TADM: Transformer-based Any-step Dynamics Model

Tianyou Li

Abstract: We introduce the Transformer-based Any-step Dynamics Model (TADM), which combines the flexible, variable-length backtracking of Any-step Dynamics Models (ADM) with a non-recurrent Transformer encoder to improve long-horizon state prediction in model-based reinforcement learning. By drawing an any-step index and attending over an anchor state plus its intervening action subsequence, TADM avoids the compounding errors inherent in step-wise bootstrapping and eliminates the need for recurrent backpropagation. We give a brand new potential theoretical upper error bound—showing this model greatly reduces the worst-case per-step next-state prediction error from kL_s to L_s—and demonstrate that TADM achieves both faster convergence and lower negative log-likelihood on held-out dynamics data. In online experiments on the Hopper-v5 benchmark, TADM surpasses ADM-GRU and hybrid GRU+attention baselines, reaching a return of 2000 in 44 k environment steps and yielding over twice the final performance of ADM-GRU. These results highlight the promise of integrating any-step prediction with Transformer architectures for more accurate and sample-efficient world modeling.

Keywords: Model-based Reinforcement Learning, World Models, Transformer

1 Introduction

Model-based reinforcement learning (MBRL) has shown impressive performance improvements in both online and offline settings by improving sample efficiency [1], which is the *pain point* in applying RL algorithms in robotics. The idea of MBRL originates from an analogy to human intelligence. Human beings have the ability to predict how things could happen by imagining actions taken within an imagined world. By leveraging this world, better actions can be determined by targeting the action sets that achieve the lowest cost in the imagined world, thus reducing trial-and-error costs. The phrase "model" in MBRL refers to the environment model (often called the World Model [2]), which is expected to serve the same function as the imagined world in the human mind.

The most critical idea in MBRL, the model refers to the abstraction of the dynamics of the environment, with which the learning agent interacts [1]. Thus, under the Markovian property of the environment, learning the model corresponds to recovering the mapping from the current state-action pair to the next state and the reward of the next state. However, in many cases, the reward function is explicitly defined, thus the major task of model learning is to learn the state transition dynamics. With a well-trained model, one can easily create numerous trajectories to optimize the policy without letting the agent interact with the real environment, which is time-consuming and computationally expensive.

To propose high-fidelity dynamics models, many efforts have been devoted by previous researchers, such as adversarial models [3, 4], causal models [5], and ensemble dynamics models [6, 7], which are employed by the majority of MBRL algorithms. Although these algorithms greatly improved the model precision in short rollouts, it is still challenging to predict the transition dynamics in long-horizon rollouts due to the inevitable bootstrapping prediction. However, one could find that the majority of the models require the history of states and actions up to the current step, which should be sufficient for predicting the next state under the Markovian property assumption [8]. The

common idea shared by the explanations made by the authors proposing this kind of model is that the system dynamics are determined by a Markov Decision Process (MDP), but the agent can only directly observe part of the underlying state, therefore, the prediction of s_{t+1} can leverage earlier information s_t, a_t.

Inspired by the Any-step Dynamics Model (ADM) framework proposed in [9] and the Transformer State-Space Model (TSSM) proposed in [10], this work proposes a combined version of these two methods, Transformer-based Any-step Dynamics (TADM), which integrates the any-step prediction that mitigates compounding errors in model rollouts by enabling flexible, variable-length backtracking for direct state prediction and the independence of TSSM on relying on recurrent neural networks (RNNs) to model sequential action-state dependencies. This architectural innovation eliminates the need for bootstrapping and ensemble models, offering improved predictive fidelity and more efficient uncertainty estimation.

The proposed TADM framework offers enhanced long-horizon prediction accuracy and reduced compounding error in model rollouts through a more direct relation-capturing ability brought by the attention mechanism. Due to the limited time and computation power, this work demonstrates the effectiveness of this approach only in online RL benchmarks with modified network architectures, showing consistent improvements over the original ADM framework.

