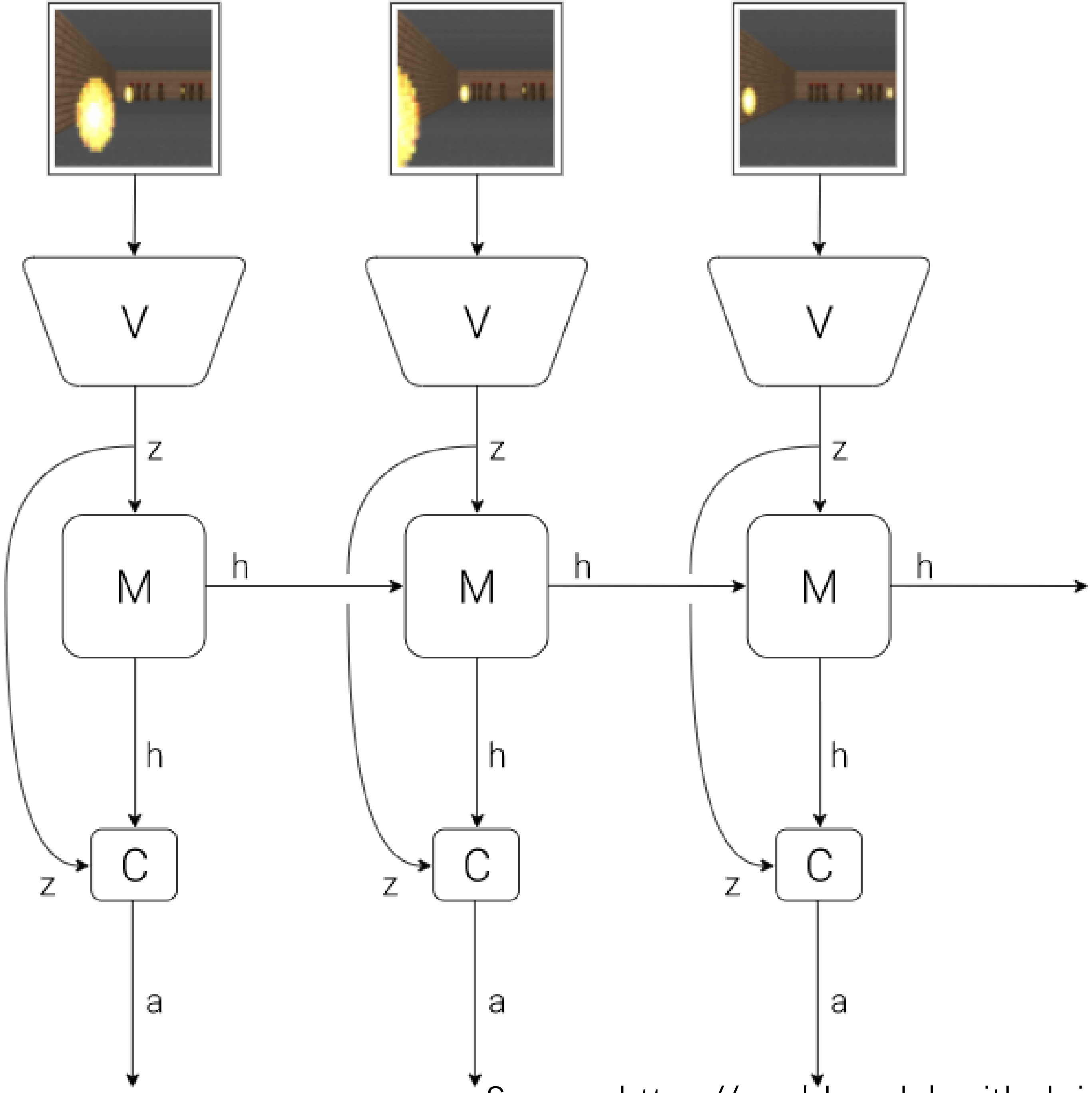
### World Model

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.

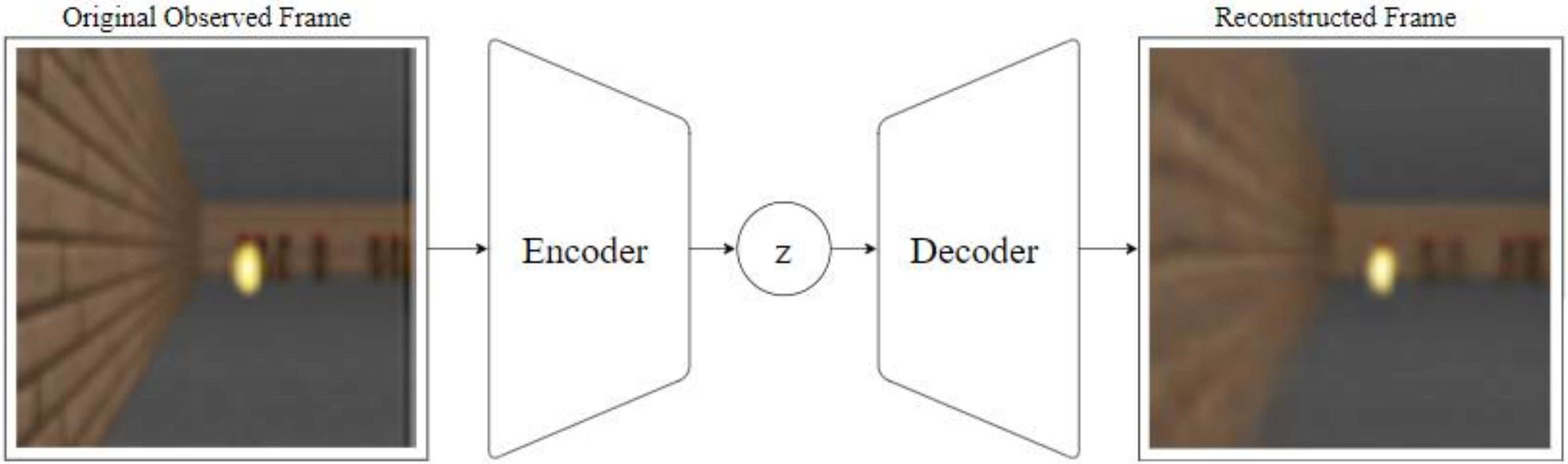
A small **Controller (C)** uses the representations from both **V** and **M** to select good actions.

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




Source: https://worldmodels.github.io/

# V: Variational AutoEncoder (VAE)




Flow diagram of a Variational Autoencoder. [31, 32]

Source: https://worldmodels.github.io/

# Demo

- https://worldmodels.github.io/

At each time step, our agent receives an **observation** from the environment.

