STORM: Efficient Stochastic Transformer based World Models for Reinforcement Learning

March 12, 2025

Reference

Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. Storm: Efficient stochastic transformer based world models for reinforcement learning, 2023. URL https://arxiv.org/abs/2310.09615.

Introduction

- Previous world model approaches:
  - SimPLe: Uses LSTM
  - DreamerV3: Employs GRU
  - RNNs excel at sequence modeling but lack parallel computing capabilities
- Transformer architecture advantages:
  - Superior performance over RNNs
  - Better handles long-term dependencies
  - Enables efficient parallel computing
- STORM (Stochastic Transformer-based wORld Model):
  - Uses categorical VAE as image encoder
  - Incorporates Transformer for sequence modeling
  - Achieves 126.7% mean human normalized score on Atari 100k
  - Training requires only 4.3 hours on RTX 3090

Comparison with Recent Approaches

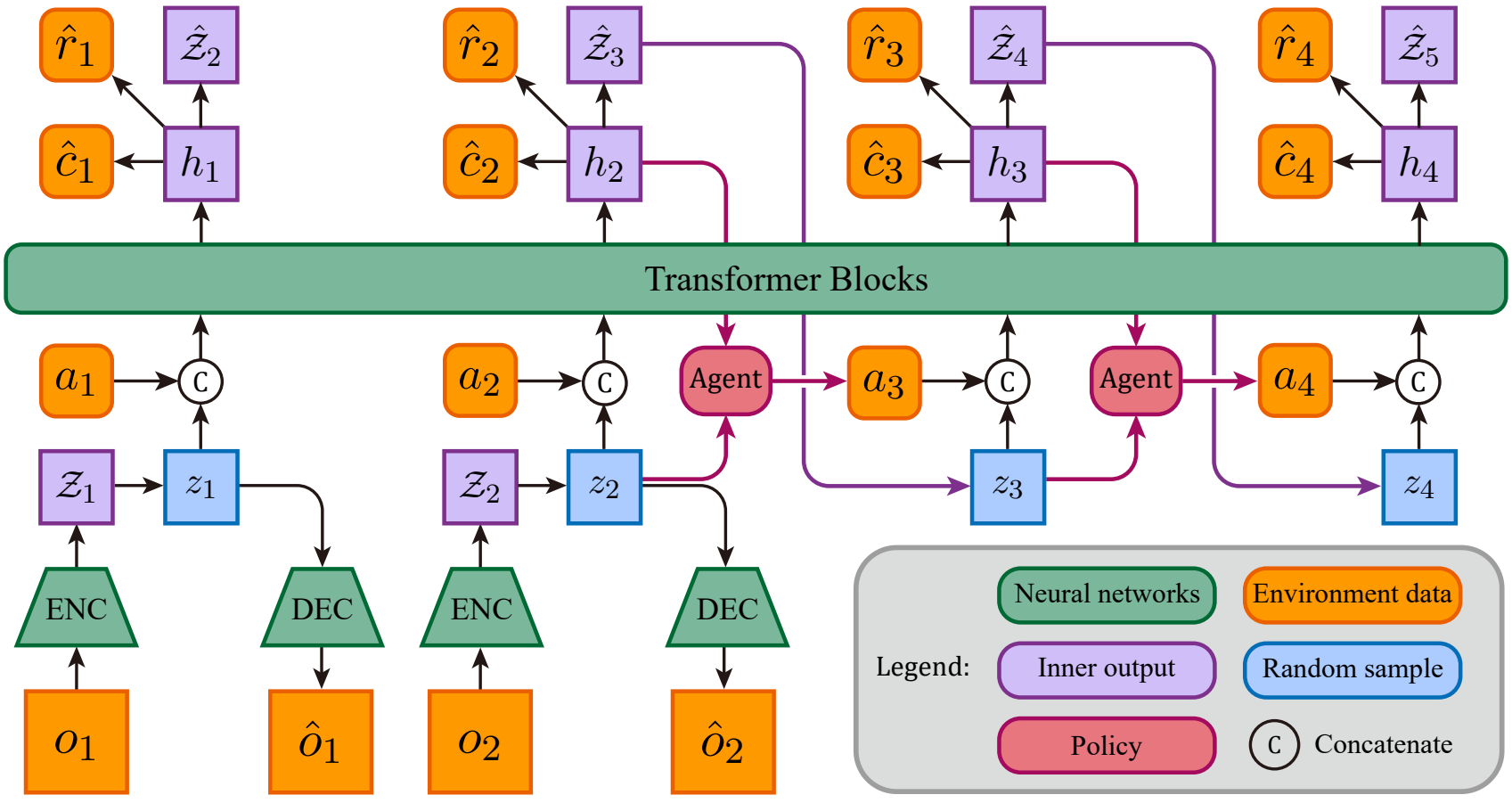
Comparison between STORM and recent approaches. "Tokens" refers to the input tokens introduced to the sequence model during a single timestep. "Historical information" indicates whether the VAE reconstruction process incorporates historical data, such as the hidden states of an RNN.