2 Related Work

2.1 ANY-STEP DYNAMICS MODEL (ADM)

In the regime of sequence models, the Any-step Dynamics Model (ADM) framework proposed in [9] provides a brand new framework allowing the model to predict the next state with an earlier state s_{t-k+1} and the action sequence along the trajectory $a_{t-k+1:t}$ corresponding to any integer k within a specified range m. This flexible trajectory window helps reduce dependency on step-wise bootstrapping. However, ADM still relies on recurrent architectures to encode the state-action trajectory, which may limit scalability and parallelism.

2.2 TRANSFORMER STATE SPACE MODEL (TSSM)

Recent work on Transformer State Space Models (TSSM) [10] explores the application of Transformer architectures to model state-action sequences. Unlike RNN-based approaches, Transformers can attend to all previous states and actions simultaneously, enabling more effective modeling of long-horizon dependencies and better scalability. TSSM demonstrates that Transformers when combined with state-space structure, outperform traditional recurrent methods in sequential prediction tasks. However, since this architecture is modified from the Recurrent State Space Model (RSSM) proposed in [11], it still employs all the state-action pairs along the trajectory, which is demonstrated to be harmful for the long-horizon prediction [9].

This work builds on both ADM and TSSM. Specifically, this work proposes TADM, a Transformer-based Any-step Dynamics Model, which leverages the transformer encoder structure from [10] to alleviate the long-horizon uncertainty and information loss inherent in the RNN-based design of ADM. Meanwhile, by adopting the any-step prediction framework from [9]-predicting s_{t+1} using an earlier state s_{t+1-k} and the action sequence $(a_{t+1-k}, \ldots, a_t)$ for any integer k within a specified range-TADM mitigates the compounding error caused by the bootstrapped prediction mechanism typically used in transformer-based rollouts. This design enables more accurate long-horizon dynamics modeling without relying on ensembles or recurrent architectures.

3 Preliminaries

3.1 Markov Decision Process from Multi-step Perspective

Continuing with the notation used in [9], we consider a continuous-state and continuous-action MDP specified by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, \rho_0, \gamma)$, where S, A are the state-action space, $T(s_{t+1}, r_{t+1}|s_t, a_t) : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the underlying transition kernel that gives the distribution of next state and reward s_{t+1}, r_{t+1} given the current state-action pair (s_t, a_t), ρ_0 is the initial state distribution, and γ is the discount factor. We define ρ^π as the occupancy measure of state-action pair (s_t, a_t) over the states induced by the transition kernel T and the policy π. In a multi-step view, by marginalizing the trajectory from previous state s_{t-k+1} and the action sequence $a_{t-k+1:t}$ induced by policy π, this kind of relationship can be expressed by the k-step transition model [9]

$$T^k(s_{t+1}, r_{t+1}|s_{t-k+1}, a_{t-k+1:t}) = \sum_{(s_{t-k+2:t}, r_{t-k+2:t}) \in \mathcal{S}^{k-1} \times \mathbb{R}^{k-1}} \prod_{i=0}^{k-1} T(s_{t-i+1}, r_{t-i+1}|s_{t-i}, a_{t-i}) \tag{1}$$

Similarly, we define the inverse distribution induced by transition kernel T and the policy π over $(s_{t-k+1:t}, r_{t-k+1:t})$ given the agent has reached state s_{t+1} as $\Gamma^k_\pi(s_{t-k+1:t}, r_{t-k+1:t}|s_{t+1})$.

To formulate the problem for k-step prediction, the occupancy measure of state-action pair (s_t, a_t) is given as $\rho^\pi(s, a) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t P_{\pi,t}(s, a)$, where $P_{\pi,t}(s, a)$ is the probability of policy π reaches state s and choose action a after time t. Then, by utilizing this term, we have the RL's goal

$$\max \sum_s \sum_a \left[\sum_t \gamma^t \cdot r(s, a) \cdot P_{\pi,t}(s, a) \right] = \max \mathbb{E}_{\rho^\pi} \left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \right] \tag{2}$$

According to [9], such a policy can be achieved by estimating the state-action value function $Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, r_{t+1} \sim T(\cdot|s_t, a_t)} [r_{t+1} + \gamma V^\pi(s_{t+1})]$, where $V^\pi(s_{t+1})$ is the expectation $Q^\pi(s_t, a_t)$ of all the action $a_{t+1} \sim \pi(\cdot|s_{t+1})$.