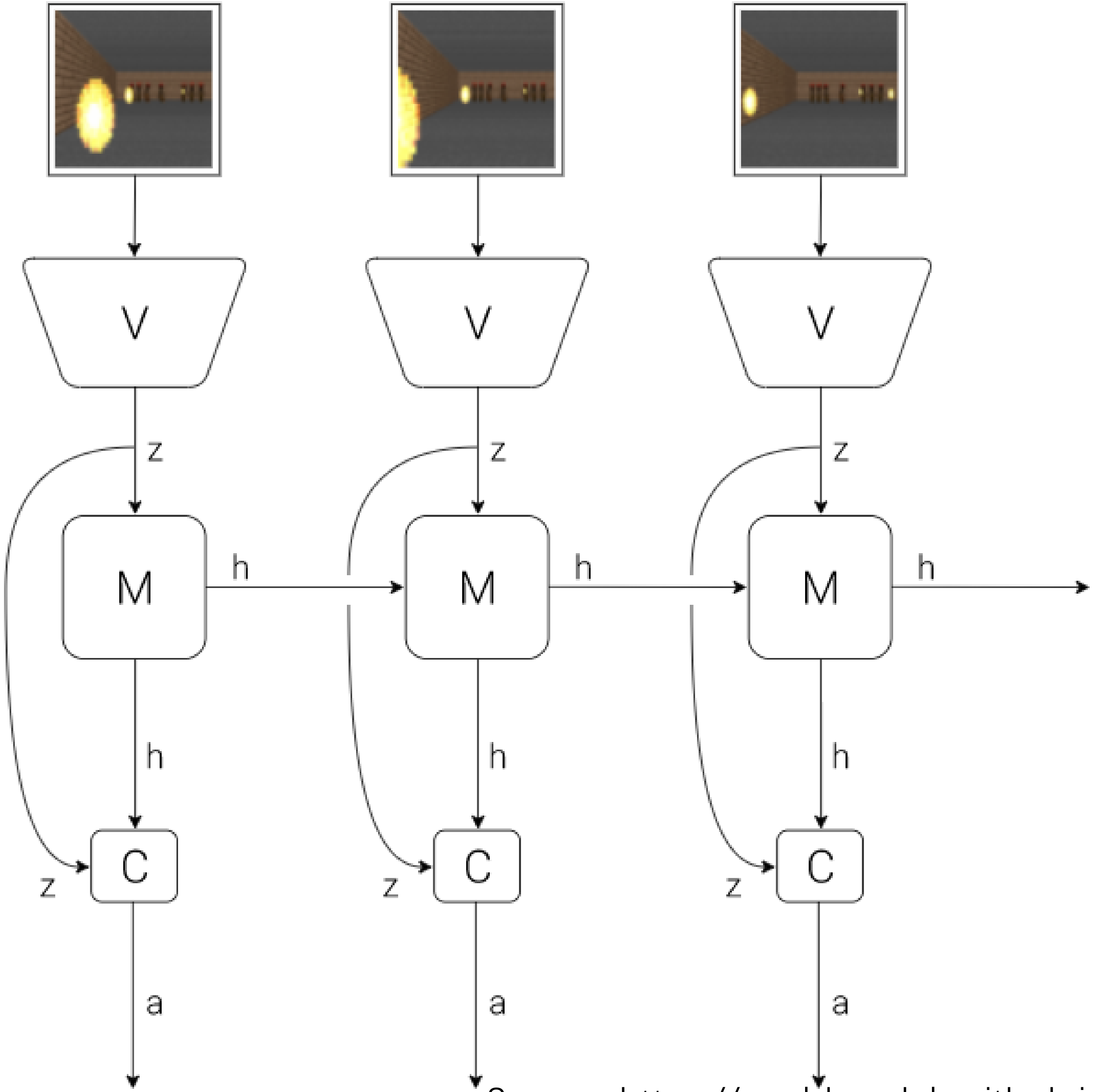
**World Model**

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

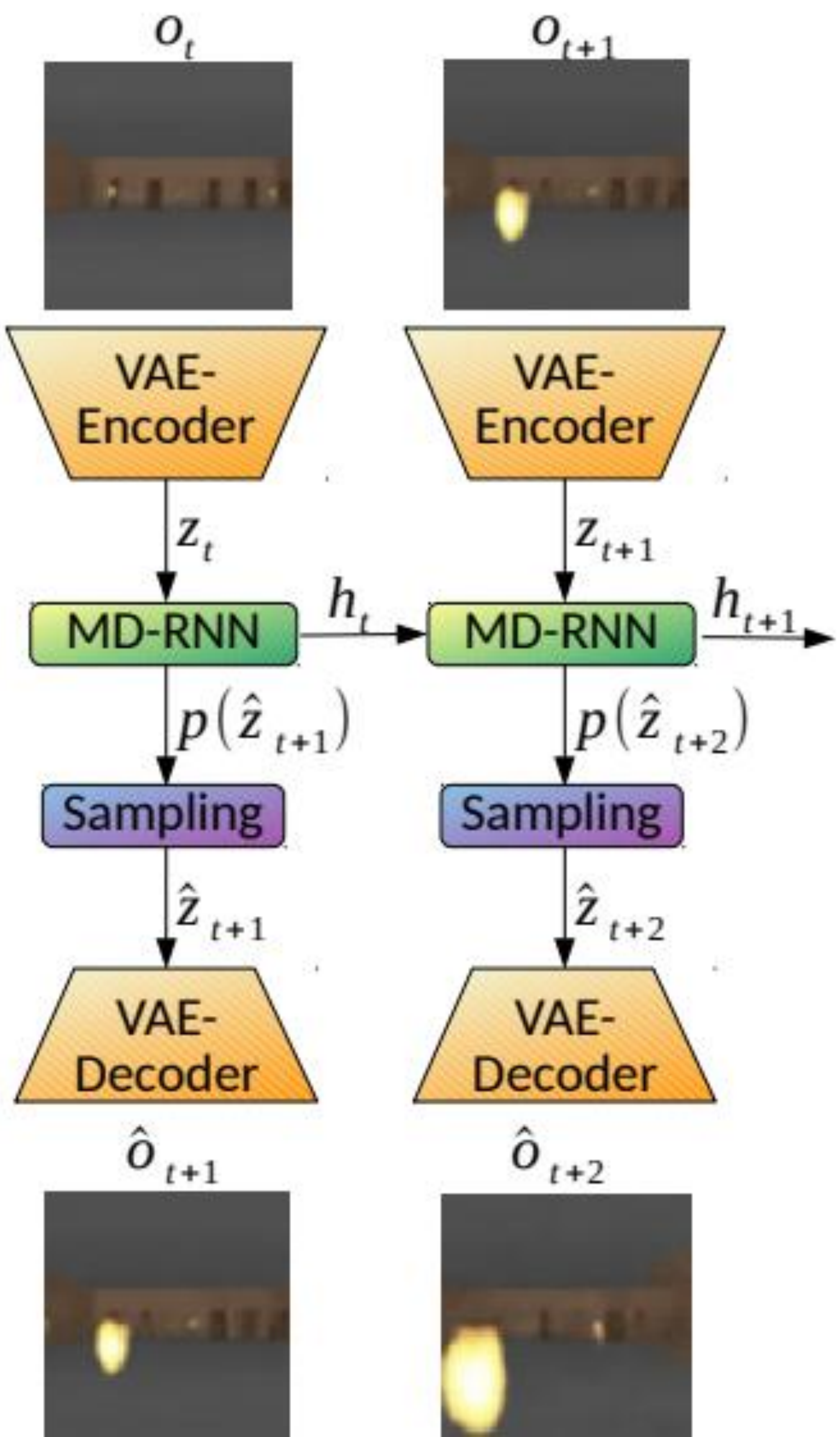
The **Memory RNN (M)** integrates the historical codes to create a representation that can predict future states.

