Hieros: Hierarchical Imagination on Structured State Space Sequence World Models

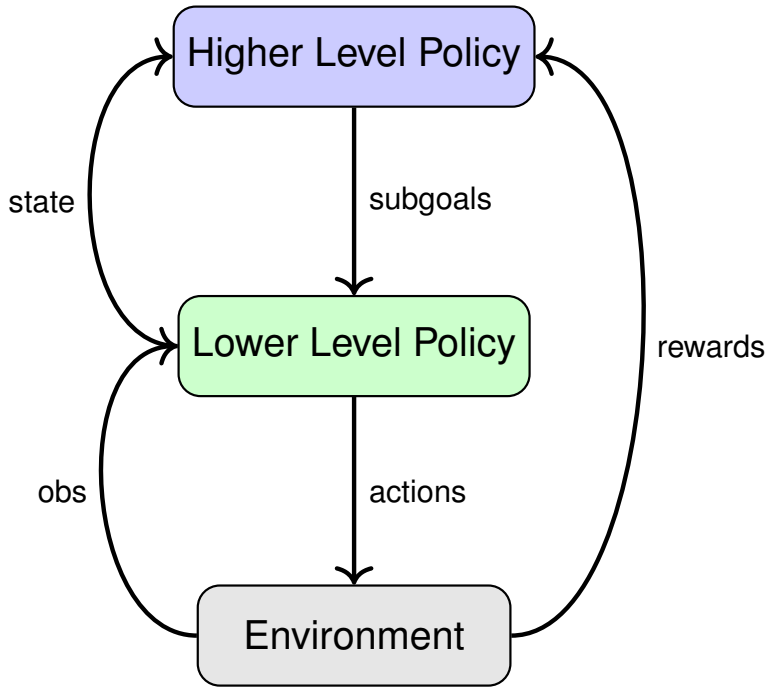
May 21, 2025

Reference

Paul Mattes, Rainer Schlosser, and Ralf Herbrich. Hieros: Hierarchical imagination on structured state space sequence world models, 2024. URL https://arxiv.org/abs/2310.05167.

Hierarchical Reinforcement Learning (HRL)

- Improves RL sample efficiency by operating at different time scales
- Agent learns across multiple levels of abstraction
- Higher level policies create subtasks/subgoals
- Lower level policies fulfill these subgoals
- Benefits:
  - Reduces decision complexity
  - Improves exploration
  - Better transfer learning
  - More interpretable behavior
- Successfully applied in various complex environments including Atari games
- Enables agents to solve long-horizon tasks more efficiently



Contributions

- HIEROS [Mattes et al., 2024]: A Hierarchical RL agent with multi-layered world models enabling long-term planning and precise predictions
- S5WM: A world model using S5 layers with efficient design and internal state access
- ETBS: $O(1)$ time complexity experience sampling method
- SOTA performance on Atari100k benchmark
- Comprehensive ablation studies validating their hierarchical approach

