LEARNING TRANSFORMER-BASED WORLD MODELS WITH CONTRASTIVE PREDICTIVE CODING

Maxime Burchi, Radu Timofte
Computer Vision Lab, CAIDAS & IFI, University of Würzburg, Germany
maxime.burchi@uni-wuerzburg.de

ABSTRACT

The DreamerV3 algorithm recently obtained remarkable performance across diverse environment domains by learning an accurate world model based on Recurrent Neural Networks (RNNs). Following the success of model-based reinforcement learning algorithms and the rapid adoption of the Transformer architecture for its superior training efficiency and favorable scaling properties, recent works such as STORM have proposed replacing RNN-based world models with Transformer-based world models using masked self-attention. However, despite the improved training efficiency of these methods, their impact on performance remains limited compared to the Dreamer algorithm, struggling to learn competitive Transformer-based world models. In this work, we show that the next state prediction objective adopted in previous approaches is insufficient to fully exploit the representation capabilities of Transformers. We propose to extend world model predictions to longer time horizons by introducing TWISTER (Transformer-based World model wIth contraSTivE Representations), a world model using action-conditioned Contrastive Predictive Coding to learn high-level temporal feature representations and improve the agent performance. TWISTER achieves a human-normalized mean score of 162% on the Atari 100k benchmark, setting a new record among state-of-the-art methods that do not employ look-ahead search. We release our code at https://github.com/burchim/TWISTER.

1 INTRODUCTION

Deep Reinforcement Learning (RL) algorithms have achieved notable breakthroughs in recent years. The growing computational capabilities of hardware systems have allowed researchers to make significant progress, training powerful agents from high-dimensional observations like images (Mnih et al., 2013) or videos (Hafner et al., 2020) using deep neural networks (LeCun et al., 2015) as function approximations. Following the rapid adoption of Convolutional Neural Networks (CNNs) (LeCun et al., 1989) in the field of Computer Vision for their efficient pattern recognition ability, neural networks were applied to visual reinforcement learning problems and achieved human to superhuman performance in challenging and visually complex domains like Atari games (Mnih et al., 2015; Hessel et al., 2018), the game of Go (Silver et al., 2018; Schrittwieser et al., 2020), StarCraft II (Vinyals et al., 2019) and more recently, Minecraft (Baker et al., 2022; Hafner et al., 2023).

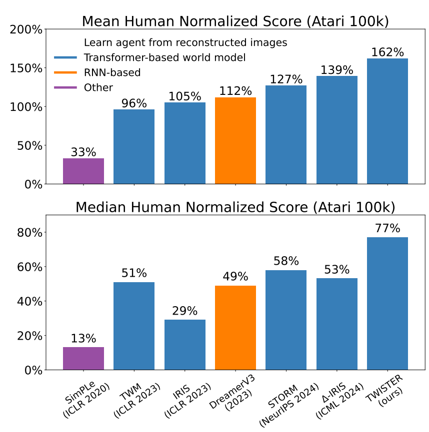


Figure 1: Human-normalized mean and median scores of recently published model-based methods on the Atari 100k benchmark. TWISTER outperforms other model-based approaches. TWM, IRIS, STORM and Δ-IRIS employ a Transformer-based world model while DreamerV3 uses a RNN-based model.

Following the success of neural networks in solving reinforcement learning problems, model-based approaches learning world models using gradient backpropagation were proposed to reduce the amount of necessary interaction with the environment to achieve strong results (Kaiser et al., 2020; Hafner et al., 2020; 2021; 2023; Schrittwieser et al., 2020). World models (Sutton, 1991; Ha & Schmidhuber, 2018) summarize an agent's experience into a predictive model that can be used in place of the real environment to learn complex behaviors. Having access to a model of the environment enables the agent to simulate multiple plausible trajectories in parallel, improving generalization, sample efficiency and decision-making via planning.

Design choices for the world model have tended toward Recurrent Neural Networks (RNNs) (Hafner et al., 2019) for their ability to model temporal relationships effectively. Following the success of the Dreamer algorithm (Hafner et al., 2020) and the rapid adoption of the Transformer architecture (Vaswani et al., 2017) for its superior training efficiency and favorable scaling properties compared to RNNs, research works have proposed replacing the one-layer recurrent-based world model of Dreamer with a Transformer-based world model using masked self-attention (Chen et al., 2022; Micheli et al., 2023; Robine et al., 2023). However, despite the improved training efficiency of these methods, their impact on performance remains limited compared to the Dreamer algorithm, struggling to learn competitive Transformer-based world models. Zhang et al. (2024) suggested that these findings may be attributed to the subtle differences between consecutive video frames. The task of predicting the next video frame in latent space may not require a complex model in contrast to other fields like Neural Language Modeling (Kaplan et al., 2020) where a deep understanding of the past context is essential to accurately predict the next tokens. As shown in Figure 2, the cosine similarity between adjacent latent states of the world model is very high, making it relatively straightforward for the world model to predict the next state compared to more distant states. These findings motivate our work to complexify the world model objective by extending predictions to longer time horizons in order to learn higher quality feature representations and improve the agent performance.

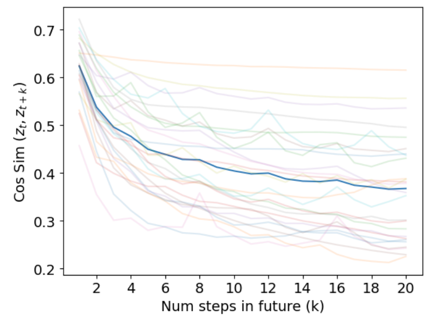


Figure 2: Cosine Similarities between TWISTER latent state z_t and future states z_{t+k} aggregated over all 26 games of the Atari 100k benchmark. We show average similarities over 5 seeds.

In this work, we show that the next latent state prediction objective adopted in previous approaches is insufficient to fully exploit the representation capabilities of Transformers. We introduce TWISTER, a Transformer model-based reinforcement learning algorithm using action-conditioned Contrastive Predictive Coding (AC-CPC) to learn high-level temporal feature representations and improve the agent performance. CPC (Oord et al., 2018) was initially applied to speech, image, and text domains as a pretraining pretext task. It also showed promising results on DeepMind Lab tasks (Beattie et al., 2016) being used as an auxiliary loss for the A3C agent (Mnih et al., 2016). Motivated by these findings, we apply the CPC objective to model-based reinforcement learning by conditioning CPC predictions on the sequence of future actions. This approach enables the world model to accurately predict the feature representations of future time steps using contrastive learning. As shown in Figure 1, TWISTER sets a new record on the commonly used Atari 100k benchmark (Kaiser et al., 2020) among state-of-the-art methods that do not employ look-ahead search, achieving a human-normalized mean and median score of 162% and 77%, respectively.

2 Related Works

2.1 Model-based Reinforcement Learning

Model-based reinforcement learning approaches use a model of the environment to simulate agent trajectories, improving generalization, sample efficiency, and decision-making via planning. Following the success of deep neural networks for learning function approximations, researchers proposed to learn world models using gradient backpropagation. While initial works concentrated on simple environments like proprioceptive tasks (Silver et al., 2017; Henaff et al., 2017; Wang et al., 2019; Wang & Ba, 2020) using low-dimensional observations, more recent works focus on learning world models from high-dimensional observations like images (Kaiser et al., 2020; Hafner et al., 2019).

One of the earliest model-based algorithms applied to image data is SimPLe (Kaiser et al., 2020), which proposed to learn a world model for Atari games in pixel space using a convolutional autoencoder. The world model learns to predict the next frame and environment reward given previous observation frames and selected action. It is then used to train a Proximal Policy Optimization (PPO) agent (Schulman et al., 2017) from reconstructed images and predicted rewards. Concurrently, PlaNet (Hafner et al., 2019) introduced a Recurrent State-Space Model (RSSM) using a Gated Recurrent Unit (GRU) (Cho et al., 2014) to learn a world model in latent space, planning using model predictive control. PlaNet learns a convolutional variational autoencoder (VAE) (Kingma & Welling, 2013) with a pixel reconstruction loss to encode observation into stochastic state representations. The RSSM learns to predict the next stochastic states and environment rewards given previous stochastic and deterministic recurrent states. Following the success of PlaNet on DeepMind Visual Control tasks (Tassa et al., 2018), Dreamer (Hafner et al., 2020) improved the algorithm by learning an actor and a value network from the world model representations. DreamerV2 (Hafner et al., 2021) applied the algorithm to Atari games, utilizing categorical latent states with straight-through gradients (Bengio et al., 2013) in the world model to improve performance, instead of Gaussian latents with reparameterized gradients (Kingma & Welling, 2013). DreamerV3 (Hafner et al., 2023) mastered diverse domains using the same hyper-parameters with a set of architectural changes to stabilize learning across tasks. The agent uses symlog predictions for the reward and value function to address the scale variance across domains. The networks also employ layer normalization (Ba et al., 2016) to improve robustness and performance while scaling to larger model sizes. It stabilizes policy learning by normalizing the returns and value function using an Exponential Moving Average (EMA) of the returns percentiles. With these modifications, DreamerV3 outperformed specialized model-free and model-based algorithms in a wide range of benchmarks.