3.2 Transformer Attention Mechanism

The Transformer encoder [12] relies on self-attention to model sequence dependencies in a single layer. Given an input token sequence $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{n \times d}$, we compute three projections:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d \times d_k}$. The attention output is

$$\text{Attention}(Q, K, V) = \text{softmax}\Big(\frac{QK^\top}{\sqrt{d_k}}\Big) V$$

enabling each token to aggregate information from all others weighted by similarity in the d_k-dimensional query–key space. To enrich representational capacity, *multi-head attention* splits each of Q, K, V into h subspaces (heads), applies attention in parallel, and concatenates the results. Formally,

$$\text{MultiHead}(X) = \Big[\text{Attention}(Q_i, K_i, V_i)\Big]_{i=1}^{h} W_O$$

where $W_O \in \mathbb{R}^{hd_k \times d}$ projects back to dimension d. Since pure self-attention lacks any notion of sequence order, we add a fixed *positional encoding* to the inputs [12]:

$$\text{PE}_{(pos,2i)} = \sin\Big(\frac{pos}{10000^{2i/d}}\Big), \quad \text{PE}_{(pos,2i+1)} = \cos\Big(\frac{pos}{10000^{2i/d}}\Big)$$

These sinusoidal signals allow the model to distinguish token positions and generalize to longer sequences.

4 Method

We introduce the **Transformer Any-step Dynamics Model** (TADM), an architecture that couples the *any-step backtracking* principle of ADM [9] with a *non-recurrent* Transformer encoder, illustrated in Figure 1. TADM preserves ADM's ability to predict the next state from an *earlier* anchor state and the intervening action sequence, while reducing the compounding-error amplification that arises from repeatedly feeding model-generated states into a recurrent network. In what follows we (i) formalise the model definition, (ii) give the training objective, (iii) describe the online imagination procedure, and (iv) derive an upper bound on the prediction error growth.

4.1 Any-step Input Construction

At time step t we draw an *any-step index* $k \sim \mathcal{U}\{1, \dots, m\}$ and form the token sequence

$$\mathbf{z}_i = \mathrm{Embed}\big([\, s_{t-k+1};\, a_{t-k+i-1}\,]\big) \quad \in \mathbb{R}^d, \qquad i = 1, \dots, k+1,$$

where $\mathbf{z}_1$ encodes the *anchor* state s_{t-k+1} and the remaining tokens encode the action subsequence $a_{t-k+1:t}$. We add a fixed sinusoidal positional encoding $\mathbf{p}_i \in \mathbb{R}^d$ and obtain the input matrix $X = [\mathbf{z}_1 + \mathbf{p}_1, \dots, \mathbf{z}_{k+1} + \mathbf{p}_{k+1}] \in \mathbb{R}^{(k+1)\times d}$.

4.2 Sequence Encoder

TADM replaces ADM's GRU with an L-layer *Transformer encoder* $\mathcal{T}_\theta : \mathbb{R}^{(k+1)\times d} \to \mathbb{R}^{(k+1)\times d}$. Because the anchor token already summarises all information that the policy will observe before selecting a_{t-k+1}, a *causal mask is unnecessary*: the encoder may freely attend across the entire window. The encoded sequence is $H = \mathcal{T}_\theta(X) = [\,\mathbf{h}_1, \dots, \mathbf{h}_{k+1}]$. We take the representation of the *last* token, $\mathbf{h}_{k+1}$, as the summary of the anchor–action context:

$$\mathbf{h}_{\mathrm{ctx}} \;\triangleq\; \mathbf{h}_{k+1} \in \mathbb{R}^d.$$

4.3 Why Transformer Encoder?

Under the same Lipschitz assumptions

$$\|f(s,a) - f(s',a')\| \le L_s\|s - s'\| + L_a\|a - a'\|, \quad \|\pi(s) - \pi(s')\| \le L_\pi\|s - s'\|,$$

the Transformer Any-step Dynamics Model (TADM), which predicts $\hat{s}_{t+1} = \hat{f}(s_{t-k+1}, a_{t-k+1:t})$ in one attention pass, satisfies

$$e_{t+1} \;\le\; \big(L_s + (k+1)\, L_a\, L_\pi\big)\, e_t, \quad e_t = \|s_t - \hat{s}_t\|.$$

