TransDreamerV3: Implanting Transformer In DreamerV3

Shruti Sadanand Dongare dshruti20@vt.edu
Amun Kharel akharel@vt.edu
Jonathan Samuel samueljon17@vt.edu
Xiaona Zhou xzhou1@vt.edu

Abstract

This paper introduces TransDreamerV3, a reinforcement learning model that enhances the DreamerV3 architecture by integrating a transformer encoder. The model is designed to improve memory and decision-making capabilities in complex environments. We conducted experiments on Atari-Boxing, Atari-Freeway, Atari-Pong, and Crafter tasks, where TransDreamerV3 demonstrated improved performance over DreamerV3, particularly in the Atari-Freeway and Crafter tasks. While issues in the Minecraft task and limited training across all tasks were noted, TransDreamerV3 displays advancement in world model-based reinforcement learning, leveraging transformer architectures.

1 Introduction

Reinforcement learning (RL) has emerged as a powerful paradigm, allowing agents to autonomously learn optimal strategies by interacting with their environments [14]. A particularly noteworthy concept in the RL domain is the idea of World Models [4]—a structured representation of the environment that enables agents to simulate potential future outcomes, facilitating learning within an imagined environment. These models have played a crucial role in addressing tasks that require foresight and planning [4, 16], thereby bridging the gap between traditional model-free methods, which learn solely from past interactions, and model-based approaches that leverage predictive models for enhanced efficiency and performance [14].

One of the most notable advancements in the field of reinforcement learning with World Models is the introduction of DreamerV3 [8]. This state-of-the-art algorithm stands out for its capacity to generalize and master a diverse array of domains using fixed hyperparameters. DreamerV3's approach represents the first algorithm to successfully address the complex challenge of collecting diamonds in the popular video game Minecraft from scratch. While DreamerV3 has achieved significant milestones, it heavily relies on Recurrent State Space Models (RSSM), which combine the strengths of recurrent neural networks (RNN) with structured latent variable models [6]. Despite the power of RNNs in capturing sequential dependencies, they have historically faced challenges related to memory decay, especially concerning long-term memory retention [4]. This inherent limitation creates difficulties in environments where agents need to remember and act upon events from distant past interactions. As tasks become more complex and demand enhanced memory capabilities, there is a clear need to explore alternative approaches.

Transformers, originally introduced for natural language processing, have consistently exhibited significant advantages over other architectures, particularly in memory management and parallel processing [15]. Their incorporation into reinforcement learning has uncovered remarkable potential, especially in tasks requiring substantial memory capabilities [3]. In this context, the TransDreamer [2] architecture comes to the forefront, leveraging the strengths of world models while capitalizing on the transformer's benefits. This architecture builds upon the foundation of the original Dreamer architecture [5]. At the core of this innovation is the Transformer State-Space Model (TSSM)—a stochastic transformer-based state-space model.

In this research, we introduce TransDreamerV3, an evolution of the DreamerV3 framework. Our model aims to harness the advantages of transformers by integrating components from the TSSM framework of TransDreamer into the DreamerV3 architecture. Through this integration, we hypothesize improvements not only in memory capabilities but also in the overall performance and adaptability of the model across a variety of tasks.

We implemented several modifications in our model, transitioning from RSSM to TSSM. Firstly, for the deterministic state model, we substituted the existing gated recurrent unit with a transformer encoder. This encoder is independent of prior deterministic states but dependent on all representation states and actions. Regarding the representation model, we modified the belief to exclude the deterministic state. For trajectory roll-out, we limited imagined trajectories to three per training sample, and the replay buffer prioritized trajectories with higher rewards. Lastly, we trained the policy by keeping the transformer parameters frozen during training.

While implementing DreamerV3, the team substituted the GRU in the RSSM module with a transformer, utilizing the relatively recent programming languages JAX and Ninjax. Ninjax, constructed on the JAX foundation, facilitated the development of intricate neural network architectures. To ensure harmony with DreamerV3's existing codebase, the team adopted a naive transformer, encountering obstacles due to the scarcity of documentation and community knowledge about Ninjax. The naive transformer features a manually constructed attention mechanism, omits positional encoding and dropout layers, and applies layer normalization after the attention and feed-forward network steps.

We summarize the key contributions of this paper as follows:

- We introduce TransDreamerV3, a world model agent based on transformers, extending the success of TransDreamer and DreamerV3.
- We enhanced the DreamerV3 codebase by substituting the RSSM architecture with the Naive TSSM architecture, developed using the modern programming languages JAX and Ninjax.