Attributes	SimPLe	TWM	IRIS	DreamerV3	STORM
Sequence model	LSTM	Transformer-XL	Transformer	GRU	Transformer
Tokens	Latent	Latent, action, reward	Latent(4×4)	Latent	Latent
Latent representation	Binary-VAE	Categorical-VAE	VQ-VAE	Categorical-VAE	Categorical-VAE
Historical information	Yes	No	Yes	Yes	No
Agent state	Reconstructed image	Latent	Reconstructed image	Latent, hidden	Latent, hidden
Agent training	PPO	As DreamerV2	As DreamerV2	DreamerV3	As DreamerV3

Detailed Comparison with Prior Work

- SimPLe and Dreamer rely on RNN-based models, whereas STORM employs a GPT-like Transformer as the sequence model.
- In contrast to IRIS that employs multiple tokens, STORM utilizes a single stochastic latent variable to represent an image.
- STORM follows a vanilla Transformer structure, while TWM adopts a Transformer-XL structure.
- In the sequence model of STORM, an observation and an action are fused into a single token, whereas TWM treats observation, action, and reward as three separate tokens of equal importance.
- Unlike Dreamer and TransDreamer, which incorporate hidden states, STORM reconstructs the original image without utilizing this information.

STORM (Zhang et al. [2023]) Architecture Overview



Data Structure and Training Process

- At each timestep t, a data point consists of:
  - Observation o_t
  - Action a_t
  - Reward r_t
  - Continuation flag c_t (indicates if episode is ongoing)
- Replay buffer:
  - Maintains first-in-first-out queue structure
  - Enables sampling of consecutive trajectories
- Training process:

S1 Execute policy to gather environment data into replay buffer
S2 Update world model using sampled trajectories
S3 Improve policy using imagined experiences from world model

Model Structure: VAE Formulation

- Convert image observations to latent space:

$$\text{Encoder: } z_t \sim q_\phi(z_t|o_t) = \mathcal{Z}_t$$
$$\text{Decoder: } \hat{o}_t = p_\phi(z_t)$$

- Latent structure $\mathcal{Z}_t$:
  - Stochastic categorical distribution
  - 32 categories × 32 classes
  - Sampled variable $z_t \sim \mathcal{Z}_t$ represents o_t
- Implementation details:
  - CNN-based encoder q_ϕ and decoder p_ϕ
  - Straight-through gradient estimation for backpropagation

Model Structure: Sequence Model

$$\text{Action mixer: } e_t = m_\phi(z_t, a_t)$$
$$\text{Sequence model: } h_{1:T} = f_\phi(e_{1:T})$$
$$\text{Dynamics predictor: } \hat{z}_{t+1} = g_\phi^D(\hat{z}_{t+1}|h_t)$$
$$\text{Reward predictor: } \hat{r}_t = g_\phi^R(h_t)$$
$$\text{Continuation predictor: } \hat{c}_t = g_\phi^C(h_t)$$

