TWISTED: ENHANCING TRANSFORMER WORLD MODELS WITH SPATIO-TEMPORAL ENCODING AND GRAPH-BASED OPTIMAL DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Model-based reinforcement learning improves sample efficiency by using learned world models to simulate experiences for training agents. Recent world models that leverage transformers demonstrate high quality simulations, leading to better agent performance. However, transformer world models underutilize spatial relationships between visually adjacent tokens, which are critical when interacting in visual environments. Additionally, current models rely on sampling methods for transformer decoding that do not leverage visual similarities among subsequent frames. To address these limitations, we introduce TWISTED, a transformer world model with 3D spatio-temporal positional encoding and a graph-based optimal decoding strategy specific to visual environments. Our experiments show state-of-the-art performance on the Craftax-classic, Craftax, and MinAtar benchmarks, challenging visual environments requiring long-horizon object recall and interaction. The proposed method achieves a *return* of 72.5% and a *score* of 35.6% on Craftax-classic, significantly surpassing the previous best of 67.4% and 27.9%. We plan to release our source code on GitHub upon acceptance.

1 INTRODUCTION

Reinforcement learning (RL) provides a framework for training agents to interact with their environment through reward signals (Sutton & Barto, 2018). To avoid heavy reliance on costly environment interactions, model-based RL learns a predictive model of the environment dynamics, enabling the agent to simulate future trajectories called "imaginations" (Hafner et al., 2023; Micheli et al., 2022). Recently, transformers have emerged as powerful world models (Micheli et al., 2022; Dedieu et al., 2025). They treat sequences of past states and actions as token streams and predict the next state token-by-token. However, adapting transformers to world modeling faces challenges that have not been fully explored. First, transformers typically rely on one-dimensional positional encodings to capture token order, which may be insufficient for visual environments. Second, effective token decoding strategies tailored to world models have yet to be investigated.

The standard choice for positional encoding is Rotary Position Embedding (RoPE), which captures the relative distance between tokens (Su et al., 2024). While effective in text domains, RoPE is less suited to visual environments, where data is naturally structured in three dimensions—two spatial (within frames) and one temporal (across frames). RoPE encodes only one-dimensional relationships, leading to a loss of fine-grained spatio-temporal structure when applied naively to vision-based tasks.

For transformer decoding, common practice is to sample tokens from the output probabilities in parallel or sequentially (Micheli et al., 2022; Dedieu et al., 2025). Parallel decoding, while efficient, ignores dependencies between output tokens, often leading to hallucinations in complex environments such as duplicated objects. On the other hand, sequential decoding incurs higher computational costs and has been shown to degrade generation quality due to auto-regressive drift (Dedieu et al., 2025). Moreover, neither decoding scheme leverages the fact that sequential frames in visual environments are highly similar (see Figure 1(a)). Meanwhile, exploiting similarities between images has been studied extensively in computer vision, leading to the development of optical flow-based techniques. (Brox et al., 2004; Vedula et al., 2005; Perazzi et al., 2016).

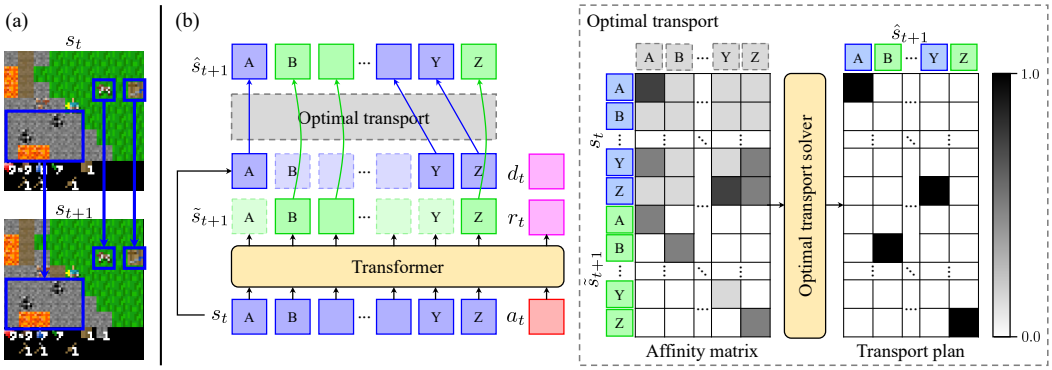

Figure 1: (a) Sequential frames in visual environments like Craftax-classic preserve most of the information from the previous frame. (b) Our proposed world model enhances next state prediction by solving an optimal transport problem with previous state tokens (s_t, blue) and the transformer's output for candidate next-state tokens ($\tilde{s}_{t+1}$, green) to generate the final next-state tokens ($\hat{s}_{t+1}$). Optimal transport defines an affinity matrix from the s_t and $\tilde{s}_{t+1}$ tokens to the positions for $\hat{s}_{t+1}$. A solver takes the affinity matrix and produces a transport plan, assigning a token from s_t or $\tilde{s}_{t+1}$ to each final next-state token in $\hat{s}_{t+1}$. This approach enables effective reuse of relevant past tokens.

In this regard, we propose TWISTED (Transformer World model with Informed Spatio-Temporal Encoding and Decoding), a transformer world model designed for visual RL environments. TWISTED introduces two key innovations:

1. A 3D spatio-temporal positional encoding that combines absolute and relative encodings across space and time, preserving both spatial structure and temporal structure.
2. A graph-based optimal decoding scheme grounded in optimal transport, which formulates decoding as a transport problem between the previous frame's optimally decoded tokens and the transformer's predictions for next tokens. This enables partial reuse of previous tokens, reducing hallucinations and improving object persistence over time.

We evaluate TWISTED on the Craftax-classic, Craftax, and MinAtar benchmarks. Craftax-classic is a challenging 2D open-world game featuring long-horizon tasks and dynamic enemies (Matthews et al., 2024). TWISTED achieves a return of 72.5% and a score of 35.6%, setting a new state-of-the-art and outperforming the previous best results of 67.4% and 27.9%, respectively (Dedieu et al., 2025). Craftax is a harder environment based on Craftax-classic, in which TWISTED also exceeds baselines (Matthews et al., 2024). MinAtar is a suite of 4 Atari games with simplified representations, which tests generality across different game dynamics (Young & Tian, 2019). TWISTED surpasses the previous state of the art for model-based RL in all 4 games (Dedieu et al., 2025).

2 PRELIMINARIES

2.1 MODEL-BASED REINFORCEMENT LEARNING

Reinforcement learning considers a Partially Observable Markov Decision Process, characterized by $(\mathbb{S}, \mathbb{A}, \Omega, T, O, R, \gamma)$, where $\mathbb{S}$ is a set of states, $\mathbb{A}$ is a set of discrete actions, Ω is a set of observations, T gives the transition probabilities between states $T(s' \mid s, a)$, O gives the observation probabilities $O(o \mid s)$, and R is a reward function $R(s, a)$ (Sutton & Barto, 2018). The goal is to find a policy π which chooses actions for each state that maximizes the expected discounted return $\mathbb{E}_\pi \left[\sum_{t \geq 0} \gamma^t r_t\right]$, where γ is a discount factor. A world model takes an input of previous state s_t and action a_t, then returns a predicted output of next state $\hat{s}_{t+1}$, reward r_t, and done signal d_t, similar to the real environment. The agent collects real environment trajectories during training by interacting with the environment using the policy π. Then the world model trains on the trajectories saved in

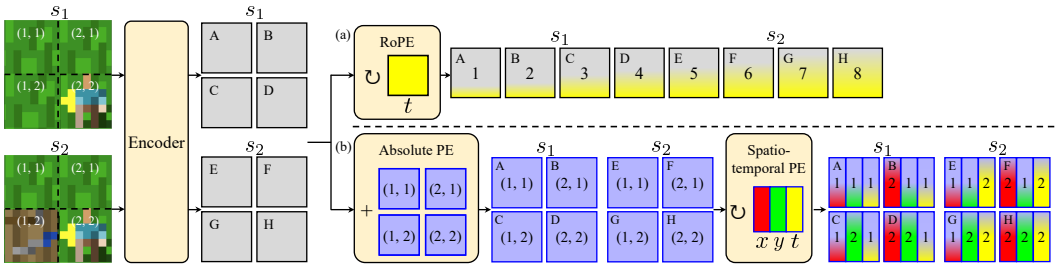

Figure 2: Use of positional encoding in (a) original RoPE vs. (b) our method. Original RoPE makes vertically adjacent (A and C) and temporally adjacent (A and E) tokens far from each other, while our method brings them together along two out of three axes. Moreover, original RoPE makes spatially distant tokens (D and E) adjacent, while our method separates them across all axes.

the replay buffer. Over the course of training, the agent is trained on both the trajectories collected from the real environment and generated trajectories from the world model, called *imaginations*.

