World Models

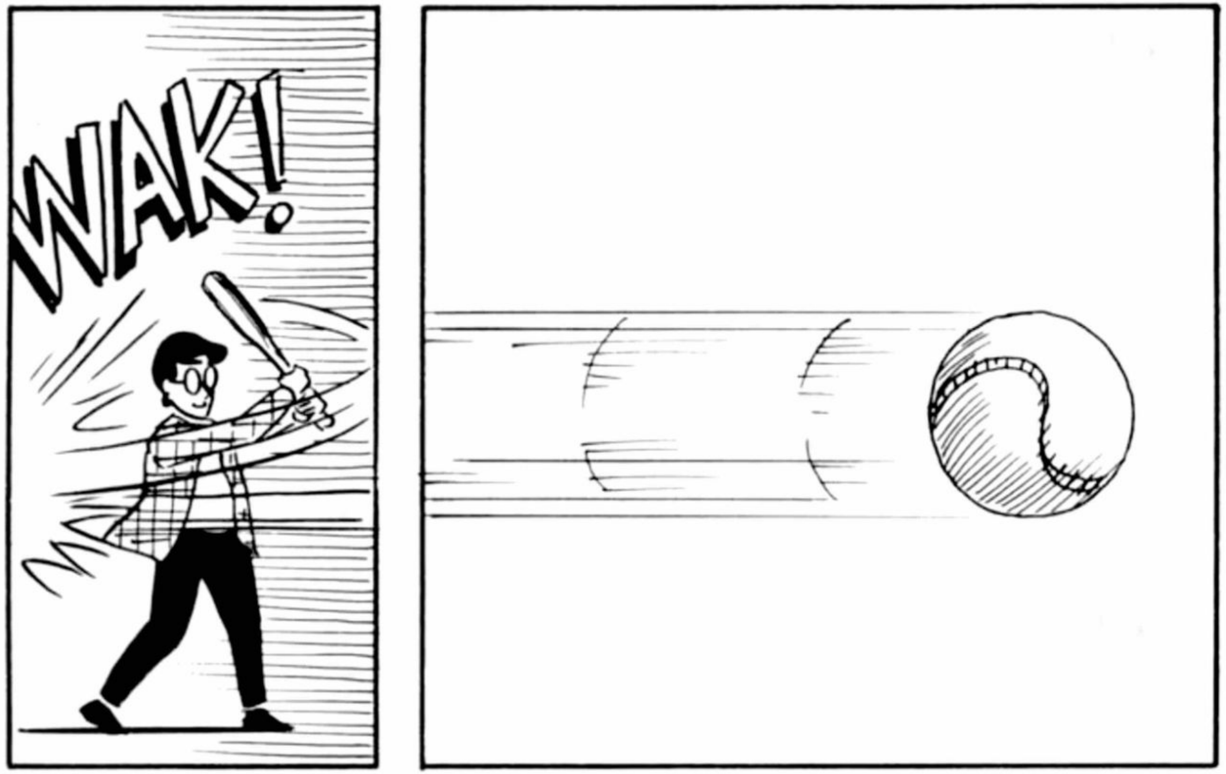
David Ha and Jurgen Schmidhuber

NIPS, 2018, Oral Presentation

… What we perceive at any given moment is governed by our brain’s prediction of the future based on our internal model ...

… We are able to instinctively act on this predictive model and perform fast reflexive behaviours when we face danger, without the need to consciously plan out a course of action ...

Take baseball for example.