- Model components and their roles:
  - m_ϕ: Combines latent states and actions into unified tokens through MLP-based fusion
  - f_ϕ: Processes temporal dependencies using a GPT-style Transformer with causal masking
  - g_ϕ^D: Predicts the next latent distribution to capture environment dynamics
  - g_ϕ^R: Estimates immediate rewards based on current hidden states
  - g_ϕ^C: Determines episode termination probability from current context
- Key feature: Each token e_t can only attend to its causal history $e_1, e_2, \ldots, e_t$

Model Structure: Loss Functions

- Total loss function with hyperparameters $\beta_1 = 0.5$, $\beta_2 = 0.1$:

$$\mathcal{L}(\phi) = \frac{1}{BT} \sum_{n=1}^{B} \sum_{t=1}^{T} \Big[\mathcal{L}_t^{\text{rec}}(\phi) + \mathcal{L}_t^{\text{rew}}(\phi) + \mathcal{L}_t^{\text{con}}(\phi) + \beta_1 \mathcal{L}_t^{\text{dyn}}(\phi) + \beta_2 \mathcal{L}_t^{\text{rep}}(\phi) \Big]$$

- Environment prediction losses:
  - $\mathcal{L}_t^{\text{rec}}(\phi) = ||\hat{o}_t - o_t||_2$
  - $\mathcal{L}_t^{\text{rew}}(\phi) = \mathcal{L}^{\text{sym}}(\hat{r}_t, r_t)$
  - $\mathcal{L}_t^{\text{con}}(\phi) = c_t \log \hat{c}_t + (1 - c_t) \log(1 - \hat{c}_t)$
- Latent dynamics losses (KL-divergence based):
  - $\mathcal{L}_t^{\text{dyn}}(\phi) = \max\big(1, \text{KL}\big[\text{sg}(q_\phi(z_{t+1}|o_{t+1})) \;||\; g_\phi^D(\hat{z}_{t+1}|h_t)\big]\big)$
  - $\mathcal{L}_t^{\text{rep}}(\phi) = \max\big(1, \text{KL}\big[q_\phi(z_{t+1}|o_{t+1}) \;||\; \text{sg}(g_\phi^D(\hat{z}_{t+1}|h_t))\big]\big)$
  - Both losses are KL divergences clamped to minimum value of 1 with selective gradient stopping

Agent Learning

- During inference:
  - Sample z_t from prior $\hat{\mathcal{Z}}_t$ rather than posterior $\mathcal{Z}_t$
  - Use KV cache in Transformer for faster inference
- Agent state and policy:

$$\text{State: } s_t = [z_t, h_t]$$

$$\text{Critic: } V_\psi(s_t) \approx \mathbb{E}_{\pi_\theta, p_\phi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k}\right]$$

$$\text{Actor: } a_t \sim \pi_\theta(a_t|s_t)$$

Agent Learning: Actor-Critic Algorithm

- Actor-critic loss functions (from DreamerV3):

$$\mathcal{L}(\theta) = \frac{1}{BL} \sum_{n=1}^{B} \sum_{t=1}^{L} \left[-\mathrm{sg}\left(\frac{G_t^\lambda - V_\psi(s_t)}{\max(1, S)} \right) \ln \pi_\theta(a_t|s_t) - \eta H\big(\pi_\theta(a_t|s_t)\big) \right] \tag{1a}$$

$$\mathcal{L}(\psi) = \frac{1}{BL} \sum_{n=1}^{B} \sum_{t=1}^{L} \left[\Big(V_\psi(s_t) - \mathrm{sg}\big(G_t^\lambda\big) \Big)^2 + \Big(V_\psi(s_t) - \mathrm{sg}\big(V_{\psi^{\mathrm{EMA}}}(s_t)\big) \Big)^2 \right] \tag{1b}$$

- λ-return calculation:

$$G_t^\lambda \doteq r_t + \gamma c_t \Big[(1-\lambda) V_\psi(s_{t+1}) + \lambda G_{t+1}^\lambda \Big] \tag{2a}$$

$$G_L^\lambda \doteq V_\psi(s_L) \tag{2b}$$

Agent Learning: Training Details

- Normalization ratio S for actor loss:

$$S = \text{percentile}(G_t^\lambda, 95) - \text{percentile}(G_t^\lambda, 5) \tag{3}$$