2.2 RoPE

Rotary Position Embedding (RoPE) is a positional encoding method that injects positional information into a transformer's attention mechanism by applying rotations to query and key vectors (Su et al., 2024). These rotations cause the attention operation to naturally encode relative offsets between tokens. Concretely, each input token embedding is partitioned into pairs of coordinates, with each pair forming a 2D subspace where a rotation is applied according to the token's 1D position index. Owing to its simplicity and scalability, RoPE has become the standard positional encoding in modern transformer architectures.

2.3 Optimal transport

Optimal transport is a family of optimization problems that compares and aligns probability distributions based on a given cost of moving mass between elements (Peyré & Cuturi, 2019). Optimal transport considers probability distributions $\mathbf{a} \in \Delta^{n-1}$ and $\mathbf{b} \in \Delta^{m-1}$ over the source and target domains, respectively. It seeks a transport plan $\mathbf{\Pi} \in \mathbb{R}_{+}^{n \times m}$ that minimizes the cost $\langle \mathbf{\Pi}, \boldsymbol{C} \rangle = \sum_{i=1}^{n} \sum_{j=1}^{m} \Pi_{ij} C_{ij}$, subject to the marginal constraints $\mathbf{\Pi} \mathbf{1}_m = \mathbf{a}$ and $\mathbf{\Pi}^\top \mathbf{1}_n = \mathbf{b}$.

To solve optimal transport problems efficiently, regularized variants of optimal transport have been proposed. One popular approach introduces an entropic regularization term to the objective, leading to the *Sinkhorn distance*, which can be computed efficiently using iterative matrix scaling (Cuturi, 2013). The Sinkhorn algorithm solves the regularized problem in $O(n^2/\epsilon^2)$ time for a desired approximation error ϵ, making it practical for large-scale problems.

3 Method

Based on the concepts presented in Section 2, our method centers around a transformer world model that exploits spatial relationships within frames and temporal relationships between frames. First, states and actions are converted into tokens using a tokenizer. Then, the token embeddings are augmented with positional encodings that capture spatio-temporal information, before being fed into a transformer. Finally, the transformer output tokens are used by an optimal transport solver to produce the next state tokens, as shown in Figure 1(b). Through this process, the world model generates imagined trajectories for policy training.

3.1 Tokenizer

Following the practice in Dedieu et al. (2025), we represent states and actions as discrete tokens to interface with the transformer, using a tokenizer that converts the visual observation to tokens using

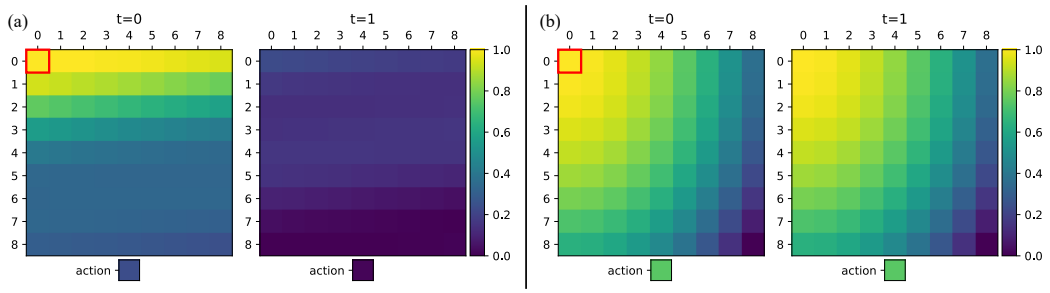

Figure 3: Attention pattern comparison between (a) original RoPE and (b) our relative positional encoding. The heatmaps show a normalized attention score given identical query and key matrices, with respect to the top left query at $(x, y, t) = (0, 0, 0)$, marked with red boxes. Original RoPE tends to attend to x-adjacent tokens more than y-adjacent and t-adjacent tokens. Our spatio-temporal positional encoding preserves adjacency of all three axes.

nearest neighbor patch lookup. Each token represents a particular visual patch of the image state. First, each frame is divided into a grid of L visual patches $\{p_1, \ldots, p_L\}$, where $p_i \in [0, 1]^{h \times w \times 3}$ with height h and width w. The tokenizer maintains a codebook $C = \{c_1, \ldots, c_K\}$, consisting of K codes $c_i \in [0, 1]^{h \times w \times 3}$. Each patch p is mapped to a token q by finding its nearest neighbor in the codebook:

$$q = \text{enc}(p) = \underset{1 \le i \le K}{\text{argmin}} \, \|p - c_i\|_2^2 \, .$$

The codebook is constructed by sampling patches from the replay buffer. A patch is added if it is sufficiently far away from all existing codes: when $\min_{1 \le i \le K} \|p - c_i\|_2^2 > \tau$ for a chosen threshold τ. To convert tokens back to images, the tokenizer retrieves the corresponding code for each token $\text{dec}(q) = c_q$ and reassembles the grid into the full image.

3.2 Spatio-temporal positional encoding

After converting image frames into tokens, positional encodings are added to the token embeddings for input to the transformer. Previous transformer world models employ Rotary Position Embedding (RoPE) for positional encoding (Su et al., 2024). However, RoPE uses a single-dimensional position index, which is unable to distinguish between temporal differences (i.e., tokens from different time steps) and spatial differences (i.e., tokens from different positions within the same frame). To incorporate both spatial and temporal information into the model, our method employs a two-fold positional encoding strategy that uses both absolute position and relative position (see Figure 2). First, each token receives a trainable embedding according to its absolute spatial coordinates (x, y) in the grid of patches. This embedding is added to the original token embedding, anchoring each token to a specific semantic location in the observation.

To capture relative positional relationships, our method applies RoPE across spatial and temporal axes. Each token's embedding is divided into three sub-vectors corresponding to its temporal, vertical, and horizontal coordinates. RoPE is then applied independently along each axis, enabling the attention mechanism to capture localized relational structure across both space and time. This formulation allows the model to generalize over local interactions (e.g., neighboring pixels or frames), regardless of absolute location. It preserves adjacency in both spatial and temporal dimensions, while original RoPE loses the adjacency of the y-axis and temporal axis, as visualized in Figure 3.

A special indexing strategy handles action tokens, since actions do not have inherent spatial coordinates. To apply relative spatio-temporal encoding, the action at timestep t is assigned the spatial coordinates (t, t). This simplifies action representation to a 1D RoPE formulation, while placing an action token adjacent to the state tokens in the same timestep. Additional indexing details are described in Appendix C.

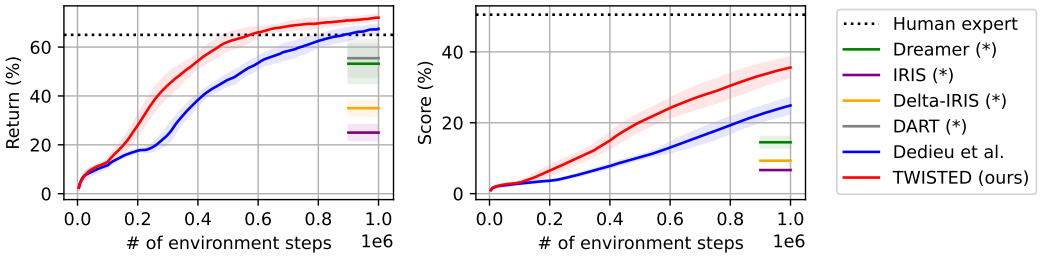

Figure 4: TWISTED achieves state-of-the-art return and score in Craftax-classic, with significantly faster convergence (Matthews et al., 2024). Shading indicates standard deviation among seeds. *Baselines with reported results at 1M steps are displayed with horizontal lines from 900K to 1M steps. DART does not report score, and IRIS and Δ-IRIS do not report standard deviation for score.