Proposed Model

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

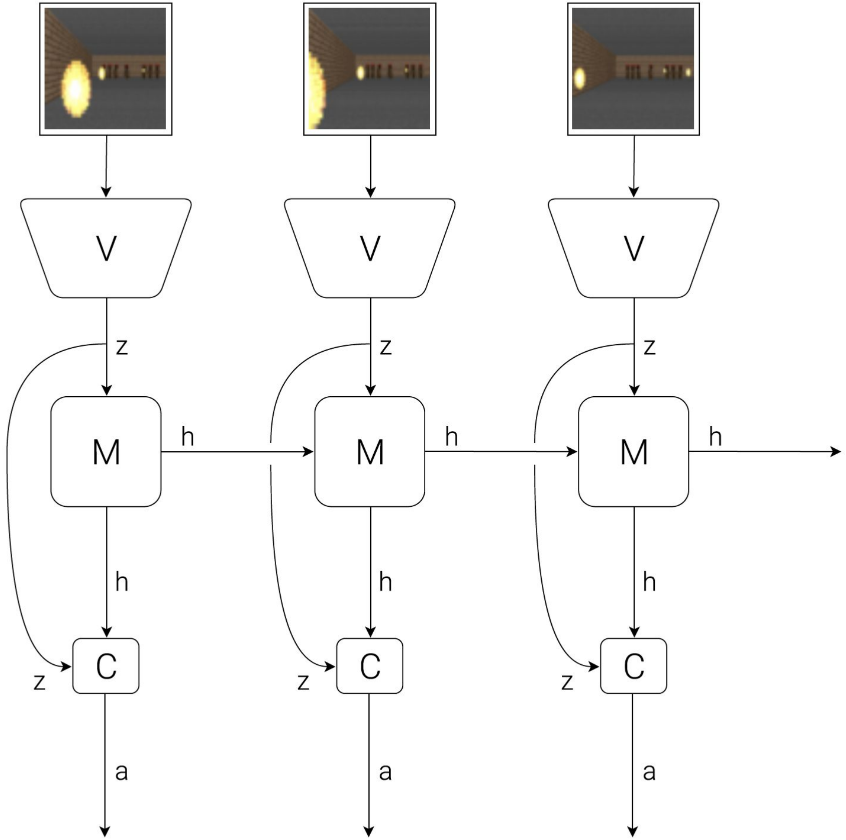
World Model

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.

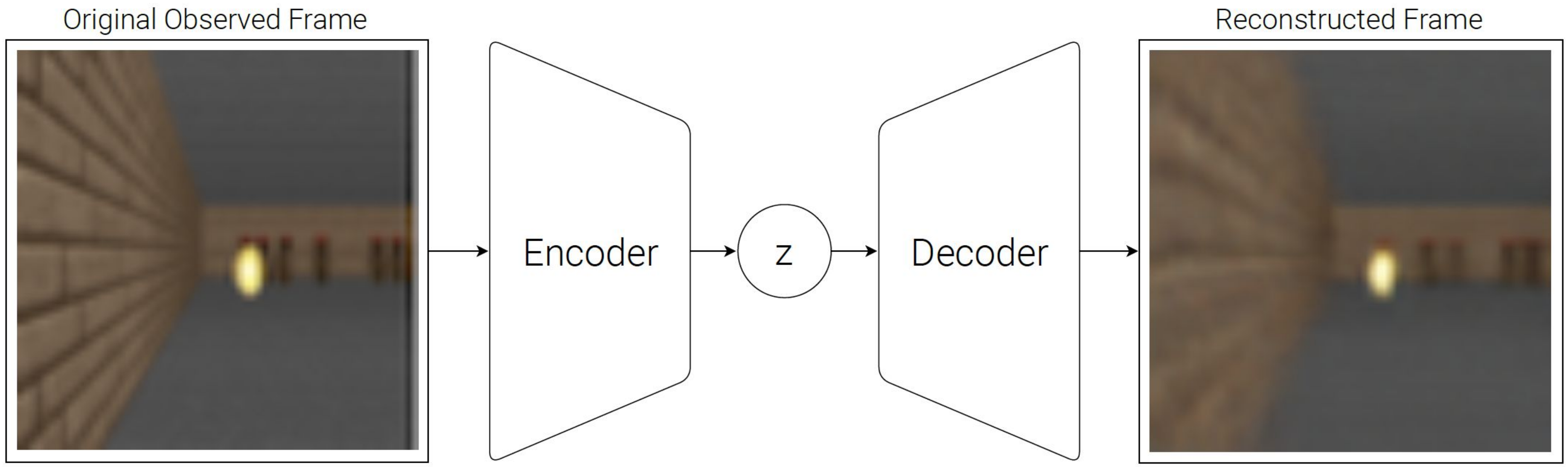
A small **Controller (C)** uses the representations from both **V** and **M** to select good actions.

