Recurrent Autoregressive Diffusion: Global Memory Meets Local Attention

Taiye Chen[1], Zihan Ding[2], Anjian Li[2], Christina Zhang[2], Zeqi Xiao[3], Yisen Wang[1], and Chi Jin[2]

[1] Peking University, China yeyutaihan@stu.pku.edu.cn,yisen.wang@pku.edu.cn
[2] Princeton University, USA {zihand,anjianl,christinaz,chij}@princeton.edu
[3] Nanyang Technological University, Singapore zeqixiao1@gmail.com

Abstract. Recent advancements in video generation has shifted from bidirectional models for short videos to autoregressive ones for ultra long video generation. Previous models, which usually use sliding window attention to restrict inference cost, lack effective memory compression and retrieval for long-term generation beyond the window size, leading to issues of forgetting and spatiotemporal inconsistencies. To enhance the retention of historical information with a fixed memory budget, we additionally incorporate temporal recurrent neural network (RNN) layers into the diffusion transformer (DiT) model. Specifically, we found that a LSTM layer after attention at each DiT layer achieves comparable performance to other state-of-the-art RNN blocks, such as Test-Time Training (TTT) and Mamba2. Moreover, existing diffusion-RNN approaches often suffer from performance degradation due to training-inference gap or the lack of overlap across windows. To address these limitations, we propose a novel Recurrent Autoregressive Diffusion (RAD) framework, which leverages recurrent blocks for memory update and retrieval and preserves local details by full attention on overlapping sliding windows, with no training and inference gap. Experiments on Memory Maze and Minecraft datasets demonstrate the superiority for long video generation by our framework with global memory and local attention.
Project page: https://yeyutaihan.github.io/recurrent-autoregressive-diffusion/

Keywords: World model · RNN · Diffusion model

1 Introduction

World models have attracted considerable interest from the research community for their pivotal roles in data synthesis, model-based planning, simulation, and beyond. Leveraging the recent great progress in generative models, video diffusion models have become one of the most promising approaches for efficient and scalable world models. Opposed to representation world models [1, 5] that learn latent representations of the environment, video diffusion models [3, 8, 32] directly achieve world modeling in high-dimensional video pixel space.

Despite remarkable advancements, existing video world models face significant challenges in maintaining spatiotemporal consistency. For instance, models like Oasis [16]—centered on Minecraft gameplay—and foundational models such as Cosmos [31] both struggle with severe forgetting issues. This difficulty primarily stems

from the inherently limited attention window of diffusion transformer (DiT) [34] architecture. Given the low information density of video data, tokenizing video sequences often results in context lengths that quickly exceed these attention limits. Consequently, frames outside the active attention window are effectively disregarded, resulting in visible temporal and spatial inconsistencies. While recent research has explored alternative approaches to mitigate this, such as leveraging 3D representations [44] to bolster spatial-temporal consistency, they usually lack the interactiveness and scalability of pixel-based video diffusion models.

A central challenge in applying DiT to long video generation is the ineffective compression of historical context through key-value (KV) caching in standard attention mechanisms. As video sequences extend, this approach results in scalability bottlenecks, since the memory and computation requirements grow proportionally to sequence length. Recent efforts have sought to address this limitation by incorporating recurrent neural networks (RNNs)—including Mamba [15, 17, 36, 46] and Test-Time Training (TTT) approaches [43, 56]—within the DiT framework. However, these integrations often suffer from two major issues: (1) chunk-wise autogressive processing, common in TTT-style models, relies heavily on hidden state propagation, causing the model to lose direct access to the dense contextual information present in recent frames and resulting in pixel-level inconsistencies across chunk boundaries; and (2) the introduction of recurrency breaks the parallelizable nature of attention during training, thereby compromising computational efficiency and exacerbating the gap between training and inference procedures.

In this work, we introduce **Recurrent Autoregressive Diffusion (RAD)**, a unified framework for long-term video generation with global memory and local attention, with the following core designs to address previous challenges:

- We incorporate an RNN layer along temporal dimension after the attention in each DiT layer for retrieving global memory. Through systematic comparison among different RNN architectures—including LSTM [23], Mamba2, and TTT—within autoregressive video generation framework, we reveal that LSTM, despite its simplicity, delivers robust performance and often surpasses more recent RNN variants on challenging long video benchmarks.
- To preserve local context details, we choose a largely overlapped sliding window attention. Through a comprehensive comparison between **chunk-wise** and **frame-wise** autoregressive inference paradigms, we find that frame-wise autoregressive rollout with stride size 1 of sliding window substantially enhances spatiotemporal consistency by leveraging immediate contextual information, and reduces reliance on persistent hidden state propagation—thereby mitigating pixel-level inconsistencies and improving the fidelity of long video synthesis.
- For compute efficiency, we introduce a **hidden-state prefetch mechanism** to recover parallelism in attention computation, which overcomes the inherent trade-off between recurrency and training parallelism. This mechanism enables fully parallel attention computation during training while retaining the benefits of recurrent historical compression, significantly improving efficiency for large-scale, long-sequence modeling.

2 Related work

Video Diffusion Model The remarkable success of diffusion models originated in image synthesis [37,39] and was later extended to video generation [7,22,24,35,51,57]. Current state-of-the-art video diffusion models typically employ VAEs [29] to map videos from the pixel level to a latent space, while the model architecture has evolved from the UNet [6,11,40] to diffusion transformer (DiT) [34].

Video World Model World models [20,21,48] predict future states based on the current state and input actions. They hold broad application prospects in fields such as autonomous driving [25,38], navigation [4], and robotic manipulation [2,30,49] and games [9,16,18,45,54]. The capability of video diffusion models to synthesize high-quality videos makes them promising candidates as video world models. Since the proposal of "video generation models as world simulators" by Sora [8], a series of foundational world models have emerged—such as Genie2 [32], Genie3 [3], and Cosmos [31]—demonstrating remarkable video generation quality and interactivity. To achieve video world models, recent advancement improves video generation models in multiple aspects: autoregressive inference for extending video durations [10,13,26,52], memory mechanism for improving spatiotemporal consistency [12,50,53], physics modeling [28], etc.