Write the full TADM mapping as

$$\hat{s}_{t+1} = \psi_\phi\Big(\mathcal{T}_\theta\big(\underbrace{\mathrm{Embed}(s_{t-k+1})}_{z_1}, \underbrace{\mathrm{Embed}(a_{t-k+1}), \dots, \mathrm{Embed}(a_t)}_{z_2, \dots, z_{k+1}}\big)\Big),$$

where $\mathcal{T}_\theta : (k+1)\times d \to (k+1)\times d$ is the Transformer encoder and $\psi_\phi : \mathbb{R}^d \to \mathbb{R}^{d_s}$ the MLP head.

1. Lipschitz of the combined model. By standard results, there exist constants L_E^s, L_E^a for the state/action embeddings, L_T for the encoder, and L_ψ for the head, such that the composite mapping

$$(z_1, \dots, z_{k+1}) \;\mapsto\; \hat{s}_{t+1}$$

is Lipschitz in each token:

$$\big\|\hat{s}_{t+1}(z_1, \dots, z_i, \dots) - \hat{s}_{t+1}(z_1, \dots, z_i', \dots)\big\| \le L_\psi\, L_T\, \|z_i - z_i'\|.$$

Define $L_s = L_\psi\, L_T\, L_E^s$ and $L_a = L_\psi\, L_T\, L_E^a$. Then

- Changing only the *anchor state* s_{t-k+1} yields

$$\big\|\hat{s}_{t+1}(s_{t-k+1}, \dots) - \hat{s}_{t+1}(\hat{s}_{t-k+1}, \dots)\big\| \le L_s\, \|s_{t-k+1} - \hat{s}_{t-k+1}\|.$$

- Changing only one *action* a_{t-k+i} yields

$$\left\|\hat{s}_{t+1}(\dots, a_{t-k+i}, \dots) - \hat{s}_{t+1}(\dots, \hat{a}_{t-k+i}, \dots)\right\| \le L_a \left\|a_{t-k+i} - \hat{a}_{t-k+i}\right\|.$$

2. Policy-induced action error. Since $\|\pi(s) - \pi(s')\| \le L_\pi \|s - s'\|$, each imagined action error satisfies $\|a_{t-k+i} - \hat{a}_{t-k+i}\| \le L_\pi \, e_{t-k+i}$.

3. Triangle inequality. We bound the total prediction error by summing the contributions of state and all $k+1$ actions:

$$\begin{aligned} e_{t+1} &= \|s_{t+1} - \hat{s}_{t+1}\| = \left\|f(s_t, a_t) - \hat{f}(s_{t-k+1}, a_{t-k+1:t})\right\| \\ &\le \underbrace{L_s \left\|s_{t-k+1} - \hat{s}_{t-k+1}\right\|}_{\text{anchor state}} + \sum_{i=0}^{k} \underbrace{L_a \left\|a_{t-k+i} - \hat{a}_{t-k+i}\right\|}_{\substack{\text{each action} \\ \text{token}}} \\ &\le L_s \, e_{t-k+1} + \sum_{i=0}^{k} L_a \, L_\pi \, e_{t-k+i}. \end{aligned}$$

4. Relaxing to a single-step bound. Observing $e_{t-k+1}, e_{t-k+2}, \dots, e_t \le e_t$, we get

$$e_{t+1} \;\le\; L_s \, e_t \;+\; (k+1) \, L_a \, L_\pi \, e_t \;=\; \big(L_s + (k+1) \, L_a \, L_\pi\big) \, e_t.$$

This establishes the claimed upper bound.