3.3 Optimal transport-based decoding

Having established the spatio-temporal positional encoding for transformer inputs, the next key mechanism involves decoding the transformer outputs. In existing approaches, the output of the transformer world model is directly used to predict each token in the next frame (Micheli et al., 2022; 2024; Agarwal et al., 2024; Dedieu et al., 2025). However, in most visual environments, two adjacent frames are often very similar, e.g. the same tiles but shifted when the player moves right. Our intuition is closely related to the notion of optical flow from classic computer vision tasks (Brox et al., 2004; Vedula et al., 2005; Perazzi et al., 2016). This relation allows tokens to be taken directly from the previous frame into the next frame, rather than offloading the burden of regenerating all next-state tokens to the transformer. To exploit this, the final next-state token predictions are formulated as an optimal transport problem.

Let L be the number of tokens for each frame state. Our method constructs a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the vertices $\mathcal{V} = \mathcal{V}_S \cup \mathcal{V}_D$ consist of source vertices $\mathcal{V}_S$ that correspond to previous state tokens and candidate next-state tokens, and destination vertices $\mathcal{V}_D$ that represent the finalized next-state tokens ($|\mathcal{V}_S| = 2L$ and $|\mathcal{V}_D| = L$). The edges $\mathcal{E} = \{(u, v) \mid u \in \mathcal{V}_S, v \in \mathcal{V}_D\}$ connect all sources to all destinations. We now define affinities on these edges for transport.

Let K be the size of the codebook. Given transformer predictions $\mathbf{p}_j \in [0, 1]^K$ for the next state tokens, and previous state tokens $\mathbf{u}_i \in \{0, 1\}^K$ for all $i, j \in \{0, \dots, L-1\}$, we define an affinity matrix $\boldsymbol{A}^{(prev)} \in \mathbb{R}^{L \times L}$ that scores the affinity between previous state tokens and predicted next-state tokens. Each entry is computed as:

$$A_{ij}^{(prev)} = \langle \mathbf{p}_j, \mathbf{u}_i \rangle - c_d D\left((x_i, y_i), (x_j, y_j)\right), \forall i, j \in \{0, \dots, L-1\}, \tag{1}$$

where c_d is a coefficient of cost for distance, $D(\cdot)$ is a distance function for 2D coordinates, and (x_i, y_i) and (x_j, y_j) are the 2D coordinates of the i-th and j-th tokens, respectively. To allow the model to generate new content not present in the previous frame, the graph includes wildcard tokens. The matrix $\boldsymbol{A}^{(gen)} \in \mathbb{R}^{L \times L}$ scores the bonus of admitting newly generated tokens instead of reusing the previous ones, using diagonal entries:

$$A_{kj}^{(gen)} = \begin{cases} \|\mathbf{p}_j\|_\infty - c_w, & \text{if } k = j, \\ -\infty & \text{otherwise,} \end{cases} \quad \forall k, j \in \{0, \dots, L-1\}, \tag{2}$$

where c_w is a constant penalty for using a wildcard token. With the matrices defined above, an optimal transport plan $\boldsymbol{P}^{(prev)}$ and $\boldsymbol{P}^{(gen)}$ is computed by optimizing the following equation:

$$\begin{aligned} \underset{\substack{\boldsymbol{P}^{(prev)}\in[0,1]^{L\times L}\\ \boldsymbol{P}^{(gen)}\in[0,1]^{L\times L}}}{\text{minimize}} \quad & \left\langle -\begin{pmatrix}\boldsymbol{A}^{(prev)}\\ \boldsymbol{A}^{(gen)}\end{pmatrix}, \begin{pmatrix}\boldsymbol{P}^{(prev)}\\ \boldsymbol{P}^{(gen)}\end{pmatrix} \right\rangle \\ \text{subject to} \quad & \boldsymbol{P}^{(prev)}\mathbf{1}_L \le \mathbf{1}_L, \\ & \boldsymbol{P}^{(gen)}\mathbf{1}_L \le \mathbf{1}_L, \\ & \left(\boldsymbol{P}^{(prev)} + \boldsymbol{P}^{(gen)}\right)^\top \mathbf{1}_L = \mathbf{1}_L. \end{aligned} \tag{3}$$

Solving the optimal transport problem yields a partial transport plan, represented by a matrix with continuous values in the range $[0, 1]$. However, our application requires a strict one-to-one mapping between discrete tokens. To address this, we convert the partial transport plan into a binary assignment matrix with values $\{0, 1\}$ using a greedy binarization procedure based on column-wise argmax. Specifically, for each column in the transport matrices $\boldsymbol{P}^{(prev)}$ and $\boldsymbol{P}^{(gen)}$, we identify the row with the highest transport weight, selecting that row in either $\boldsymbol{P}^{(prev)}$ or $\boldsymbol{P}^{(gen)}$, whichever yields the larger value. In the event of a conflict where multiple columns select the same row, we retain the assignment corresponding to the column with the higher transport value and reassign the conflicting column using argmax again, excluding rows that have already been assigned. The complete binarization procedure is described in Algorithm 3 in Appendix D.

Let $\boldsymbol{\Pi}^{(prev)} \in \{0,1\}^{L\times L}$ and $\boldsymbol{\Pi}^{(gen)} \in \{0,1\}^{L\times L}$ denote the binarized versions of $\boldsymbol{P}^{(prev)}$ and $\boldsymbol{P}^{(gen)}$, respectively. The j-th token of the next state is determined by copying the i-th token of the previous state where $\Pi_{ij}^{(prev)} = 1$. If no such i exists, which occurs only when $\Pi_{jj}^{(gen)} = 1$, the model instead samples from the transformer's predicted distribution. The overall decoding rule is thus defined as

$$\mathbf{u}'_j = \begin{cases} \mathbf{u}_i, & \text{where } \Pi_{ij}^{(prev)} = 1, \\ \text{sample}(\mathbf{p}_j) & \text{where } \Pi_{jj}^{(gen)} = 1, \end{cases} \quad \forall j \in \{0, \dots, L-1\}. \tag{4}$$

Solving this optimization problem involves the Sinkhorn algorithm. By default, the Sinkhorn algorithm minimizes the objective given by a cost matrix rather than an affinity matrix, so the cost matrix is set as the negative of the computed affinity matrix. The end-to-end decoding process is characterized in Algorithm 1 in Appendix D, which also contains additional algorithmic details.

4 Experiments

4.1 Craftax-classic

Environment We evaluate our method on the Craftax-classic environment (Matthews et al., 2024). Craftax-classic is a fast implementation of Crafter, a challenging procedurally generated, partially observable environment featuring stochastic transitions and a complex hierarchy of achievements (Hafner, 2021). These attributes demand both strong generalization and the ability to model object interactions across time.

Experiment configuration Each method is trained on Craftax-classic for 1M environnment steps, using 10 different seeds per method. The baseline methods consist of DreamerV3 (Hafner et al., 2023), IRIS (Micheli et al., 2022), Δ-IRIS (Micheli et al., 2024), DART (Agarwal et al., 2024), and Dedieu et al. (2025)[1], which had the previous state-of-the-art return on Craftax-classic. Each experiment runs on a single Nvidia RTX 3090 GPU for 57.7 hours. See Appendix B for all hyperparameters and Appendix G for details on compute time.

Results Figure 4 shows that our proposed world model leads to substantially higher return and score, along with faster convergence compared to baseline methods.[2] Return and score are reported

[1] We use the (fast) variant from Dedieu et al. (2025), as the (slow) variant is prohibitively expensive to train.

[2] Score is a metric defined as the geometric mean of the success rates for each achievement (Hafner, 2021). Score puts more emphasis on unlocking a variety of achievements, in contrast to return, which is simply the sum of rewards for each episode.

Table 1: Results on Craftax-classic after 0.5M and 1M environment interactions. Return is averaged over episodes of the final 50,000 environment interactions to smooth out variance. The final value for Score is reported directly, as it is already a cumulative metric and does not require additional smoothing. Metrics not reported by baselines are marked as —. † uses hyperparameters of TWISTED.

	@ 0.5M		@ 1M	
Method	Return (%)	Score (%)	Return (%)	Score (%)
Human expert	—	—	65.0 ± 10.5	50.5 ± 6.8
DreamerV3 (Hafner et al., 2023)	—	—	53.2 ± 8.0	14.5 ± 1.6
IRIS (Micheli et al., 2022)	—	—	25.0 ± 3.2	6.66
Δ-IRIS (Micheli et al., 2024)	—	—	35.0 ± 3.2	9.30
DART (Agarwal et al., 2024)	—	—	55.45 ± 3.39	—
Dedieu et al. (2025)	—	—	67.42 ± 0.55	27.91 ± 0.63
Dedieu et al. (2025) (reproduced)	48.17 ± 0.82	10.22 ± 0.20	68.14 ± 0.42	24.89 ± 0.74
Dedieu et al. (2025) (reproduced)†	54.32 ± 0.60	13.06 ± 0.39	68.55 ± 0.72	27.24 ± 0.86
TWISTED (ours)	**63.10** ± 1.24	**20.12** ± 0.80	**72.46** ± 0.45	**35.60** ± 0.92

in Table 1, as the mean and standard error over 10 seeds. After 1M environment interactions, our method achieves a final return and score surpassing all baselines. It also outperforms the previous best baseline during training at 0.5M environment interactions, demonstrating superior sample efficiency in a more data-constrained setting.