2 Related Work

2.1 World Models

In the context of World Models, DreamerV3 is a notable advancement, improving upon DreamerV2 [7] in several ways. It introduces discrete regression for the critic, enhancing learning efficiency in environments with sparse rewards [7]. The actor network in DreamerV3 efficiently maximizes returns & maintains exploration balance within different reward conditions. This model has shown success in over 150 tasks demonstrating its wide applicability, generalizability and robust performance with fixed hyperparameters [7]. DreamerV3 achieved the groundbreaking feat of autonomously collecting diamonds in the MineCraft environment, without any human guidance. Despite DreamerV3's success, its performance is not entirely consistent across different tasks, as it only occasionally solves the Minecraft diamond challenge, indicating limitations in long horizon tasks possibly due to a lack of long-term memory retention in the Recurrent State Space Model (RSSM) framework.

2.2 Transformers

Transformers find applications in a wide array of tasks, yet within the realm of Reinforcement Learning (RL), they exhibit unique emerging potential, alongside some distinct challenges from an embodied AI perspective. Prior works [12], [9] have tackled the integration of transformers in RL, often employing strategies like introducing gating layers on top of transformer layers to enhance training stability. Diverse approaches are explored across different works; for instance, the 'Decision Transformer' [3], and work like [10] represent a concurrent sequence-modeling approach, wherein it re-frames the RL problem as a sequence modeling task and employs transformers to predict actions, eliminating the need for additional networks for actor or critic roles.

2.3 World Models With Transformers

IRIS [11] employs a discrete autoencoder & an autoregressive Transformer as the World model framework, demonstrating great sample efficiency in complex environments like the Atari 100k

benchmark, outperforming humans in 10 out of 26 games with limited data. Despite its efficiency, it encounters challenges in tasks requiring extensive memory. TransDreamer [2] introduces the Transformer State-Space Model (TSSM), specifically designed to handle long-range memory access and memory-based reasoning in both 2D and 3D visual RL tasks. It addresses the limitations of RSSM frameworks employed within DreamerV2's architecture and demonstrates improved performance in complex tasks requiring sophisticated temporal understanding.

3 Methods

Transformers have been shown to excel in tasks requiring complex long-term temporal dependencies such as memory based reasoning and learning complex interactions between historical states [2],[13],[1]. These are important and desirable capabilities that a World Model needs. We hypothesize that a Transformer based World Model in DreamerV3 will outperform RNN-based Dreamer agent for tasks requiring complex and long-term memory dependency.

We use two main papers [8] and [2] to replace the backbone of the world model used in Dreamer, which is a stochastic recurrent neural network called Recurrent State-Space Model (RSSM) by a Transformer State-Space Model (TSSM).

3.1 Transformer State Space Model (TSSM)

We get the idea of TSSM as shown from the paper [2]. We believe using TSSM instead of RSSM in DreamerV3 would increase efficiency for various tasks such as Atari 100K, Proprio Control, BSuite, Visual Control, Atari 200M and Crafter. Because of its ability to do complex interactions and learning long-term dependencies, it would also perform the Minecraft Diamond task faster and more efficiently. TSSM has following desirable qualities based on paper [2]: i) Direct access to past states, ii) Can update each time step in parallel during training, iii) Ability to roll out sequentially for trajectory imagination for test time, and iv) Stochasticity for latent variable. We are going to make two primary contributions as stated in the Problem Statement using the TSSM backbone.

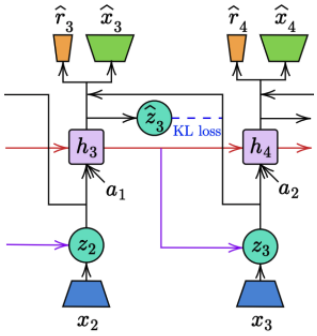

Figure 1: RSSM model where red arrow makes sequential computational necessary

3.2 Architectural changes from RSSM to TSSM

Figure 1 shows the architecture of RSSM and Figure 2 depicts the architecture of TSSM, which will be used to update the backbone of Dreamer-v3. RSSM has sequentially dependent computation when we have to update state $h_t = f_{gru}(h_{t-1}, z_{t-1}, q_{t-1},)$, which is shown in red arrows in Figure 1. TSSM removes this sequential computation by employing a transformer as a replacement for the RNN. RSSM accesses the past indirectly via compression of h_{t-1}. However, transformer directly accesses the sequence of stochastic states and actions of the past at every time step as $h_t = f_{transformer}(z_{1:t-1}, a_{1:t-1})$. Using a modified Dreamer using $q(z_t|x_t)$ instead of $q(z_t|x_t, h_t)$ would perform similarly to the original Dreamer as shown by experiment in paper [2]. We use the modified equation for the representation model in our work. Stochastic state model, Image Predictor, Reward Predictor and Discount Predictor all remain the same as Dreamer-v3 for this project. To summarize:

