Transformer-based World Models Are Happy With 100k Interactions

April 30, 2025

Reference

Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan Harmeling. Transformer-based world models are happy with 100k interactions, 2023. URL https://arxiv.org/abs/2303.07109.

Transformer-based World Model [Robine et al., 2023]

- Transformer-based world model (TWM) generates meaningful, new experience
- Used to train a policy that outperforms previous model-free and model-based RL algorithms on Atari 100k benchmark
- Based on Transformer-XL architecture:
  - Multiple self-attention layers with residual connections
  - Causal masking prevents accessing future time steps
  - Computationally efficient at inference time
  - Uses relative positional encodings (removes dependence on absolute time steps)

Contributions

1. New autoregressive world model based on Transformer-XL
   - Model-free agent trained in latent imagination
   - Transformer not needed at inference time
2. Improved world model with reward feedback
3. Rewritten balanced KL divergence loss
4. New thresholded entropy loss
   - Stabilizes policy entropy during training
5. Effective sampling procedure for growing experience dataset
6. Excellent results on Atari 100k benchmark
   - Comparison with recent sample-efficient methods
   - Reporting of empirical confidence intervals

TWM Architecture

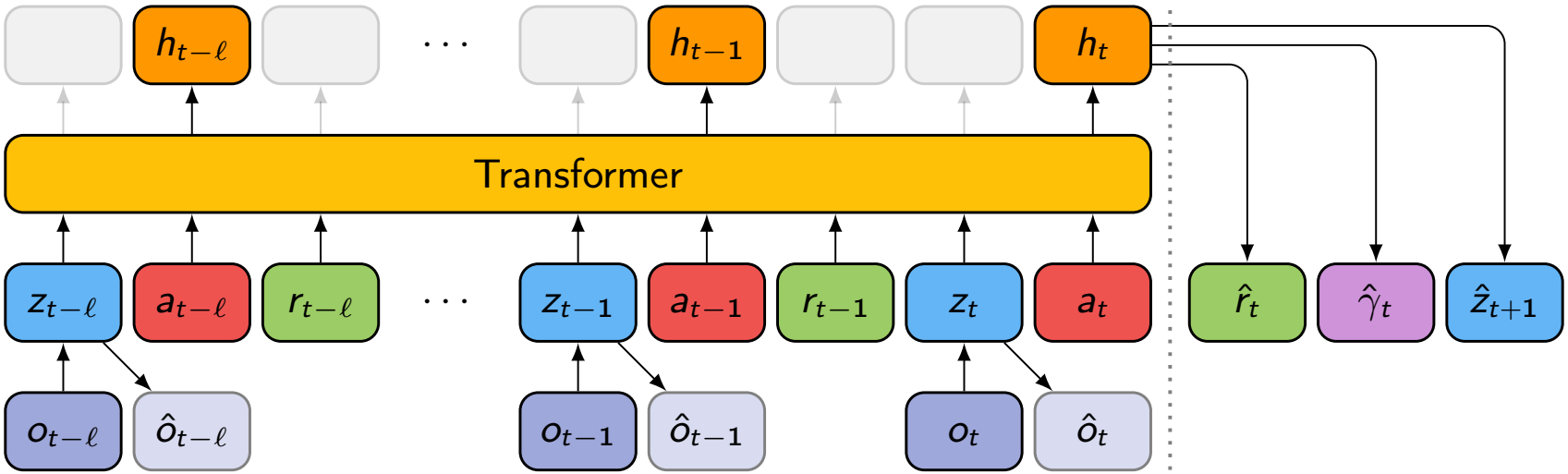


Figure: TWM architecture. Observations $o_{t-\ell:t}$ are encoded using a CNN. Linear embeddings of stochastic, discrete latent states $z_{t-\ell:t}$, actions $a_{t-\ell:t}$, and rewards $r_{t-\ell:t}$ are fed into a transformer, which computes a deterministic hidden state h_t at each time step. Predictions of the reward r_t, discount factor γ_t, and next latent state z_{t+1} are computed based on h_t using MLPs.