In parallel to the Dreamer line of work, Schrittwieser et al. (2020) proposed MuZero, a model-based algorithm combining Monte-Carlo Tree Search (MCTS) (Coulom, 2006) with a powerful world model to achieve superhuman performance in precision planning tasks such as Chess, Shogi and Go. The model is learned by being unrolled recurrently for K steps and predicting environment quantities relevant to planning. The MCTS algorithm uses the learned model to simulate environment trajectories and output an action visit distribution over the root node. This potentially better policy compared to the neural network one is used to train the policy network. More recently, Ye et al. (2021) proposed EfficientZero, a sample efficient version of the MuZero algorithm using self-supervised learning to learn a temporally consistent environment model and achieve strong performance on Atari games.

2.2 Transformer-based World Models

Recent works have proposed replacing RNN-based world models by Transformer-based architectures using self-attention to process past context. TransDreamer (Chen et al., 2022) replaced DreamerV3's RSSM by a Transformer State-Space Model (TSSM) using masked self-attention to imagine future trajectories. The agent was evaluated on Hidden Order Discovery tasks requiring long-term memory and reasoning. They also experimented on a few Visual DeepMind Control (Tassa et al., 2018) and Atari (Bellemare et al., 2013) tasks, showing comparable performance to DreamerV2. TWM (Robine et al., 2023) (Transformer-based World Model) proposed a similar approach, encoding states, actions and rewards as distinct successive input tokens for the autoregressive Transformer. The decoder also reconstructed input images without the world model hidden states, discarding past context temporal information for image reconstruction. More recently, STORM (Zhang et al., 2024) (Stochastic Transformer-based wORld Model) achieved results comparable to DreamerV3 with better training efficiency on the Atari 100k benchmark. STORM proposed to fuse state and action into

Table 1: Comparison between TWISTER and other recent model-based approaches learning a world model in latent space. *Tokens* refers to tokens used by the autoregressive world model. *Latent* (z_t) is image representation while *hidden* (h_t) is world model hidden state carrying historical information.

Attributes	TWM	IRIS	DreamerV3	STORM	Δ-IRIS	TWISTER (ours)
World Model	Transformer	Transformer	GRU	Transformer	Transformer	Transformer
Prediction Horizon	Next state	Next state	Next state	Next state	Next state	K = 10 steps
Tokens	Latent, action, reward	Latent (4×4)	Latent	Latent	Latent (2×2)	Latent
Latent Representation	Categorical-VAE	VQ-VAE	Categorical-VAE	Categorical-VAE	VQ-VAE	Categorical-VAE
Decoder Inputs	Latent	Latent	Latent, hidden	Latent	Latent, action, image	Latent
Agent State (s_t)	Latent	Image	Latent, hidden	Latent, hidden	Image	Latent, hidden

a single token for the transformer network compared to TWM which uses distinct tokens. This led to better training efficiency with state-of-the-art performance.

Another line of work focused on designing Transformer-based world model to train agents from reconstructed trajectories in pixel space. Analogously to SimPLe, the agent's policy and value functions are trained from image reconstruction instead of world model hidden state representations. This requires learning auxiliary encoder networks for the policy and value functions. Contrary to Dreamer-inspired works that learn agents from world model representations, these approaches also require accurate image reconstruction to train agents effectively. IRIS (Micheli et al., 2023) first proposed a world model composed of a VQ-VAE (Van Den Oord et al., 2017) to convert input images into discrete tokens and an autoregressive transformer to predict future tokens. IRIS was evaluated on the Atari 100k benchmark (Kaiser et al., 2020) showing promising performance in a low data regime. More recently, Micheli et al. (2024) proposed Δ-IRIS, encoding stochastic deltas between time steps using previous action and image as conditions for the encoder and decoder. This increased VQ-VAE compression ratio and image reconstruction capabilities, achieving state-of-the-art performance on the Crafter (Hafner, 2021) benchmark and better results on Atari 100k.

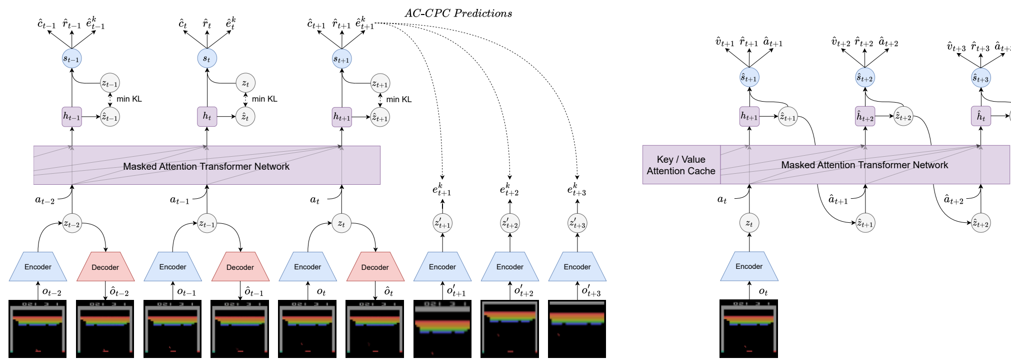
Table 1 compares the architectural details of recent model-based approaches learning a world model in latent space with our proposed method. Following preceding Transformer-based approaches, we reconstruct image observation from the encoder stochastic state z_t instead of s_t, which prevents the world model from using temporal information to facilitate reconstruction. The Transformer network uses relative positional encodings (Dai et al., 2019), which simplifies the use of the world model during imagination and evaluation. Absolute positional encodings require the Transformer network to reprocess past latent states with adjusted positional encodings when the current position gets larger than the ones seen during training. We also use the agent state s_t as input for predictor networks during the world model training phase to make actor-critic learning more straightforward.

2.3 Contrastive Predictive Coding

Contrastive Predictive Coding (CPC) was introduced by Oord et al. (2018) as a representation learning method based on contrastive learning for autoregressive models. CPC encodes a temporal signal into hidden representations and trains an autoregressive model to maximize the mutual information between the autoregressive model output features and future encoded representations using an InfoNCE loss based on Noise-Contrastive Estimation (Gutmann & Hyvärinen, 2010). CPC was able to learn useful representations achieving strong performance on four distinct domains: speech phoneme classification, image classification, text classification tasks, and reinforcement learning with DeepMind Lab 3D environments (Beattie et al., 2016). While CPC was applied to speech, image, and text domains as a pretraining pretext task, it showed promising results on DeepMind Lab tasks being used as an auxiliary loss for the A3C (Mnih et al., 2016) agent. In this work, we propose to apply CPC to model-based reinforcement learning. We introduce action-conditioned CPC (AC-CPC) that conditions CPC predictions on the sequence of future actions to help the world model to make more accurate predictions and learn higher quality representations. We describe our use of action-conditioned CPC in more detail in section 3.1.

3 Method

We introduce TWISTER, a Transformer model-based reinforcement learning algorithm using action-conditioned Contrastive Predictive Coding to learn high-level feature representations and improve the agent performance. TWISTER comprises three main neural networks: a world model, an actor network and a critic network. The world model learns to transform image observations into discrete stochastic states and simulate the environment to generate imaginary trajectories. The actor and critic networks are trained in latent space with imaginary trajectories generated from the world model to select actions maximizing the expected sum of future rewards. The three networks are trained concurrently using a replay buffer sampling sequences of past experiences collected during training. This section describes the architecture and optimization process of our proposed Transformer-based world model with contrastive representations. Analogously to previous approaches, we also detail the learning process of the critic and actor networks taking place in latent space. Figure 3 shows an overview of our Transformer-based world model trained with AC-CPC. It also illustrates the imagination process undertaken during the agent behavior learning phase.



(a) World Model Learning

(b) Agent Behavior Learning

Figure 3: Transformer-based world model with contrastive representations. The world model learns temporal feature representations by maximizing the mutual information between model states s_t and future stochastic states $z'_{t:t+K}$ obtained from augmented views of image observations. The encoder network converts image observations into stochastic states z_t, from which a decoder network learns to reconstruct images while the masked attention Transformer network predicts next episode continuations, rewards and stochastic states conditioned on selected actions.