Ablations To further understand our method's performance, we conduct ablation studies to evaluate the individual contributions of the spatio-temporal positional encoding (STPE) and the optimal transport mechanism. Table 2 shows that both components make independent improvements to policy return and score, but using them together leads to the best result. Furthermore, for spatio-temporal positional encoding, including absolute spatial embeddings improves the performance compared to only using relative encoding (Relative PE only).

Table 2: Ablations on Craftax-classic with 1M interactions. † uses hyperparameters of TWISTED.

Method	Return (%)	Score (%)
Dedieu et al. (2025)†	68.55 ± 0.72	27.24 ± 0.86
Relative PE only	69.26 ± 0.50	29.37 ± 0.80
STPE only	71.85 ± 0.63	33.94 ± 1.10
Optimal transport only	69.77 ± 0.65	31.08 ± 0.88
TWISTED (ours)	**72.46** ± 0.45	**35.60** ± 0.92

Accuracy evaluation To assess the contribution of optimal transport to world model prediction, we measure the prediction accuracy with and without the transport mechanism. Our evaluation uses 10,000 environment transitions and counts how many next states are predicted perfectly (where every predicted token is correct). Table 3 shows that adding optimal transport improves accuracy. The transformer alone has particularly low accuracy in cases involving randomly moving creatures, which optimal transport helps with. By improving the accuracy of world model prediction, optimal transport decoding leads to higher quality imaginations and improved policy performance as seen in the ablations.

Qualitative analysis Figure 5 compares imaginations generated by our method vs. Dedieu et al. (2025). Our method excels in situations where tiles in the generated frame are correlated. For example, a creature in Craftax-classic can move to adjacent tiles, but it should only move to one destination tile and should not be duplicated to multiple destination tiles. However, because the transformer generates output tokens for a state in parallel, it cannot capture this constraint naturally. Therefore, during imagination, duplication or disappearance of creatures occurs, which is a critical defect of modeling environment dynamics. Optimal transport-based decoding eliminates this issue by capturing the appropriate constraint between output tiles. Solving this issue is particularly important because similar hallucinations arise in non-transformer world models as well (see Appendix A).

Table 3: Prediction accuracy on a dataset of 10,000 transitions. The first column reports overall accuracy, while the latter two break down accuracy based on whether the input state contains a randomly moving creature. Applying the optimal transport mechanism to the STPE-only transformer outputs increases accuracy by 3.39%. The transformer accuracy is especially low for transitions involving creatures, which optimal transport improves by 3.45%. † uses hyperparameters of TWISTED.

Method	Accuracy (%)	w/ creatures (%)	w/o creatures (%)
Dedieu et al. (2025)†	46.94	33.83	61.03
STPE only	47.74	34.92	62.13
TWISTED (ours)	**51.13**	**38.37**	**65.46**

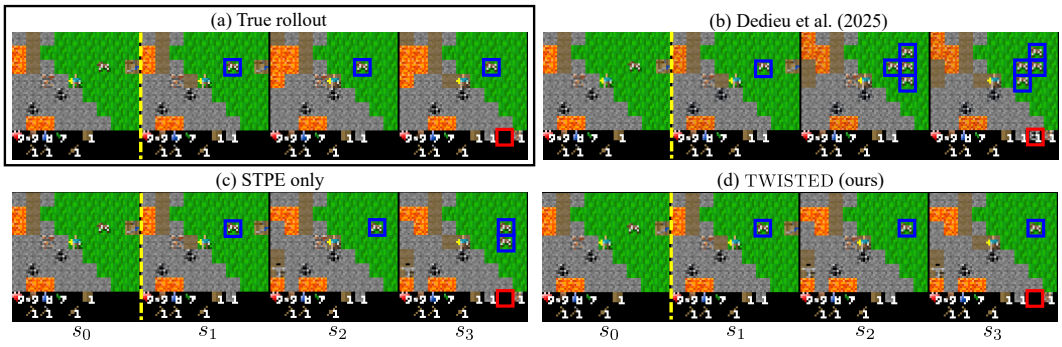

Figure 5: Comparison of imagined rollouts from different world models. (a) shows the ground-truth environment trajectory, while (b), (c), and (d) illustrate imagined rollouts generated by the baseline, the baseline with spatio-temporal positional encoding (STPE only), and TWISTED, respectively. All rollouts begin from the same initial state s_0 (left of the yellow dashed line). TWISTED fixes inaccurate dynamics (red boxes) and duplication errors (blue boxes) produced by the baseline.

4.2 CRAFTAX

We also evaluate our method on Craftax, a more complex and difficult environment that builds on Craftax-classic (Matthews et al., 2024). Craftax features a larger screen, more items, more enemies, and more levels compared to Craftax-classic (details in Appendix E). We compare against Simulus (Cohen et al., 2025) and Dedieu et al. (2025)[3], which set the previous best return and score, respectively. Table 4 reports return and score on Craftax, as the mean and standard error over 5 seeds. TWISTED achieves a return of 7.09% and a score of 2.40%, surpassing the baselines. These results demonstrate that TWISTED can generalize to more difficult environments.

Table 4: Results on Craftax after 1M environment interactions. Simulus does not report Score (—).

Method	Return (%)	Score (%)
Dedieu et al. (2025)	5.44 ± 0.25	1.53 ± 0.10
Simulus	6.59	—
TWISTED (ours)	$\mathbf{7.09} \pm 0.20$	$\mathbf{2.40} \pm 0.04$

4.3 MINATAR

To further validate the generalization performance of our approach, we also evaluate on the MinAtar benchmark (Young & Tian, 2019; Lange, 2022). MinAtar consists of 4 Atari games with simplified symbolic observations of size 10×10. We compare against the previous state of the art for model-based RL, Dedieu et al. (2025), and the recent model-free Artificial Dopamine (AD) agent (Guan et al., 2023). Each method is trained on each game in MinAtar for 1M environment steps (except AD uses 5M steps), using 10 seeds per game. Table 5 shows that TWISTED outperforms the

[3]We report the (fast) variant from version arXiv:2502.01591v1 of Dedieu et al. (2025).

Table 5: Returns on MinAtar after 1M environment interactions (or 5M for AD). Return is evaluated on 1,000 evaluation episodes at the end of training.

Method	Asterix	Breakout	Freeway	SpaceInvaders
AD (Guan et al., 2023)	21.05 ± 0.65	27.78 ± 0.16	57.68 ± 0.07	140.36 ± 1.70
Dedieu et al. (2025)	44.81 ± 3.54	93.92 ± 1.44	71.12 ± 0.13	186.16 ± 1.25
TWISTED (ours)	**50.04** ± 2.98	**99.53** ± 2.31	**71.34** ± 0.07	**188.85** ± 0.62

baselines in all 4 games. Return graphs for each game can be found in Appendix F. By improving in every game, TWISTED demonstrates that spatio-temporal encoding and optimal transport-based decoding confer robust benefits across a variety of environments.

5 Related works

Transformer world models Transformer architectures have been effectively utilized in model-based RL. The concept of transformer world models was first introduced by IRIS (Micheli et al., 2022). Building upon IRIS, Δ-IRIS proposed an agent architecture that encodes stochastic deltas between time steps, enhancing token efficiency by exploiting similarities between adjacent frames (Micheli et al., 2024). TWM, STORM, DART, and TWISTER also utilized the transformer architecture for world models, demonstrating its efficacy across different benchmarks (Robine et al., 2023; Zhang et al., 2023; Agarwal et al., 2024; Burchi & Timofte, 2025). Transformer world models further advanced with techniques including nearest neighbor tokenization and block teacher forcing, achieving state-of-the-art performance on Craftax-classic (Dedieu et al., 2025). Outside of transformers, other world models have used GRUs (DreamerV3), diffusion (DIAMOND), decoder-free latent spaces (TD-MPC2), and discrete codebook latent spaces (DC-MPC) (Hafner et al., 2023; Alonso et al., 2024; Hansen et al., 2024; Scannell et al., 2025).