World Model (1/3): Observation Model

- The observation model is a VAE that compresses raw observations into compact latent states:

$$\text{Encoder:} \quad z_t \sim p_\phi(z_t \mid o_t) \tag{1}$$

$$\text{Decoder:} \quad \hat{o}_t \sim p_\phi(\hat{o}_t \mid z_t) \tag{2}$$

- **Why a VAE?**
  - **Compression:** Extracts essential features into a small latent representation
  - **Stochasticity:** Sampling adds regularizing noise that prevents overfitting
- **Discrete latents:** z_t consists of 32 categorical variables with 32 categories
  - This "discrete bottleneck" improves representation learning
  - Forces clear distinctions between different latent states
- **Frame stacking:** Although the VAE only processes o_t, each observation is already a stack of four recent frames, providing limited temporal context

World Model (2/3): Autoregressive Dynamics Model

The dynamics model predicts "what happens next" given everything it has produced so far:

$$\text{Aggregation:} \quad h_t = f_\psi(z_{t-\ell:t}, a_{t-\ell:t}, r_{t-\ell:t-1}) \tag{3}$$

$$\text{Reward predictor:} \quad \hat{r}_t \sim p_\psi(\hat{r}_t \mid h_t) \tag{4}$$

$$\text{Discount predictor:} \quad \hat{\gamma}_t \sim p_\psi(\hat{\gamma}_t \mid h_t) \tag{5}$$

$$\text{Latent state predictor:} \quad \hat{z}_{t+1} \sim p_\psi(\hat{z}_{t+1} \mid h_t) \tag{6}$$

Transformer-XL Architecture:

- **Causal masking:** Ensures attention only to past tokens
- **Recurrence cache:** Remembers information from arbitrarily far in the past
- **Relative positional encodings:** Remove dependence on absolute time indices
- **Input tokens:** Interleaved latent states, actions, and rewards for past ℓ steps

World Model (3/3): Why Fully Autoregressive Dynamics?

Beneficial Properties:

- **Autoregression:** Model directly sees its own past predictions
- **Parallel training:** Process all time steps at once (unlike RNNs)
- **Cached inference:** Only process newest tokens at test time
- **Long-term dependencies:** Capture far-past influences efficiently

Advantages over traditional approaches:

- **Richer dependencies:** Each past latent can be directly attended to
- **Robustness to prediction errors:** Model knows about past errors and can correct course
- **Adaptation to own noise:** Can react to randomness introduced by its own sampling

Intuition: By giving the dynamics model direct access to everything it has generated so far, it learns complex, self-aware temporal patterns. Transformers + recurrence provide scalability and memory, while autoregression enables adaptability to its own predictions.

Loss Function (1/2): Observation Model Training

Observation Model Loss:

$$\mathcal{L}_{\phi}^{\text{Obs.}} = \mathbb{E}\Bigg[\sum_{t=1}^{T} \underbrace{-\ln p_{\phi}(o_t \mid z_t)}_{\text{decoder}} \underbrace{- \alpha_1 H(p_{\phi}(z_t \mid o_t))}_{\text{entropy regularizer}} + \underbrace{\alpha_2 H(p_{\phi}(z_t \mid o_t), p_{\psi}(\hat{z}_t \mid h_{t-1}))}_{\text{consistency}}\Bigg] \tag{7}$$

, while H denotes the cross entropy loss function.

Components Explained:

- **Decoder loss:** Teaches the model to reconstruct observations from latent states
- **Entropy regularizer:** Ensures latent space diversity by preventing collapse
- **Consistency loss:** Forces encoder to produce latents similar to what dynamics predicts
- Hyperparameters α_1, α_2 let us balance exploration vs. consistency

Loss Function (2/2): Dynamics Model Training

Dynamics Model Loss:

$$\mathcal{L}_{\psi}^{\text{Dyn.}} = \mathbb{E}\left[\sum_{t=1}^{T} \underbrace{H(p_{\phi}(z_{t+1} \mid o_{t+1}), p_{\psi}(\hat{z}_{t+1} \mid h_t))}_{\text{latent state predictor}} - \underbrace{\beta_1 \ln p_{\psi}(r_t \mid h_t)}_{\text{reward predictor}} - \underbrace{\beta_2 \ln p_{\psi}(\gamma_t \mid h_t)}_{\text{discount predictor}}\right] \tag{8}$$

, while H denotes the cross entropy loss function.