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

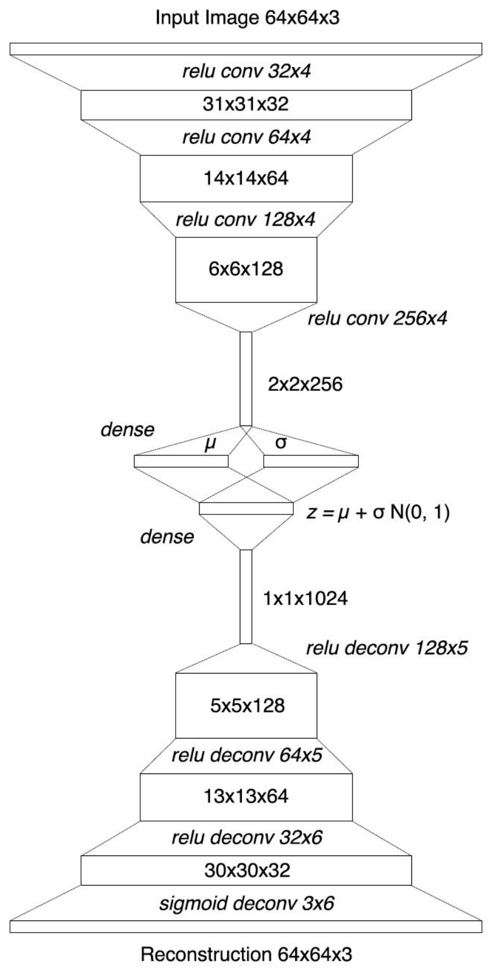


VAE (V) Model

- V model learns an abstract, compressed representation of each observed input frame.

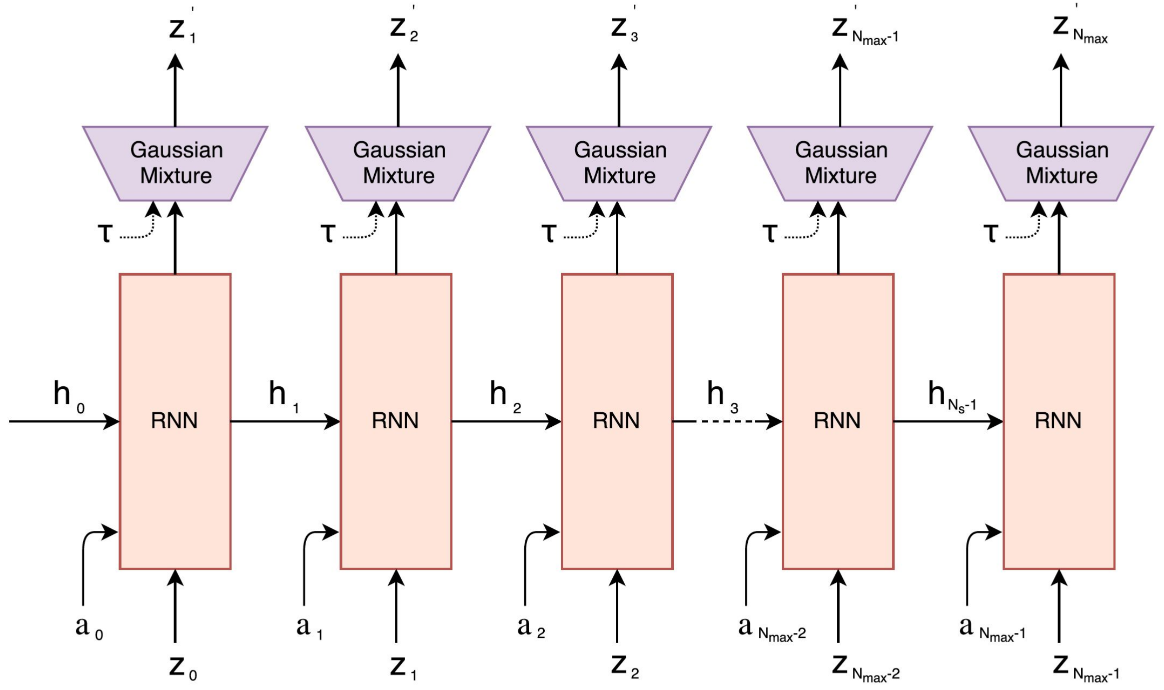


VAE (V) Model


Input Image 64x64x3
relu conv 32x4
31x31x32
relu conv 64x4
14x14x64
relu conv 128x4
6x6x128
relu conv 256x4
2x2x256
dense
μ
σ
z = μ + σ N(0, 1)
dense
1x1x1024
relu deconv 128x5
5x5x128
relu deconv 64x5
13x13x64
relu deconv 32x6
30x30x32
sigmoid deconv 3x6
Reconstruction 64x64x3

MDN-RNN (M) Model

- M model serves as a predictive model of the future z vectors that V is expected to produce.



MDN-RNN (M) Model

- Since many complex environments are stochastic in nature, we train our RNN to output a probability density function p(z) instead of a deterministic prediction of z.
  $\rightarrow$ The MDN outputs the parameters of a mixture of Gaussian distribution used to sample a prediction of the next latent vector z.