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.




Source: https://worldmodels.github.io/



$o_t$
$o_{t+1}$
VAE-
Encoder
VAE-
Encoder
$z_t$
$z_{t+1}$
MD-RNN
$h_t$
MD-RNN
$h_{t+1}$
$p(\hat{z}_{t+1})$
$p(\hat{z}_{t+2})$
Sampling
Sampling
$\hat{z}_{t+1}$
$\hat{z}_{t+2}$
VAE-
Decoder
VAE-
Decoder
$\hat{o}_{t+1}$
$\hat{o}_{t+2}$

At each time step, our agent receives an **observation** from the environment.

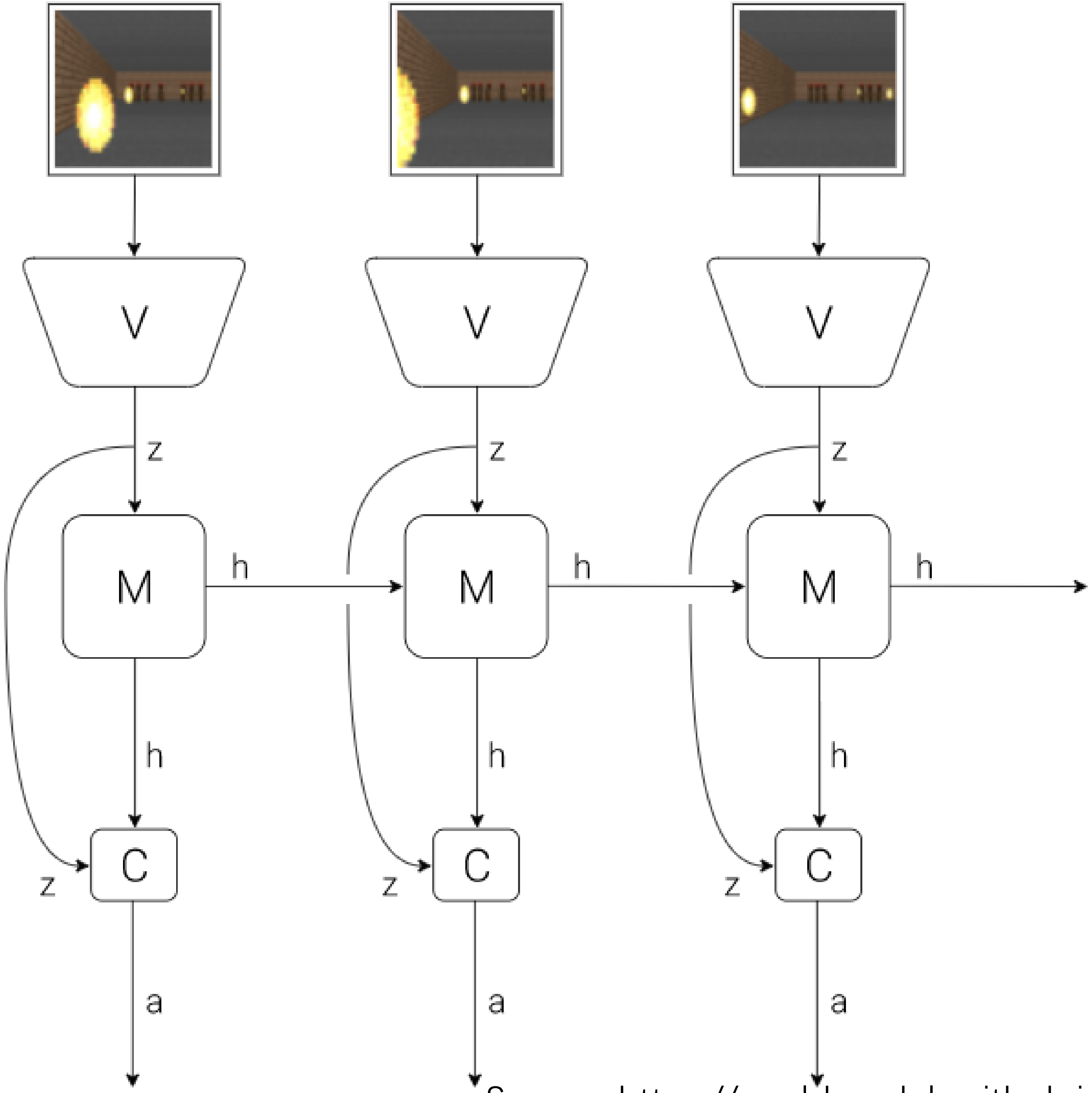
**World Model**

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.

