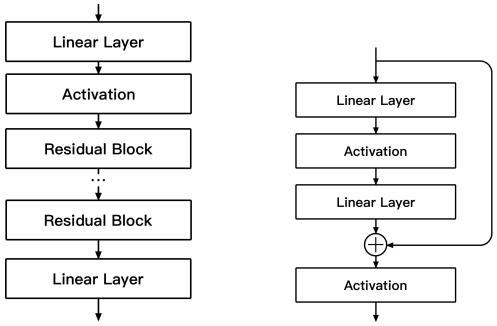
Neural Motion Simulator
Pushing the Limit of World Models in Reinforcement Learning

Supplementary Material

A. Architecture



(a) Residual Network (b) Residual Block

Figure 7. Network architecture of residual networks.

Basic NN	Layers	Input Shape	Output Shape
Residual Network	Fully-connected layer	(B, D_{input})	(B, D_h)
	Activation	(B, D_h)	(B, D_h)
	Residual Blocks	(B, D_h)	(B, D_h)
	Fully-connected layer	(B, D_h)	(B, D_{output})
MLP	Fully-connected layer	(B, D_{input})	(B, D_h)
	Activation	(B, D_h)	(B, D_h)
	Hidden Fully-connected Blocks	(B, D_h)	(B, D_h)
	Fully-connected layer	(B, D_h)	(B, D_{output})

Table 8. Basic NNs

Module	Basic NN	Input Shape	Output Shape
Position Encoder (M)	Residual Network	(B, D_q)	$(B, D_v \times (D_v+1)/2)$
	Rearrange	$(B, D_v \times (D_v+1)/2)$	(B, D_v, D_v)
State Encoder (B)	Residual Network	$(B, D_q + D_v)$	(B, D_v)
Action Encoder (τ)	MLP	(B, D_a)	(B, D_v)
Corrector	Residual Network	$(B, D_q + D_v + D_a)$	(B, D_v)

Table 9. MoSim Modules

Model Size	Position Encoder		State Encoder		Action Encoder		Correctors		
	N_B	D_h	N_B	D_h	N_B	D_h	N_C	N_B	D_h
Small	3	64	3	64	1	32	1	5	64
Medium	3	64	3	64	3	32	1	5	128
Large	3	128	3	128	3	128	3	5	128

Table 10. MoSim Parameters

For the position encoder and state encoder in the predictor, as well as the corrector, we use residual networks as shown in Figure 7a, with the residual block design illustrated in Figure 7b. For the action encoder in the predictor, we use a standard MLP composed of multiple linear layers and activation layers. In the position encoder, the output vector of length $n(n+1)/2$ (where n is the dimension of velocity) from the residual network is rearranged into an $n \times n$ lower triangular matrix L, and the symmetric positive definite matrix M is then obtained by computing LL^T.

The number of correctors can be adjusted based on the complexity of the task. We use one corrector for most tasks, while two correctors are used for the humanoid task.

The specific architectural parameters of the models are detailed in Tables 8, 9, and 10, in which B refers to batch size, D_{input}, D_{output}, D_h, D_q, D_v, D_a respectively refer to dimension of input, output, latent variable, position, velocity, action.

Prediction Horizon	Cheetah	Reacher	Acrobot	Panda	Hopper	Humanoid	Go2
DreamerV3	16	16	16	16	16	16	16
MoSim	60	>1000	99	>1000	51	42	200

Table 11. Prediciton Horizon

B. Experimental Details for Latent Evaluation

In subsection 3.2, we compared the predictive capabilities in the latent space of MoSim and TD-MPC2 using expert data, as described in the main text. For the comparison on random data in table4, we still used the provided TD-MPC2 checkpoints, since training TD-MPC2 on random data leads to a meaningless latent space.

In this experiment, 1 step for MoSim is equivalent to `control step * action repeat`. Here, the control step refers to the duration of a single action in terms of physics timesteps in the DM Control environment, while the action repeat follows the default TD-MPC setting of 2. For example, in the Humanoid environment (`control step = 5`, `action repeat = 2`), 1 step actually requires us to recursively predict `2 * 5 = 10` physical steps, for experience evaluation, we set action repeat = 1, control step = 1.

C. Prediction Horizon

To more directly demonstrate the improvement in MoSim's prediction capability, Table 11 uses the MSE loss of DreamerV3 at a 16-step prediction horizon as a reference. It presents the prediction horizon at which MoSim achieves the same loss, providing a more intuitive measure of MoSim's predictive performance.

Task Name	Humanoid-walk	Hopper-hop
Horizon	1000	1000

Table 12. The minimum step limitation required for zero-shot learning in two additional tasks.(1000 is typically the default step limit in reinforcement learning simulation environments.)

D. Zero-Shot Learning Requirements

In subsection3.4, we explored the minimal step limitation required to achieve zero-shot learning. We additionally provide the requirements for two more tasks. The results are shown in Table 12.

E. MyoSuite Dataset Prediction Results

We also conducted predictions on the MyoSuite dataset. To satisfy MoSim's assumption of the Markov property, instead of directly using the muscle-tendon model in MyoSuite (where actions in this case form a second-order dynamic system, causing the overall system to violate the Markov property), we directly used the motor-tendon model. Specifically, we obtained the force on each tendon and learned its effect on joint states.

F. Experimental Settings for Residual Flow Penalty

To ensure the penalty term is on the same scale as the original reward, we use the log probability density of the current state under the flow-based model as the penalty term and add it directly to the original reward. To normalize the penalty term within the range of $[-1, 1]$, we apply a sigmoid function with a custom inflection point and then subtract 1. This transformation ensures that the penalty term is appropriately scaled for reinforcement learning.

Thus, the final reward function is given by:

$$R = R_{\text{original}} + R_{\text{penalty}} \tag{9}$$

where the penalty term R_{penalty} is defined as:

$$R_{\text{penalty}} = \sigma\left(\frac{\log P(\mathbf{s}) - \tau}{\alpha}\right) - 1 \tag{10}$$

where $\log P(\mathbf{s})$ is obtained using the normalizing flow model as:

$$\log P(\mathbf{s}) = \log P_{\text{base}}(x) + \log|\det J| \tag{11}$$

with:

$$x, \log|\det J| = f^{-1}(\mathbf{s}) \tag{12}$$

where:

- f^{-1} is the inverse transformation of the normalizing flow model,

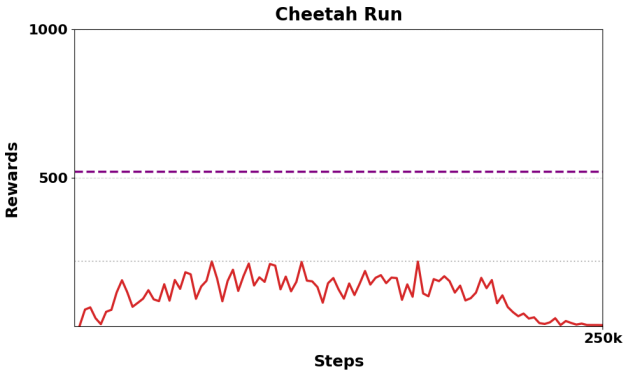


Figure 8. offline cheetah run.

- $P_{\text{base}}(x)$ is the probability density function of the base distribution (e.g., Gaussian),
- J is the Jacobian matrix of the transformation,
- $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function,
- τ is the custom inflection point,