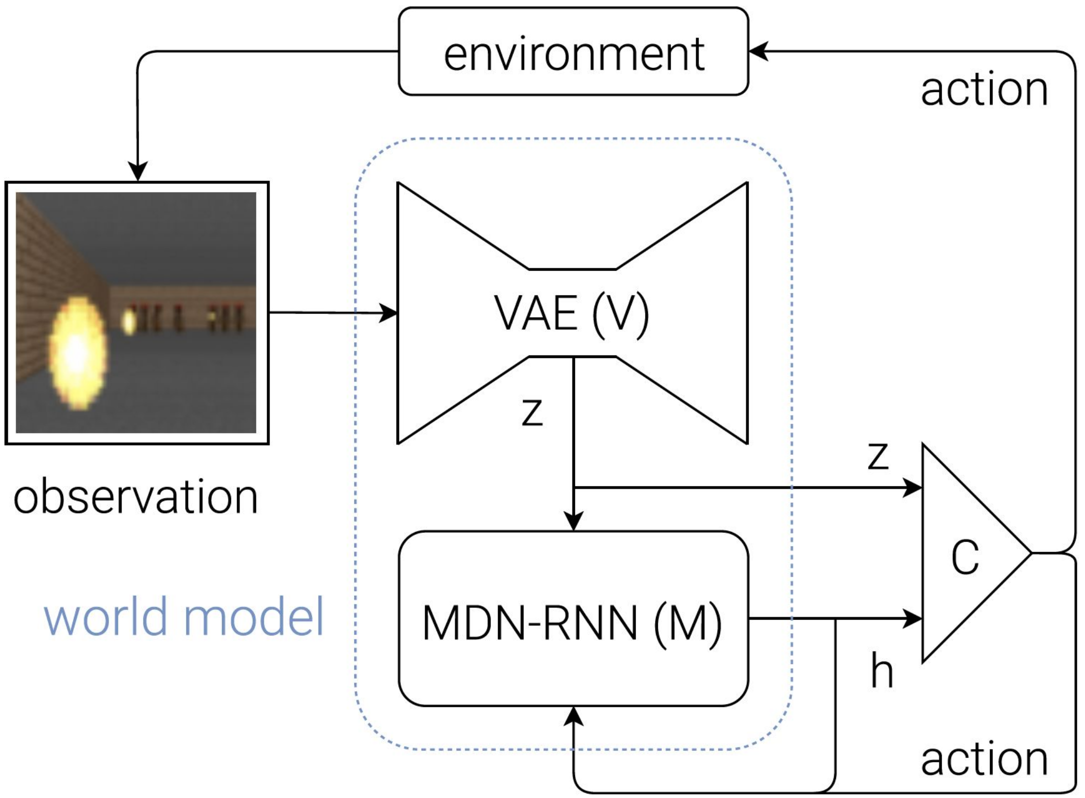
- During sampling, we can adjust a temperature parameter to control model uncertainty.

Controller (C) Model

- C is a simple single layer linear model that maps z and h directly to action a at each time step.

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

Flow Diagram



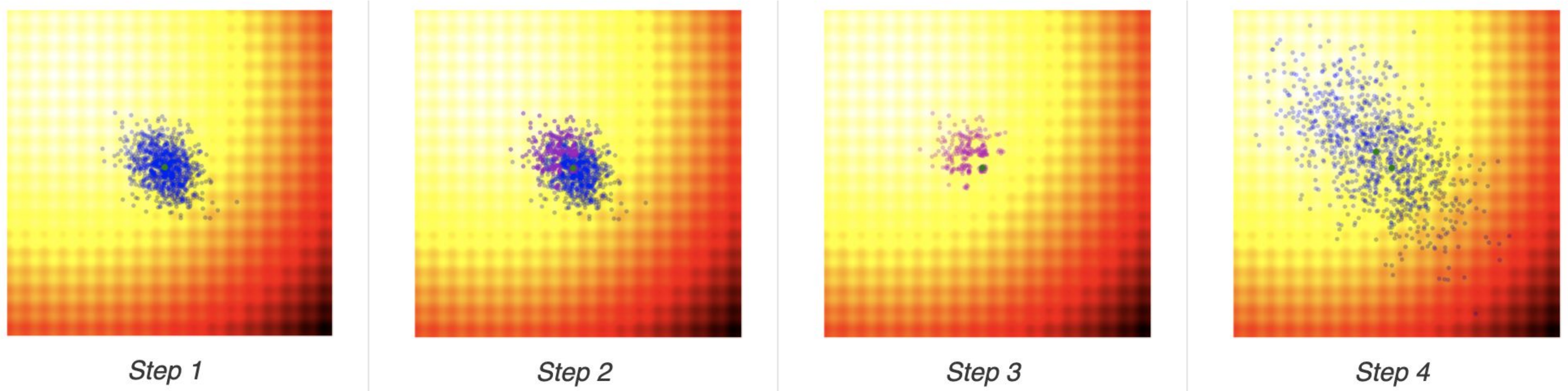
Optimization: CMA-ES (Covariance-Matrix Adaptation Evolution Strategy)