Components Explained:

- **Latent state predictor:** Learns to predict the next latent state that the encoder would produce
- **Reward predictor:** Learns to anticipate rewards from current history
- **Discount predictor:** Learns to predict episode terminations ($\gamma_t = 0$ at episode end)
- Coefficients β_1, β_2 balance importance of reward vs. termination prediction

Key Benefits:

- The balanced approach prevents either distribution from dominating
- Dynamics model can adapt to changing environments while maintaining stability
- Explicit cross-entropy terms give finer control than standard VAE training
- Self-supervised nature allows learning from unlabeled experience

Policy Learning (1/3): Actor-Critic on Imagined Trajectories

Key Components:

- **Imagined Rollouts:** Generate trajectories entirely inside the world model
- **Actor-Critic Architecture:**
  - Actor: $\pi_\theta(a_t \mid \hat{z}_t)$ with parameters θ
  - Critic: $V_\xi(\hat{z}_t)$ with parameters ξ
- **Advantage Estimation:** Use GAE with predicted discounts $\hat{\gamma}_t$

$$A_t = \sum_{k=0}^{\infty} \left(\prod_{i=0}^{k-1} \hat{\gamma}_{t+i} \right) r_{t+k} \; - \; V_\xi(\hat{z}_t) \tag{9}$$

- **Discount-Weighted Losses:** Multiply by cumulative product of $\hat{\gamma}$ to softly down-weight terms after predicted episode end

Policy Learning (2/3): Thresholded Entropy

Thresholded Entropy Loss:

$$\mathcal{L}_\theta^{\text{Ent.}} = \max\left(0,\ \Gamma\ -\ \frac{H(\pi_\theta)}{\ln m}\right) \tag{10}$$

where $0 \leq \Gamma \leq 1$ is the threshold hyperparameter, $H(\pi_\theta)$ is the entropy of the policy, m is the number of discrete actions, and $\ln(m)$ is the maximum possible entropy of the categorical action distribution.

Benefits:

- **Normalized:** Entropy term lives in $[0, 1]$ regardless of action-space size
- **Hinge effect:** No penalty when $H/\ln m \geq \Gamma$, linear penalty otherwise
- **Guaranteed exploration:** Maintains minimum exploration level without temperature schedules

Policy Learning (3/3): Efficient Inputs

The network must take some representation x_t of state. Options:

Input x_t	**Pros**	**Cons**
o_t	Stable—no drift	Heavy: requires conv-nets
z_t	Light: small vectors	Drift between encodings
$[z_t, h_t]$	Balance of stability/efficiency	Complex implementation

Policy Input Choices:

- **At inference (real env.):** $x_t = z_t$ with last 4 frames fed into VAE
- **At training (imagined rollouts):** $x_t = \hat{z}_t$ (model's predicted latents)
- **Efficiency gains:** Fast inference (single VAE encode per step) and fast training (small MLPs on $\hat{z}$)

Summary:

- Actor-critic leverages imagined futures with model's own $\hat{\gamma}_t$
- Thresholded entropy ensures fixed minimum exploration rate
- Lean inputs (compact discrete latent z_t) keep both training and inference efficient

Training (1/2): Overall Loop

Three-phase training loop:

1. **Collect real experience**
   - Run policy π_θ in actual environment
   - Record transitions $(o_t,\, a_t,\, r_t,\, d_t)$ where d_t is done-flag
2. **Update World Model**
   - Sample N short sequences of length ℓ from dataset $\mathcal{D}$
   - Compute observation loss $\mathcal{L}_\phi^{\text{Obs.}}$ and dynamics loss $\mathcal{L}_\psi^{\text{Dyn.}}$
   - Update world-model parameters (ϕ, ψ)
3. **Update Policy via Imagination**
   - From same $N \times \ell$ observations, pick M starting points
   - "Roll out" each latent for H steps inside world model:

$$\begin{aligned} \hat{z}_{t+1} &\sim p_\psi(\hat{z}_{t+1} \,|\, h_t) \\ \hat{r}_t &\sim p_\psi(\hat{r}_t \,|\, h_t) \\ \hat{\gamma}_t &\sim p_\psi(\hat{\gamma}_t \,|\, h_t) \\ a_t &\sim \pi_\theta(a_t \,|\, \hat{z}_t) \end{aligned}$$