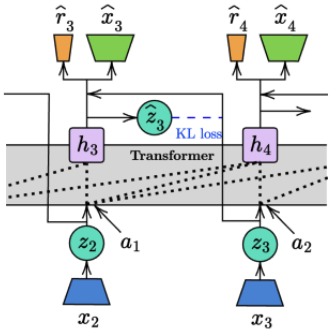

Figure 2: Modified TSSM model by eliminating deterministic state dependency (red arrows depicted in Figure 1), facilitating parallel updates across all time steps

- Deterministic State Model: Replace the current gated recurrent unit with a transformer encoder that is independent of prior deterministic states h_t and dependent of all representation states z_t and actions a_t
- Representation Model: Adjust belief to omit the deterministic state h_t
- Trajectory Roll-out: Cap imagined trajectories to 3 per training sample and repay buffer prioritizes higher reward trajectories
- Policy Learning: Freeze transformer parameters during training

3.3 Implementation in DreamerV3

In the implementation of DreamerV3, the team integrated a transformer into the world model architecture, replacing the GRU within the RSSM module. This task was undertaken using JAX and Ninjax, two relatively new programming languages in the field of machine learning. Ninjax is a neural network library built on top of JAX, providing a more streamlined and flexible approach to building complex neural network architectures.

The decision to implement a naive transformer, where the input and output dimensions mirror those of the original GRU, aimed to focus attention on the prior deterministic state and action only. This approach was intended as a baseline to ensure compatibility with DreamerV3's broader codebase. However, the team encountered challenges due to the nascent state of documentation and community knowledge surrounding Ninjax. This led to a slower-than-anticipated development and integration process for the transformer.

In our naive transformer implementation, the attention mechanism is manually constructed with separate linear transformations for queries, keys, and values, followed by a scaled dot-product attention calculation without an attention mask. Our approach omits positional encoding due to only considering a the prior state. In the feed-forward network within each transformer layer, we omit a dropout layer due to conflicts with Ninjax. Layer normalization in our model is applied post-attention and post-feed-forward network addition steps.

4 Experiments & Results

To assess the performance of DreamerV3 [8], TransDreamer [2], and our model, we conducted training experiments on Atari-boxing, Atari-freeway, Atari-pong, and Crafter tasks. Due to time and resource constraints, we compare performances at different training steps. Nevertheless, it's important to note that within each task, the comparison is fair as all models are trained under the same settings. Performance comparisons for Atari tasks and Crafter are visualized in Figure 3.

4.1 Atari and Crafter Tasks

For the Atari-Boxing task, TransDreamer [2] exhibited superior performance after 1.5 million experimental steps. In the Atari-Freeway task, DreamerV3 [8] achieved non-zero rewards after

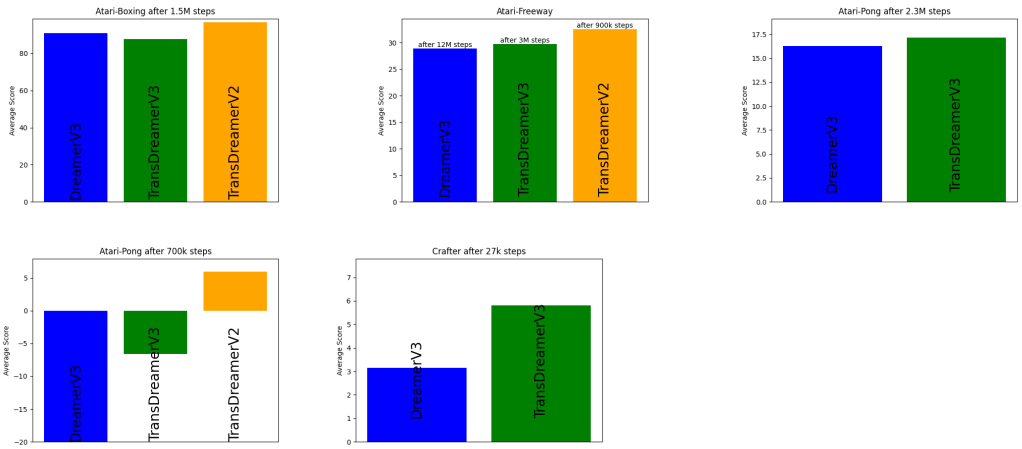

Figure 3: Performance comparison of DreamerV3, TransDreamer, and TransDreamerV3

12 million steps, while our model, TransDreamerV3, accomplished the same in 3 million steps. Conversely, TransDreamer [2] achieved positive rewards within 1 million steps, surpassing the other two models. In the Atari-Pong task, both DreamerV3 [8] and our model initially received negative rewards for the first 700k steps. However, our model eventually outperformed DreamerV3 [8] after 2.3 million steps. Although we didn't complete training for TransDreamer [2] up to 2.3 million steps, we will provide updates when available. With the Crafter task, our model significantly outperforms DreamerV3[8]. In every tested environment except boxing, our model, employing a naive transformer, surpasses DreamerV3 in performance. This outcome indicates that even though both our model and the GRU-based state space model concentrate on the most recent data, the attention mechanism in our model has advantages over the GRU's gated mechanism. However, our model underperforms compared to the original TransDreamer, which aligns with expectations given that our model does not fully utilize the context of previous states.