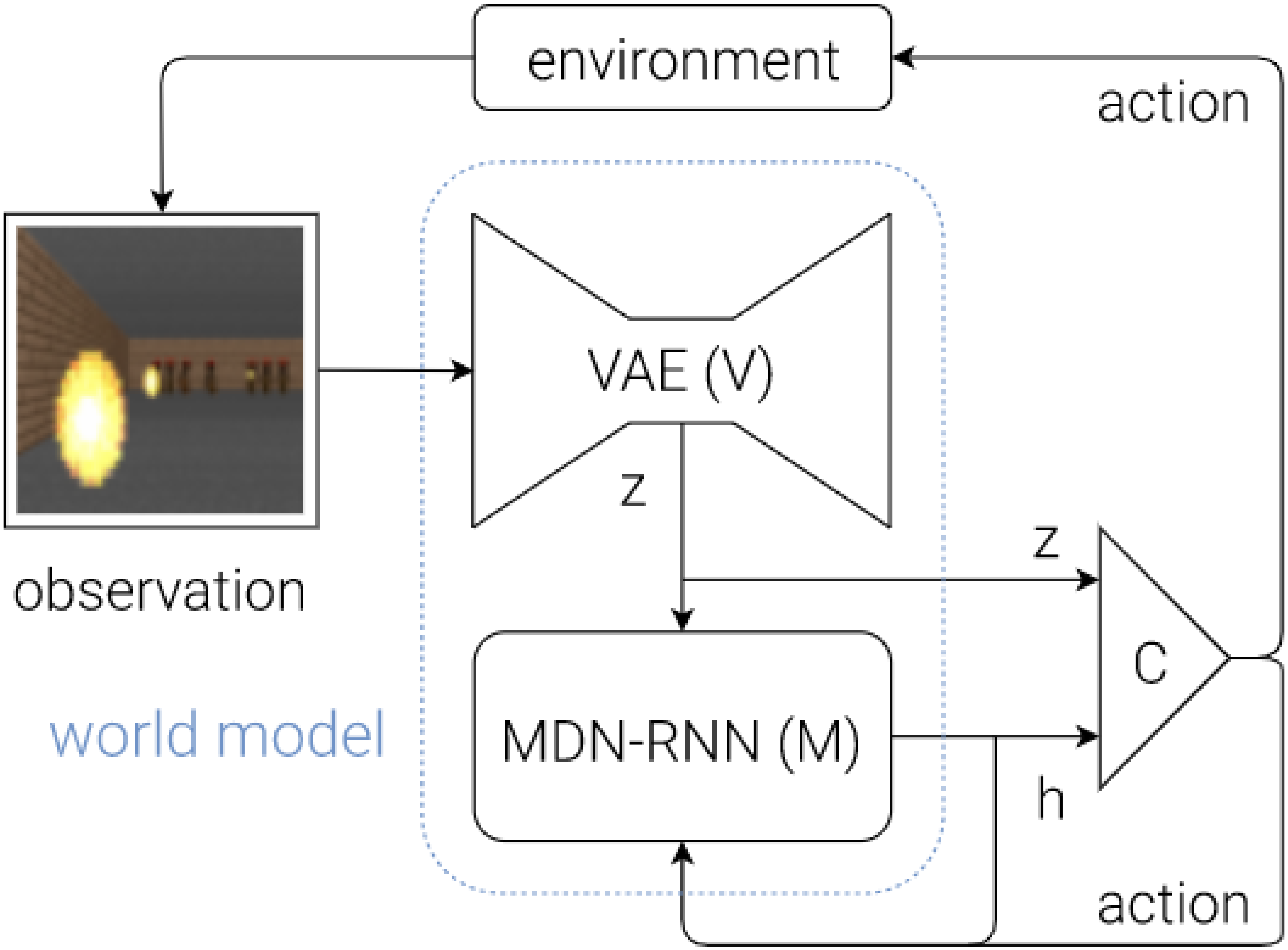



Source: https://worldmodels.github.io/

C is a simple single layer linear model that maps $z_t$ and $h_t$ directly to action $a_t$ at each time step:

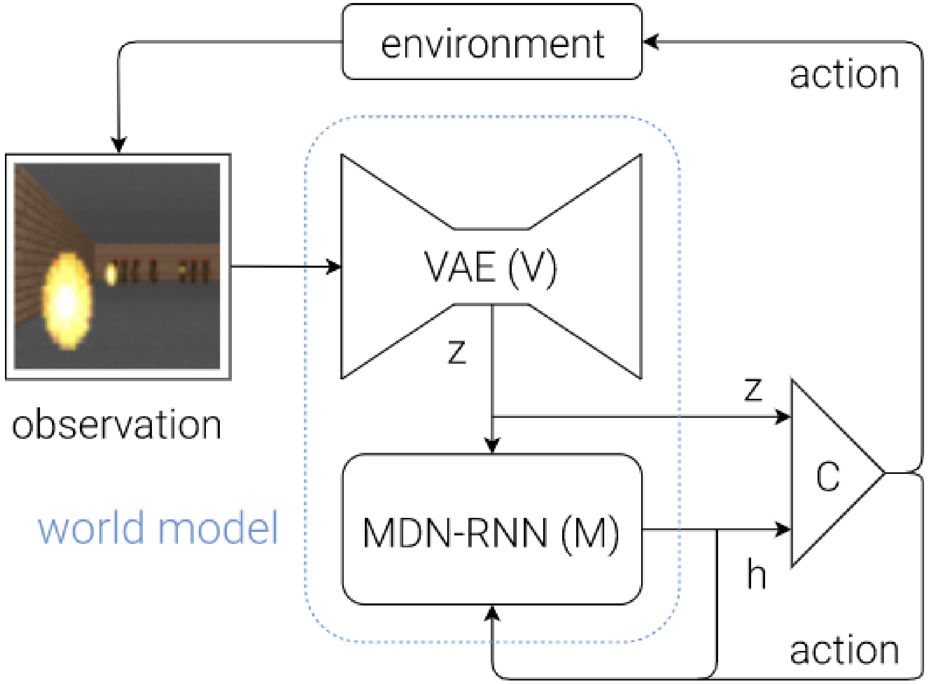
$$a_t = W_c \, [z_t \; h_t] \; + \; b_c$$

In this linear model, $W_c$ and $b_c$ are the weight matrix and bias vector that maps the concatenated input vector $[z_t \; h_t]$ to the output action vector $a_t$.[3]