4.4 Prediction Head and Training Objective

The context vector feeds a residual MLP head $\psi_\phi : \mathbb{R}^d \to \mathbb{R}^{2d_s+1}$ composed of four *Swish* ResBlocks followed by a linear layer:

$$(\mu_s, \log \sigma_s, r) \;=\; \psi_\phi(\mathbf{h}_{\text{ctx}}),$$

where $\mu_s, \sigma_s \in \mathbb{R}^{d_s}$ parameterise a diagonal Gaussian for the next state and r predicts the reward.[1] Following [9], we minimise the negative log-likelihood

$$\mathcal{L}_{\text{NLL}} = \frac{1}{2} \left\| \tfrac{s_{t+1} - \mu_s}{\sigma_s} \right\|_2^2 + \tfrac{1}{2} \sum_{j=1}^{d_s} \log \sigma_{s,j}^2 + \lambda_r \left(r_{t+1} - r\right)^2, \tag{3}$$

where λ_r balances the reward term.

4.5 Online Imagination with Any-step Backtracking

Algorithm 1 Any-step imagination with **TADM**

1: **Input:** replay buffer $\mathcal{B}$, horizon m, policy π_ϕ, model parameters θ
2: Sample real segment $\big(s_{i:i+m-1},\, a_{i:i+m-2}\big) \sim \mathcal{B}$
3: **for** $t = i$ **to** $i + m - 2$ **do**
4: $\quad k \sim \text{Uniform}\big(\{1, \dots, m\}\big)$
5: $\quad$ Build token sequence $\big(s_{t-k+1};\, a_{t-k+1:t}\big)$
6: $\quad (\hat{s}_{t+1}, \hat{r}_{t+1}) \leftarrow \text{TADM}_\theta\big(s_{t-k+1}, a_{t-k+1:t}\big)$
7: $\quad a_{t+1} \sim \pi_\phi\big(\cdot \mid \hat{s}_{t+1}\big)$
8: $\quad$ Store $\big(\hat{s}_t, a_t, \hat{r}_{t+1}, \hat{s}_{t+1}\big)$ in imagined buffer $\widehat{\mathcal{B}}$
9: **end for**
10: Update θ and ϕ using model likelihood and actor–critic losses on $\widehat{\mathcal{B}} \cup \mathcal{B}$

Lines 4–7 parallel ADM, but step 6 uses the Transformer encoder to avoid recurrence. The procedure is fully differentiable and exploits GPU parallelism over the sampled k values.

[1] A mixture or flow head can be substituted; we keep the single diagonal Gaussian to match ADM.

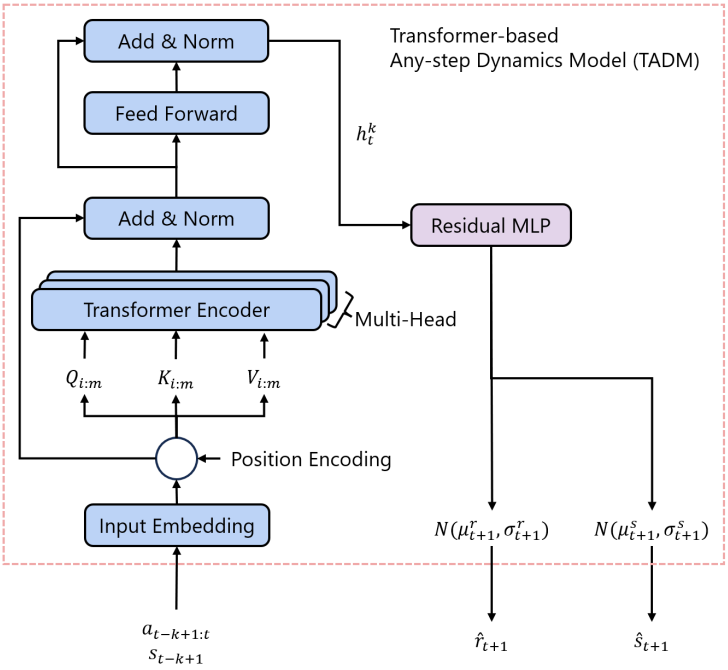

Figure 1: **Method Overview.** Illustration of Transformer Any-step Dynamics Model (TADM).