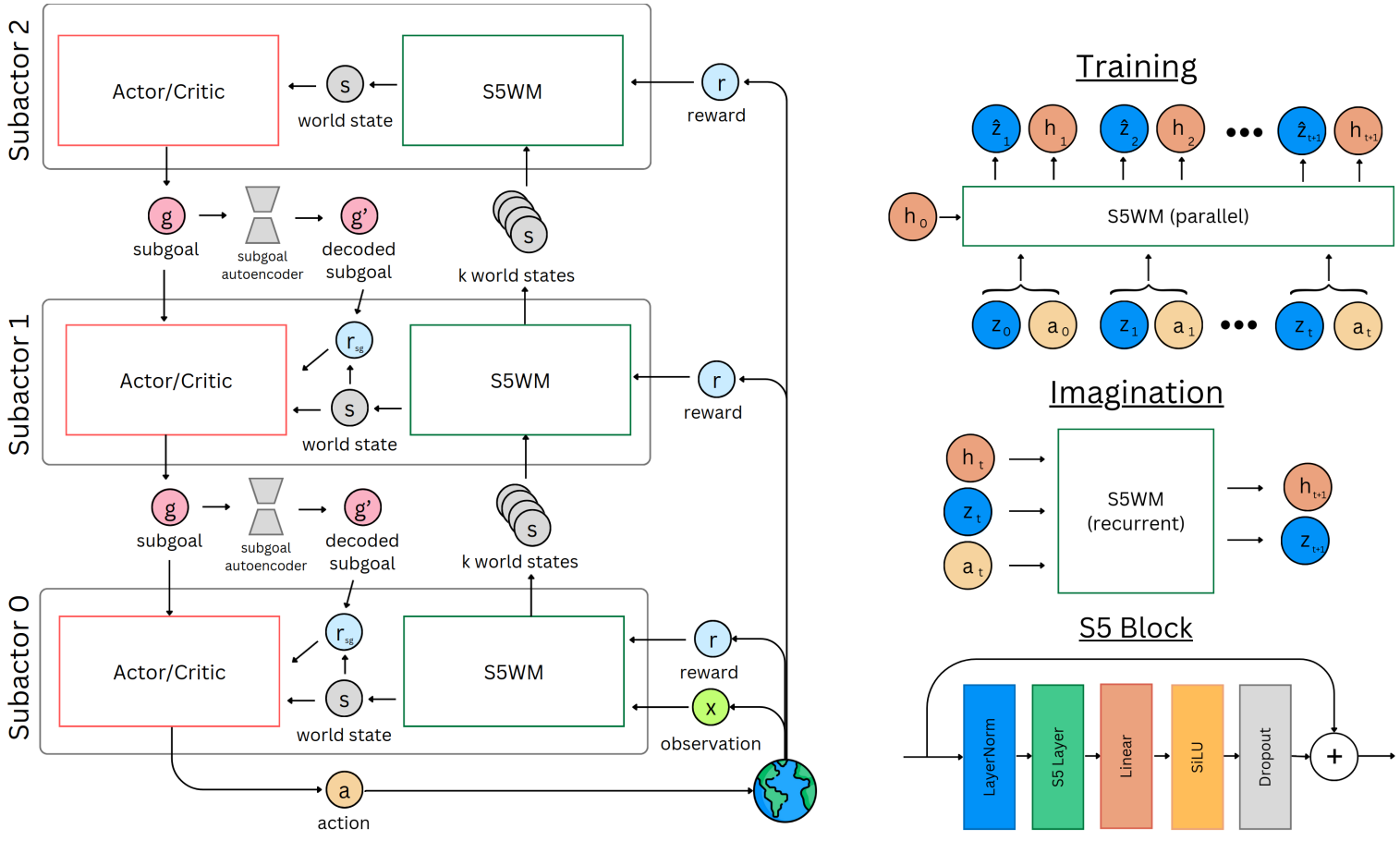


Figure: On the left: Hierarchical subactor structure of HIEROS. Each layer of the hierarchy learns its own latent state world model and interacts with the other layers via subgoal proposal. The action outputs of each actor/critic is the subgoal input of the next lower layer. The output of the lowest level actor/critic is the actual action in the real environment. On the right: Training and imagination procedure of the S5WM. HIEROS uses a stack of S5 blocks with their architecture shown above.

Overall Architecture

- **Hierarchy of subactors**: $L+1$ levels (indexed $i = 0, \ldots, L$)
  - Level 0 (the "worker") acts in the real environment, taking primitive actions
  - Higher levels ($i \geq 1$) output *subgoals* for the level immediately below ($i - 1$)
- **Time-scale separation**: Each higher level holds its subgoal constant for k lower-level steps

At each level i, three modules:

1. A *world model* w_θ^i
2. An *actor–critic* π_ϕ^i
3. A *subgoal autoencoder* g_ψ^i

World Model & Imagined Rollouts

Each subactor trains its own S5-style world model w_θ^i to predict future latent states:

- **Input:** either real env. observations (for $i = 0$) or the lower-level's imagined states aggregated over k steps (for $i \geq 1$)
- **Usage:** the world model then *imagines* trajectories, allowing actor–critic training without further environment calls

