Simulus, SGF

July 2, 2025

Reference

Lior Cohen, Kaixin Wang, Bingyi Kang, Uri Gadot, and Shie Mannor. Uncovering untapped potential in sample-efficient world model agents, 2025. URL https://arxiv.org/abs/2502.11537.

Uncovering Untapped Potential in Sample-Efficient World Model Agents (Simulus)

Simulus [Cohen et al., 2025]

- **Simulus** builds on top of REM and following the same $\mathcal{V}$-$\mathcal{M}$-$\mathcal{C}$ structure.
- Tokenizer $\mathcal{V}$ encodes raw observations o_t and actions a_t into *fixed-length* sequence.
- World model $\mathcal{M}$ parameterized by θ, would embed and stream into a *RetNet* world model f_θ. (POP lets the RetNet predict the whole next observation in one go instead of token-by-token.)
- An ensemble of 4 identical heads predicts the same tokens. Their Jensen-Shannon disagreement is the intrinsic reward i_t.
- A LSTM Actor-Critic $\mathcal{C}$ is optimized with:

$$\bar{r}_t = \alpha_{\text{ext}} \hat{r}_t + \alpha_{\text{int}} i_t.$$

- Prioritized WM replay: world-model batches are 30% high-loss frames, 70% uniform.

Tokenizer $\mathcal{V}$

- Each modality would have its own encoder/decoder pair.

Modality	**Encoder**	**Tokens per obs**	**Vocab**
Image 64×64	3-level VQ-VAE	$8 \times 8 = 64$	512
Vector (cont.)	Scalar $\rightarrow$ 125-bin quantiser $\pm$symlog	len(vector)	125
Categorical	identity (already integer)	len(cat)	native
2-D grid	flatten, embed & average	mn	sizes per channel

Table: Tokenizer modalities and their encoding specifications

World Model $\mathcal{M}$

World Model input:

- Tokenizer converts raw observations o_t into tokens $z_t = (z_{t,1}, \ldots, z_{t,K})$.
- Each token $z_{t,j}$ is a small integer $(0 \ldots \text{vocab\_size} - 1)$
- Tokens are mapped to fixed-size embeddings $e_{t,j} \in \mathbb{R}^d$
- Concatenate K vectors to form observation block $E_t = (e_{t,1}, \ldots, e_{t,K})$
- Action tokens A_t are appended to E_t, forming input pair $\underbrace{E_t}_{\text{observation}}, \underbrace{A_t}_{\text{action tokens}}$.
- The world model's input stream follows the pattern: $E_1 A_1\ E_2 A_2\ E_3 A_3\ \ldots$

World Model architecture:

- **RetNet Architecture**: Transformer-like network where expensive self-attention is replaced by cheaper **Retention** operation
- **Sequential Operation**: At each step consumes one full (E, A) block and updates hidden state:

$$(h_t, x_t) = f_\theta\big(h_{t-1}, (\mathbf{E}_t, \mathbf{A}_t)\big)$$

  where h_t: recurrent state, x_t: prediction of next observation.
- **POP**: Uses learnable query sequence $\mathbf{U}$ to predict all K tokens in parallel
- **POP Implementation**:
  1. After processing block t, hold hidden state h_t
  2. Call RetNet with learnable $\mathbf{U}$: $(\_, y_t^u) = f_\theta(h_t, \mathbf{U})$
  3. Feed $y_t^u \in \mathbb{R}^{K \times d}$ rows into MLP heads to get $p_\theta(\hat{z}_{t+1,j} \mid y_{t,j}^u)$
- **Result**: Parallel K predictions enable $K\times$ faster generation

Intrinsic Reward Signal

Motivation: Extrinsic reward is sparse early in training. Model-based agents can measure their own ignorance - prediction disagreements indicate where more data is needed.

Implementation:

- **4 Independent Heads**: $p_{\phi_1}, p_{\phi_2}, p_{\phi_3}, p_{\phi_4}$ all predict from same RetNet
- **Jensen-Shannon Divergence (JSD)** measures disagreement:

$$u_j = H\left(\frac{1}{4}\sum_{i=1}^{4} p_{\phi_i}\right) - \frac{1}{4}\sum_{i=1}^{4} H(p_{\phi_i})$$

- **Average over tokens**: $i_t = \frac{1}{K}\sum_{j=1}^{K} u_j$
- **Combined reward**: $\bar{r}_t = \alpha_{\text{ext}}\hat{r}_t + \alpha_{\text{int}} i_t$

Architecture:

- All heads see **stop-gradient** inputs for JSD computation
- JSD bounded in $[0, \log V]$ - mixes well with any reward scale
- High JSD frames get high intrinsic reward and prioritized replay
- Creates feedback loop: explore $\rightarrow$ learn $\rightarrow$ reduce uncertainty $\rightarrow$ shift exploration