3.1 World Model Learning

Consistent with prior works (Hafner et al., 2023; Robine et al., 2023; Zhang et al., 2024), we learn a world model in latent space by encoding input image observations o_t into hidden representations using a convolutional VAE with categorical latents. The hidden representations are linearly projected to categorical distribution logits comprising 32 categories, each with 32 classes, from which discrete stochastic states z_t are sampled. The world model is implemented as a Transformer State-Space Model (TSSM) (Chen et al., 2022) using masked self-attention to predict next stochastic states $\hat{z}_{t+1}$ given previous states $z_{1:t}$ and actions $a_{1:t}$. The Transformer network outputs hidden states h_t that are concatenated with stochastic states z_t to form the model states $s_t = \{h_t, z_t\}$. The world model predicts environment reward $\hat{r}_t$, episode continuation $\hat{c}_t$ and AC-CPC features $\hat{e}_t^k$ using simple Multi Layer Perceptron (MLP) networks. The trainable world model components are the following:

$$
\begin{array}{ll}
\text{TSSM} \left\{ \begin{array}{l} \text{Encoder Network:} \\ \text{Transformer Network:} \\ \text{Dynamics Predictor:} \end{array} \right. & \begin{array}{l} z_t \sim q_\phi(z_t \mid o_t) \\ h_t = f_\phi(z_{1:t-1}, a_{1:t-1}) \\ \hat{z}_t \sim p_\phi(\hat{z}_t \mid h_t) \end{array} \\
\quad\quad\quad \text{Decoder Network:} & \hat{o}_t \sim p_\phi(\hat{o}_t \mid z_t) \\
\quad\quad\quad \text{Reward Predictor:} & \hat{r}_t \sim p_\phi(\hat{r}_t \mid s_t) \\
\quad\quad\quad \text{Continue Predictor:} & \hat{c}_t \sim p_\phi(\hat{c}_t \mid s_t) \\
\text{AC-CPC} \left\{ \begin{array}{l} \text{Representation Network:} \\ \text{AC-CPC Predictor:} \end{array} \right. & \begin{array}{l} e_t^k = q_\phi^k(z'_{t+k}) \\ \hat{e}_t^k = p_\phi^k(s_t, a_{t:t+k}) \end{array}
\end{array}
\tag{1}
$$

Transformer State-Space Model We train an autoregressive Transformer network using masked self-attention with relative positional encodings (Dai et al., 2019). During both training, exploration and evaluation, the hidden state sequence computed for the previous segment or state is cached to be reused as an extended context when the model processes the next state. This encoding and caching mechanism allows the world model to imagine future trajectories from any state, eliminating the need to reprocess latent states with adjusted positional encodings.

World model losses Given an input batch containing B sequences of T image observations $o_{1:T}$, actions $a_{1:T}$, rewards $r_{1:T}$, and episode continuation flags $c_{1:T}$, the world model parameters (ϕ) are optimized to minimize the following loss function:

$$
L(\phi) = \frac{1}{BT} \sum_{b=1}^{B} \sum_{t=1}^{T} \Big[L_{rew}(\phi) + L_{con}(\phi) + L_{rec}(\phi) + L_{dyn}(\phi) + L_{cpc}(\phi) \Big] \tag{2}
$$

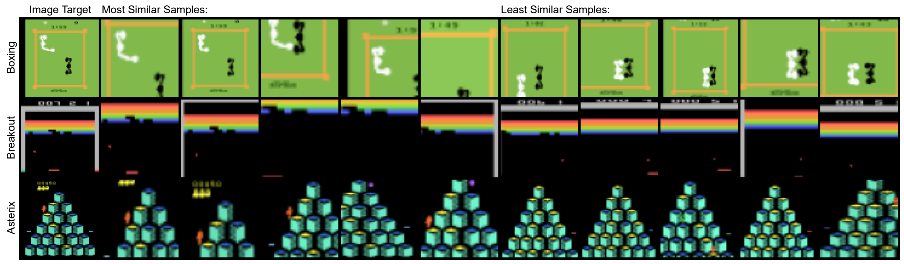


Figure 4: AC-CPC predictions made by the world model. We show the target positive sample without augmentation and the predicted most/least similar samples among the batch of augmented image views. We observe that TWISTER learns to identify most/least similar samples to the future target state using observation details such as the ball position, game score or agent movements. AC-CPC necessitates the agent to focus on observation details to accurately predict future samples, thereby preventing common failure cases where small objects are ignored by the reconstruction loss.

L_{rew} and L_{con} train the world model to predict environment rewards and episode continuation flags, which are used to compute the returns of imagined trajectories during the behavior learning phase. We adopt the symlog cross-entropy loss from DreamerV3 (Hafner et al., 2023), which scales and transforms rewards into twohot encoded targets to ensure robust learning across games with different reward magnitudes. The reconstruction loss L_{rec} trains the categorical VAE to learn stochastic representations z_t for the world model by reconstructing input visual observations o_t:

$$L_{rew}(\phi) = \mathrm{SymlogCrossEnt}(\hat{r}_t, r_t) \tag{3a}$$

$$L_{con}(\phi) = \mathrm{BinaryCrossEnt}(\hat{c}_t, c_t) \tag{3b}$$

$$L_{rec}(\phi) = ||\hat{o}_t - o_t||_2^2 \tag{3c}$$

The world model dynamics loss L_{dyn} trains the dynamics predictor network to predict the next stochastic states representations from transformer hidden states by minimizing the Kullback–Leibler (KL) divergence between the predictor output distribution $p_\phi(\hat{z}_t \mid h_t)$ and the next encoder representation $q_\phi(z_t \mid o_t)$. We also add a regularization term to avoid spikes in the KL loss and stabilize learning by training the encoder representations to become more predictable. Both loss terms use the stop gradient operator $sg(\cdot)$ to prevent the gradients of targets from being backpropagated and are scaled with loss weights $\beta_{dyn} = 0.5$ and $\beta_{reg} = 0.1$, respectively:

$$\begin{aligned} L_{dyn}(\phi) = \; & \beta_{dyn} \; \max\big(1, \mathrm{KL}\big[\, sg(q_\phi(z_t \mid o_t)) \; || \quad p_\phi(\hat{z}_t \mid h_t) \;\big]\big) \\ + \; & \beta_{reg} \; \max\big(1, \mathrm{KL}\big[\quad q_\phi(z_t \mid o_t) \; \; || \, sg(p_\phi(\hat{z}_t \mid h_t))\big]\big) \end{aligned} \tag{4}$$

The Transformer network learns feature representations using action-conditioned Contrastive Predictive Coding. The representations are learned by maximizing the mutual information between model states s_t and future stochastic states $z'_{t:t+K}$ obtained from augmented views of image observations. We adopt a simple strategy to generate negative samples: Given the sequence batch of augmented stochastic states Z' containing one positive sample, we treat the other $B \times T - 1$ samples as negatives. The world model learns to distinguish positive samples from negatives using InfoNCE:

$$L_{cpc}(\phi) = -\frac{1}{K} \sum_{k=0}^{K-1} \log \frac{exp(sim(z'_{t+k}, s_t))}{\sum_{z'_j \in Z'} exp(sim(z'_j, s_t))} \tag{5}$$

The world model learns to predict $K = 10$ future stochastic states among the batch of augmented samples. We compute similarities as dot products: $sim(z'_j, s_t) = q^k_\phi(z'_j)^T p^k_\phi(s_t, a_{t:t+k})$, learning two MLP networks q^k_ϕ and p^k_ϕ for each step k. Contrary to the original CPC paper, which experiments with continuous feature states, we use discrete latent states for the world model. This requires learning a representation network q^k_ϕ to project discretized stochastic states z'_j to contrastive feature representations e^k_t. The AC-CPC predictor p^k_ϕ uses the concatenated sequence of future actions $a_{t:t+k}$ as condition to reduce uncertainty and learn quality representations.

3.2 Agent Behavior Learning

The agent critic and actor networks are trained with imaginary trajectories generated from the world model. In order to compare TWISTER with previous approaches that train agents using world model representations, we adopt the agent behavior learning settings from DreamerV3 (Hafner et al., 2023). Learning takes place entirely in latent space, which allows the agent to process large batch sizes and increase generalization. We flatten the model states of the sampled sequences along the batch and time dimensions to generate $B^{img} = B \times T$ sample trajectories using the world model. The self-attention keys and values features computed during the world model training phase are cached to be reused during the agent behavior learning phase and preserve past context. As shown in Figure 3b, the world model imagines $H = 15$ steps into the future using the Transformer network and the dynamics network head, selecting actions by sampling from the actor network categorical distribution. Analogously to world model predictor networks, the actor and critic networks are designed as simple MLPs with parameter vectors (θ) and (ψ), respectively.

$$\begin{array}{ll} \text{Actor Network:} & a_t \sim \pi_\theta(a_t|s_t) \\ \text{Critic Network:} & v_t \sim V_\psi(v_t|s_t) \end{array} \tag{6}$$

Critic Learning Following DreamerV3, the critic network learns to minimize the symlog cross-entropy loss with discretized λ-returns obtained from imagined trajectories with rewards and episode continuation flags predicted by the world model:

$$R_t^\lambda = \hat{r}_{t+1} + \gamma\hat{c}_{t+1}\Big((1-\lambda)V_\psi(s_{t+1}) + \lambda R_{t+1}^\lambda\Big) \qquad R_{H+1}^\lambda = V_\psi(s_{H+1}) \tag{7}$$

The critic does not use a target network but relies on its own predictions for estimating rewards beyond the prediction horizon. This requires stabilizing the critic by adding a regularizing term toward the outputs of its own EMA network $V_{\psi'}$. Equation 8 defines the critic network loss:

$$L_{critic}(\psi) = \frac{1}{BH}\sum_{b=1}^{B}\sum_{t=1}^{H}\Big[\underbrace{\text{SymlogCrossEnt}\big(v_t, R_t^\lambda\big)}_{\text{discrete returns regression}} + \underbrace{\text{SymlogCrossEnt}\big(v_t, V_{\psi'}(s_t)\big)}_{\text{critic EMA regularizer}}\Big] \tag{8}$$