Subgoal Autoencoder

The autoencoder at level i compresses high-dimensional model states into a small discrete space of subgoals:

$$\begin{aligned} \text{Encoder:} \quad & g_t \sim p_\psi(g_t \mid h_t)\,, \\ \text{Decoder:} \quad & \hat{h}_t = f_\psi(g_t) \approx h_t. \end{aligned}$$

- h_t is the *deterministic* part of the world model state
- The discrete code g_t is much lower-dimensional than h_t

Training loss (a standard β-VAE):

$$\mathcal{L}_\psi = \|f_\psi(z) - h_t\|_2 + \beta\, \mathrm{KL}\big[p_\psi(g_t \mid h_t) \,\big\|\, q(g_t)\big],$$

where $q(g)$ is a uniform prior over codes.

Action & Policy Inputs

At each time step the actor π_ϕ^i receives:

$$a_t \sim \pi_\phi^i\big((h_t, z_t),\, g_t,\, r_t,\, c_t,\, H\big(p_\theta(z_t \mid m_t)\big)\big).$$

- (h_t, z_t) - full latent state from the world model
- g_t - the *current* subgoal
- r_t - the reward signal at level i
- c_t - continue flag (e.g. episode not done)
- $H\big(p_\theta(z_t \mid m_t)\big)$ - the entropy of the stochastic latent

Reward Composition

Each subactor maximizes a *mixed* reward:

$$r = r_{\text{extr}} + w_g\, r_g + w_{\text{nov}}\, r_{\text{nov}}$$

where:

- r_{extr} is the environment's raw reward (only nonzero at level 0)
- **Subgoal reward** r_g measures alignment with the intended subgoal:

$$r_g = \frac{g_t^\top\, h_t}{\max\big(\|g_t\|, \|h_t\|\big)}$$

- **Novelty reward** r_{nov} is the reconstruction error:

$$r_{\text{nov}} = \left\| h_t - g_\psi^i(h_t) \right\|_2$$

Multilayered Hierarchical Imagination

- **Multi-Level Interaction**
  - **Level 1** observes *all* of the worker's imagined states for k steps, then proposes a new subgoal g^1
  - **Level 2** does the same on level 1's imagined sequences, and so on
  - Earlier works like Director only feed every kth state to the higher policy
- **Explainability & Visualization**
  - Each subgoal code g^i can be decoded back into a world-model latent
  - Makes hierarchical decisions interpretable
- **Why it works**
  - **Time abstraction** via holding subgoals constant for k steps
  - **State abstraction** via compressing h_t into a low-dimensional discrete code
  - **Intrinsic motivation** from novelty and subgoal rewards
  - **Scalability** to more than two levels

S5-based World Model (S5WM)

- **Core Components** (at each hierarchical level):
  - **Sequence model:** $(m_t, h_t) = f_\theta\left(h_{t-1},\, z_{t-1},\, a_{t-1}\right)$
  - **Encoder:** $z_t \sim q_\theta\left(z_t \mid o_t\right)$ (depends only on o_t, enabling parallel computation)
  - **Dynamics predictor:** $\hat{z}_t \sim p_\theta\left(\hat{z}_t \mid m_t\right)$ (used during imagination)
  - **Reward & Continue predictors:** $\hat{r}_t \sim p_\theta\left(\hat{r}_t \mid h_t, z_t\right)$, $\hat{c}_t \sim p_\theta\left(\hat{c}_t \mid h_t, z_t\right)$
  - **Decoder:** $\hat{o}_t \sim p_\theta\left(\hat{o}_t \mid h_t, z_t\right)$
- **Training vs. Imagination**
  - **Training:** Uses posterior $z_t \sim q(z_t \mid o_t)$ from real observations
  - **Imagination:** Samples $\hat{z}_t \sim p(\hat{z}_t \mid m_t)$ without real observations
- **Advantages of S5**
  - Parallel & autoregressive processing
  - Long-range memory via structured state-space kernels
  - Linear (not quadratic) memory cost in sequence length
  - Direct use of recurrent hidden state as world-model component
- **Episode-Reset Mechanism**
  - Binary continue flags $c_0, \ldots, c_n$ passed to S5 blocks
  - When $c_t = 0$ (episode start), internal state h_t resets to learned initial h_0