- Value function regularization using EMA:

$$\psi_{t+1}^{\text{EMA}} = \sigma\psi_t^{\text{EMA}} + (1-\sigma)\psi_t \tag{4}$$

where:

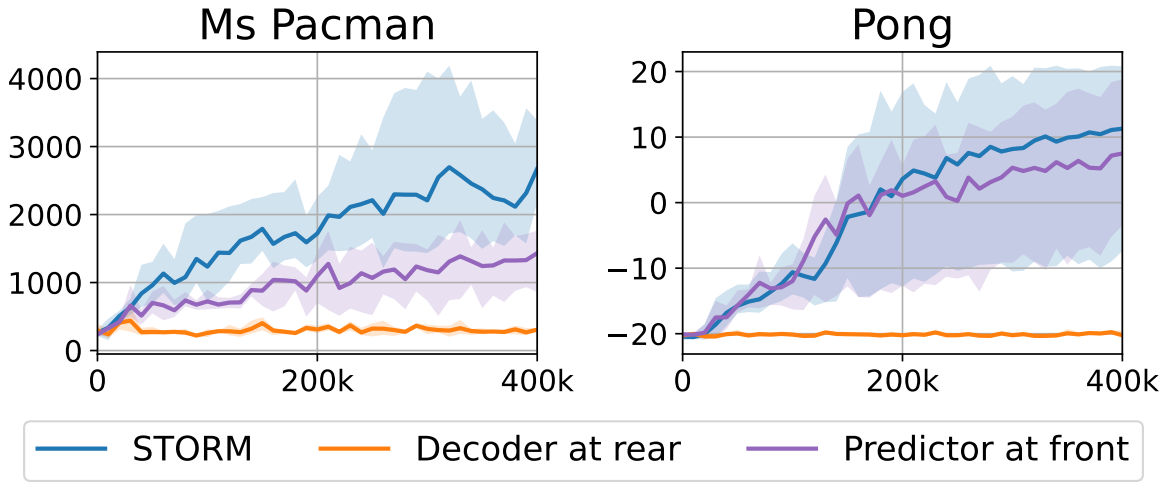
- σ is the decay rate
- ψ_t represents current critic parameters
- ψ_{t+1}^{EMA} denotes updated critic parameters

Game Scores

Game	Random	Human	SimPLe	TWM	IRIS	DreamerV3	STORM
Alien	228	7128	617	675	420	**959**	**984**
Amidar	6	1720	74	122	143	139	**205**
Assault	222	742	527	683	**1524**	706	801
Asterix	210	8503	**1128**	**1116**	854	932	1028
Bank Heist	14	753	34	467	53	**649**	**641**
Battle Zone	2360	37188	4031	5068	**13074**	12250	**13540**
Boxing	0	12	8	**78**	70	**78**	**80**
Breakout	2	30	16	20	**84**	31	16
Chopper Command	811	7388	979	1697	1565	420	**1888**
Crazy Climber	10780	35829	62584	71820	59234	**97190**	66776
Demon Attack	152	1971	208	350	**2034**	303	165
Freeway	0	30	17	24	**31**	0	**34**
Freeway w/o traj	0	30	17	24	**31**	0	0
Frostbite	65	4335	237	**1476**	259	909	1316
Gopher	258	2413	597	1675	2236	3730	**8240**
Hero	1027	30826	2657	7254	7037	**11161**	**11044**
James Bond	29	303	101	362	463	445	**509**
Kangaroo	52	3035	51	1240	838	**4098**	**4208**
Krull	1598	2666	2204	6349	6616	7782	**8413**
Kung Fu Master	256	22736	14862	24555	21760	21420	**26182**
Ms Pacman	307	6952	1480	1588	999	1327	**2673**
Pong	-21	15	13	**19**	15	**18**	11
Private Eye	25	69571	35	87	100	882	**7781**
Qbert	164	13455	1289	3331	746	3405	**4522**
Road Runner	12	7845	5641	9109	9615	15565	**17564**
Seaquest	68	42055	683	**774**	661	618	525
Up N Down	533	11693	3350	**15982**	3546	7667	7985
Human Mean	0%	100%	33%	96%	105%	112%	**126.7**%
Human Median	0%	100%	13%	51%	29%	49%	**58.4**%