Diffusion Model with Memory While scalable DiT-based video diffusion models have shown strong performance on short clips, applying full attention to long videos incurs prohibitive computational and memory costs. To enable long-video generation without global attention, additional memory mechanisms are required, which can be classified into *context memory*, *hidden-state memory*, and *weight memory*. Context memory approaches treat past frames as conditions for autoregressive (AR) prediction. With a limited context window, they either adopt recency-biased frame selection—as in diffusion forcing [10,41] and self-forcing [13,26]—or employ similarity-based retrieval, such as VRAG [12] and WorldMem [50]. Hidden-state memory methods compress historical information into recurrent states, e.g., by inserting Mamba layers before attention [36]. However, the recurrent structure disrupts the temporal parallelism of the original DiT, adding extra computational overhead. Weight memory techniques use inner-loop losses to update specific weights as memory while sliding over small chunks, as seen in Test-Time Training (TTT) blocks [14,56]. Since adjacent attention windows do not overlap, the updated weights must fully encode all necessary history from prior chunks, imposing high demands on memory capacity for storage and retrieval. In contrast, our approach performs efficient memory compression only for history beyond the context window. It bridges context memory and hidden-state memory through frame-wise sliding during autoregressive generation, balancing efficiency and long-range consistency.

3 Preliminaries

3.1 Recurrent Neural Networks

RNN Assuming $x_t \in \mathbb{R}^D$ denotes the input of time t, The general form of a recurrent neural network (RNN) can be described as:

$$h_t = f(h_{t-1}, x_t, y_{t-1}; \theta)$$
$$y_t = g(h_t, x_t)$$

where h_t is the hidden state at time t, x_t is the input, y_t is the output, $f(\cdot)$ is a parameterized nonlinear function (typically involving affine transformation and activation), θ represents all trainable parameters, and $g(\cdot)$ maps the hidden state to the output.

LSTM

$$\begin{bmatrix} f_t \\ i_t \\ o_t \\ g_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \begin{bmatrix} y_{t-1} \\ x_t \end{bmatrix}$$
$$C_t = f_t \odot C_{t-1} + i_t \odot g_t$$
$$y_t = o_t \odot \tanh(C_t)$$

where f_t,i_t,o_t,g_t denote forget gate, input gate, output gate, candidate cell state at time t, σ,$tanh$ denote sigmoid activation function and hyperbolic tangent activation function, W denote weight and bias matrices for respective gates. The vectors C_t and y_t denote the cell state and output at time t, respectively, encapsulating all compressed memory information accumulated up to and including time step t.

Mamba

$$H_t = AH_{t-1} + BX_{t-1}; \quad X_t = CH_t + DX_{t-1}$$

where H_t are latent states, and A,B,C,D are linear projections of input X_t, i.e., $A_t := \text{Linear}_{\theta_A}(X_t)$ and similarly for B_t, C_t, and D_t. This is the mathematical formulation of Mamba for autoregressive tasks. In a multi-layer setting, the output from the previous timestep does not serve as the input for the next timestep, but rather as the input for the next layer. Therefore, it can be expressed as:

$$H_t = AH_{t-1} + BX_t; \quad Y_t = CH_t + DX_t$$

TTT

$$W_t = W_{t-1} - \eta \nabla \mathcal{L}(W_{t-1}; x_t)$$
$$y_t = f(\theta_Q x_t; W_t)$$

where the self-supervised loss $\mathcal{L}$ is often defined as $\mathcal{L}(W;x_t) = ||f(\theta_K x_t;W) - \theta_V x_t||^2$, θ_Q,θ_K,θ_V are trainable parameters, and W_t is a hidden state matrix at time t. TTT-linear learns per-instance weights $W_t \in \mathbb{R}^{d \times d}$, but with often small MLPs or projections. TTT updates global weights per step; typically, there is no recurrent hidden state transfer beyond $\mathcal{L}(W;x_t)$.

Comparison As presented in Tab. 1, we provide a comparison of different RNN types, where h is LSTM hidden size, n is Mamba SSM state size.

3.2 Video Diffusion model

Latent Video Diffusion Model We adopt a latent video diffusion model [6] that first encodes pixel space into a latent representation $\boldsymbol{z} = \mathcal{E}(\boldsymbol{x})$ using a pretrained variational

Table 1: Comparison of computational complexity between LSTM, Mamba (SSM), and Test-Time Training (TTT-linear with $d\times d$ memory weights) for input tensor (B,H,W,L,d).

Aspect	**LSTM**	**Mamba (SSM)**	**TTT**
Computation (FLOPs)	$O(BHWL(dh+h^2))$	$O(BHWLdn)$	$O(BHWL(d^2+C))$
Parameter count	$O(h(d+h))$	$O(dn)+O(d^2)$ (if projected)	$O(d^2)$
Training Memory	$O(BHWLh)$	$O(BHWL(d+n))$	$O(BHWd)$
Inference Memory	$O(BHWh)$	$O(BHW(d+n))$	$O(BHWd)$
Sequence scaling	Linear in L	Linear in L	Linear in L

autoencoder (VAE). The forward process gradually adds Gaussian noise to the latent according to a variance schedule $\{\beta_t\}_{t=1}^T$:

$$q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})=\mathcal{N}(\boldsymbol{z}_t;\sqrt{1-\beta_t}\boldsymbol{z}_{t-1},\beta_t\mathbf{I}) \tag{1}$$

The model learns to reverse this process by predicting the noise $\boldsymbol{\epsilon}_\theta$ at each step:

$$\mathcal{L}=\mathbb{E}_{t,\boldsymbol{\epsilon},\boldsymbol{z}}[\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t,t)\|_2^2] \tag{2}$$

where $\boldsymbol{z}_t=\sqrt{\bar{\alpha}_t}\boldsymbol{z}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})$.