S5WM Loss Functions

All parts trained end-to-end under a DreamerV3-style loss:

$$\mathcal{L}(\theta) = \underbrace{\mathcal{L}_{\text{pred}}}_{\text{reward, cont., obs.}} + \alpha_{\text{dyn}} \underbrace{\mathcal{L}_{\text{dyn}}}_{\text{KL}\left[\text{sg}\, q \| p\right]} + \alpha_{\text{rep}} \underbrace{\mathcal{L}_{\text{rep}}}_{\text{KL}\left[q \| \text{sg}\, p\right]}$$

- **Prediction loss**

$$\mathcal{L}_{\text{pred}} = -\ln p_\theta(r_t \mid h_t, z_t) - \ln p_\theta(c_t \mid h_t, z_t) - \ln p_\theta(o_t \mid h_t, z_t)$$

- **Dynamics loss**

$$\mathcal{L}_{\text{dyn}} = \max\Big(1,\ \mathrm{KL}\big[\mathrm{sg}(q_\theta(z_t \mid o_t)) \,\big\|\, p_\theta(z_t \mid m_t)\big]\Big)$$

- **Representation loss**

$$\mathcal{L}_{\text{rep}} = \max\Big(1,\ \mathrm{KL}\big[q_\theta(z_t \mid o_t) \,\big\|\, \mathrm{sg}(p_\theta(z_t \mid m_t))\big]\Big)$$

Efficient Time-balanced Sampling (ETBS)

Problem: Uniform sampling from growing replay buffer leads to imbalance

- Older entries (small index i) get oversampled
- Recent experiences are under-represented

Quantifying the imbalance:

- Expected draw count for entry x_i after n samplings:

$$\mathbb{E}[N_{x_i}] = \sum_{k=i}^{n} \frac{1}{k} = H_n - H_{i-1} \approx \ln(n) - \ln(i-1)$$

- Normalized probability:

$$p_i = \frac{\mathbb{E}[N_{x_i}]}{n} \approx \frac{\ln(n) - \ln(i)}{n}$$

Solution, from Imbalanced to Uniform via the CDF:

- **Key insight (Probability Integral Transform):** If $u \sim p(i)$ and $v = \mathrm{CDF}_p(u)$, then v is uniformly distributed on $[0, 1]$
- **Practical implementation:**
  - Precompute the CDF of $p(i) \approx (\ln n - \ln i)/n$
  - To sample: draw uniform $u \in [0, 1]$, then compute $i = \mathrm{CDF}_p^{-1}(u)$
  - This remaps the skewed distribution back to (nearly) uniform

ETBS Temperature-controlled Interpolation

Temperature-controlled interpolation:

$$p_{\text{ETBS}}(i) = \tau \underbrace{\text{CDF}_p\big(p(i)\big)}_{\text{"uniformized"}} + (1-\tau)\, p(i)$$

where $\tau \in [0, 1]$ controls uniformity vs. original bias ($\tau = 0.3$ in practice)

Constant-time implementation:

- Precompute lookup table for CDF_p over $i = 1, \ldots, n$
- Sampling via table lookup or binary search to invert CDF
- All steps run in $O(1)$ time per sample

Benefits:

- Balances recency and coverage
- Light bias toward older transitions improves stability
- $O(1)$ efficiency vs. $O(n)$ in prior time-balanced methods