Actor Learning The actor network learns to select actions that maximize the predicted returns using Reinforce (Williams, 1992) while maximizing the policy entropy to ensure sufficient exploration during both data collection and imagination. The actor network loss is defined as follows:

$$L_{actor}(\theta) = \frac{1}{BH}\sum_{b=1}^{B}\sum_{t=1}^{H}\Big[\underbrace{-\,sg(A_t^\lambda)\log\pi_\theta(a_t \mid s_t)}_{\text{reinforce}}\underbrace{-\,\eta\text{H}\big(\pi_\theta(a_t \mid s_t)\big)}_{\text{entropy regularizer}}\Big] \tag{9}$$

Where $A_t^\lambda = \big(\hat{R}_t^\lambda - V_\psi(s_t)\big)/\max(1, S)$ defines advantages computed using normalized returns. The returns are scaled using exponentially moving average statistics of their 5^{th} and 95^{th} batch percentiles to ensure stable learning across all Atari games:

$$S = \text{EMA}(\text{Per}(R_t^\lambda, 95) - \text{Per}(R_t^\lambda, 5), momentum = 0.99) \tag{10}$$

4 Experiments

In this section, we describe our experiments on the commonly used Atari 100k benchmark. We compare TWISTER with SimPLe, DreamerV3 and recent Transformer model-based approaches in Table 2. We also perform several ablation studies on the principal components of TWISTER.

4.1 Atari 100k Benchmark

The Atari 100k benchmark was proposed in Kaiser et al. (2020) to evaluate reinforcement learning agents on Atari games in low data regime. The benchmark includes 26 Atari games with a budget of 400k environment frames, amounting to 100k interactions between the agent and the environment using the default action repeat setting. This amount of environment steps corresponds to about two hours (1.85 hours) of real-time play, representing a similar amount of time that a human player would need to achieve reasonably good performance. The current state-of-the-art is held by EfficientZero V2 (Wang et al., 2024), which uses Monte-Carlo Tree Search to select the best action at every time step. Another recent notable work is BBF (Schwarzer et al., 2023), a model-free agent using learning techniques that are orthogonal to our work such as periodic network resets and hyper-parameters annealing to improve performance. In this work, to ensure fair comparison and demonstrate the effectiveness of AC-CPC for learning world models, we compare our method with model-based approaches that do not utilize look-ahead search techniques. Combining these additional components with TWISTER would nevertheless be an interesting research direction for future works.

4.2 Results

Table 2: Agent scores and human-normalized metrics on the 26 games of the Atari 100k benchmark. We show average scores over 5 seeds. Bold numbers indicate best performing method for each game.

Game	Random	Human	SimPLe	TWM	IRIS	DreamerV3	STORM	Δ-IRIS	TWISTER (ours)
Alien	228	7128	617	675	420	959	**984**	391	970
Amidar	6	1720	74	122	143	139	**205**	64	184
Assault	222	742	527	683	**1524**	706	801	1123	721
Asterix	210	8503	1128	1116	854	932	1028	**2492**	1306
Bank Heist	14	753	34	467	53	649	641	**1148**	942
Battle Zone	2360	37188	4031	5068	13074	12250	**13540**	11825	9920
Boxing	0	12	8	78	70	78	80	70	**88**
Breakout	2	30	16	20	84	31	16	**302**	35
Chopper Command	811	7388	979	1697	1565	420	**1888**	1183	910
Crazy Climber	10780	35829	62584	71820	59324	**97190**	66776	57854	81880
Demon Attack	152	1971	208	350	**2034**	303	165	533	289
Freeway	0	30	17	24	31	0	**34**	31	32
Frostbite	65	4335	237	**1476**	259	909	1316	279	305
Gopher	258	2412	597	1675	2236	3730	8240	6445	**22234**
Hero	1027	30826	2657	7254	7037	**11161**	11044	7049	8773
James Bond	29	303	100	362	463	445	509	309	**573**
Kangaroo	52	3035	51	1240	838	4098	4208	2269	**6016**
Krull	1598	2666	2205	6349	6616	7782	8413	5978	**8839**
Kung Fu Master	258	22736	14862	24555	21760	21420	**26182**	21534	23442
Ms Pacman	307	6952	1480	1588	999	1327	**2673**	1067	2206
Pong	–21	15	13	19	15	18	11	**20**	**20**
Private Eye	25	69571	35	87	100	882	**7781**	103	1608
Qbert	164	13455	1289	3331	746	3405	**4522**	1444	3197
Road Runner	12	7845	5641	9107	9615	15565	17564	10414	**17832**
Seaquest	68	42055	683	**774**	661	618	525	827	532
Up N Down	533	11693	3350	**15982**	3546	7600	7985	4072	7068
# Superhuman	0	N/A	1	8	10	9	10	11	**12**
Normed Mean (%)	0	100	33	96	105	112	127	139	**162**
Normed Median (%)	0	100	13	51	29	49	58	53	**77**

Table 2 compares TWISTER with SimPLe (Kaiser et al., 2020), DreamerV3 (Hafner et al., 2023) and recent Transformer model-based approaches (Robine et al., 2023; Micheli et al., 2023; Zhang et al., 2024; Micheli et al., 2024) on the Atari 100k benchmark. Following preceding works, we use human-normalized metrics and compare the mean and median returns across all 26 games. The human-normalized scores are computed for each game using the scores achieved by a human player and the scores obtained by a random policy: $normed\ score = \frac{agent\ score - random\ score}{human\ score - random\ score}$. We also show stratified bootstrap confidence intervals of the human-normalized mean and median in Figure 5. Performance curves corresponding to individual games can be found in the appendix 9.

TWISTER achieves a human-normalized mean score of 162% and a median of 77% on the Atari 100k benchmark, setting a new record among state-of-the-art model-based methods that do not employ look-ahead search techniques. Analogously to STORM, we find that TWISTER demonstrates superior performance in games where key objects related to rewards are numerous, such as *Amidar*, *Bank Heist*, *Gopher* and *Ms Pacman*. Furthermore, we observe that TWISTER benefits from increased performance in games with small moving objects like *Breakout*, *Pong* and *Asterix*. We suppose that the AC-CPC objective requires the agent to focus on the ball's position in these games to accurately predict future samples, thereby preventing failure cases where small objects are ignored by the reconstruction loss. Alternatively, IRIS and Δ-IRIS solve this issue by learning agents from high-quality reconstructed images. They encode image observations into spatial latent spaces through a VQ-VAE structure, which allows these approaches to better capture details and achieve lower reconstruction errors with good results for these games. We show CPC predictions made by the world model for diverse Atari games in the appendix A.4.

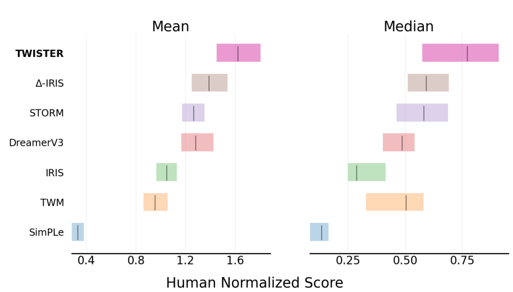


Figure 5: Mean and median scores, computed with stratified bootstrap confidence intervals (Agarwal et al., 2021). TWISTER achieves a normalized mean of 1.62 and a median of 0.77.

4.3 Ablation Studies

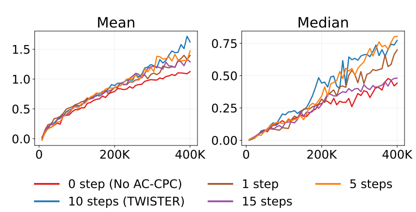
In order to study the impact of AC-CPC on TWISTER performance, we perform ablation studies on all 26 games of the Atari 100k benchmark, applying one modification at a time. We experiment with the number of CPC steps predicted by the world model. We show that data augmentation helps to complexify the AC-CPC objective and improve its effectiveness. We find that conditioning CPC predictions on the sequence of future actions leads to more accurate predictions and improves the quality of representations. We also study the effect of world model design on AC-CPC effectiveness. Table 3 shows the aggregated scores obtained for the main ablations after 400k environment steps.

Table 3: Ablations of the AC-CPC loss, contrastive samples augmentation, conditioning on future actions and using DreamerV3's RSSM. We perform one modification at a time and evaluate on the 26 Atari games. The detailed results obtained for individual games can be found in the appendix A.6.

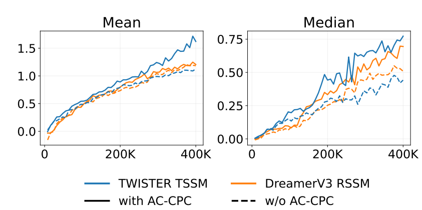
Metrics	TWISTER	No AC-CPC	DreamerV3 World Model	No Action Conditioning	No Data Augmentation
Normed Mean (%)	**162**	112	121	111	120
Normed Median (%)	**77**	44	69	42	68

Number of Contrastive Steps We experiment with several numbers of CPC steps, comparing human-normalized metrics over all 26 games of the Atari100k benchmark. Figure 6a shows that TWISTER achieves the best human-normalized mean score when predicting 10 steps into the future, corresponding to 0.67 seconds of game time. We find that AC-CPC has a significant effect on TWISTER performance up to a certain amount of steps. We observe an increase in human-normalized mean and median scores with the number of predicted CPC steps. However, a degradation of the results is noticed when predicting 15 steps into the future. The difference in median score indicates a decrease in performance for middle-scoring games.