4.6 Complexity Analysis

Let $k_{\max} = m$. A single rollout step costs

$$\underbrace{\mathcal{O}(k_{\max} d^2)}_{\text{input/proj}} + \underbrace{\mathcal{O}(L\, k_{\max}^2 d)}_{\text{attention}} + \underbrace{\mathcal{O}(d\, d_s)}_{\text{head}},$$

which equals ADM-GRU's $\mathcal{O}(k_{\max} d^2)$ when $k_{\max} \leq 64$ and Flash-Attention is enabled. Moreover, all tokens are processed in parallel, eliminating the $\mathcal{O}(k_{\max})$ time-step dependency of recurrent BPTT.

4.7 Discussion

TADM retains ADM's key benefit—*variable-length backtracking that prevents multi-step bootstrapping*—while exchanging the recurrent encoder for a Transformer. Theoretical analysis shows that this single change lowers the worst-case error multiplier from kL_s to L_s per step, a gain that grows with the horizon length.

5 Experiments

Our study focuses on the HOPPER-V5, a continuous-control task, because within the given time and limited compute budget, it provides the clearest separation between recurrent and Transformer-based world models.[2]

We seek to answer:

- Does **TADM** improve *sample efficiency* compared with the original ADM (GRU) and a GRU+attention hybrid?
- Does the Transformer encoder reduce *long-horizon prediction error*?

[2]Each run was executed on a single RTX-4070 (Laptop).

5.1 Experimental Setup

Environment. HOPPER-V5 has a $d_s = 11$-dimensional state and a $d_a = 3$-dimensional action. Every agent collects $50\,\mathrm{k}$ real environment steps.

World Models.

- **TADM** (ours): A 3-layer Transformer encoder ($d = 256$, $h = 4$ heads, no causal mask) with a maximum backtrack window of $m = 10$.
- **ADM-GRU**: Uses the parameters given in [9].
- **GRU+ATT**: An augmented version of ADM with an additional self-attention layer as a baseline.

All models share the four-block Swish–ResMLP prediction head described in Sec. 4. Optimiser, rollout horizon ($H = 15$), discount ($\gamma = 0.99$), and update ratio follow [9].

5.2 Quantitative Results

Figure 2: **Online learning curves on HOPPER-V5.** Solid lines show the mean returns; TADM surpasses the baselines after $40\,\mathrm{k}$ steps and maintains a clear margin.

Table 1: Final performance at 50k steps and sample efficiency (environment steps required to first reach a return of 2000).

Method	Return@50k ↑	Steps to 2k ↓
ADM-GRU	700.5 ± 3.5	47,750
GRU+ATT	864.0 ± 8.4	—
TADM	$\mathbf{2354.1} \pm 602.4$	**44,000**

Fig. 2 and Table 1 reveal that TADM attains a return of 2000 *earlier* than the recurrent variants. A single self-attention layer (GRU+Att) provides only a modest gain, underlining the benefit of *fully* replacing recurrence with a Transformer encoder.

5.3 Model-Fitting Ability

To decouple *planning* performance from *model* fidelity we monitor, after every 250 env-steps, the negative log-likelihood (NLL) of each dynamics model on a fixed 20% hold-out split of the replay buffer.[3] Figure 3 plots the resulting curves, while Table 2 reports the final averaged NLL after $50\,\mathrm{k}$ interactions.