Experiments - Summary

- **Setup:** Evaluated on Atari100k benchmark (25 games, 100k steps)
- Compared against SimPLe and imagination-based methods (TWM, IRIS, DreamerV3)
- **Results:** New state-of-the-art in mean/median human-normalized scores
  - Matches or beats other methods on 9 of 25 games
  - Trains in $\sim$14h per game (comparable to DreamerV3/TWM, faster than IRIS)
- **Strengths:** Excels in games with shifting dynamics across levels
  - Hierarchical subgoals direct exploration effectively
  - Examples: Frostbite, James Bond, PrivateEye
- **Limitations:** Weaker on simple-dynamics games (Breakout, Pong)
  - S5WM struggles with uniform dynamics
  - Hierarchy less beneficial when subgoals aren't informative
- **Analysis:** S5WM better models long-term, shifting dynamics
  - Successfully predicts level transitions in complex games
  - Trade-off between model size and environment complexity

Experiments - Atari100k Results

Task	Random	Human	SimPLe	TWM	IRIS	DreamerV3	Hieros
Alien	228	7 128	617	675	420	**959**	828
Amidar	6	1 720	74	123	143	139	127
Assault	222	742	527	683	1 524	706	**1 764**
Asterix	210	8 503	**1 128**	1 117	854	932	899
BankHeist	14	753	34	467	53	**649**	177
Battle Zone	2 360	37 188	4 031	5 068	13 074	12 250	**15 140**
Boxing	0	12	8	78	70	**78**	65
Breakout	2	30	16	20	**84**	31	10
Chop.Command	811	7 388	979	**1 697**	1 565	420	1 475
CrazyClimber	10 780	35 829	62 584	71 820	59 324	**97 190**	50 857
DemonAttack	152	1 971	208	350	**2 034**	303	1 480
Freeway	0	30	17	24	**31**	0	**31**
Frostbite	65	4 335	237	1 476	259	909	**2 901**
Gopher	258	2 412	597	1 675	2 236	**3 730**	1 473
Hero	1 027	30 826	2 657	7 254	7 037	**11 161**	7 890
JamesBond	29	303	100	362	463	445	**939**
Kangaroo	52	3 035	51	1 240	838	4 098	**6 590**
Krull	1 598	2 666	2 205	6 349	6 616	7 782	**8 130**
KungFuMaster	258	22 736	14 862	**24 555**	21 760	21 420	18 793
Ms.Packman	307	6 952	1 480	1 588	999	1 327	**1 771**
Pong	-21	15	13	**18**	15	**18**	5
PrivateEye	25	69 571	35	86	100	882	**1 507**
Qbert	164	13 455	1 289	3 331	746	**3 405**	770
RoadRunner	12	7 845	5 641	9 109	9 615	15 565	**16 950**
Seaquest	68	42 055	683	**774**	661	618	560
Mean	0	100	34	96	105	112	**120**
Median	0	100	11	50	29	49	**56**
IQM	0	100	13	46	50	N/A	**53**
Optimality Gap	100	0	73	52	51	N/A	**49**