At inference time, we can sample new videos by starting from random noise $\boldsymbol{z}_T\sim\mathcal{N}(0,\mathbf{I})$ and iteratively denoising:

$$\boldsymbol{z}_{t-1}=\frac{1}{\sqrt{\alpha_t}}(\boldsymbol{z}_t-\frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t,t))+\sigma_t\boldsymbol{\epsilon} \tag{3}$$

where $\alpha_t=1-\beta_t$ and $\bar{\alpha}_t=\prod_{s=1}^t\alpha_s$. The final latent sequence $\boldsymbol{z}_0$ is decoded back to pixel space using the decoder $\mathcal{D}$ to obtain the generated video.

Diffusion Forcing To enable long video generation, we apply the Diffusion Forcing [10] technique. During training, we randomly add noise to each frame in the entire input video sequence according to the diffusion schedule: $z_t^i=\sqrt{\bar{\alpha}_t}z_0^i+\sqrt{1-\bar{\alpha}_t}\epsilon^i,\epsilon^i\sim\mathcal{N}(0,\mathbf{I})$, where z_t^i represents the noised latent of the i-th frame, and the training objective for action-conditioned autoregressive video models become:

$$\mathcal{L}_{\text{DF}}=\mathbb{E}_{[t],\boldsymbol{\epsilon},\mathbf{z},a}[\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta(\mathbf{z}_{[t]},[t],\boldsymbol{a})\|_2^2]$$
$$\boldsymbol{\epsilon}=\{\epsilon^i\}_{i=1}^L,\mathbf{z}_{[t]}=\{z_t^i\}_{i=1}^L$$

where $[t]$ is vector of L timesteps with different $t\in[T]$ for each frame and $\boldsymbol{a}$ is an action sequence $\boldsymbol{a}\in\mathbb{R}^{L\times A}$. The noise prediction model $\boldsymbol{\epsilon}_\theta$ conditioned on both the action sequence $\boldsymbol{a}$ and noised frames $\mathbf{z}_{[t]}$.

4 Methodology

To address the limitations of fixed-size context windows in video diffusion models, we propose the integration of **global memory with local attention** mechanisms. This approach enables the model to effectively capture long-term dependencies in video sequences while maintaining high fidelity in generated frames.

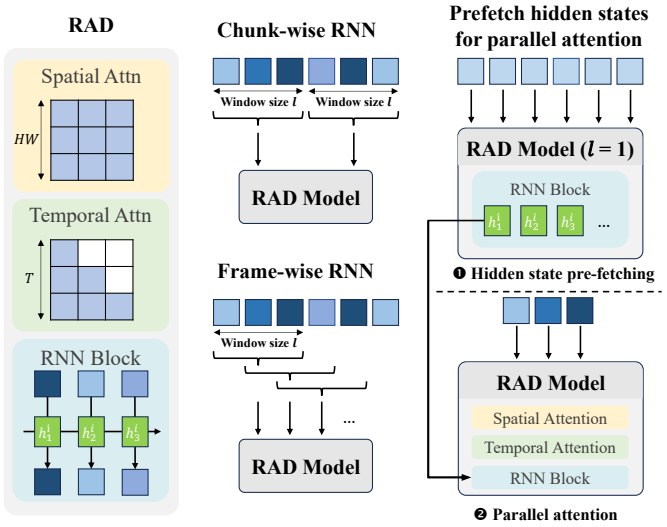

Fig. 1: Training paradigm for Recurrent Autoregressive Diffusion with global memory and local attention: The model has three components in each DiT block, including spatial attention, temporal attention and RNN memory block. It supports both chunk-wise and frame-wise autoregressive generation with different attention mechanisms. For efficient training of frame-wise RNN, we (1) pre-fetch the hidden states from clean sample sequence to enable parallel attention computation across entire long sequences, and (2) conduct diffusion model forward in standard manner to get diffusion loss. This improves efficiency and fidelity for long-sequence video modeling. h_j^i is the i-th layer hidden state for j-th frame.

4.1 Recurrent Autoregressive Diffusion

We introduce the Recurrent Autoregressive Diffusion (RAD) model, which integrates a Recurrent Neural Network (RNN) block into the Diffusion Transformer (DiT) architecture [34] to carry global memory information. The overall architecture of the RAD model is illustrated in Fig. 2. Following the designs by previous work [16, 57], we decompose the attention mechanism in our DiT into two distinct modules: Spatial Axis Attention and Temporal Axis Attention. Each layer of the DiT therefore comprises three primary components—the two attention modules as standard architecture and an additional RNN block.

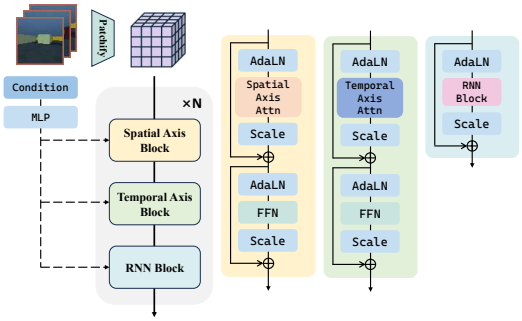

Fig. 2: Recurrent Autoregressive Diffusion model architecture: The RAD model consists of N DiT blocks, each of which contains three blocks: a Spatial Axis Block, a Temporal Axis Block, and an RNN Block. The timestep and action condition are processed through an MLP and then used to control each block via adaLN.

The RNN block performs retrieval and update operations solely along the temporal axis, meaning computations are carried out on a per-frame basis rather than a per-patch basis. The Rotary Position Embedding (RoPE) [42] is applied in both spatial and temporal dimensions, to enhance the capacity of both attention modules to capture positional dependencies. Conditioning information, specifically the timestep and action condition, is incorporated into the RAD model via adaptive Layer Normalization (adaLN). RAD also additionally applies the action conditioning on RNN blocks, apart from the attention modules. This design is verified to effectively enhance the action control and improve video fidelity.