Prioritized Replay for World Model

Implementation:

- Store latest obs-loss with each transition
- WM batch sampling: 70% uniform, 30% softmax(loss)
- New frames: high dummy loss for guaranteed sampling

Benefits:

- Focus on hard-to-predict transitions
- New experiences always sampled
- Balanced learning via uniform component

Controller $\mathcal{C}$

Overview: Outputs stochastic policy $\pi_\psi(a_t \mid \tau_t)$ and value estimate $V_\psi(\tau_t)$

Recurrent Backbone:

- Single-layer LSTM
- $h_t, c_t = \text{LSTM}(v_t, h_{t-1}, c_{t-1})$
- Constitutes majority of controller parameters ψ

Input Pipeline:

- **Modality Encoders**: Each token type $z_t^{(i)}$ gets encoded via $E^{(i)} : \{0, \ldots, V_i - 1\}^{K_i} \to \mathbb{R}^{d_i}$
- **Fusion MLP**: $v_t = g_{\text{fuse}}([v_t^{(0)} \| v_t^{(1)} \| \cdots]) \in \mathbb{R}^{d_{\text{fuse}}}$

Training:

- Generate imagined trajectory $\hat{\tau} = (z_1, a_1, \bar{r}_1, \ldots, z_H, a_H, \bar{r}_H)$
- λ-returns + REINFORCE with value baseline + entropy regularization like Dreamer.

Results

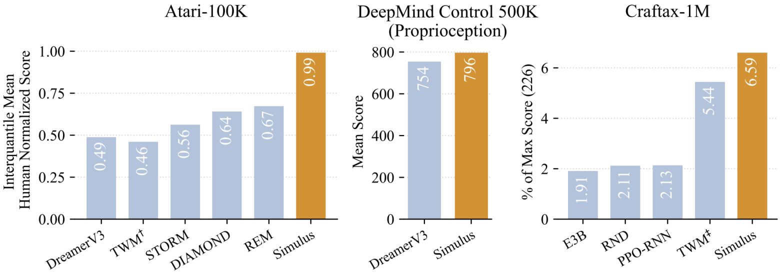

Figure: Simulus results

Simple, Good, Fast: Self-Supervised World Models Free of Baggage (SGF)

Aim: Strip world-models to bare essentials - can we do well without RNNs/Transformers, discrete latents, or pixel reconstructions?

Core Ingredients:

- Self-supervised representations (VICReg-style)
- Two losses: temporal consistency + information maximization
- Frame + action stacking (captures short-term dependencies)
- Strong image augmentations (injects stochasticity)
- Deterministic, feed-forward dynamics (MLP predicts Δ-latent, reward, terminal)

Why it works:

- Learned latents are informative and locally smooth $\rightarrow$ simple dynamics suffices
- Augmentations stand in for stochastic modelling

Ingredients Leading to Simplicity

Stacking instead of memory:

- Traditional: RNNs/attention for long-term dependencies
- Their approach: *frame and action stacking*
- Stack m recent frames $\rightarrow$ captures velocity, partial observability
- Stack m recent actions $\rightarrow$ handles action delays
- Much faster than recurrent/attention mechanisms

Augmentations instead of stochasticity:

- Prior models: stochastic even in deterministic POMDPs
- Our approach: *data augmentation* for stochasticity
- Random augmentations during training $\rightarrow$ robustness
- Avoids computational overhead of stochastic predictions
- Similar to successful model-free approaches (DrQ, RAD)

Key insight: Simple alternatives achieve similar performance with much less complexity