Experiments - Rewards Graph

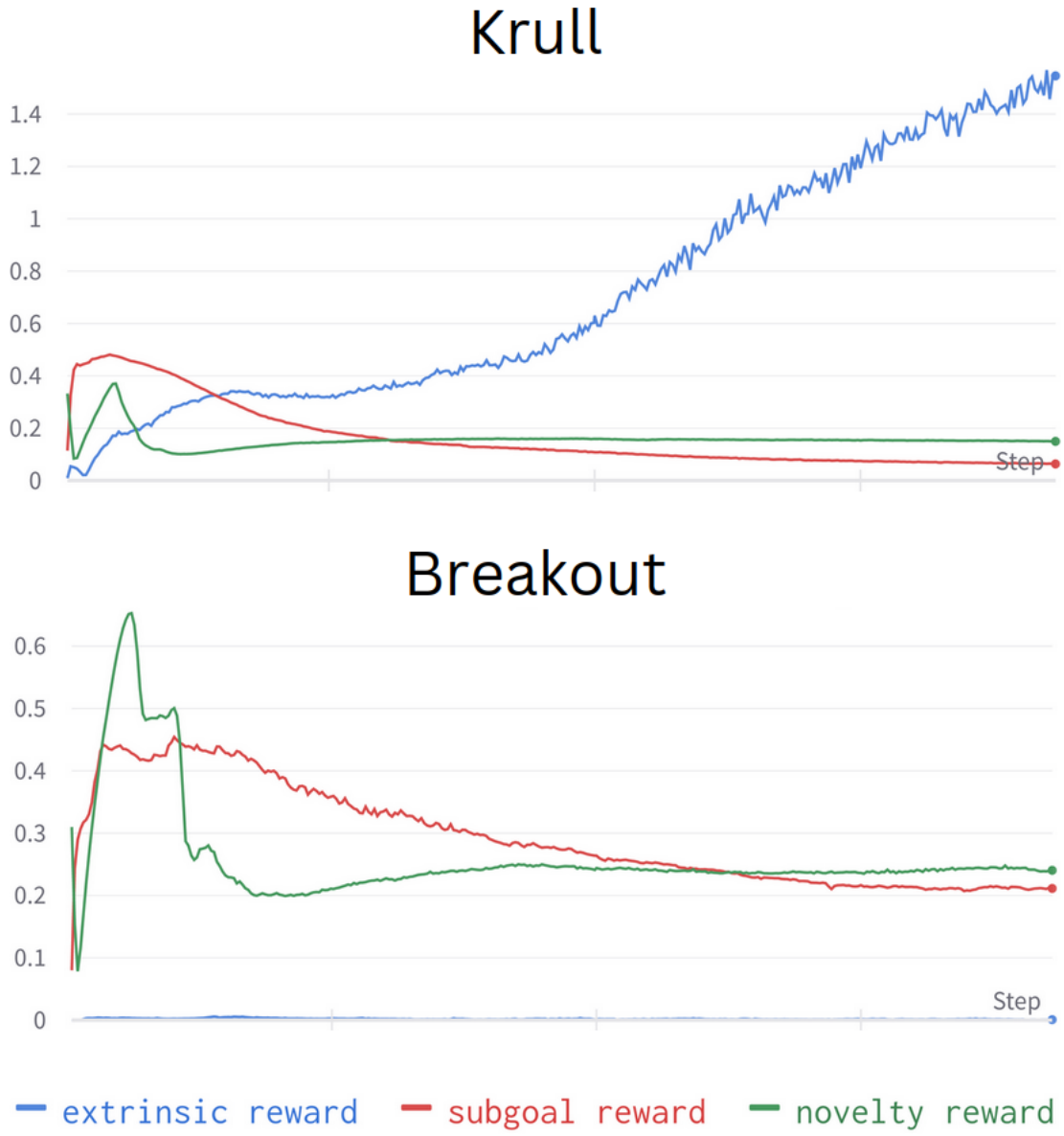


Figure: Extrinsic, subgoal, and novelty rewards per step for Krull (top) and Breakout (bottom) for the lowest level subactor.

Experiments - Reward Analysis

Analysis of partial rewards (r_{extr}, r_{nov}, and r_{sg}) for the lowest level subactor in Breakout and Krull reveals:

- Breakout: Sparse extrinsic rewards provide little guidance for score improvement
  - Subactor relies heavily on higher-level subgoals
  - Strict adherence to subgoals hinders performance
- Krull: More frequent extrinsic rewards offer clearer direction
  - Subactor treats higher-level subgoals as flexible hints
  - Leads to better overall performance