For the RNN block, we compare three alternatives including LSTM, TTT and Mamba's SSM block [14], and find that LSTM performs the best in our RAD model, shown in experiment Sec. 5. In the following sections we discuss two key design choices made to optimize the integration of RNNs within the DiT framework: (a). frame-wise autoregression for better context consistency (Sec. 4.2) and (b). hidden state pre-fetching for parallel attention computation (Sec. 4.3).

4.2 Chunk-wise and Frame-wise Autoregression

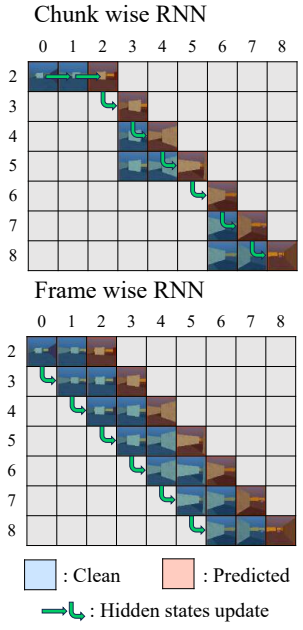

Fig. 3: Effective temporal attention maps for chunk-wise and frame-wise RNNs, with chunk size 3 and 2 initial context frames.

Algorithm 1 Inference Pipeline of Frame-wise RAD

Require: Context video $\mathbf{x}_{1:T_c}$, target length T, RAD layers of attention and RNN block $\{A_d, R_d\}_{d=1}^{D}$, window size C, denoising steps K

$\mathbf{v}_{out} \leftarrow \mathbf{x}_{1:T_c}$
Initialize hidden states $\{h^d\}_{d=1}^{D}$
 # Prefill hidden states from context
for $d=1$ to D **do** ▷ DiT layer
  $h_0^d \leftarrow \mathbf{0}$
  for $t=1$ to T_c-C+1 **do** ▷ Frame sequence
    $\mathbf{z}_t^d \leftarrow A_d(\mathbf{x}_t)$
    $h_t^d \leftarrow R_d(\mathbf{z}_t^d, h_{t-1}^d)$
  end for
  $h^d \leftarrow h_{T_c-C+1}^d$
end for
 # Sliding window with 1-frame stride
while $|\mathbf{v}_{out}| < T$ **do** ▷ Frame sequence
  Sample $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
  $\mathbf{z} \leftarrow \text{cat}(\mathbf{v}_{out}[-C+1:], \epsilon)$
  for $k=1$ to K **do** ▷ Denoising step
    for $d=1$ to D **do** ▷ DiT layer
      $\mathbf{z}^d \leftarrow A_d(\mathbf{z})$
      $\mathbf{z}, \hat{h}^d \leftarrow R_d(\mathbf{z}^d, h^d)$
    end for
    $\mathbf{z} \leftarrow \text{DenoiseStep}(\mathbf{z}, k)$
  end for
  $h^d \leftarrow \hat{h}^d, d=1,\ldots,D$ ▷ Clean hidden state
  $\mathbf{v}_{out} \leftarrow \text{Append}(\mathbf{v}_{out}, \mathbf{z}_{-1})$
end while
return $\mathbf{v}_{out}$

In the RAD framework, the RNN block processes temporal information in an autoregressive manner, which can be implemented via two distinct modes: chunk-wise and frame-wise autoregression. Both approaches are seamlessly integrated with the

temporal attention mechanisms of the DiT architecture, as illustrated in Fig. 1 and Fig. 3.

For chunk-wise autoregression, the attention windows are non-overlapping. During training, the input video sequence is divided into chunks based on the model's window size. Within each chunk, local attention is computed, while global temporal dependencies are preserved by propagating the RNN hidden states across chunk boundaries. To ensure that the model operates autoregressively within each chunk, we apply a causal mask to the temporal dimension of the attention mechanism. Importantly, this approach contrasts with existing methods [14, 56] that focus on fine-tuning diffusion models trained for full-sequence denoising. Instead, our method ensures both architectural and procedural consistency between training and inference: the model processes data during inference in exactly the same manner as during training, thereby supporting robust temporal generalization and faithful sequence modeling.

For the frame-wise RNN mode, building on Diffusion Forcing [10], we employ a frame-by-frame autoregressive generation scheme. This method allows the model to fully leverage the attention mechanism for transmitting pixel-level information across frames. However, this comes at the cost of significant computational overhead when naively applying sliding window-based training. To circumvent this inefficiency, we introduce a Hidden State Pre-fetching strategy, detailed in a subsequent section. During inference, as depicted in Fig. 3, we slide the window one frame at a time, with only the first frame in each window responsible for updating the RNN's hidden state. This procedure directly mirrors the window size of 1 used in the hidden state pre-fetch step at training. Additionally, hidden state updates are performed only at the final step of the DDIM process, ensuring that all memory inputs consist of clean frames—thereby maintaining consistency with the training process and promoting more stable generation quality.

4.3 Hidden State Pre-fetch for Parallel Attention

Our training strategy for frame-wise RNN is depicted in the right panel of Fig. 1 and Fig. 4. A well-known limitation of RNNs is their inherent sequential dependence: each timestep's output is tightly linked to the previous hidden state. This strict temporal recursion inhibits parallelization and considerably slows training, posing a particular challenge for long sequences and large-scale datasets. When RNNs are further integrated with attention mechanisms, the subsequent attention computation needs to follow the same recurrence along the input sequence during training, resulting in excessive computational bottlenecks. It becomes expensive to maintain a fully sequential temporal loop during training.

To mitigate these challenges, we employ a **hidden state pre-fetching** scheme that partially decouples the RNN from the attention modules. In this framework, the RNN maintains its temporal recurrence, but the attention operations can proceed in parallel, significantly improving training efficiency. Without this technique, a frame-wise sliding with window size l over a sequence length L requires $(L-l+1)$ times sequential attention computation, while parallel attention only conducts once at training in our case. Another pivotal benefit of this approach is that only clean frames, rather than noised frames from the diffusion process [36], are used as inputs to RNN

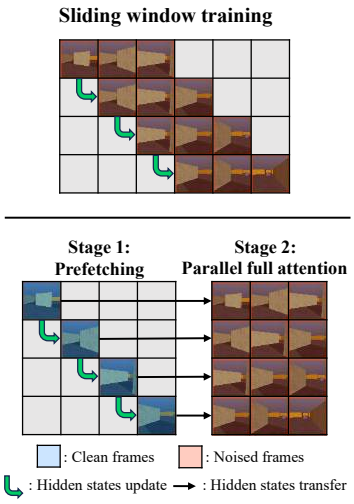

Fig. 4: Comparison of standard sliding-window forward and hidden-state prefetching.