Positional embeddings in video modeling Positional encoding for multi-dimensional information has been developed in the context of video modeling. The Qwen2-VL series introduced Multimodal Rotary Position Embedding (M-RoPE), decomposing positional embeddings into components capturing 1D textual and 3D video information (Wang et al., 2024). Complementing this, VideoRoPE proposed enhancements such as Low-frequency Temporal Allocation and Adjustable Temporal Spacing to improve video rotary position embeddings, demonstrating superior performance in video understanding tasks Wei et al. (2025).

Optimal transport in RL Optimal transport theory has been applied to RL in other contexts, specifically for curriculum and offline reinforcement learning. CurrOT framed curriculum generation as a constrained optimal transport problem between task distributions (Klink et al., 2022). GRADIENT formulated curriculum reinforcement learning as an optimal transport problem with a tailored distance metric between tasks (Huang et al., 2022). Additionally, Achievement Distillation introduced a contrastive learning method using optimal transport to enhance the discovery of hierarchical achievements, leading to improved sample efficiency (Moon et al., 2023).

6 Conclusion

In this paper, we present TWISTED, a transformer world model tailored for visual RL environments. TWISTED captures the inherent structure of visual environment inputs by encoding both spatial and temporal dimensions using a combination of absolute and relative positional encodings. By selectively reusing tokens from preceding frames with optimal transport-based decoding, it effectively leverages frame-to-frame similarities to model next-state tokens instead of solely relying on the transformer to regenerate each one. These innovations enable TWISTED to achieve new state-of-the-art performance on the challenging Craftax-classic, Craftax, and MinAtar benchmarks.

Reproducibility statement

For full reproducibility, all source code is included in the supplementary materials. The source code will also be released on GitHub upon acceptance. All implementation details are described in Appendix B for the world model and policy, Appendix C for spatio-temporal encoding, and Appendix D for optimal transport. All hyperparameters are listed in Appendix B for Craftax-Classic, Appendix E for Craftax, and Appendix F for MinAtar.

References

Pranav Agarwal, Sheldon Andrews, and Samira Ebrahimi Kahou. Learning to play atari in a world of tokens. In *ICML*, 2024.

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. In *NeurIPS*, 2024.

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

Thomas Brox, Andres Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, 2004.

Maxime Burchi and Radu Timofte. Learning transformer-based world models with contrastive predictive coding. In *ICLR*, 2025.

Lior Cohen, Kaixin Wang, Bingyi Kang, Uri Gadot, and Shie Mannor. Uncovering untapped potential in sample-efficient world model agents. *arXiv preprint arXiv:2502.11537*, 2025.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NeurIPS*, 2013.

Antoine Dedieu, Joseph Ortiz, Xinghua Lou, Carter Wendelken, J Swaroop Guntupalli, Wolfgang Lehrach, Miguel Lazaro-Gredilla, and Kevin Patrick Murphy. Improving transformer world models for data-efficient rl. In *ICML*, 2025.

Jonas Guan, Shon Eduard Verch, Claas Voelcker, Ethan C. Jackson, Nicolas Papernot, and William A. Cunningham. Temporal-difference learning using distributed error signals. In *NeurIPS*, 2023.

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

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

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *ICLR*, 2024.

Peide Huang, Mengdi Xu, Jiacheng Zhu, Laixi Shi, Fei Fang, and Ding Zhao. Curriculum reinforcement learning using optimal transport via gradual domain adaptation. In *NeurIPS*, 2022.

Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *ICML*, 2020.

Diederik Kingma and Jimmy Ba. Adam: a method for stochastic optimization". In *ICLR*, 2015.

Pascal Klink, Haoyi Yang, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. Curriculum reinforcement learning via constrained optimal transport. In *ICML*, 2022.

Robert Tjarko Lange. gymnax: A JAX-based reinforcement learning environment library, 2022. URL `http://github.com/RobertTLange/gymnax`.

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson, Samuel Coward, and Jakob Foerster. Craftax: a lightning-fast benchmark for open-ended reinforcement learning. In *ICML*, 2024.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. *arXiv preprint arXiv:2209.00588*, 2022.

Vincent Micheli, Eloi Alonso, and François Fleuret. Efficient world models with context-aware tokenization. In *ICML*, 2024.

Seungyong Moon, Junyoung Yeom, Bumsoo Park, and Hyun Oh Song. Discovering hierarchical achievements in reinforcement learning via contrastive learning. In *NeurIPS*, 2023.

Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016.

Gabriel Peyré and Marco Cuturi. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11:355–206, 01 2019. doi: 10.1561/2200000073.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan Harmeling. Transformer-based world models are happy with 100k interactions. *arXiv preprint arXiv:2303.07109*, 2023.

Aidan Scannell, Mohammadreza Nakhaei, Kalle Kujanpää, Yi Zhao, Kevin Sebastian Luck, Arno Solin, and Joni Pajarinen. Discrete codebook world models for continuous control. In *ICLR*, 2025.

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

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Richard Sutton and Andrew Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Sundar Vedula, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):475–480, 2005. doi: 10.1109/TPAMI.2005.63.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Pan Zhang, Yuhang Cao, Jian Tong, Haodong Duan, Qipeng Guo, Jiaqi Wang, et al. Videorope: What makes for good video rotary position embedding? *arXiv preprint arXiv:2502.05173*, 2025.

Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. Storm: Efficient stochastic transformer based world models for reinforcement learning. In *NeurIPS*, 2023.

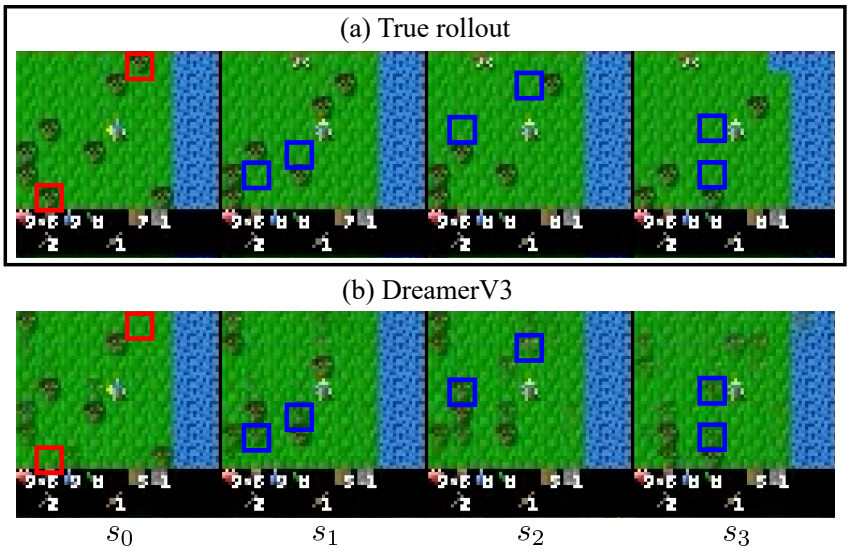

Figure 6: An imagination rollout of DreamerV3 compared to the ground-truth trajectory. DreamerV3's imagination includes disappearance of trees (red boxes) and duplication of trees (blue boxes) over time, similar to duplication issues shown in Figure 5 for Dedieu et al. (2025)

A Additional qualitative analysis

The qualitative analysis in Section 4.1 discusses how parallel decoding in baseline transformer world models leads to duplication and disappearance of objects during imagination. However, this issue is not isolated to transformers and can also be observed in non-transformer world models, like DreamerV3 (Hafner et al., 2023). Figure 6 shows duplication and disappearance artifacts in an imagination rollout generated by DreamerV3, demonstrating that this issue occurs across different types of world models. Thus, by eliminating these hallucinations, optimal transport-based decoding resolves a problem that is widespread among world models.