- Evolution Strategy: Iterate the following process.
    1. sample parameters from the current distribution.
    2. get the N best parameters in the sampled parameters.
    3. update the distribution from the N best parameter.

- CMA-ES: Adaptively increase or decrease the search space for the next generation.

https://blog.otoro.net/2017/10/29/visual-evolution-strategies/

Optimization: CMA-ES



1. Calculate the fitness score of each candidate solution in generation (g).
2. Isolates the best 25% of the population in generation (g), in purple.
3. Using only the best solutions, along with the mean $\mu^{(g)}$ of the current generation (the green dot), calculate the covariance matrix $C^{(g+1)}$ of the next generation.
4. Sample a new set of candidate solutions using the updated mean $\mu^{(g+1)}$ and covariance matrix $C^{(g+1)}$.

https://blog.otoro.net/2017/10/29/visual-evolution-strategies/

Optimization: CMA-ES

$$\mu_x^{(g+1)} = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} x_i,$$

$$\mu_y^{(g+1)} = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} y_i.$$

$$\sigma_x^{2,(g+1)} = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} (x_i - \mu_x^{(g)})^2,$$

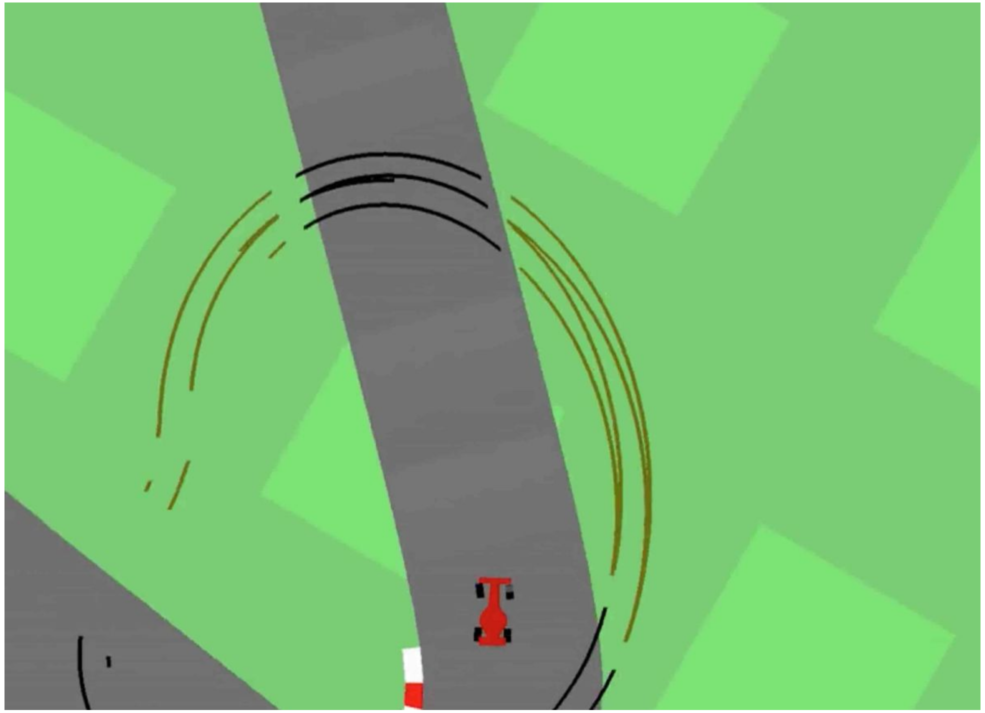
$$\sigma_y^{2,(g+1)} = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} (y_i - \mu_y^{(g)})^2,$$

$$\sigma_{xy}^{(g+1)} = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} (x_i - \mu_x^{(g)})(y_i - \mu_y^{(g)}).$$

https://blog.otoro.net/2017/10/29/visual-evolution-strategies/

Car Racing Experiment

- The agent is rewarded for visiting as many tiles as possible in the least amount of time.