   - Compute actor-critic losses on imagined trajectories
   - Update policy parameters θ

Training (2/2): Balanced Dataset Sampling

Problem: Uniform sampling gives equal weight to early (poor) and recent (better) experience

Solution: Softmax over "visitation counts"

- Track visitation count v_i for each timestep i used as sequence start
- Convert counts to sampling probabilities via softmax:

$$p_i = \frac{\exp\left(-v_i/\tau\right)}{\sum_{j=1}^{T} \exp\left(-v_j/\tau\right)} \tag{11}$$

Temperature τ controls recency bias:

- $\tau \to \infty$: Approaches uniform sampling
- Lower τ: Penalizes frequently-sampled steps, favors newer data

Benefits:

- Emphasizes fresh experience when data is scarce
- Prevents overfitting to early, untrained rollouts
- Can increase τ as replay buffer grows for better stability

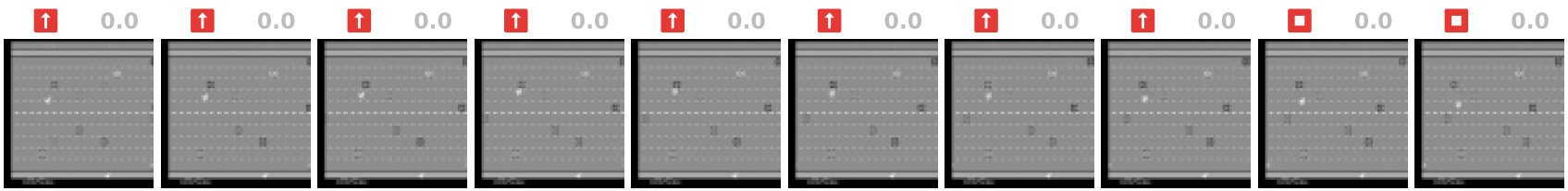
Experiment: Atari 100k

			Model-free				Imagination	
Game	Random	Human	DER	CURL	DrQ(ε)	SPR	SimPLe	TWM (ours)
Alien	227.8	7127.7	802.3	711.0	**865.2**	841.9	616.9	674.6
Amidar	5.8	1719.5	125.9	113.7	137.8	**179.7**	74.3	121.8
Assault	222.4	742.0	561.5	500.9	579.6	565.6	527.2	**682.6**
Asterix	210.0	8503.3	535.4	567.2	763.6	962.5	**1128.3**	1116.6
BankHeist	14.2	753.1	185.5	65.3	232.9	345.4	34.2	**466.7**
BattleZone	2360.0	37187.5	8977.0	8997.8	10165.3	**14834.1**	4031.2	5068.0
Boxing	0.1	12.1	-0.3	0.9	9.0	35.7	7.8	**77.5**
Breakout	1.7	30.5	9.2	2.6	19.8	19.6	16.4	**20.0**
ChopperCommand	811.0	7387.8	925.9	783.5	844.6	946.3	979.4	**1697.4**
CrazyClimber	10780.5	35829.4	34508.6	9154.4	21539.0	36700.5	62583.6	**71820.4**
DemonAttack	152.1	1971.0	627.6	646.5	**1321.5**	517.6	208.1	350.2
Freeway	0.0	29.6	20.9	**28.3**	20.3	19.3	16.7	24.3
Frostbite	65.2	4334.7	871.0	1226.5	1014.2	1170.7	236.9	**1475.6**
Gopher	257.6	2412.5	467.0	400.9	621.6	660.6	596.8	**1674.8**
Hero	1027.0	30826.4	6226.0	4987.7	4167.9	5858.6	2656.6	**7254.0**
Jamesbond	29.0	302.8	275.7	331.0	349.1	**366.5**	100.5	362.4
Kangaroo	52.0	3035.0	581.7	740.2	1088.4	**3617.4**	51.2	1240.0
Krull	1598.0	2665.5	3256.9	3049.2	4402.1	3681.6	2204.8	**6349.2**
KungFuMaster	258.5	22736.3	6580.1	8155.6	11467.4	14783.2	14862.5	**24554.6**
MsPacman	307.3	6951.6	1187.4	1064.0	1218.1	1318.4	1480.0	**1588.4**
Pong	-20.7	14.6	-9.7	-18.5	-9.1	-5.4	12.8	**18.8**
PrivateEye	24.9	69571.3	72.8	81.9	3.5	86.0	35.0	**86.6**
Qbert	163.9	13455.0	1773.5	727.0	1810.7	866.3	1288.8	**3330.8**
RoadRunner	11.5	7845.0	11843.4	5006.1	11211.4	**12213.1**	5640.6	9109.0
Seaquest	68.4	42054.7	304.6	315.2	352.3	558.1	683.3	**774.4**
UpNDown	533.4	11693.2	3075.0	2646.4	4324.5	10859.2	3350.3	**15981.7**
Normalized Mean	0.000	1.000	0.350	0.261	0.465	0.616	0.332	**0.956**
Normalized Median	0.000	1.000	0.189	0.092	0.313	0.396	0.134	**0.505**