Model Structure

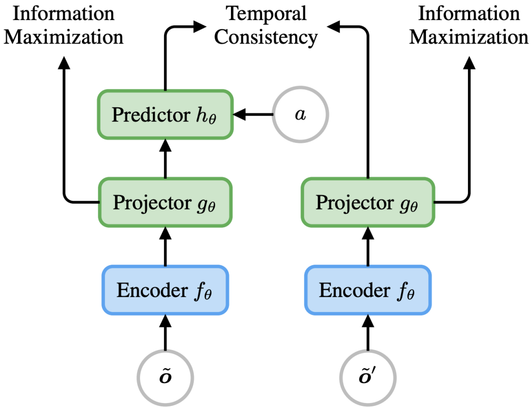

Figure: Model Structure

Representation Learning

- Given transition $(\mathbf{o}, \mathbf{a}, \mathbf{o}', r, e)$, apply random augmentations $t, t' \sim \mathcal{T}$ to get $\tilde{\mathbf{o}} = t(\mathbf{o})$ and $\tilde{\mathbf{o}}' = t'(\mathbf{o}')$
- Encoder f_{enc} computes representations $\tilde{\mathbf{y}} = f_{\text{enc}}(\tilde{\mathbf{o}})$ and $\tilde{\mathbf{y}}' = f_{\text{enc}}(\tilde{\mathbf{o}}')$
- Projector f_{proj} computes embeddings $\tilde{\mathbf{z}} = f_{\text{proj}}(\tilde{\mathbf{y}})$ and $\tilde{\mathbf{z}}' = f_{\text{proj}}(\tilde{\mathbf{y}}')$
- Action-conditioned predictor f_{pred} predicts $\hat{\mathbf{z}}' = f_{\text{pred}}(\tilde{\mathbf{z}}, \mathbf{a})$

Loss Function:

$$\mathcal{L}_{\text{Repr.}}(\theta) = \mathbb{E}_{\tau}\Big[\underbrace{\tfrac{\eta}{D}\|f_{\text{pred}}(\tilde{\mathbf{z}}, \mathbf{a}) - \tilde{\mathbf{z}}'\|_2^2}_{\text{Temporal Consistency}} + \underbrace{\mathrm{VC}(\tilde{\mathbf{Z}}) + \mathrm{VC}(\tilde{\mathbf{Z}}')}_{\text{Information Maximization}}\Big]$$

Variance-Covariance Regularization:

$$\mathrm{VC}(\mathbf{Z}) = \frac{1}{D}\sum_{j=1}^{D}\Big[\underbrace{\rho \max\Big(0, 1 - \sqrt{\mathrm{Cov}(\mathbf{Z})_{j,j} + \varepsilon}\Big)}_{\text{Variance}} + \underbrace{\nu \textstyle\sum_{k \neq j} \mathrm{Cov}(\mathbf{Z})_{j,k}^2}_{\text{Covariance}}\Big]$$

Where $\mathbf{Z}$ is the set of all embeddings in the batch.

Variance regularization keeps the standard deviation of each embedding feature across the batch above 1 using a hinge loss. **C**ovariance regularization decorrelates the embedding features by attracting their covariances towards zero

Dynamics Model

Maximum likelihood estimation:

$$\mathcal{L}_{\text{Dyn.}}(\theta) = \mathbb{E}_{\tau}\left[\underbrace{-\log p_\theta(\mathrm{sg}(\mathbf{y}')|\mathrm{sg}(\mathbf{y}), \mathbf{a})}_{\text{Transition Distribution}} - \underbrace{\log p_\theta(r|\tilde{\mathbf{y}}, \mathbf{a}, \tilde{\mathbf{y}}')}_{\text{Reward Distribution}} - \underbrace{\log p_\theta(e|\tilde{\mathbf{y}}, \mathbf{a}, \tilde{\mathbf{y}}')}_{\text{Terminal Distribution}} \right]$$

Key design choices:

- Stop-gradient $\mathrm{sg}(\cdot)$ on representations in transition loss
- Prevents moving targets from representation model updates
- Rewards/terminals provide stable POMDP signals
- Learn transitions with non-augmented observations: $\mathbf{y} = f_{\text{enc}}(\mathbf{o})$, $\mathbf{y}' = f_{\text{enc}}(\mathbf{o}')$

Results

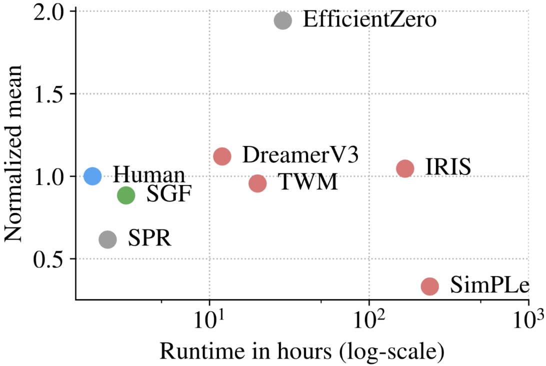

Figure: Score and runtime comparison in the Atari 100k benchmark. SPR is model-free, EfficientZero performs lookahead.

Limitations

1. Limited to MDPs, not POMDPs:

- Only works with **deterministic MDPs**
- Cannot handle **non-deterministic POMDPs** because:
  - Transition distribution needs to be **stochastic**
  - Predictor must handle **uncertainty** between observations
  - Both networks need **stochastic predictions** (mean/variance or Gaussian mixtures)
- Limited to **short-term dependencies** (avoided RNN/Transformers)
- This might explain why we didn't reach **state-of-the-art performance**

2. VICReg Image Requirement:

- VICReg currently requires **image observations**
- Could be applied to **other modalities** if reasonable augmentations are available
- Could combine SGF with **other self-supervised learning methods**