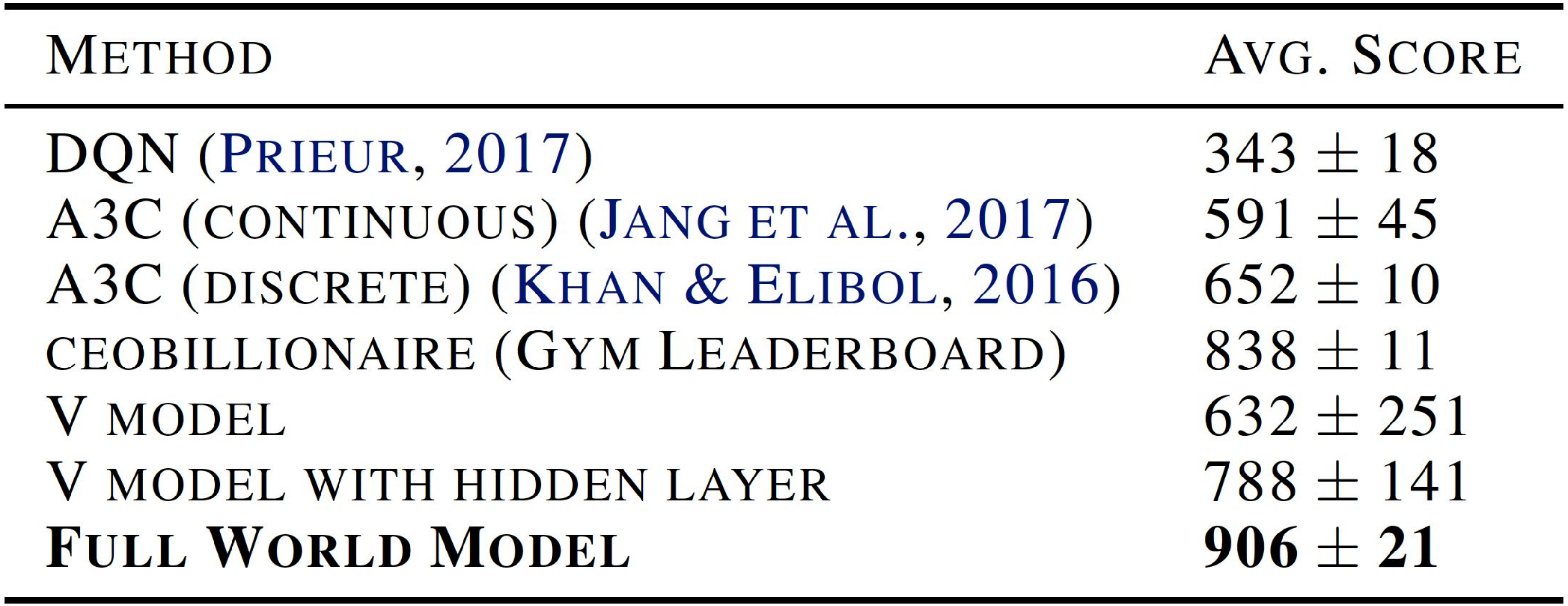


Car Racing Experiment

1. Collect 10,000 rollouts from a random policy.
2. Train VAE (V) to encode frames into $z \in \mathcal{R}^{32}$.
3. Train MDN-RNN (M) to model $P(z_{t+1} \mid a_t, z_t, h_t)$.
4. Define Controller (C) as $a_t = W_c \; [z_t \; h_t] \; + \; b_c$.
5. Use CMA-ES to solve for a W_c and b_c that maximizes the expected cumulative reward.