Table: Mean scores on Atari 100k benchmark with human normalized mean and median. Results averaged over 5 runs per game, 100 episodes per run. Bold indicates best scores.

Analysis (1/2): Imagined Trajectories

(a) Boxing. The player (white) presses *fire*, hits the opponent, and gets a reward.



(b) Freeway. The player moves up and bumps into a car. The world model correctly pushes the player down, although *up* is still pressed. The movement of the cars is modeled correctly.

Figure: Trajectories imagined by TWM. Above each frame they show the performed action and the produced reward.

Analysis (2/2): Attention Maps

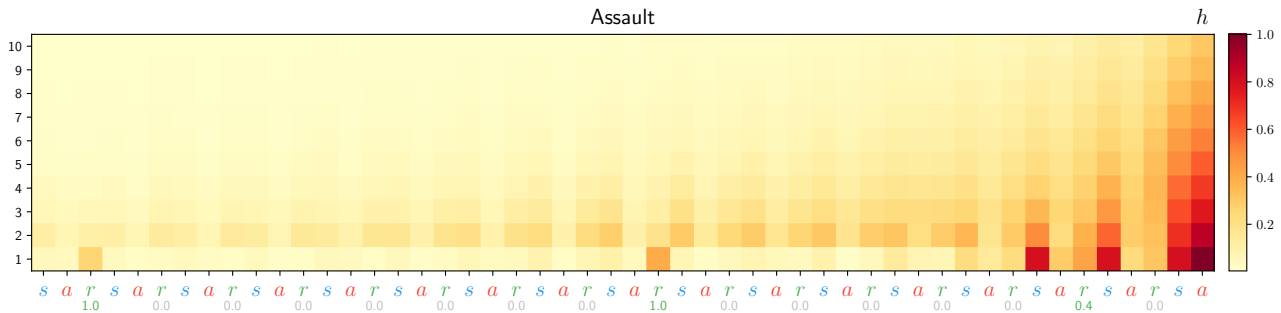


Figure: Attention map of the learned transformer for the current hidden state h, computed on an imagined trajectory for the game Assault. The x-axis corresponds to the input sequence with the three modalities (states, actions, rewards), where the two rightmost columns are the current state and action. The y-axis corresponds to the layer of the transformer.

Ablation Studies (1/2): Uniform Sampling

- We evaluate the effectiveness of our balanced sampling procedure by comparing it with uniform sampling ($\tau = \infty$ in Eqn. (11))
- Balanced sampling significantly improves performance across tested games
- Dynamics loss (Eqn. (8)) is lower with balanced sampling at training completion
- Possible explanation: uniform sampling causes the world model to overfit on early training data, leading to poor performance in later training stages