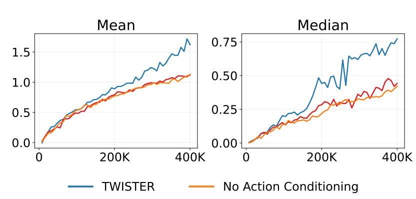
World Model Architecture We study the impact of world model design on AC-CPC effectiveness to learn feature representations. Figure 6b shows the effect of AC-CPC on performance when replacing the TSSM of TWISTER with DreamerV3's RSSM (Hafner et al., 2023). While the two approaches achieve similar results without the AC-CPC objective, we find that AC-CPC has a significant effect on TWISTER, improving performance on most games. These findings can be attributed to the fact that Transformers are generally more effective than RNNs at learning feature representations due to several key architectural differences. The capacity of self-attention to model temporal relationships without recurrence makes the Transformer architecture highly effective at capturing context and learning hierarchical features. On the other hand, the recurrent nature of RNNs can lead to vanishing gradients and slower convergence, particularly with long sequences.



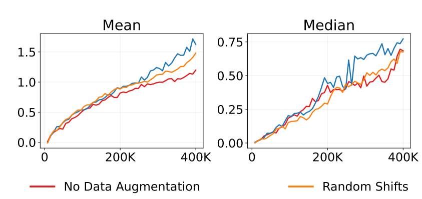
(a) Number of Contrastive steps



(b) World Model Architecture



(c) Action Conditioning



(d) Data Augmentation

Figure 6: Ablations made on the Atari 100k benchmark. The results are averaged over 5 seeds. We study the effect of data augmentation, action conditioning and the number of predicted CPC steps on TWISTER performance. We also study the effect of world model design on AC-CPC effectiveness.

Actions Conditioning We find that conditioning the CPC prediction head on the sequence of future actions leads to more accurate predictions and higher quality representations. Figure 7 shows the aggregated CPC loss and prediction accuracy for training and validation sequences over all Atari games. We report the average number of times the similarity for the positive sample is higher than for the negative samples in the contrastive loss. Without knowing the sequence of future actions, the world model cannot predict future environment states accurately, which makes the task almost insolvable and counterproductive beyond a certain amount of CPC steps. We observe a decrease in accuracy compared to TWISTER when predicting multiple steps without knowing the sequence of future actions. Figure 6c shows the aggregated human-normalized scores over the 26 games when removing the condition of future actions for CPC predictions. We find that the CPC objective does not bring notable performance improvements when removing future actions conditioning.

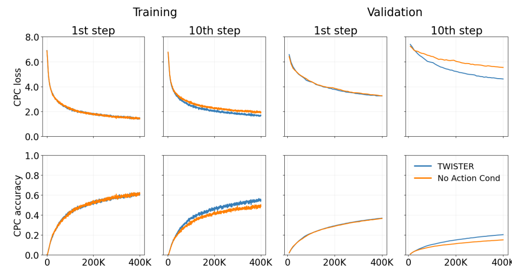


Figure 7: Aggregated CPC loss and prediction accuracy over all 26 games. We use a validation replay buffer of 100k samples to compare CPC loss on unseen trajectories. The trajectories are obtained from a collection of DreamerV3 and TWISTER agents pretrained with 5 seeds.

Effect of Data Augmentation The effect of data augmentation on CPC performance was studied by Kharitonov et al. (2021). In their work, they propose to introduce data augmentation for CPC to learn higher quality speech representations, yielding better performances. In this work, we apply image augmentation to contrastive samples in order to complexify the AC-CPC objective and make the representation learning task more challenging. We apply the commonly used *random crop and resize* augmentation during training for its effectiveness in the area of image-based contrastive learning (Chen et al., 2020). The use of random crops requires the world model to identify several key elements in the observations in order to accurately predict positives samples. We also experiment with *random shifts* (Yarats et al., 2021), shifting the image up to 4 pixels in height and width but found it to have a lesser impact on the learning objective. Figure 6d shows the aggregated human-normalized scores for studied augmentation techniques. We find that *random crop and resize* helps the best to improve final performance. Not using image augmentations for negative and positive samples reduces the impact of AC-CPC on TWISTER performance, achieving lower mean and median scores. We show the impact of data augmentation on the AC-CPC objective complexity for different time horizons in Figure 8.

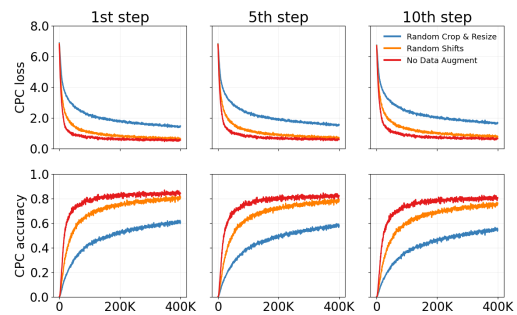


Figure 8: Effect of data augmentation on AC-CPC objective complexity. We aggregate CPC loss and prediction accuracy over all Atari games for different time horizons.

5 Conclusion

We propose TWISTER, a Transformer model-based reinforcement learning agent learning high-level temporal feature representations with action-conditioned Contrastive Predictive Coding. TWISTER achieves new state-of-the-art results on the Atari 100k benchmark among model-based approaches that do not employ look-ahead search with a human-normalized mean and median score of 162% and 77%, respectively. We study the impact of learning contrastive representations on Transformer-based world models and find that the AC-CPC objective significantly helps to improve the agent performance. We also show that data augmentation and future actions conditioning play an important role in the learning of representations to complexify the AC-CPC objective and help the model to make accurate future predictions. Following our early findings, we hope that this work will inspire researchers to further study the benefits of self-supervised learning techniques for model-based reinforcement learning.

6 Acknowledgments

This work was partly supported by The Alexander von Humboldt Foundation (AvH).

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.

Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020.

Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. Transdreamer: Reinforcement learning with transformer world models. *arXiv preprint arXiv:2202.09481*, 2022.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pp. 72–83. Springer, 2006.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Fei Deng, Ingook Jang, and Sungjin Ahn. Dreamerpro: Reconstruction-free model-based reinforcement learning with prototypical representations. In *International conference on machine learning*, pp. 4956–4975. PMLR, 2022.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 297–304. JMLR Workshop and Conference Proceedings, 2010.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

Danijar Hafner. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*, 2021.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.

Mikael Henaff, William F Whitney, and Yann LeCun. Model-based planning with discrete and continuous actions. *arXiv preprint arXiv:1705.07177*, 2017.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. In *International Conference on Learning Representations*, 2020.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Eugene Kharitonov, Morgane Rivière, Gabriel Synnaeve, Lior Wolf, Pierre-Emmanuel Mazaré, Matthijs Douze, and Emmanuel Dupoux. Data augmenting contrastive learning of speech representations in the time domain. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pp. 215–222. IEEE, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *International Conference on Learning Representations*, 2023.

Vincent Micheli, Eloi Alonso, and François Fleuret. Efficient world models with context-aware tokenization. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=BiWIERWBFX`.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan Harmeling. Transformer-based world models are happy with 100k interactions. In *International Conference on Learning Representations*, 2023.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pp. 30365–30380. PMLR, 2023.

David Silver, Hado Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, et al. The predictron: End-to-end learning and planning. In *International Conference on Machine Learning*, pp. 3191–3199. PMLR, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Shengjie Wang, Shaohuai Liu, Weirui Ye, Jiacheng You, and Yang Gao. Efficientzero v2: Mastering discrete and continuous control with limited data. *arXiv preprint arXiv:2403.00564*, 2024.

Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. In *International Conference on Learning Representations*, 2020.

Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Denis Yarats, Rob Fergus, and Ilya Kostrikov. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *9th International Conference on Learning Representations, ICLR 2021*, 2021.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. *Advances in Neural Information Processing Systems*, 34:25476–25488, 2021.

Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. Storm: Efficient stochastic transformer based world models for reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

A Appendix

A.1 Atari 100k Evaluation Curves

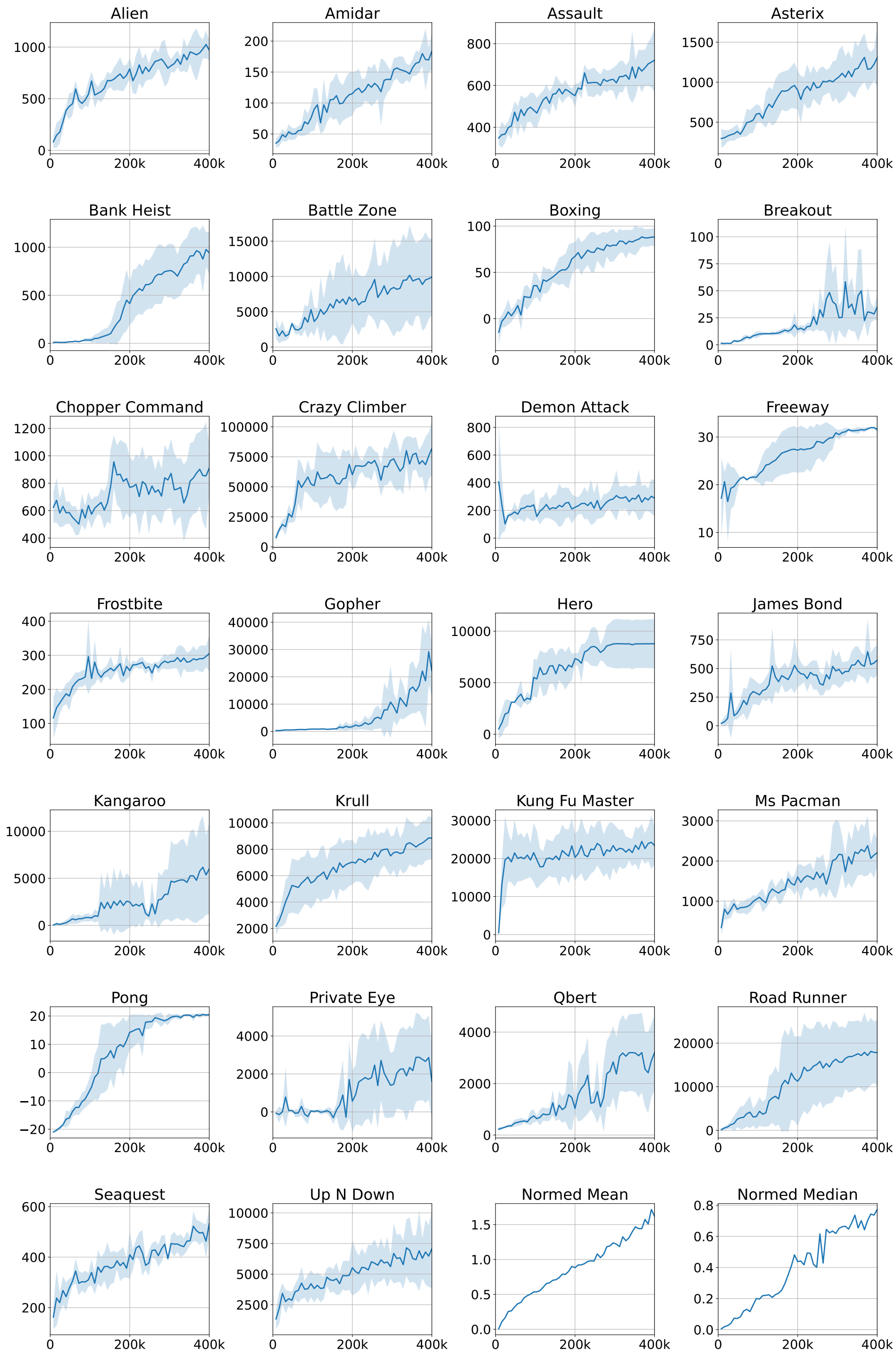


Figure 9: Evaluation curves of TWISTER on the Atari100k benchmark for individual games (400K environment steps). The solid lines represent the average scores over 5 seeds, and the filled areas indicate the standard deviation across these 5 seeds.

A.2 MODEL ARCHITECTURE

Table 4: Architecture of the encoder network. The size of submodules is omitted and can be derived from output shapes. Each convolution layer (Conv) is followed by a layer normalization (LN) and a SiLU activation layer. The encoder downsamples images with strided convolutions layers using a kernel size of 4, a stride of 2 and a padding of 1. We flatten output features and project them to categorical distribution logits using a Linear layer. Stochastic states z_t are sampled from Softmax probabilities and encoded to one hot vectors.

Submodule	Output shape
Input image (o_t)	$3 \times 64 \times 64$
Conv + LN + SiLU	$32 \times 32 \times 32$
Conv + LN + SiLU	$64 \times 16 \times 16$
Conv + LN + SiLU	$128 \times 8 \times 8$
Conv + LN + SiLU	$256 \times 4 \times 4$
Flatten	4096
Linear	1024
Reshape + Softmax	32×32
Sample + One Hot (outputs z_t)	32×32

Table 5: Architecture of the decoder network. Images are reconstructed from stochastic states. Each transposed convolution layer (ConvTrans) uses a kernel size of 4, a stride of 2 and padding of 1.

Submodule	Output shape
Input stochastic state (z_t)	32×32
Flatten	1024
Linear	4096
Reshape	$256 \times 4 \times 4$
ConvTrans + LN + SiLU	$128 \times 8 \times 8$
ConvTrans + LN + SiLU	$64 \times 16 \times 16$
ConvTrans + LN + SiLU	$32 \times 32 \times 32$
ConvTrans (outputs $\hat{o}_t$)	$3 \times 64 \times 64$

Table 6: Transformer block. Dropout (Srivastava et al., 2014) is used in each Transformer submodule to reduce overfitting. We also apply Dropout to attention weights in the MHSA module.

Submodule	Module alias	Output shape
Input features (label as x_1) Multi-head self-attention Linear + Dropout Residual (add x_1) LN (label as x_2)	MHSA	$T \times 512$
Linear + ReLU Linear + Dropout Residual (add x_2) LN	Feed Forward	$T \times 1024$ $T \times 512$ $T \times 512$ $T \times 512$

Table 7: Transformer network. The stochastic states $z_{0:T-1}$ and one-hot encoded actions $a_{0:T-1} \in \mathbb{R}^{T \times A}$ are combined using an action mixer network (Zhang et al., 2024). The features are processed by the Transformer network to compute hidden states $h_{1:T}$.

Submodule	Module alias	Output shape
Inputs stochastic states ($z_{0:T-1}$) Flatten Concat actions $a_{0:T-1}$ Linear + LN + SiLU Linear + LN	Action Mixer	$T \times 32 \times 32$ $T \times 1024$ $T \times (1024 + A)$ $T \times 512$ $T \times 512$
Transformer block $\times$ K Outputs hidden states ($h_{1:T}$)	Transformer Network	$T \times 512$

Table 8: Networks with Multi Layer Perceptron (MLP) structure. Inputs are first flattened and concatenated along the feature dimension. Each MLP layer is followed by a layer normalization and SiLU activation except for the last layer which outputs distribution logits.

Network	MLP layers	Inputs	Hidden dimension	Output dimension	Output Distribution
Reward predictor	3	s_t	512	255	Symlog Discrete
Continue predictor	3	s_t	512	1	Bernoulli
Representation network	2	z'_{t+k}	512	512	N/A
AC-CPC predictor	2	$s_t, a_{t:t+k}$	512	512	N/A
Critic network	3	s_t	512	255	Symlog Discrete
Actor network	3	s_t	512	A	One hot Categorical

A.3 Hyper-parameters

Table 9: TWISTER hyper-parameters. We apply the same hyper-parameters to all Atari games.

Parameter	Symbol	Setting
General		
Batch Size	B	16
Sequence Length	T	64
Optimizer	—	Adam (Kingma & Ba, 2014)
Image Resolution	—	64×64 (RGB)
Training Step per Policy Step	—	1
Environment Instances	—	1
Transformer Network		
Transformer Blocks	N	4
Number of Attention Heads	—	8
Dropout Probability	—	0.1
Attention Context Length	—	8
World Model		
Stochastic State Features	—	32
Classes per Feature	—	32
Dynamics Loss Scale	β_{dyn}	0.5
Representation Loss Scale	β_{reg}	0.1
AC-CPC Steps	K	10
Random Crop & Resize Scale	—	(0.25, 1.0)
Random Crop & Resize Ratio	—	(0.75, 1.33)
Learning Rate	α	10^{-4}
Adam Betas	β_1, β_2	0.9, 0.999
Adam Epsilon	ϵ	10^{-8}
Gradient Clipping	—	1000
Actor Critic		
Imagination Horizon	H	15
Return Discount	γ	0.997
Return Lambda	λ	0.95
Critic EMA Decay	—	0.98
Return Normalization Momentum	—	0.99
Actor Entropy Scale	η	$3 \cdot 10^{-4}$
Learning Rate	α	$3 \cdot 10^{-5}$
Adam Betas	β_1, β_2	0.9, 0.999
Adam Epsilon	ϵ	10^{-5}
Gradient Clipping	—	100