Algorithm 2 Frame-wise RAD Training

Require: Clean video $\mathbf{x}$, target $\mathbf{v}_{gt}$, sequence length T, RAD layers $\{A_d, R_d\}_{d=1}^{D}$

Initialize hidden bank $\mathbf{H} \leftarrow \emptyset$
 # Prefetch hidden states with 1-frame RAD
for $d=1$ to D **do** ▷ DiT layer
  $h_0^d \leftarrow \mathbf{0}$
  for $t=1$ to T **do** ▷ Frame sequence
    $\mathbf{z}_t^d \leftarrow A_d(\mathbf{x}_t)$
    $h_t^d \leftarrow R_d(\mathbf{z}_t^d, h_{t-1}^d)$
    $\mathbf{H}[d,t] \leftarrow h_t^d$ ▷ Clean state
  end for
end for
 # Train RAD on full sequence
$\mathbf{z} = \mathbf{x} + \epsilon$, $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
for $d=1$ to D **do** ▷ DiT layer
  $\mathbf{z}^d \leftarrow A_d(\mathbf{z})$ ▷ Parallel attention
  $\mathbf{z} \leftarrow R_d(\mathbf{z}^d, \mathbf{H}[d,:])$ ▷ Parallel RNN
end for
return MSE$(\mathbf{z}, \mathbf{v}_{gt})$

memory during prefetching, a design choice that accelerates training convergence (see Sec. 6.1 for empirical analysis).

Concretely, for frame-wise autoregression, we first apply the RAD model with a window size of 1 across the clean frames of each training sequence. This step independently computes all RNN hidden states $\{h_t^d\}_{t\in[T]}^{d\in[D]}$ across frames in sequence and DiT layers, where each h_t^d aggregates contextual information up to t-th frame and d-th DiT layer. Assuming the previous hidden state stacks sufficient context information, the pre-fetched hidden states are equivalent to the standard DiT with RNN by sliding-window. However, after pre-fetching all hidden states, the attention modules within each window—despite the sequential RNN update—can now be processed in parallel. By setting the prefetched hidden states for RNN layers at corresponding positions, we can compute the attention modules in RAD with a normal window size as usual to get the diffusion loss for the entire sequence. In our experiments, we do not calculate the diffusion losses for all sub-sequences (length l) in given sample (length L), but randomly sample partial of them to reduce computation cost.

5 Experiments

5.1 Datasets and Evaluation Protocol

Maze Dataset For our small-scale experimental setting, we employ the Memory Maze dataset [33], which consists of approximately 30000 training videos, each depicting agent navigation within a 15×15 maze environment and comprising 1000 frames. For the Maze Dataset, all models were trained from scratch during the training process.

For evaluation, we collect an additional set of 200 maze videos, with frame counts ranging from 100 to 300. In each sequence, the agent traverses from a designated start position to a target location and then returns along the same path. We define the initial 60% of frames—corresponding to the outbound trajectory—as the contextual input, while the remaining 40% serve as the prediction targets for model evaluation. This setup is specifically designed to rigorously test the model's capacity for long-term memory and context utilization. The aerial visualization of the maze data can be found in the Appendix.

Minecraft Dataset For large-scale experiments, we utilize the MineRL [19] to generate 20000 training sequences following the protocols in VRAG [12]. Each video contains 1200 frames. For evaluation, we collect 60 sequences incorporating distinct action patterns, such as rotation in place, which are intended to probe the model's memory capabilities under diverse behaviors. To ensure fair comparison, we first train a standard diffusion forcing model on the entire 20000-sample training set. Subsequently, this pretrained base model is fine-tuned with different RAD-RNN architectures and training paradigms, each for the same 5000 optimization steps. This procedure enables a controlled assessment of architectural and methodological differences under consistent pretraining conditions.

Evaluation Metrics We employ three widely adopted metrics to quantitatively evaluate model performance: Structural Similarity Index (SSIM) [47], which assesses spatial consistency in generated frames; Peak Signal-to-Noise Ratio (PSNR), which measures pixel-level reconstruction fidelity; and Learned Perceptual Image Patch Similarity (LPIPS) [55], which evaluates perceptual similarity. Since our evaluation datasets are specially tailored for long-term memory tasks, these metrics that directly compare against ground-truth videos can accurately reflect the capacity of different methods in memory retention and maintaining spatiotemporal consistency.

Baselines For the baselines, we select Diffusion Forcing [10], Teacher Forcing, and VRAG [12]. Additionally, TTT-c can be regarded as an implementation of LaCT [56]. All baselines and our proposed RAD method adopt identical data settings: models are trained from scratch on the Maze dataset, while on the Minecraft dataset, they are fine-tuned starting from a pre-trained model.

5.2 Maze Results

RNN Methods We compare LSTM, Mamba2, and TTT within the RAD architecture as described in Sec. 4.2, trained under identical settings for three epochs. Table 2 summarizes the results for both chunk-wise ("-c") and frame-wise ("-f") autoregressive modes.