Experiments - RSSM vs. S5WM

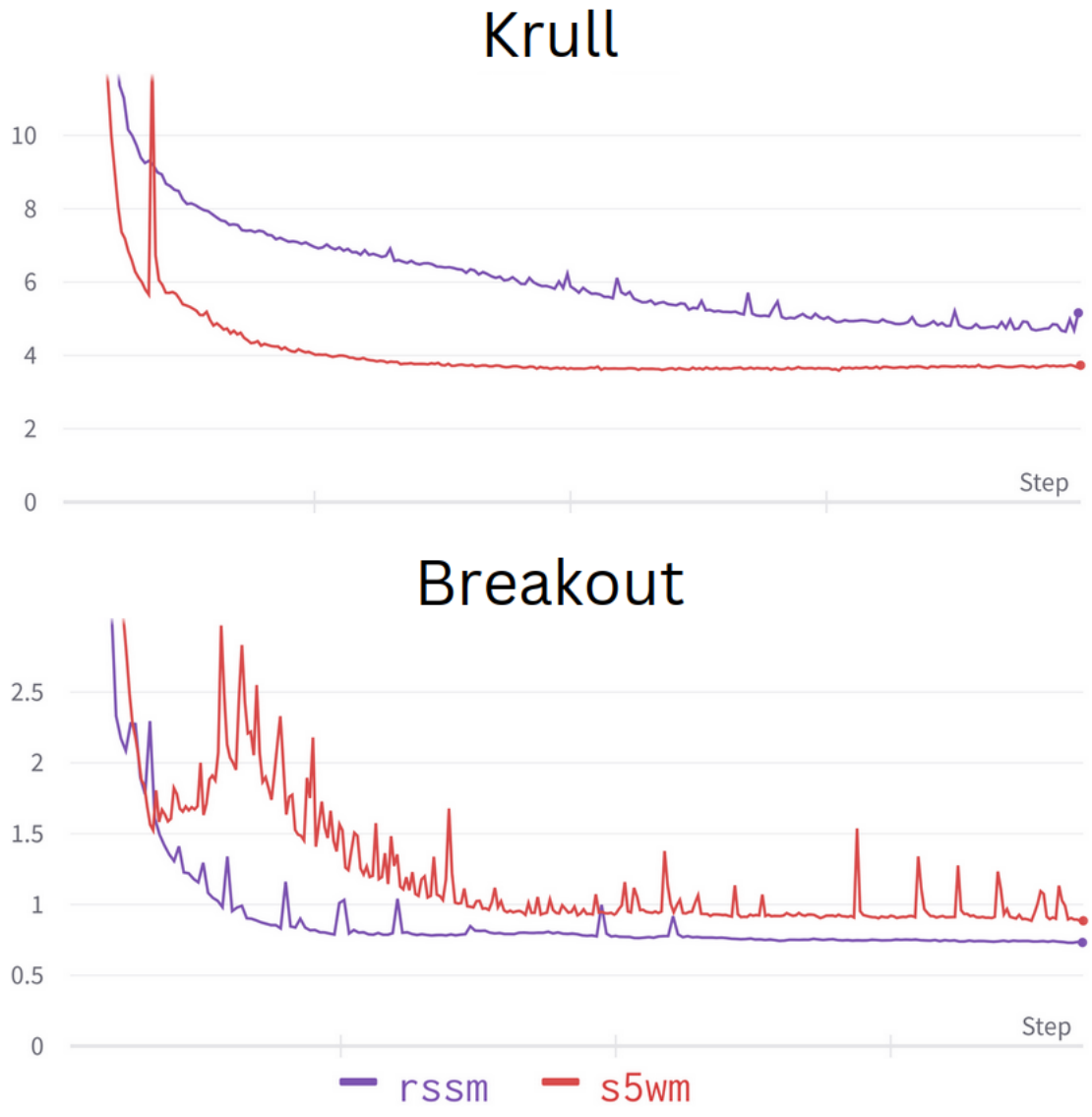


Figure: World model losses for the S5WM and RSSM for Krull and Breakout. The S5WM is able to achieve an overall lower world model loss compared to the RSSM for Krull, while those roles are reversed for Breakout.

Experiments - S5WM vs. RSSM Analysis

To directly compare our S5WM architecture with the RSSM used in DreamerV3, the S5WM was replaced with an RSSM and trained HIEROS on four different games:

- RSSM underperforms S5WM in Krull, Battle Zone, and Freeway
- RSSM slightly outperforms S5WM in Breakout
- S5WM achieves lower overall world model loss in Krull
- RSSM achieves lower loss in Breakout

Key insights:

- S5WM excels in complex environments with substantial input distribution shifts
- RSSM performs better in simpler environments with stable input distributions
- The difference is particularly notable given Krull's higher absolute loss values