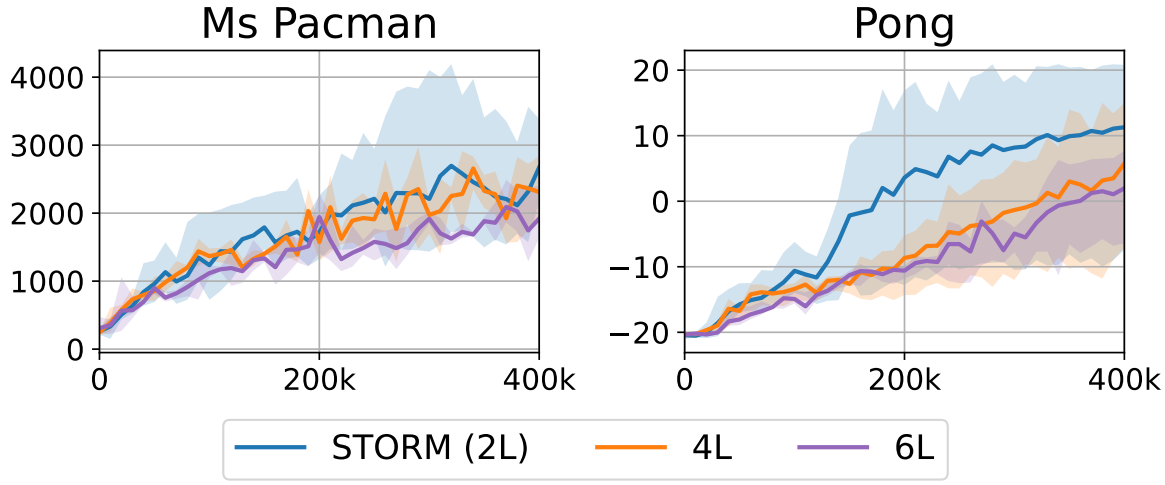
World Model Architecture Variants

- Two key architectural variants explored:
  - "Decoder at rear": Using $z_t \sim \hat{\mathcal{Z}}_t$ instead of $z_t \sim \mathcal{Z}_t$ for observation reconstruction. It shows that reconstruction loss should be applied directly to encoder output
  - "Predictor at front": Using z_t instead of h_t as input for prediction functions - minimal impact on single-frame reward tasks (e.g. *Pong*), but performance drops on multi-frame reward tasks (e.g. *Ms. Pacman*)



Model Size and Layer Ablation Study

- Default configuration uses 2 Transformer layers
  - Much smaller than 10-layer models in IRIS and TWM
- Layer ablation study findings:
  - More layers do not improve performance
  - In *Pong*, 4 and 6-layer models reach max reward even with 4x samples
- Three key factors explaining this scaling behavior:
  - Frame prediction is simplified by:
    - Small frame-to-frame differences, residual connections aiding prediction
  - Limited data in Atari 100k:
    - Lacks image diversity, therefore insufficient samples for larger models
  - End-to-end training effects:
    - $\mathcal{L}^{\text{rep}}$ loss directly impacts encoder: Large sequence models may overly influence encoder



Agent State Representation Study

- Options for agent's state s_t include:
  - Predicted observation $\hat{o}_t$, hidden state h_t, latent state z_t, or combined state $[h_t, z_t]$
- Key findings from ablation studies:
  - Context-dependent performance:
    - Including h_t improves performance in context-heavy games (e.g. *Ms. Pacman*)
    - Minimal impact in simpler games like *Pong*, *Kung Fu Master*
  - Limitations of using only h_t:
    - Can lead to catastrophic forgetting in policy-dependent environments
    - World model's non-stationarity creates instability
  - Benefits of randomness:
    - Including distributions like $\mathcal{Z}_t$ helps mitigate h_t-only issues