A.4 AC-CPC PREDICTIONS

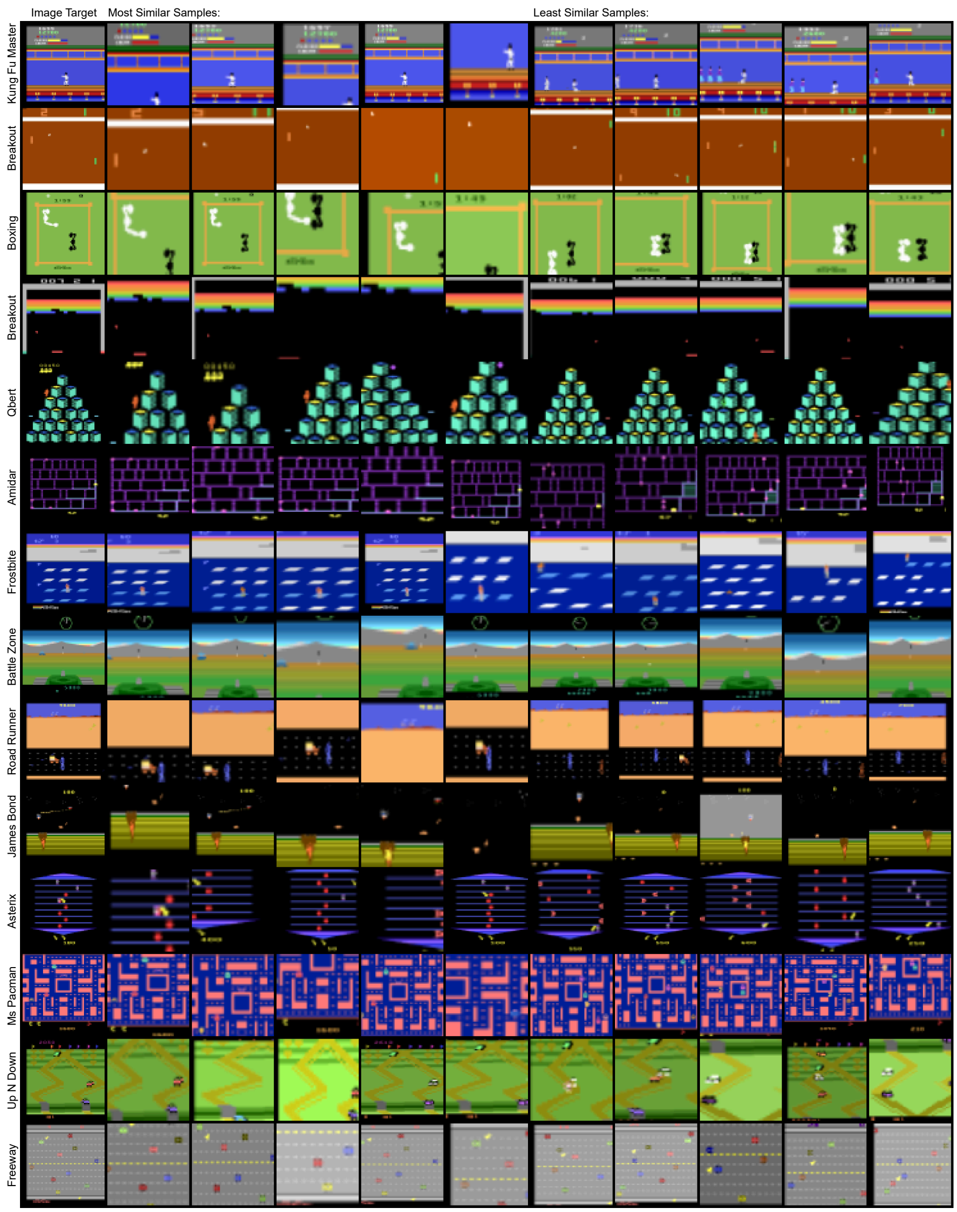


Figure 10: AC-CPC predictions made by the world model for diverse Atari games. We show the target positive sample without augmentation and predicted most/least similar samples among the batch of augmented image views. We observe that TWISTER successfully learns to identify most/least similar samples to the future target state using observation details such as the ball position in *Pong*, the game score in *Kung Fu Master* or the agent movements in *Boxing*. AC-CPC necessitates the agent to focus on observation details to accurately predict future samples, thereby preventing common failure cases where small objects are ignored by the reconstruction loss.

A.5 World Model Predictions

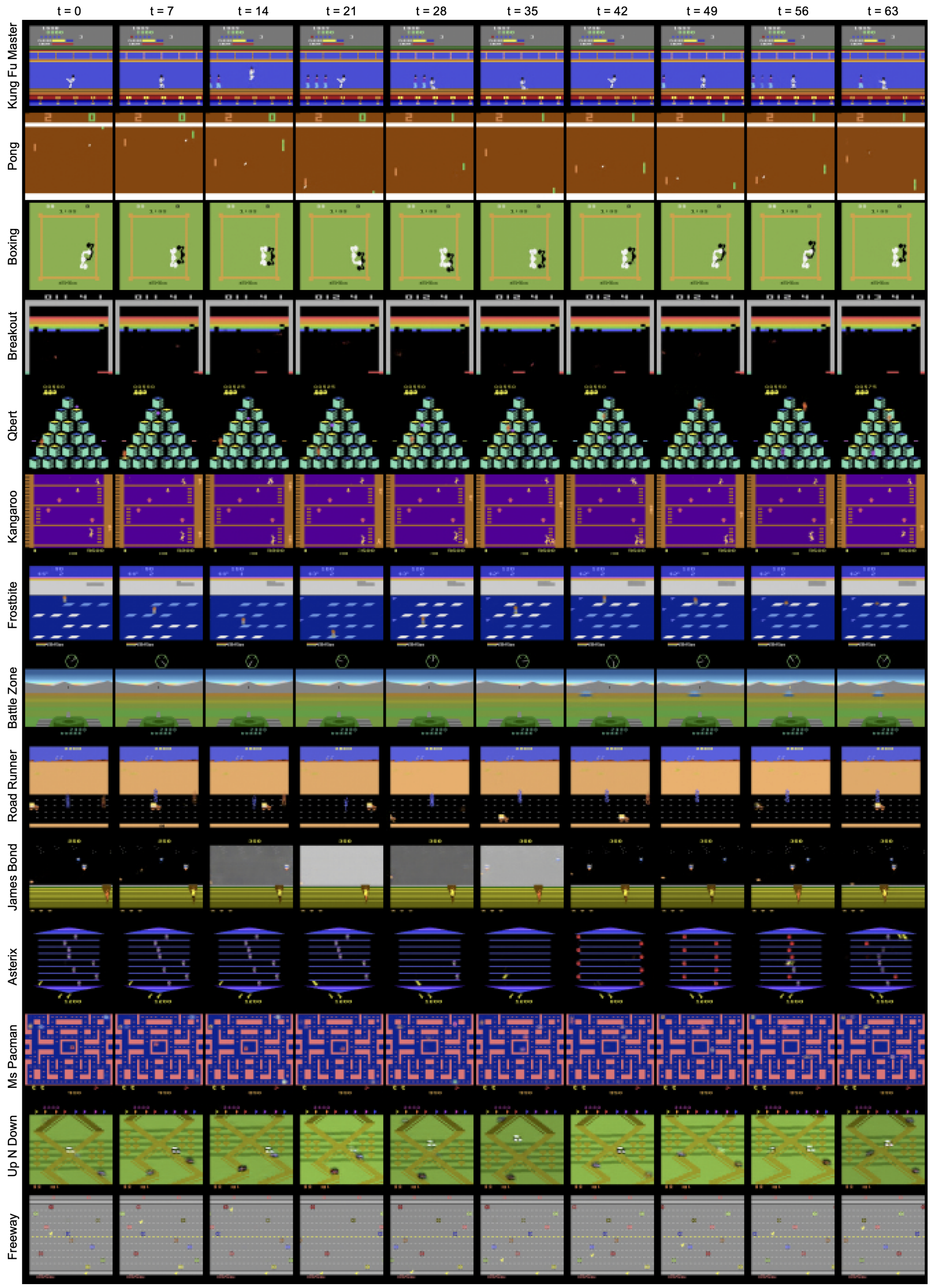


Figure 11: World Model Predictions. We show the decoder reconstruction of trajectories imagined by the world model over 64 time steps. We use 5 context frames and generate trajectories of 59 steps into the future using the Transformer network and dynamics predictor head. Actions are predicted by the actor network by sampling from the categorical distribution.

A.6 ABLATIONS RESULTS

Table 10: Ablations of the AC-CPC loss, contrastive samples augmentation, conditioning on future actions and using DreamerV3's RSSM. We show agent scores and human-normalized metrics on the 26 games of the Atari 100k benchmark. The results are averaged over 5 seeds and bold numbers indicate best performing agent for each game.