In the chunk-wise setting, LSTM delivers the strongest performance across all metrics, improving notably over both the Diffusion Forcing (DF) baseline and the other recurrent variants. Mamba2-c and TTT-c, in contrast, perform worse than the baseline in PSNR and LPIPS, which focus more on image quality. This behavior reflects the burden placed on the recurrent module: without overlapped attention, all pixel-level continuity between chunks must be carried through hidden states or memory weights alone. LSTM benefits structurally from its separation of short-term

Table 2: Experiment results on Maze Dataset: chunk-wise ("-c") and frame-wise ("-f") autoregressive modes

Model	RNN Type	PSNR ↑	SSIM ↑	LPIPS ↓
DF	None	14.73	0.32	0.51
TF	None	14.91	0.36	0.48
VRAG	None	15.55	0.41	**0.43**
RAD	Mamba2-c	13.80	0.31	0.58
	TTT-c	14.36	0.35	0.53
	LSTM-c	**15.64**	**0.43**	0.47
RAD	Mamba2-f	15.35	0.41	0.51
	TTT-f	15.50	0.41	0.52
	LSTM-f	15.50	0.41	0.45

Table 3: Experiment results on Minecraft dataset: chunk-wise ("-c") and frame-wise ("-f") autoregressive modes

Model	RNN Type	PSNR ↑	SSIM ↑	LPIPS ↓
DF	None	15.65	0.45	0.53
TF	None	16.11	0.44	0.50
VRAG	None	16.11	**0.49**	0.50
RAD	Mamba2-c	12.72	0.33	0.63
	TTT-c	14.04	0.38	0.56
	LSTM-c	14.24	0.39	0.55
RAD	Mamba2-f	16.70	0.46	0.47
	TTT-f	**16.72**	0.46	0.47
	LSTM-f	16.59	0.46	**0.46**

memory (previous output y_{t-1}) and long-term memory (cell state C_{t-1}), which aligns well with this requirement. Mamba2 and TTT—designed for global compression rather than fine-grained pixel transport—struggle in comparison.

In the frame-wise setting, all recurrent variants perform similarly. With a sliding step of 1, consecutive frames share an attention window, allowing pixel-level information to propagate directly through attention rather than via hidden states. This eliminates the main failure mode of Mamba2-c and TTT-c, enabling Mamba2-f, TTT-f, and LSTM-f to reach comparable quality with only small metric differences. Correspondingly, LSTM's structural advantage is diminished, as its explicit short-memory y_{t-1} pathway becomes less necessary when attention already provides strong local continuity. Meanwhile, compared with the baseline, the RAD model integrated with the RNN mechanism completely outperforms both Diffusion Forcing and Teacher Forcing. Even when compared to the VRAG method that explicitly models historical frames, the frame-wise RAD model achieves comparable performance.

Frame-wise vs. Chunk-wise Autoregression The cross-paradigm comparison in Tab. 2 highlights how the autoregressive design dictates the relative strength of each recurrent architecture.

In the **chunk-wise** mode, the absence of local cross-chunk attention forces hidden states to serve as the sole channel for transmitting pixel-level information. This creates excessive demands on its capacity for information storage and retrieval with the memory mechanism: it must encode both global memory and the local scene details needed to maintain visual consistency. Under this constraint, architectural differences become pronounced. Mamba2 and TTT underperform because they are not optimized to shuttle high-frequency information through memory alone, whereas LSTM's disentanglement of local (via y_{t-1}) and global (via C_{t-1}) information is highly compatible with the chunk-wise sliding training and inference paradigm. As a result, LSTM-c achieves the strongest performance.

In the **frame-wise** mode, attention spans all consecutive frames, restoring direct pixel-level communication. Therefore, Hidden states focus mostly on global information, reducing the need for local detail retention. Once this burden is lifted, Mamba2 and TTT improve substantially and converge in performance with LSTM.

The recurrent pathway in LSTM becomes partly redundant, which explains the disappearance of its earlier advantage.

Overall, the results indicate that the suitability of an RNN for autoregressive diffusion depends strongly on the temporal granularity of attention: when attention cannot bridge boundaries (chunk-wise), architectures with explicit local–global separation excel; when attention is continuous (frame-wise), architectural differences matter far less. We argue that hidden states could focus more on global information, while high-frequency temporal local information ought to be transmitted primarily through local attention.

5.3 Minecraft Results

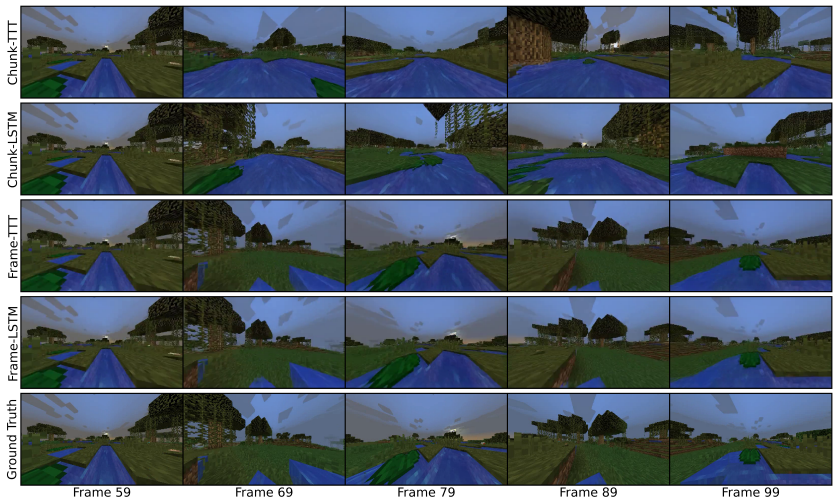

Fig. 5: Visualization results on Minecraft dataset. Frame-wise RNN can effectively memorize and reconstruct scenes, maintaining consistency with the ground truth. In contrast, due to the lack of overlap attention, Chunk-wise RNN is unable to reconstruct historical scenes in datasets with higher information density

Frame-wise vs. Chunk-wise Autoregression Table 3 presents the results of applying chunk-wise and frame-wise RAD with different RNN variants to the Minecraft dataset. The trends broadly match those observed on the Maze dataset, but with important differences driven by the substantially higher visual and structural complexity of Minecraft videos.

In the **chunk-wise** setting, LSTM again outperforms Mamba2 and TTT, consistent with its structural ability to balance short-term and long-term memory. However, unlike in the Maze experiments, all RNN variants fall short of the standard Diffusion Forcing baseline. Minecraft scenes contain dense textures, rich geometry, and rapid viewpoint changes, making it difficult for any recurrent hidden state to fully compress

and transmit pixel-level information across chunk boundaries. As a result, the architectural limitations of relying solely on hidden states for inter-chunk communication become much more pronounced, leading to degraded reconstruction quality and weaker scene consistency.