B Agent training and implementation

B.1 Training loop

This section outlines the training procedure for the world model and the policy, which are trained concurrently through alternating update steps. The overall training loop is composed of the following steps:

1. **Environment interaction:** Execute the current policy in the real environment and store the resulting experiences in a replay buffer.
2. **Policy update on real data:** Update the policy using the most recent real environment experiences collected in Step 1. The policy is trained on the data over E_{env} epochs, with each batch split into B_{policy} minibatches due to memory constraints.
3. **Tokenizer training:** Sample experiences from the replay buffer to train the nearest neighbor tokenizer. The tokenizer is updated on $U_{\text{tokenizer}}$ batches of sample trajectories.
4. **World model training:** Sample experiences from the replay buffer to train the transformer world model. The world model is updated on U_{WM} batches of sample trajectories, using B_{WM} minibatches.
5. **Policy update in imagination:** For training steps $t > T_{\text{warmup}}$, generate U_{imag} batches of imagined trajectories using the world model and the current policy, and update the policy on these synthetic rollouts. During the initial T_{warmup} real environment interactions, this step is skipped to allow the world model to reach sufficient accuracy before generating

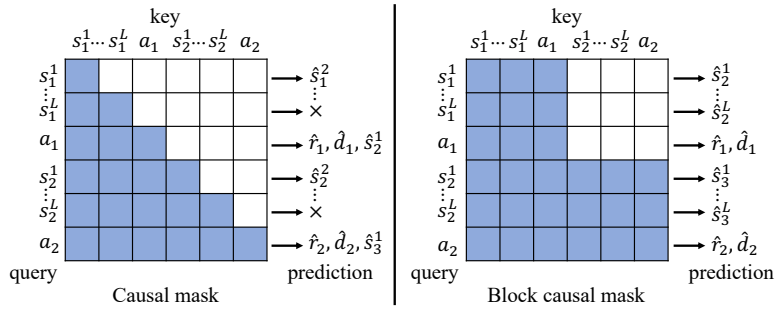

Figure 7: Comparison between the causal attention mask and the block causal attention mask. The token s_t^i denotes the i-th state token at timestep t, a_t denotes the action, $\hat{r}_t$ denotes the predicted reward, and $\hat{d}_t$ denotes the predicted done signal. Only two state tokens are shown per state for simplicity. (left) In the causal mask, each token attends to the tokens preceding it. The output embeddings of state token s_t^i are used to predict the subsequent state token s_t^{i+1}. The reward $\hat{r}_t$ and done signal $\hat{d}_t$ are predicted from a_t, and the output of s_t^L is unused. (right) In the block causal mask, all state and action tokens in the same timestep attend to each other, and they are used to predict the corresponding token in the next timestep (a_t predicts $\hat{r}_t$ and $\hat{d}_t$). This allows each frame to be predicted in parallel rather than token-by-token.

imaginations. At the start of each imagination rollout, the policy uses T_{burn} frames from the replay buffer to initialize its RNN hidden state.

The overall training loop is repeated until the agent has performed a total of T_{total} real environment interactions.

B.2 World model architecture and loss

Our transformer world model follows the GPT-2 architecture (Radford et al., 2019). The model operates over tokenized sequences that encode states and actions over T consecutive frames. These tokens are first mapped to 128-dimensional embeddings via a learned embedding layer. Absolute positional embeddings are then added, followed by an initial dropout layer. The resulting embeddings are processed through a stack of three transformer blocks. Each block consists of the following components:

1. Layer normalization
2. Multi-head attention module, comprising:
   (a) Self-attention with a block causal mask. In the block causal mask, tokens within the same timestep are decoded in parallel (see Figure 7) (Dedieu et al., 2025).
   (b) A linear projection to the 128-dimensional embedding space
   (c) Dropout
3. Residual connection with the block input
4. Layer normalization
5. Feed-forward multilayer perceptron (MLP) composed of:
   (a) A hidden layer of dimension 512
   (b) GeLU activation
   (c) Dropout

After processing through the final block, the output undergoes a final layer normalization and is then passed to three separate prediction heads: one for the next state tokens, one for the reward signal, and one for the done signal. We denote the output embeddings as

$$(E_1^1, \ldots, E_1^{L+1}, E_2^1, \ldots, E_2^{L+1}, \ldots, E_T^1, \ldots, E_T^{L+1}).$$

where L represents the number of state tokens per frame, and E_t^i corresponds to the i-th output embedding at timestep t. These embeddings are routed to prediction heads as follows:

1. For $i \leq L$, the embedding E_t^i is input to the observation head, an MLP comprising a 128-dimensional linear layer, a ReLU activation, and a final linear layer projecting to the codebook size K. The output logits define a categorical distribution over the K possible values of the predicted state token s_{t+1}^i.

2. The embedding E_t^{L+1}, corresponding to the position of the action token, is passed to both the reward and done heads. Each head is an MLP consisting of a 128-dimensional linear layer, a ReLU activation, and a final linear layer projecting to two output classes. Although the Craftax-classic environment defines reward values of -0.1, 0.1, and 1.0, we follow Dedieu et al. (2025) and binarize the reward signal to improve stability, ignoring the -0.1 and 0.1 cases.

The model is trained on trajectories of length T_{WM} sampled from the replay buffer. The total loss is the sum of three components:

1. Cross-entropy loss over next-state token predictions (across K classes).

2. Cross-entropy loss for binary reward classification (0 or 1).

3. Cross-entropy loss for done signal prediction.

Optimization is performed using the Adam optimizer with gradient norm clipping to stabilize training (Kingma & Ba, 2015). Hyperparameters for architecture and training are provided in Table 6.

Table 6: World model hyperparameters. Sweep range indicates the values tried per hyperparameter, with the final Value being chosen based on highest return.

Area	Hyperparameter	Value	Sweep range
Architecture	Sequence length T_{WM}	20	
	State tokens per frame L	81	
	Number of blocks	3	
	Number of attention heads	8	
	Embedding dimension	128	
	MLP hidden layer dimension	512	
	Dropout rate	0.1	
	Attention mask	Block causal	
	Inference with key-value caching	True	
Optimal transport	Distance cost coefficient c_d	0.6	16 values in [0, 1]
	Wildcard cost c_w	0.3	9 values in [0.2, 1]
	Sinkhorn regularization parameter ϵ	0.00001	{0.00001, 0.0001}
Training	Number of updates U_{WM}	500	
	Number of minibatches B_{WM}	3	
	Replay buffer size	128,000	
Optimization	Optimizer	Adam	
	Learning rate	0.001	
	Max norm for gradient clipping	0.5	
Tokenizer	Codebook size K	4096	
	Single patch shape	$7 \times 7 \times 3$	
	New code threshold τ	0.75	
	Number of updates $U_{\text{tokenizer}}$	25	

Table 7: Policy hyperparameters. Sweep range indicates the values tried per hyperparameter, with the final Value being chosen based on highest return.

Area	Hyperparameter	Value	Sweep range
Environment	Environment interactions T_{total}	1,000,000	
	Warmup interactions T_{warmup}	50,000	{50k, 100k, 200k}
	Number of environments (batch size)	48	
	Rollout horizon in environment	96	
	Rollout horizon in imagination T_{WM}	20	
	Burn-in horizon for RNN in imagination T_{burn}	5	
Training	Number of updates in imagination U_{imag}	300	{150, 300, 600, 1200}
	Number of epochs in environment E_{env}	4	
	Number of epochs in imagination	1	
	Number of minibatches in environment B_{policy}	8	
	Number of minibatches in imagination	1	
PPO	Discount factor γ	0.925	
	TD weight λ	0.625	
	Clipping value ϵ	0.2	
	TD loss coefficient λ_{TD}	2.0	
	Entropy loss coefficient λ_{ent}	0.01	
	PPO target discount factor α	0.95	
Optimization	Optimizer	Adam	
	Learning rate	0.00045	
	Max norm for gradient clipping	0.5	

B.3 Policy network architecture and loss

B.3.1 Architecture

We adopt the policy network architecture introduced in Dedieu et al. (2025), which comprises three primary components: a convolutional encoder, a recurrent neural network (RNN), and separate MLP heads for action and value prediction.

The convolutional encoder consists of three convolutional blocks with channel sizes [64, 64, 128]. Each block contains an instance normalization layer, a 3×3 convolutional layer with stride 1, a 3×3 max-pooling layer with stride 2, and two ResNet-style sub-blocks. Each ResNet block includes a ReLU activation, instance normalization, a 3×3 convolution with stride 1, and a skip connection to preserve the input. The encoder produces an output of shape $8 \times 8 \times 128$, which is flattened into a 8192-dimensional vector, denoted by z. The vector z is then projected into a 256-dimensional representation through a ReLU activation, a linear layer, and layer normalization. This projected representation serves as input to a GRU recurrent module, which outputs a vector $y \in \mathbb{R}^{256}$ along with the updated hidden state $h \in \mathbb{R}^{256}$.