environment
action
observation
VAE (V)
z
z
C
world model
MDN-RNN (M)
h
action

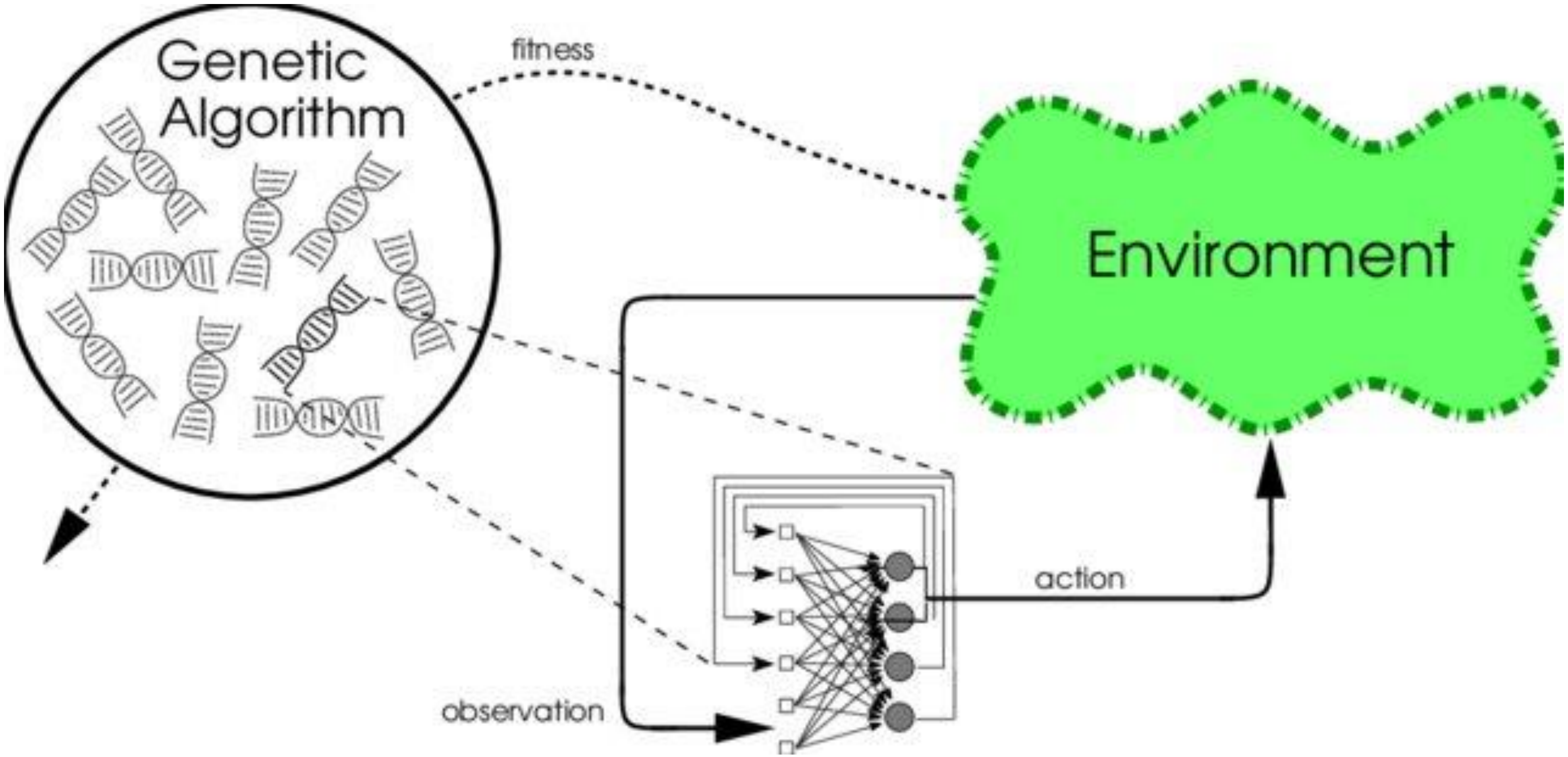
# Training procedure

1. Collect 10,000 rollouts from a random policy.
2. Train VAE (V) to encode each frame into a latent vector $z \in \mathcal{R}^{32}$.
3. Train MDN-RNN (M) to model $P(z_{t+1} \mid a_t, z_t, h_t)$.
4. Evolve Controller (C) to maximize the expected cumulative reward of a rollout.