In the **frame-wise** setting, the RAD models with three different RNN types exhibit similar performance, significantly surpassing the chunk-wise counterparts, because local visual information can propagate directly through attention, removing the need to encode high-frequency details in the hidden state. Under this more favorable regime, all three RNN types achieve significantly better performance than their chunk-wise counterparts.

Qualitative results in Fig. 5 highlight this contrast. Frame-wise TTT and frame-wise LSTM produce videos that maintain strong memory and stay aligned with ground truth across long horizons. Conversely, chunk-wise TTT and chunk-wise LSTM exhibit clear failures in memorizing scene layout and object configuration, reinforcing the difficulty of relying solely on hidden states to carry rich Minecraft-level detail across chunks.

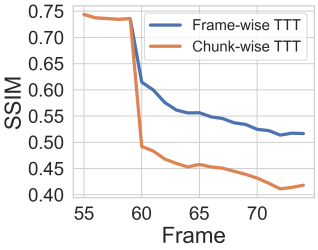

Fig. 6: Comparison of chunk-wise TTT and frame-wise TTT on SSIM metric

Overall, these experiments further support the conclusion that the granularity of the autoregressive window plays a critical role. Chunk-wise autoregression imposes an unrealistic compression burden for visually complex environments, while frame-wise autoregression leverages attention to maintain local fidelity, allowing all RNN architectures to operate on more global signals and achieve substantially better results.

To provide a more detailed comparison between chunk-wise and frame-wise RNN approaches, we present the SSIM value curves as a function of frame index in Fig. 6. Notably, both methods exhibit a decrease in performance at the 60-th frame, corresponding to the first frame predicted by the model. However, this drop is substantially more pronounced for the chunk-wise RNN, underscoring its limited capacity to effectively convey pixel-level information across chunk boundaries.

5.4 VBench Results

Table 4: VBench evaluation results for different autoregressive model variants.

Metrics	Maze Dataset					Minecraft Dataset				
	Background Consistency	Temporal Flickering	Motion Smoothness	Aesthetic Quality	Imaging Quality	Background Consistency	Temporal Flickering	Motion Smoothness	Aesthetic Quality	Imaging Quality
LSTM-c	**91.29**	**93.49**	71.4	**26.29**	51.44	97.42	93.96	95.12	57.58	**69.34**
Mamba-c	91.07	93.07	69.98	25.51	50.7	97.27	93.79	94.96	57.22	69.27
TTT-c	90.89	92.21	70.71	26	**51.6**	**97.44**	93.91	95.06	57.55	69.15
LSTM-f	91.16	93.01	**71.48**	26.11	50.94	97.38	**94.21**	**95.3**	55.71	66.72
Mamba-f	90.78	92.97	70.78	25.76	50.94	97.37	94.17	95.26	**55.73**	66.4
TTT-f	90.99	93.2	70.87	25.82	51.08	97.38	94.14	95.25	55.64	66.26

We add VBench [27] evaluation results beyond just pixel similarity. As shown in Tab. 4, the results on VBench demonstrate similar outcomes to those in Sec. 5.2 and

Sec. 5.3. The LSTM-based RNN achieves better performance, enabling the generation of higher-quality videos.

5.5 Computational Resource Analysis

Directly comparing the computational efficiency of different RNN architectures is inherently challenging, as their implementations and underlying optimizations can vary substantially. For example, LSTM benefits from extensive low-level optimizations, such as those provided by cuDNN, whereas TTT currently lacks such specialized enhancement. Despite these differences, we report a summary of computational resource costs for reference. All reported results are obtained from experiments on the Minecraft dataset, conducted using 8 NVIDIA L40 GPUs and a batch size of 3.

Table 5: Comparison of computational resource requirements among different RNN block types. "Time" denotes the duration of a single training step for an RAD model (same DiT) with the specified RNN type, while "Params" indicates the total number of parameters in the RNN block.

RNN Type	GPU Mem (GB)	Time (s/step)	Params (M)
LSTM	10.0	3.7	195
Mamba2	11.1	3.6	153
TTT	16.0	12.9	118

As presented in Tab. 5, although the LSTM architecture features the highest parameter count, its GPU memory consumption and training time per step are comparable to those of Mamba2. These discrepancies in parameterization reflect intrinsic differences in model design rather than unfair experimental conditions, permitting a reasonable assessment.

Meanwhile, to quantify the computational overhead introduced by our method, we also theoretically calculate the Flops cost of each component as in Tab. 6.

Table 6: Theoretical FLOPs analysis of one forward pass in the proposed two-stage pipeline.

Component	Formula	FLOPs	Fraction
Stage 1 (RNN)	$L(BN)Tn_{\text{layers}}8D^2$	5.1×10^{13}	35.7%
Stage 1 (Attention)	$L(4(BT)ND^2+2(BT)N^2D)$	2.5×10^{13}	17.5%
Stage 2 (Diffusion)	$L\Big(4(B_2W)ND^2+2(B_2W)N^2D\Big)$ $+L\Big(4(B_2N)WD^2+2(B_2N)W^2D\Big)$	6.7×10^{13}	46.9%

Where L denotes the number of DiT layers, N is the token length, W represents the window size, and T is the number of video frames. We can observe that the additional computational overhead introduced by prefetching mainly comes from the attention computation in Stage 1, which accounts for less than 20% of the total computational cost. Diffusion Forcing baseline has equivalent computational cost to only the Stage 2 diffusion forward pass.

6 Ablation study

6.1 Noise Level of Memory Frames

We conducted an ablation study to evaluate the impact of noise levels applied to memory frames that are fed into hidden states of RNN during training. Our

Table 7: Evaluation results for LSTM-f with clean and noised frame memory.