The action and value heads share an identical structure except for the final output projection. Each head takes the concatenated vector $[z, y]$ as input and applies a sequence of transformations: ReLU activation, layer normalization, a linear projection to 2048, another ReLU activation, and a residual block composed of two linear layers with ReLU activations. The output is passed through a final layer normalization, followed by the task-specific output projection—either to action logits or a scalar value estimate.

B.3.2 Training

We follow the policy training procedure described in Dedieu et al. (2025), using Proximal Policy Optimization (PPO) (Schulman et al., 2017) as the underlying policy gradient algorithm.

Let the trajectory be denoted as $\tau = (o_{1:T+1}, a_{1:T}, r_{1:T}, d_{1:T}, h_{0:T})$, where o_t represents the observations, a_t the actions, r_t the rewards, d_t the done signals, and h_t the hidden states of the RNN. At each timestep, PPO computes the value estimates $v_{1:T+1} = V_{\Phi_{\text{old}}}(o_{1:T+1})$ and the action probabili-

ties $\pi_{\Phi_{\text{old}}}(a_t|o_t)$ under the current fixed parameters Φ_{old}. The policy is optimized by minimizing the following PPO objective:

$$\mathcal{L}_{\text{PPO}}(\Phi) = \frac{1}{T}\sum_{t=1}^{T}\Big\{ -\min\left(p_t(\Phi)A_t, \text{clip}(p_t(\Phi))A_t\right) + \lambda_{\text{TD}}(V_\Phi(o_t) - q_t)^2 - \lambda_{\text{ent}}\mathcal{H}\left(\pi_\Phi(.|o_t)\right)\Big\}$$

where $p_t(\Phi)$ is the probability ratio $\frac{\pi_\Phi(a_t|o_t)}{\pi_{\Phi_{\text{old}}}(a_t|o_t)}$ and clip(x) is the clipping function $\min(\max(x, 1-\epsilon), 1+\epsilon)$. Here, A_t denotes a generalized advantage estimation, q_t is a temporal difference (TD) target, and $\mathcal{H}$ is the entropy operator. The advantages A_t and targets q_t are computed as

$$A_t = \delta_t + (1 - \text{done}_t)\gamma\lambda A_{t+1},$$
$$q_t = A_t + v_t,$$

where $\delta_t = r_t + (1 - \text{done}_t)\gamma v_{t+1} - v_t$.

We incorporate two modifications to the standard PPO implementation:

- Generalized advantage estimates A_t are standardized across training batches to stabilize learning.
- We track the moving average of the mean and standard deviation of q_t, with discount factor α, and train the value function to predict the standardized targets.

All policy training hyperparameters are shown in Table 7.

C Relative spatio-temporal encoding implementation

Our implementation of relative spatio-temporal encoding extends Rotary Position Embedding (RoPE) by adapting it to both spatial and temporal contexts (Su et al., 2024). While RoPE rotates pairs of embedding dimensions using frequencies based on a 1D position index, our method modulates the rotation amount based on three indices, two spatial and one temporal. Our method divides dimension pairs in a 3:1 ratio between spatial and temporal encoding, following the design principle of VideoRoPE (Wei et al., 2025). Pairs associated with lower rotation frequencies are used for temporal encoding and are rotated based on the temporal index. In contrast, pairs with higher rotation frequencies are used for spatial encoding. Given the 2D nature of spatial positions, spatial pairs are further split evenly between the horizontal and vertical axes. These are interleaved across the embedding dimension to ensure balanced representation. As a result, the axes contributing to rotation follow the pattern $(x, y, x, y, \ldots, x, y, t, t, \ldots, t)$, ordered by decreasing rotation frequency.

As action tokens lack inherent spatial coordinates, assigning them fixed spatial positions would limit the effectiveness of relative positional encoding across the majority of embedding dimensions. To address this, spatial coordinates for action tokens are defined along the diagonal, (t, t), where t represents the temporal index. State tokens are assigned spatial coordinates offset from this diagonal, $(x + t, y + t)$, ensuring temporal alignment with action tokens while preserving spatial variation.

To avoid positional collisions between state and action tokens, they are given different temporal indices. That is, the state and action tokens $s_t^1, \ldots s_t^L, a_t, s_{t+1}^1, \ldots, s_{t+1}^L, a_{t+1}$ are given temporal indices $2t, \ldots, 2t, 2t+1, 2(t+1), \ldots 2(t+1), 2(t+1)+1$. This staggered assignment ensures that each token occupies a unique spatio-temporal location, maintaining positional distinctiveness throughout the sequence.

D Optimal transport implementation

This section details various components of our optimal transport implementation. Algorithm 1 characterizes the end-to-end decoding process. Algorithm 2 describes the Sinkhorn algorithm (Cuturi, 2013). Algorithm 3 describes the process of converting the transport plan's continuous values into binary values. It is adapted from Kim et al. (2020).

Algorithm 1 Decoding with optimal transport

Input: transformer prediction $\mathbf{p}$,
previous tokens $\mathbf{u}$,
number of tokens per frame L,
Sinkhorn regularization parameter ϵ,
Number of Sinkhorn iterations T
Output: Generated tokens for next frame $\mathbf{u}'$

Compute $\boldsymbol{A}^{(prev)}$, $\boldsymbol{A}^{(gen)}$ from Equations 1 and 2

$\boldsymbol{A} = \text{concatenate} \begin{pmatrix} \boldsymbol{A}^{(prev)} & \mathbf{0} \in \mathbb{R}^{L \times L} \\ \boldsymbol{A}^{(gen)} & \mathbf{0} \in \mathbb{R}^{L \times L} \end{pmatrix} \in \mathbb{R}^{(2L) \times (2L)}$ ▷ Concatenate for Sinkhorn input
$\boldsymbol{P} = \text{SINKHORN}(-\boldsymbol{A}, \epsilon, T)$
$\boldsymbol{P}^{(prev)} = \boldsymbol{P}_{1:L,1:L}$
$\boldsymbol{P}^{(gen)} = \boldsymbol{P}_{L+1:2L,1:L}$

$\boldsymbol{\Pi}^{(prev)}, \boldsymbol{\Pi}^{(gen)} = \text{BINARIZATION}(\boldsymbol{P}^{(prev)}, \boldsymbol{P}^{(gen)})$

for $j = 0$ **to** $L - 1$ **do** ▷ Finalize the state prediction
  if $\Pi_{ij}^{(prev)} = 1$ for some i **then**
    $\mathbf{u}'_j = \mathbf{u}_i$ ▷ Take a token from the previous state
  else if $\Pi_{jj}^{(gen)} = 1$ **then**
    $\mathbf{u}'_j = \text{sample}(\mathbf{p}_j)$ ▷ Use a generated token from the transformer
  end if
end for
return $\mathbf{u}'$

Algorithm 2 SINKHORN implementation

Input: Cost matrix $\boldsymbol{C}$,
Sinkhorn regularization parameter ϵ,
Number of Sinkhorn iterations T
Output: Optimal transport plan $\boldsymbol{P}$

$\boldsymbol{K} = \exp(\boldsymbol{C}/\epsilon)$
Set uniform marginals: $\mathbf{r} = \frac{1}{\text{rows}(\boldsymbol{C})}$, $\mathbf{c} = \frac{1}{\text{cols}(\boldsymbol{C})}$
Initialize dual variables: $\mathbf{u} = \mathbf{1}$, $\mathbf{v} = \mathbf{1}$
for $t = 1$ **to** T **do**
  $\mathbf{u} = \mathbf{r} \oslash (\boldsymbol{K}\mathbf{v})$ ▷ $\oslash$ denotes element-wise division
  $\mathbf{v} = \mathbf{c} \oslash (\boldsymbol{K}^\top \mathbf{u})$
end for
return $\boldsymbol{P} = \text{diag}(\mathbf{u}) \cdot \boldsymbol{K} \cdot \text{diag}(\mathbf{v})$

Distance cost For the affinity matrix $\boldsymbol{A}^{(prev)}$ in Equation 1, we include a distance penalty $D((x_i, y_i), (x_j, y_j))$. This choice of distance cost can be adapted based on the environment. For our environments, we impose a movement constraint by defining the distance cost as

$$D((x_i, y_i), (x_j, y_j)) = \begin{cases} d, & \text{if } d \leq 4 \\ +\infty & \text{otherwise,} \end{cases}$$
$$\text{where } d = \|(x_i, y_i) - (x_j, y_j)\|_2^2.$$

This constraint reflects the fact that in environments like Craftax-classic, Craftax, and MinAtar, a token's spatial displacement between consecutive frames is limited to a maximum of two positions in any direction. For Craftax-classic and Craftax, this accounts for the potential movement of one by the player and one by a creature token, if applicable.