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

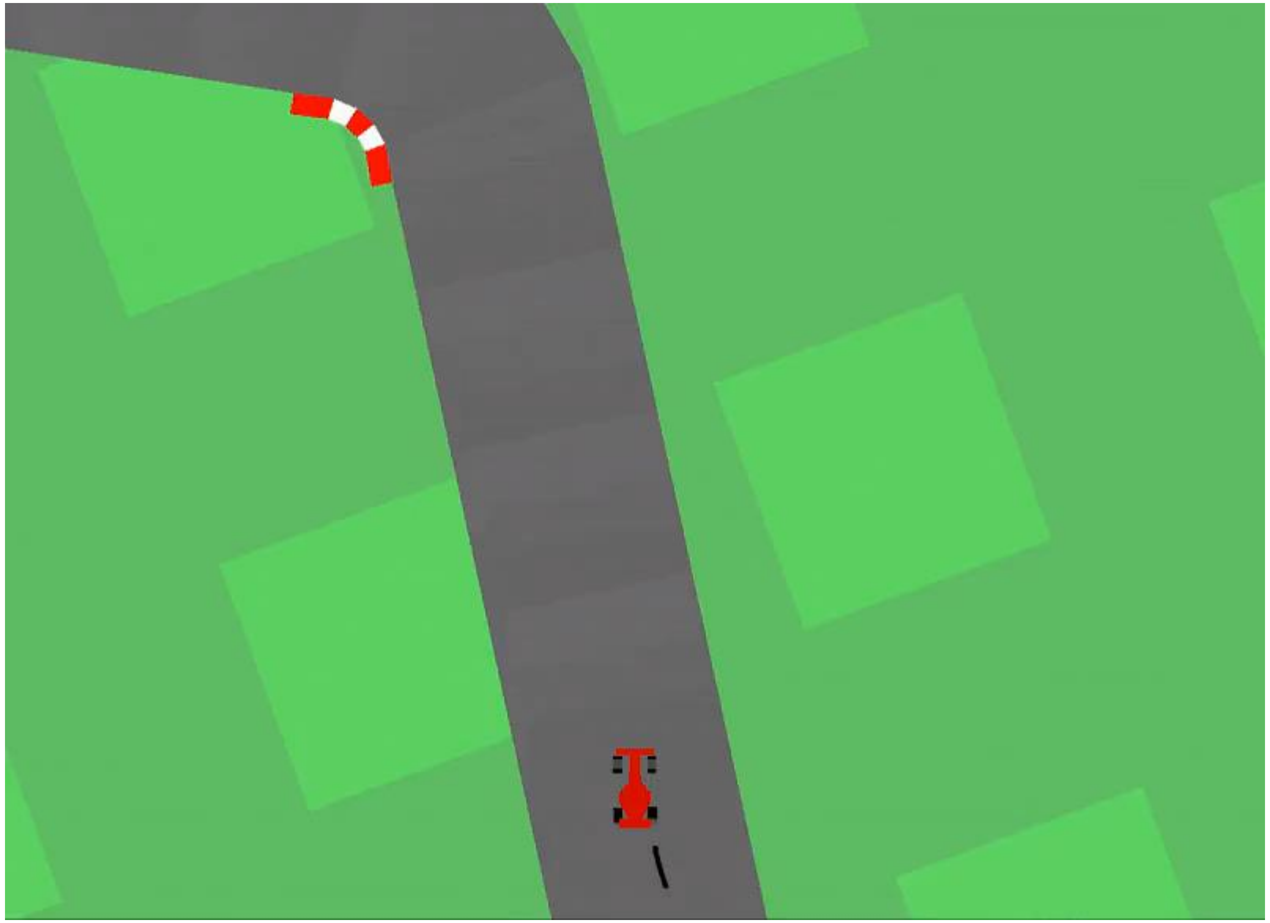
# Neuroevolution

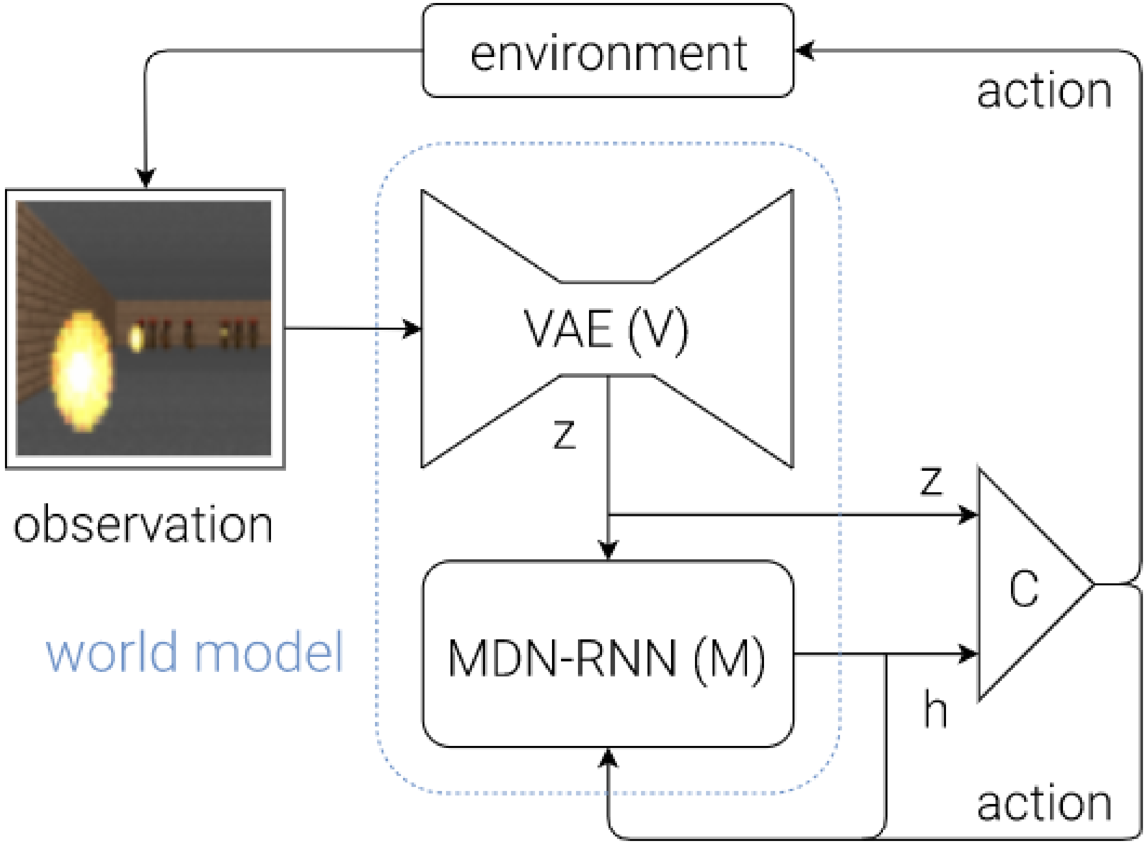



Source: https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-30164-8_589

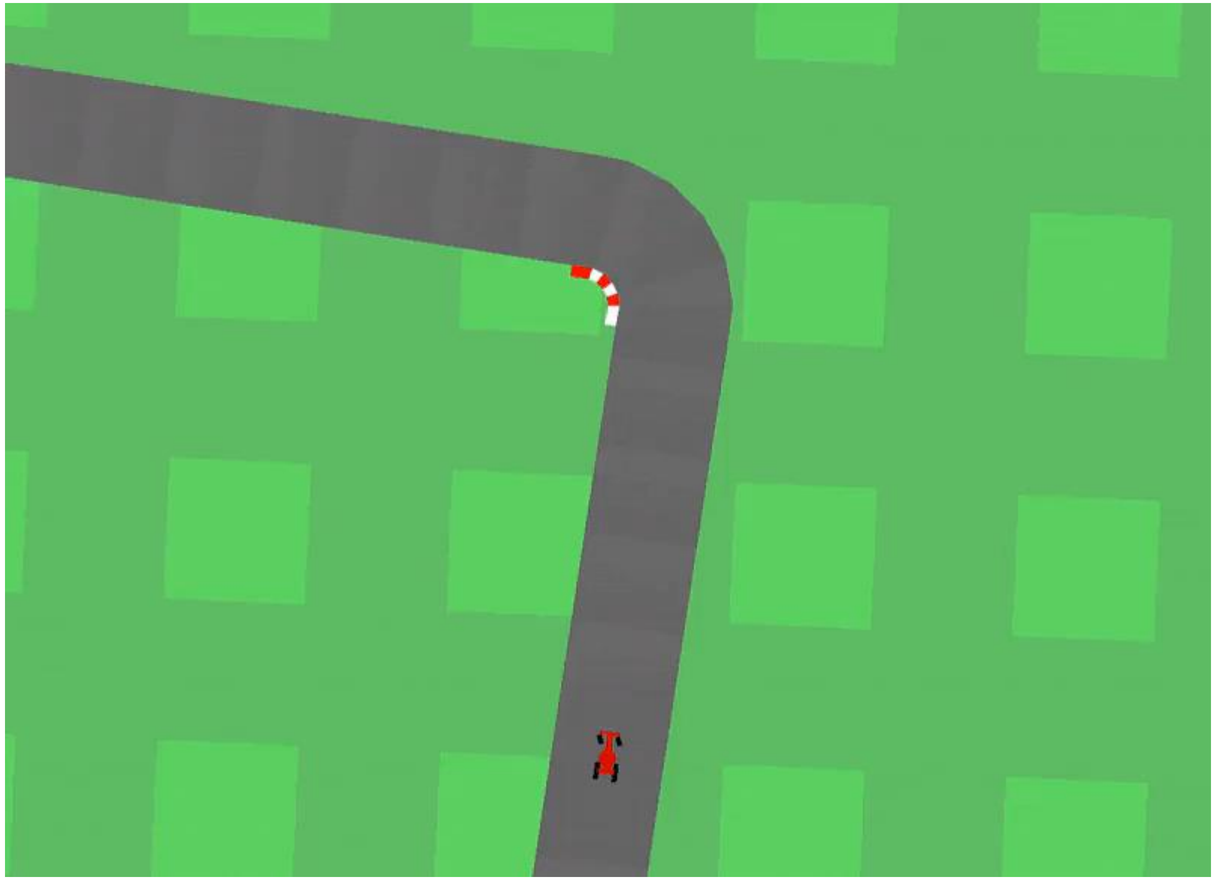
# World models - Results

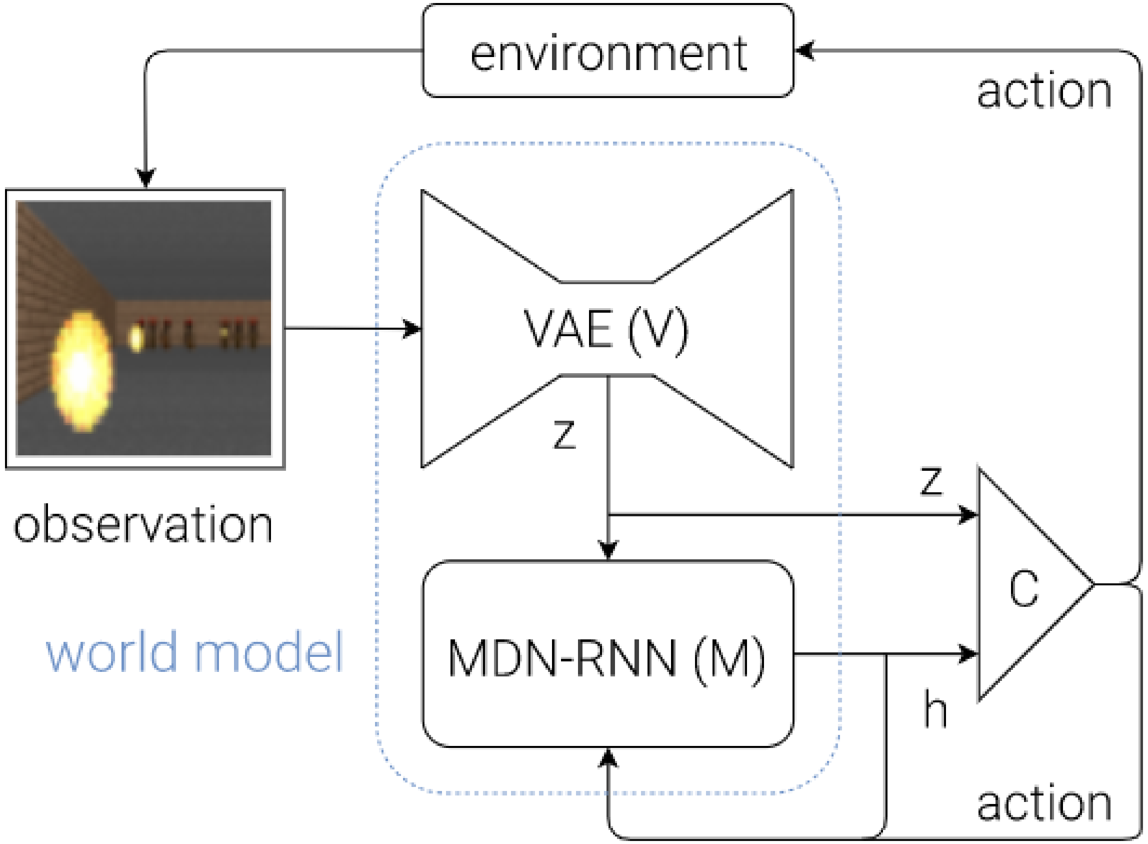
# Carracing – z only





# Carracing – z and h

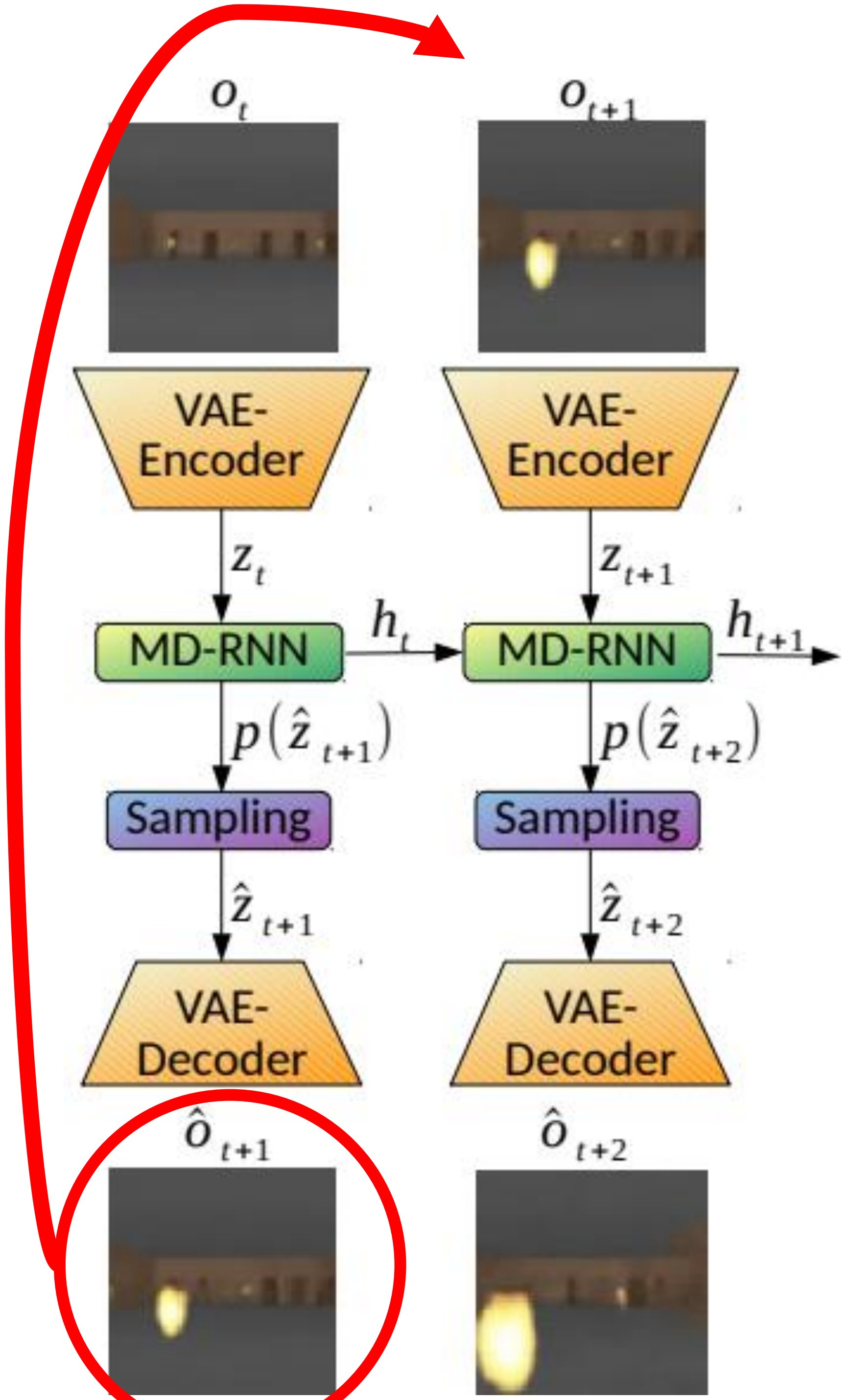




| Method | Average Score over 100 Random Tracks |
|---|---|
| DQN [53] | $343 \pm 18$ |
| A3C (continuous) [52] | $591 \pm 45$ |