MODEL	PARAMETER COUNT
VAE	4,348,547
MDN-RNN	422,368
CONTROLLER	867

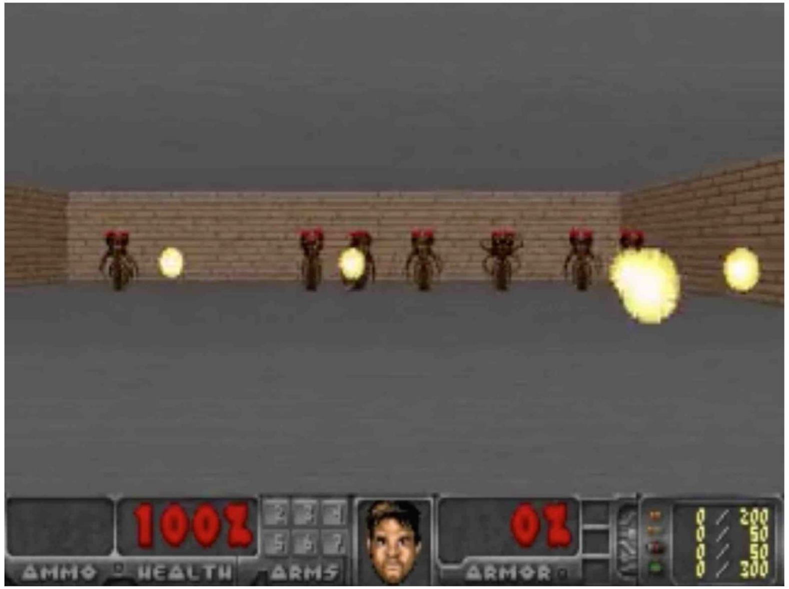
Car Racing Experiment



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	$\mathbf{906 \pm 21}$

VizDoom Experiment

- Can we train our agent to learn inside of its own dream, and transfer this policy back to the actual environment?
- The agent must learn to avoid fireballs shot by monsters from the other side of the room with the sole intent of killing the agent.

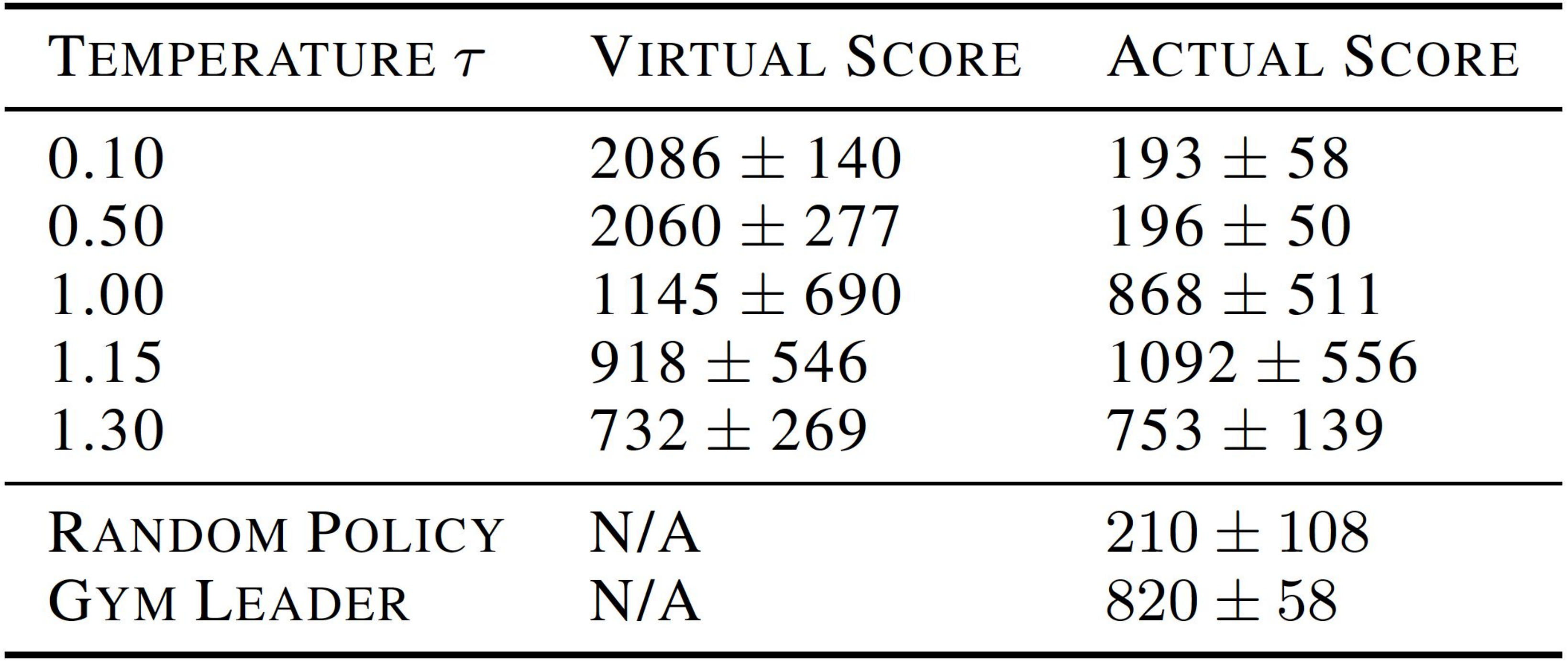


VizDoom Experiment

1. Collect 10,000 rollouts from a random policy.
2. Train VAE (V) to encode each frame into a latent vector $z \in \mathcal{R}^{64}$, and use V to convert the images collected from (1) into the latent space representation.
3. Train MDN-RNN (M) to model $P(z_{t+1}, d_{t+1} \mid a_t, z_t, h_t)$.
4. Define Controller (C) as $a_t = W_c \; [z_t \; h_t]$.
5. Use CMA-ES to solve for a W_c that maximizes the expected survival time inside the virtual environment.
6. Use learned policy from (5) on actual environment.