Algorithm 3 BINARIZATION of partial transport plan

Input: Partial transport plans $\boldsymbol{P}^{(prev)}$, $\boldsymbol{P}^{(gen)}$,
large value v
Output: Binarized transport plans $\mathbf{\Pi}^{(prev)}$, $\mathbf{\Pi}^{(gen)}$

$\boldsymbol{P}^{\text{in}}$ = concatenate $\left(\boldsymbol{P}^{(prev)}, \boldsymbol{P}^{(gen)}\right)$
Initialize $\boldsymbol{P}^{(0)} = \boldsymbol{P}^{\text{in}},\ t = 0$
repeat
  $\text{target} = \text{argmax}(\boldsymbol{P}^{(t)}, \text{dim} = 1)$
  $\mathbf{\Pi}^{\mathbf{initial}} = \mathbf{0}_{n \times m}$
  for $i = 0$ **to** $n - 1$ **do**
    $\mathbf{\Pi}^{\mathbf{initial}}[i, \text{target}[i]] = 1$
  end for
  $\boldsymbol{C} = \boldsymbol{P}^{(t)} \odot \mathbf{\Pi}^{\mathbf{initial}} - v(1 - \mathbf{\Pi}^{\mathbf{initial}})$
  $\text{source} = \text{argmax}(\boldsymbol{C}, \text{dim} = 0)$
  $\mathbf{\Pi}^{\text{out}} = \mathbf{0}_{n \times m}$
  for $j = 0$ **to** $m - 1$ **do**
    $\mathbf{\Pi}^{\text{out}}[\text{source}[j], j] = 1$
  end for
  $\mathbf{\Pi}^{\text{out}} = \mathbf{\Pi}^{\text{out}} \odot \mathbf{\Pi}^{\mathbf{initial}}$
  $\boldsymbol{R} = (1 - \mathbf{\Pi}^{\text{out}}) \odot \mathbf{\Pi}^{\mathbf{initial}}$
  $\boldsymbol{P}^{(t+1)} = \boldsymbol{P}^{(t)} - v\boldsymbol{R}$
  $t = t + 1$
until $\mathbf{\Pi}^{\text{out}} = \mathbf{\Pi}^{\text{initial}}$
$\mathbf{\Pi}^{(prev)}$ = $\mathbf{\Pi}^{\text{out}}[1 : L, 1 : L]$
$\mathbf{\Pi}^{(gen)}$ = $\mathbf{\Pi}^{\text{out}}[L + 1 : 2L, 1 : L]$
return $\mathbf{\Pi}^{(prev)}$, $\mathbf{\Pi}^{(gen)}$

Choosing between transformer and optimal transport output Optimal transport provides an effective mechanism for reusing tokens from the previous frame. However, it is less effective in scenarios where novel tokens must be introduced, such as when the agent moves to a previously unexplored area. In such cases, optimal transport may fail to consistently route wildcard entries to the appropriate newly generated tokens. Conversely, the transformer world model is capable of freely generating new tokens as needed, but lacks a mechanism for directly reusing tokens from prior frames. Rather than committing to a single output modality, we adopt a hybrid strategy for Craftax-classic and Craftax that selects between optimal transport and transformer outputs based on spatial position. In Craftax-classic and Craftax, new visual content appears along the screen boundaries as the player explores previously unseen regions. Additionally, the inventory interface—fixed at the bottom of the screen—requires updates to token values without positional shifts. To accommodate these patterns, we apply the optimal transport output to the central region of the screen, where token reuse is most appropriate, while using the transformer's predictions for the screen edges and inventory regions, where new content is more likely. For MinAtar, which does not have special behavior at the edges, the optimal transport output is used directly for the entire screen.

Table 8: Hyperparameter differences between Craftax-classic and Craftax.

Area	Hyperparameter	Craftax-classic	Craftax
Environment	Observation shape	$63 \times 63 \times 3$	$130 \times 110 \times 3$
	Number of possible actions	17	43
	Number of possible achievements	22	226
	Number of environments (batch size)	48	16
Tokenizer	Single patch shape	$7 \times 7 \times 3$	$10 \times 10 \times 3$
Architecture	State tokens per frame L	81	143
Training	Replay buffer size	$128,000$	$48,000$

Figure 8: An example observation of Craftax.

E CRAFTAX DESCRIPTION AND HYPERPARAMETERS

Craftax is a complex environment inspired by Craftax-classic, featuring a larger screen, more enemies, more items, and multiple underground levels. An example observation is shown in Figure 8. Following Dedieu et al. (2025), we change some hyperparameters for Craftax, as shown in Table 8. In particular, to accommodate the larger screen and additional tokens in memory, the batch size and replay buffer size are reduced.

F MINATAR RETURN CURVES AND HYPERPARAMETERS

Figure 9 shows the return curves of TWISTED and baseline Dedieu et al. (2025) for each game in MinAtar. TWISTED outperforms the baseline in every game. Table 9 lists the hyperparameters for MinAtar with different values compared to Craftax-classic. All hyperparameter changes follow Dedieu et al. (2025), except the hyperparameters specific to optimal transport. Also following Dedieu et al. (2025), the policy encoder uses layer normalization and the Swish activation function, and actor and value networks share weights except in their final linear layers (Ba et al., 2016; Ramachandran et al., 2017).

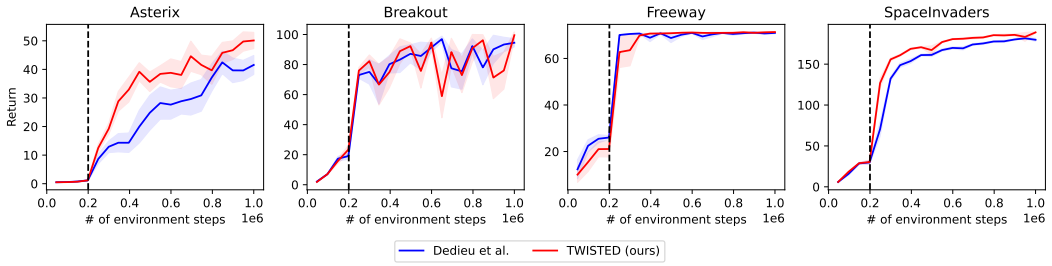

Figure 9: Return curves for MinAtar. Shading indicates standard error among multiple seeds. The vertical dashed lines indicate the start of training in imagination after T_{warmup} interactions.

Table 9: Hyperparameter differences between Craftax-classic and MinAtar. K is the number of object types for each game (4 for Asterix, 4 for Breakout, 7 for Freeway, and 6 for SpaceInvaders). The number of actions A is 5 for Asterix, 3 for Breakout, 3 for Freeway, and 4 for SpaceInvaders.

Area	Hyperparameter	Craftax-classic	MinAtar
Environment	Observation shape	$63 \times 63 \times 3$	$10 \times 10 \times K$
	Number of possible actions	17	A
	Warmup interactions T_{warmup}	50,000	200,000
Tokenizer	Single patch shape	$7 \times 7 \times 3$	$2 \times 2 \times K$
Architecture	State tokens per frame L	81	25
Optimal transport	Distance cost coefficient c_d	0.6	0.2
	Wildcard cost c_w	0.3	0.05
Training	Number of world model updates U_{WM}	500	2000
	Number of policy updates in imagination U_{imag}	300	2000
	Coefficient for reward prediction loss	1	10
	Coefficient for done prediction loss	1	10
PPO	Discount factor γ	0.925	0.95
	TD weight λ	0.625	0.75
	Entropy loss coefficient λ_{ent} in imagination	0.01	0.05
	PPO target discount factor α	0.95	0.925

Table 10: Running times on a single Nvidia RTX 3090 GPU. WM training measures one epoch of world model training. Imagination measures one epoch of policy training in imagination. Total time represents end-to-end training time for 1M environment steps.

Method	WM training (s)	Imagination (s)	Total time (hrs)
Baseline	8.08	4.48	46.3
TWISTED (ours)	8.19	7.69	57.7

G COMPUTE TIME

Table 10 reports the running time of TWISTED and its baseline Dedieu et al. (2025) using the same hyperparameters. Spatio-temporal positional encoding introduces neglible overhead, increasing world model training time by only 1%. Optimal transport-based decoding increases latency for policy training in imagination. Since imagination accounts for only one-third of total training time, the overall end-to-end training time increases by 25%.