Game	TWISTER	No CPC	DreamerV3 World Model	No Action Conditioning	No Data Augmentation
Alien	970	1147	1040	1101	**1154**
Amidar	184	173	174	**191**	154
Assault	721	**1168**	735	711	657
Asterix	1306	1165	**1401**	1082	1173
Bank Heist	942	758	**973**	651	944
Battle Zone	9920	5800	**15540**	12860	8980
Boxing	**88**	81	82	77	84
Breakout	35	14	34	**59**	31
Chopper Command	910	984	**1242**	620	646
Crazy Climber	81880	**90680**	89888	87272	77454
Demon Attack	289	215	**456**	339	356
Freeway	**32**	**32**	0	31	26
Frostbite	305	714	571	884	**953**
Gopher	**22234**	1387	3318	2972	5851
Hero	8773	8772	9944	7649	**11079**
James Bond	**573**	493	432	335	316
Kangaroo	**6016**	4724	3816	1268	2668
Krull	8839	8096	7469	8054	**9065**
Kung Fu Master	23442	22232	**25518**	19412	17566
Ms Pacman	2206	2025	1691	1927	**2294**
Pong	**20**	13	18	**20**	**20**
Private Eye	**1608**	941	535	106	489
Qbert	3197	2579	3542	**4443**	4231
Road Runner	**17832**	10556	12254	12590	13348
Seaquest	532	474	**569**	491	467
Up N Down	7068	5816	**30135**	5378	7213
# Superhuman	**12**	8	11	9	8
Normed Mean (%)	**162**	112	121	111	120
Normed Median (%)	**77**	44	69	42	68

A.7 DeepMind Control Suite Results

We assess TWISTER's performance on continuous action control tasks by evaluating on the DeepMind Control Suite (Tassa et al., 2018). The suite was designed to serve as a reliable performance benchmark for reinforcement learning agents in continuous action space, including diverse control tasks with various complexities. Similarly to DreamerV3 (Hafner et al., 2023), we evaluate on 20 tasks using only high-dimensional image observations as inputs and a budget of 1M environment steps for training.

We compare TWISTER with DreamerV3 and two other recent model-based approaches applied to continuous control. DreamerPro (Deng et al., 2022) proposed a reconstruction-free variant of the Dreamer algorithm. Similarly to SwAV Caron et al. (2020), the agent learns hidden representations by encouraging consistent cluster assignments for different augmentations of the same images Caron et al. (2020). More recently, TD-MPC2 Hansen et al. (2023) extended the TD-MPC Hansen et al. (2022) agent to multitask learning and demonstrated state-of-the-art performance on diverse continuous control tasks. TD-MPC unrolls its world model over the batch of sampled trajectories to predict the sequence of future latent states and environment quantities. The agent also learns a Q-value function to estimate long-term returns using Temporal Difference (TD) learning. It uses Model Predictive Control (MPC) for planning, selecting actions that maximize expected returns using world model value predictions.

Table 11 shows the results obtained on the 20 tasks after 1M environment steps. We obtain DreamerPro[1] and TD-MPC2[2] results using official implementations. TWISTER obtains state-of-the-art performance with a mean score of 801.8. We also experiment with removing the AC-CPC objective and find that it has a positive impact on most of the tasks. AC-CPC particularly improves performance on complex tasks such as *Acrobot Swingup*, *Quadruped Run / Walk* and *Walker Run*.

Table 11: Agent scores on the DeepMind Control Suite under visual inputs. We show average scores over 5 seeds (1M environment steps). Bold numbers indicate best performing method for each task. We also underline TWISTER numbers to indicate tasks where AC-CPC improves performance.

Task	DreamerPro	DreamerV3	TD-MPC2	TWISTER (No CPC)	TWISTER (ours)
Acrobot Swingup	**438.3**	210.0	216.0	81.6	<u>239.4</u>
Ball In Cup Catch	962.2	957.1	717.2	964.8	<u>**966.8**</u>
Cartpole Balance	**998.5**	996.4	931.1	**998.5**	997.9
Cartpole Balance Sparse	**1000.0**	**1000.0**	**1000.0**	**1000.0**	**1000.0**
Cartpole Swingup	**871.0**	819.1	808.1	759.1	<u>819.2</u>
Cartpole Swingup Sparse	**811.2**	792.9	739.0	524.4	<u>735.0</u>
Cheetah Run	**898.3**	728.7	550.3	695.1	694.0
Finger Spin	600.6	818.5	**986.0**	845.8	<u>976.3</u>
Finger Turn Easy	879.6	787.7	788.9	**956.4**	923.7
Finger Turn Hard	719.6	810.8	871.8	**960.3**	910.3
Hopper Hop	252.6	**369.6**	211.1	313.8	<u>313.9</u>
Hopper Stand	927.5	900.6	803.0	**936.5**	932.0
Pendulum Swingup	**845.2**	806.3	743.2	529.2	<u>832.1</u>
Quadruped Run	616.2	352.3	361.9	503.8	<u>**652.1**</u>
Quadruped Walk	676.9	352.6	252.6	741.6	<u>**904.9**</u>
Reacher Easy	945.5	898.9	**971.0**	886.1	<u>933.1</u>
Reacher Hard	294.0	499.2	**876.9**	380.0	<u>565.8</u>
Walker Run	750.0	**757.8**	728.1	566.2	<u>711.2</u>
Walker Stand	974.8	976.7	915.8	969.5	<u>**976.9**</u>
Walker Walk	**956.6**	955.8	945.1	961.2	951.3
Mean	770.9	739.6	720.9	728.7	<u>**801.8**</u>
Median	858.1	808.5	795.9	802.4	<u>**907.6**</u>

[1] `https://github.com/fdeng18/dreamer-pro`
[2] `https://github.com/nicklashansen/tdmpc2`

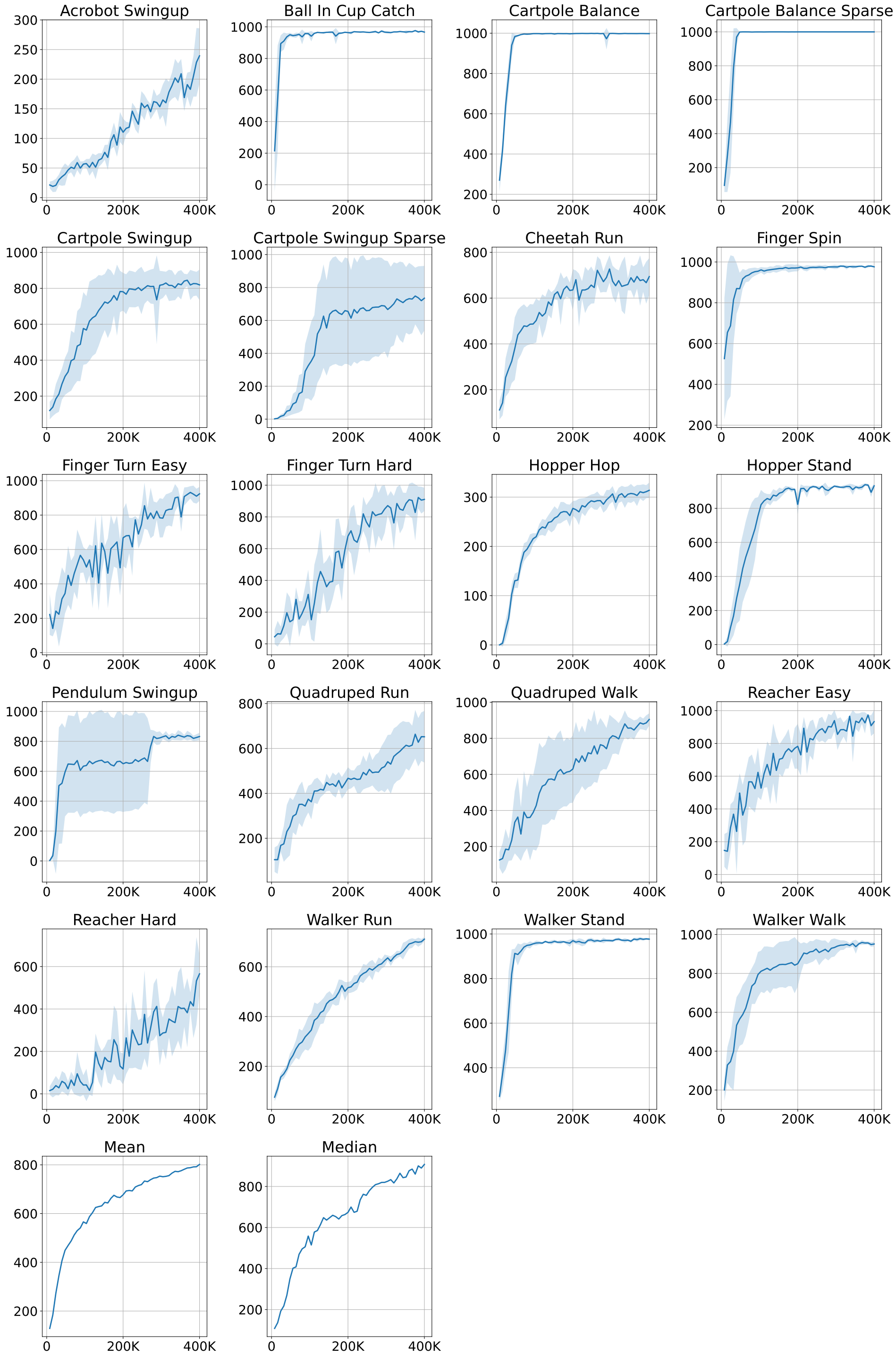


Figure 12: Evaluation curves of TWISTER on the DeepMind Control Suite for individual tasks (1M environment steps). The solid lines represent the average scores over 5 seeds, and the filled areas indicate the standard deviation across these 5 seeds.