MODEL	PARAMETER COUNT
VAE	4,446,915
MDN-RNN	1,678,785
CONTROLLER	1,088

VizDoom Experiment



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

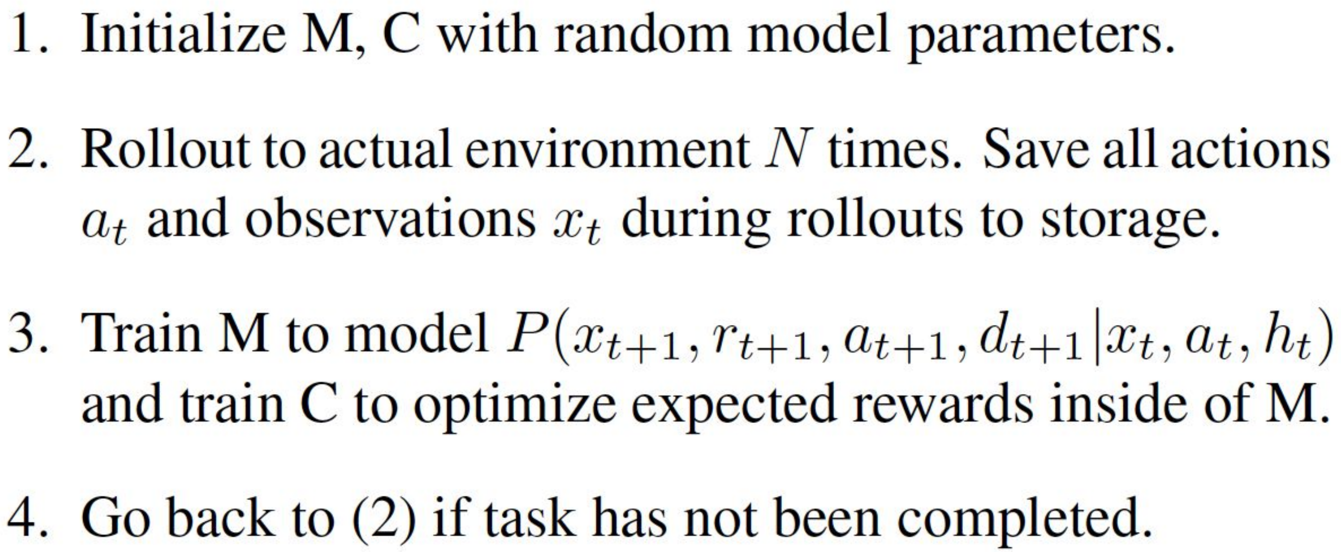
VizDoom Experiment

- Increasing the temperature parameter during the sampling process of z makes the virtual environment become more difficult.
  ex) The fireballs may move more randomly in a less predictable path compared to the actual game.

- Cheating the World Model: The controller exploits the world model, even if in the actual environment such exploits do not exist.
  ex) Even when there are signs of a fireball forming, the agent moves in a way to extinguish the fireballs magically as if it has superpowers in the environment.

Iterative Training Procedure

- For more complicated tasks, we need our agent to be able to explore its world, and constantly collect new observations so that its world model can be improved and refined over time.



1. Initialize M, C with random model parameters.
2. Rollout to actual environment N times. Save all actions a_t and observations x_t during rollouts to storage.
3. Train M to model $P(x_{t+1}, r_{t+1}, a_{t+1}, d_{t+1} | x_t, a_t, h_t)$ and train C to optimize expected rewards inside of M.
4. Go back to (2) if task has not been completed.

Iterative Training Procedure

- By flipping the sign of M’s loss function in the actual environment, the agent will be encouraged to explore parts of the world that it is not familiar with.

- The M model is required to predict the action and reward for the next time step. For instance, if the world model learns motor skills, such as walking, the smaller C model can rely on the motor skills already absorbed by the world model and focus on learning more higher level skills to navigate.