4.2 Other experiments

We encountered several challenges while attempting to run experiments with Minecraft. Despite successfully configuring the settings that yielded results for all other tasks, we consistently faced a runtime error indicating a 'Lost connection to workers.' After resolving setup issues and successfully running the training in Docker, we encountered additional difficulties with the evaluation code. Unfortunately, the issues with the evaluation code prevented us from directly comparing our model with DreamerV3 [8]. We documented the docker setup for Minecraft since it needed engineering and modifications in the originally provided Dockerfile by [8]. It's noteworthy that we were able to execute all other experiments presented in DreamerV3 [8], including Proprio Control, BSuite, and Visual Control. However, due to resource constraints, we couldn't afford to train models for each task. We have provided code and model checkpoint in the repo[1].

5 Conclusion

Our research introduces TransDreamerV3, a novel reinforcement learning model that integrates the transformer based architecture of TransDreamer into the robust framework of DreamerV3. By replacing the GRU in the RSSM module with a transformer encoder and modifying the belief representation in the model, we aimed to enhance memory capabilities and overall performance. Our model outperformed DreamerV3 in the Atari-Freeway and Crafter tasks, and showed promising results in Atari-Pong suggesting advancement in handling complex environments and decision-making scenarios. Future ablation studies should investigate the GRU gated mechanism compared with the attention mechanism of the transformer. The absence of positional embedding, dropout

[1] https://github.com/XiaonaZhou/TransDreamerV3

layers and attention to all representational and action states in our transformer architecture suggests areas for future refinement.

6 Contribution

We have completed 100% of this project without any outside help or guidance. Our contribution to this project is as follows:

- Shruti assisted in setting up DreamerV3 on Docker and planning the evaluation & visualization steps. Worked on running Minecraft experiment. Collaborated to write the related work and a part of Minecraft experiment evaluation.
- Amun assisted in implementing the naive transformer in DreamerV3's code base, replacing the GRU. Collaborated to write the introduction, methods and conclusion.
- Jonathan led developing the model architecture, planning the overall implementation, and programming & debugging the naive transformer in DreamerV3's framework. Collaborated to write the introduction, related work, methods and conclusion.
- Xiaona took charge of setting up & training DreamerV3, TransDreamer, TransDreamerV3 for various tasks, getting experimental results, making plots, and providing assistance in implementing the naive transformer. Wrote the experiments and results section, and a part of the related work.

References

[1] Andrea Banino, Adrià Puigdomènech Badia, Raphael Köster, Martin J. Chadwick, Vinicius Zambaldi, Demis Hassabis, Caswell Barry, Matthew Botvinick, Dharshan Kumaran, and Charles Blundell. Memo: A deep network for flexible combination of episodic memories, 2020.

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

[3] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[4] David R Ha and Jürgen Schmidhuber. World models. *ArXiv*, abs/1803.10122, 2018.

[5] Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *ArXiv*, abs/1912.01603, 2019.

[6] Danijar Hafner, Timothy P. Lillicrap, Ian S. Fischer, Ruben Villegas, David R Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *ArXiv*, abs/1811.04551, 2018.

[7] Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *ArXiv*, abs/2010.02193, 2020.

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

[9] Kazuki Irie, Imanol Schlag, Róbert Csordás, and Jürgen Schmidhuber. Going beyond linear transformers with recurrent fast weight programmers. *Advances in Neural Information Processing Systems*, 34:7703–7717, 2021.

[10] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[11] Vincent Micheli, Eloi Alonso, and Franccois Fleuret. Transformers are sample efficient world models. *ArXiv*, abs/2209.00588, 2022.

[12] Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pages 7487–7498. PMLR, 2020.

[13] Sam Ritter, Ryan Faulkner, Laurent Sartran, Adam Santoro, Matt Botvinick, and David Raposo. Rapid task-solving in novel environments, 2021.

[14] Peter Norvig Stuart Russell. *Artificial Intelligence: A Modern Approach*. Pearson, 4 edition, 2020.

[15] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017.

[16] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Ken Goldberg, and P. Abbeel. Daydreamer: World models for physical robot learning. In *Conference on Robot Learning*, 2022.