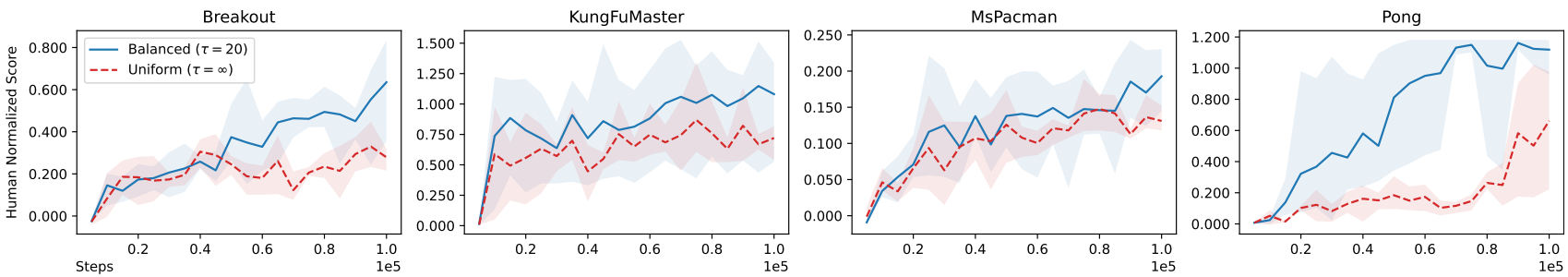


Figure: Comparison of balanced vs. uniform sampling across games. Human normalized scores during training show balanced sampling yields better performance, highlighting its importance.

Ablation Studies (2/2): No Rewards

- Predicted rewards are fed back into the transformer
- They evaluate the impact of this design choice by comparing with a variant that doesn't use rewards as input
- Figure shows this feedback mechanism significantly increases performance in several games
- In some games, performance remains equivalent, likely because the world model can make correct predictions based solely on latent states and actions

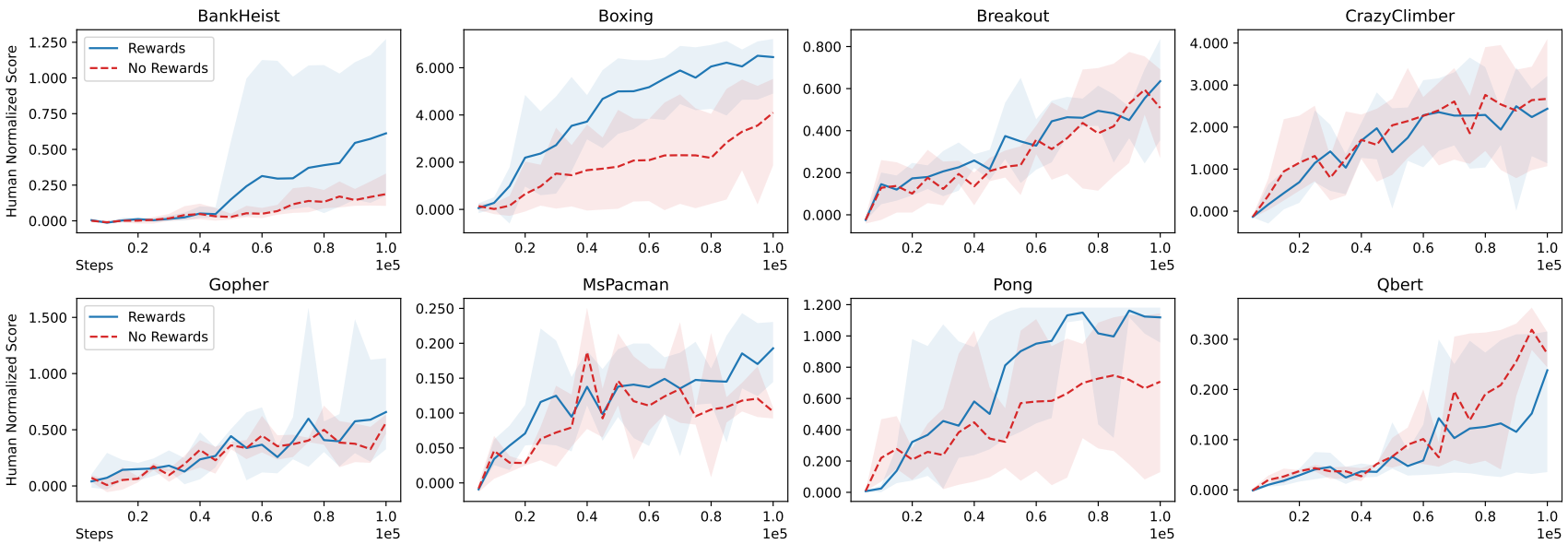


Figure: Effect of removing rewards from the input. Human normalized scores during training show conditioning on rewards significantly improves performance in some games, while others remain unaffected.