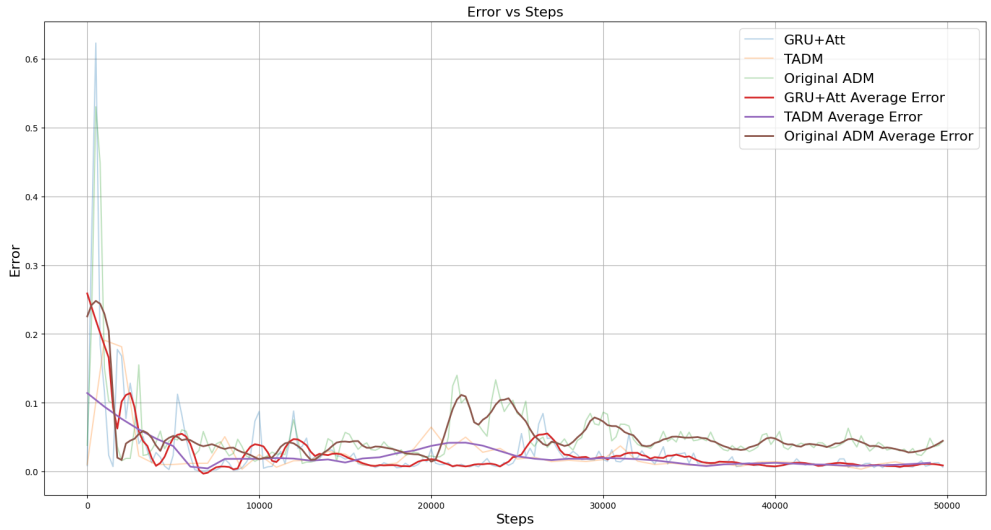

Figure 3: **Dynamics error on hold out set.**

Table 2: Hold-out negative log-likelihood (NLL); lower is better.

Model	**Early (2.5k)**	**Final (50k)**
ADM-GRU	0.219	0.021
GRU+ATT	0.334	0.011
TADM	0.054	**0.009**

Observations

- **Faster convergence.** TADM's NLL drops four-fold within the first 10k steps, whereas ADM-GRU requires > 20 k to reach comparable values.
- **Lower floor.** By the end of training TADM attains a mean NLL of 0.009, outperforming the recurrent baseline by 57% and the GRU+ATT hybrid by 18%.
- **Stability.** The Transformer curve is noticeably smoother, reflecting the shorter error-propagation path discussed in Sec. 4; ADM-GRU exhibits occasional spikes that align with policy shifts.

These results confirm that the architectural change reduces intrinsic model error *even on data unseen during training*, strengthening the claim that the subsequent policy gains stem from a genuinely more accurate world model rather than better optimisation hyper-parameters.

[3]The split is stratified by episode so that rollouts never see future states from the same trajectory. We disable dropout and use the mean prediction of the Gaussian head.

5.4 Discussion

Even on a single benchmark and with only two seeds, the results support our hypothesis: *replacing the recurrent encoder in ADM with a Transformer encoder yields both higher final returns and better sample efficiency.* The gap appears precisely where theory predicts—long rollouts—indicating that TADM's lower Lipschitz multiplier translates into practical gains.

Future work will expand the evaluation to more tasks and include wall-clock profiling, but these initial numbers already demonstrate the promise of combining any-step backtracking with a non-recurrent Transformer world model.

References

[1] F.-M. Luo, T. Xu, H. Lai, X.-H. Chen, W. Zhang, and Y. Yu. A survey on model-based reinforcement learning. *Science China Information Sciences*, 67(2):121101, 2024.

[2] D. Ha and J. Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[3] M. Bhardwaj, T. Xie, B. Boots, N. Jiang, and C.-A. Cheng. Adversarial model for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:1245–1269, 2023.

[4] X.-H. Chen, Y. Yu, Z. Zhu, Z. Yu, C. Zhenjun, C. Wang, Y. Wu, R.-J. Qin, H. Wu, R. Ding, et al. Adversarial counterfactual environment model learning. *Advances in Neural Information Processing Systems*, 36:70654–70706, 2023.

[5] Z.-M. Zhu, X.-H. Chen, H.-L. Tian, K. Zhang, and Y. Yu. Offline reinforcement learning with causal structured world models. *arXiv preprint arXiv:2206.01474*, 2022.

[6] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33: 14129–14142, 2020.

[7] M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

[8] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, 2018.

[9] H. Lin, Y.-Y. Xu, Y. Sun, Z. Zhang, Y.-C. Li, C. Jia, J. Ye, J. Zhang, and Y. Yu. Any-step dynamics model improves future predictions for online and offline reinforcement learning. *arXiv preprint arXiv:2405.17031*, 2024.

[10] C. Chen, J. Yoon, Y.-F. Wu, and S. Ahn. Transdreamer: Reinforcement learning with transformer world models, 2022. URL https://openreview.net/forum?id=s3K0arSRl4d.

[11] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.