| A3C (discrete) [51] | $652 \pm 10$ |
| ceobillionaire's algorithm (unpublished) [47] | $838 \pm 11$ |
| V model only, $z$ input | $632 \pm 251$ |
| V model only, $z$ input with a hidden layer | $788 \pm 141$ |
| **Full World Model**, $z$ and $h$ | $\mathbf{906 \pm 21}$ |

Dreaming:

Instead of an actual input, give the decoded, predicted next frame as input



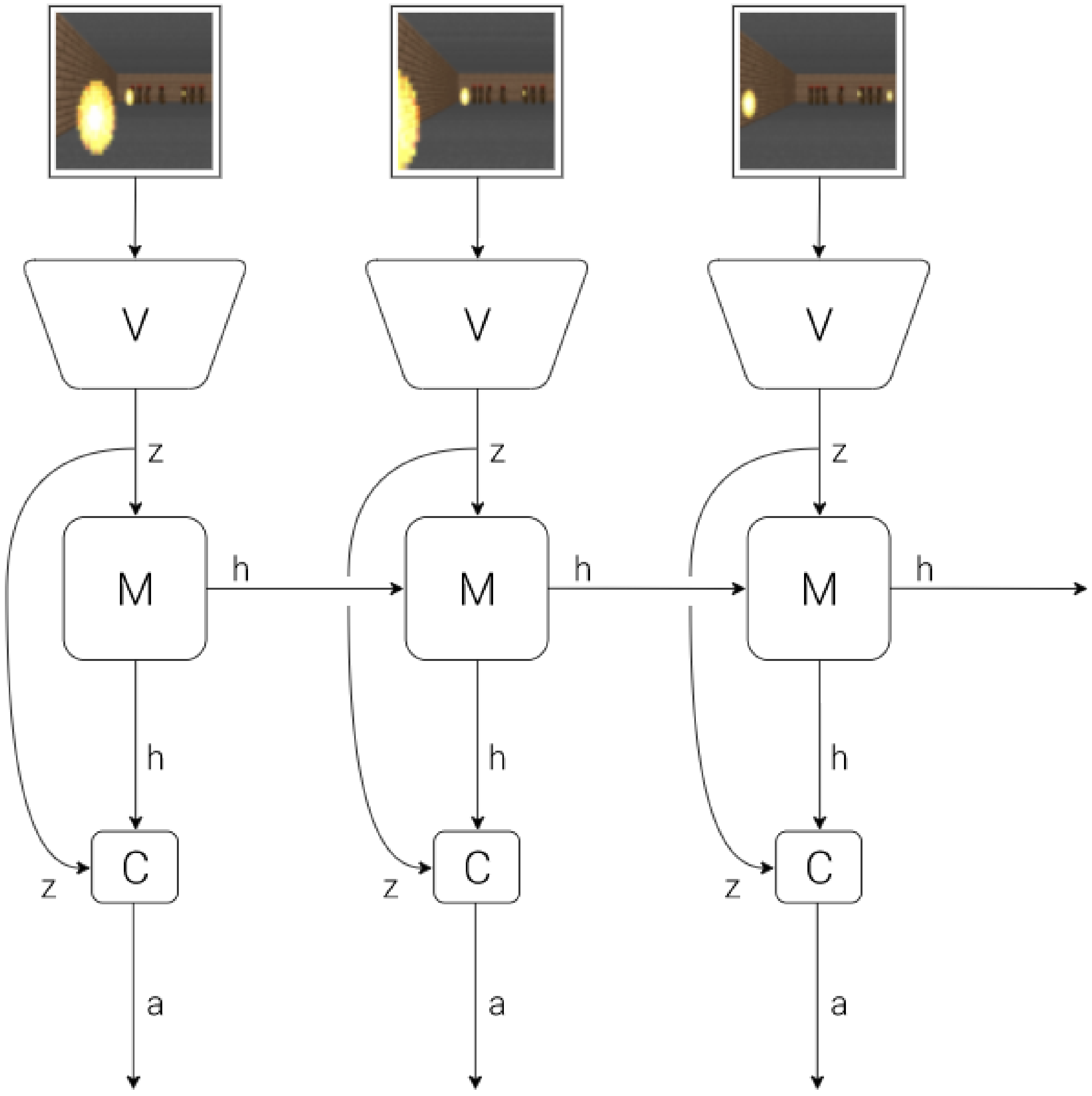
# Dreaming demo

- https://worldmodels.github.io/

# The agent can even *learn* inside its own dream!

We optimize C as before, but now every episode is a «dreamt» sequence

C is optimized to control agent inside the dream-world



# Policies learned in a dream also work in the real game!!

# Next steps: Using World Models for learning real-world tasks

A neural net's hallucination of driving on a highway:




Source: https://nv-tlabs.github.io/DriveGAN/

# Summary

- «Understanding» the world a key limitation for deep learning, making it non-robust.
- Model-based learning may be a step on the way
- Model can predict the future and consequences of actions
- We saw a world model applying Reinforcement Learning, Unsupervised Learning, an Autoencoder, an RNN and NeuroEvolution