Metrics	noised	clean
PSNR ↑	14.70	**15.30**
SSIM ↑	0.38	**0.41**
LPIPS ↓	0.52	**0.50**

Table 8: Strided chunk-wise autoregression for LSTM on Maze dataset.

RNN Type	Stride	PSNR ↑	SSIM ↑	LPIPS ↓
LSTM-c	20	**15.64**	**0.43**	0.47
LSTM-f	1	15.50	0.41	**0.45**
LSTM-partial-overlap	10	15.06	0.40	0.55

findings indicate that introducing higher noise levels makes model optimization more challenging, leading to significantly higher training loss and degraded evaluation metrics, as illustrated in Tab. 7. These results highlight the importance of our design choice: all clean frames used for memory are pre-processed in the pre-fetching stage, slid by a DiT model with a window size of 1 and decoupled from the denoising stage.

This approach shares some similarities with the strategy of Po et al. [36], where the initial n frames of each training sequence are left clean and excluded from the diffusion loss calculation. However, our method is more comprehensive, where all frames passed into the RNN hidden states remain clean, rather than solely the initial portion. Moreover, our approach does not interfere with the computation of the diffusion loss, further enhancing training efficiency.

6.2 Strided Chunk-wise Autoregression

We further investigate the partially overlap attention window in chunk-wise RNN, by keeping the same model size and training data as Sec. 5.2, while setting the stride of the sliding window to 10 frames (consistent during both training and inference). In other words, for frame-wise RNN, the stride is 1; for chunk-wise RNN, the stride equals the window size (20). The experimental results, as shown in Tab 8, indicate that this method does not show advantage compared to both frame-wise RNN and chunk-wise RNN.

7 Discussion and Conclusion

In this work, we present RAD, a unified framework for long-term video generation that augments the DiT architecture with RNN memory blocks. By systematically comparing LSTM, Mamba2, and TTT within the RAD framework, we find that LSTM achieves strong and robust performance for challenging long-range video synthesis tasks. We further analyze the importance of memory update paradigms, and demonstrate that switching from chunk-wise to explicit frame-wise autoregressive generation with overlapping context windows dramatically improves spatiotemporal consistency and video fidelity, with the RAD framework striking a good balance between local context details and global memory. Although the performance advantage is verified, we found that there are still limits of the current method: the additional RNN blocks introduce noticeable computational overhead compared to standard diffusion models, which leads to increased training and inference time. Meanwhile, the constant-sized hidden state of the RNN has a theoretical storage upper bound, which may restrict the model performance under ultra-long context scenarios.

References

1. Assran, M., Bardes, A., Fan, D., Garrido, Q., Howes, R., Mojtaba, Komeili, Muckley, M., Rizvi, A., Roberts, C., Sinha, K., Zholus, A., Arnaud, S., Gejji, A., Martin, A., Hogan, F.R., Dugas, D., Bojanowski, P., Khalidov, V., Labatut, P., Massa, F., Szafraniec, M., Krishnakumar, K., Li, Y., Ma, X., Chandar, S., Meier, F., LeCun, Y., Rabbat, M., Ballas, N.: V-jepa 2: Self-supervised video models enable understanding, prediction and planning (2025), https://arxiv.org/abs/2506.09985
2. Azzolini, A., Brandon, H., Chattopadhyay, P., Chen, H., Chu, J., Cui, Y., Diamond, J., Ding, Y., Ferroni, F., Govindaraju, R., et al.: Cosmos-reason1: From physical common sense to embodied reasoning. arXiv preprint arXiv:2503.15558 (2025)
3. Ball, P.J., Bauer, J., Belletti, F., Brownfield, B., Ephrat, A., Fruchter, S., Gupta, A., Holsheimer, K., Holynski, A., Hron, J., Kaplanis, C., Limont, M., McGill, M., Oliveira, Y., Parker-Holder, J., Perbet, F., Scully, G., Shar, J., Spencer, S., Tov, O., Villegas, R., Wang, E., Yung, J., Baetu, C., Berbel, J., Bridson, D., Bruce, J., Buttimore, G., Chakera, S., Chandra, B., Collins, P., Cullum, A., Damoc, B., Dasagi, V., Gazeau, M., Gbadamosi, C., Han, W., Hirst, E., Kachra, A., Kerley, L., Kjems, K., Knoepfel, E., Koriakin, V., Lo, J., Lu, C., Mehring, Z., Moufarek, A., Nandwani, H., Oliveira, V., Pardo, F., Park, J., Pierson, A., Poole, B., Ran, H., Salimans, T., Sanchez, M., Saprykin, I., Shen, A., Sidhwani, S., Smith, D., Stanton, J., Tomlinson, H., Vijaykumar, D., Wang, L., Wingfield, P., Wong, N., Xu, K., Yew, C., Young, N., Zubov, V., Eck, D., Erhan, D., Kavukcuoglu, K., Hassabis, D., Gharamani, Z., Hadsell, R., van den Oord, A., Mosseri, I., Bolton, A., Singh, S., Rocktäschel, T.: Genie 3: A new frontier for world models (2025), https://deepmind.google/blog/genie-3-a-new-frontier-for-world-models/, accessed: 2026-06-29
4. Bar, A., Zhou, G., Tran, D., Darrell, T., LeCun, Y.: Navigation world models. arXiv preprint arXiv:2412.03572 (2024)
5. Bardes, A., Garrido, Q., Ponce, J., Chen, X., Rabbat, M., LeCun, Y., Assran, M., Ballas, N.: Revisiting feature prediction for learning visual representations from video (2024), https://arxiv.org/abs/2404.08471
6. Blattmann, A., Dockhorn, T., Kulal, S., Mendelevitch, D., Kilian, M., Lorenz, D., Levi, Y., English, Z., Voleti, V., Letts, A., et al.: Stable video diffusion: Scaling latent video diffusion models to large datasets. arXiv preprint arXiv:2311.15127 (2023)
7. Blattmann, A., Rombach, R., Ling, H., Dockhorn, T., Kim, S.W., Fidler, S., Kreis, K.: Align your latents: High-resolution video synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 22563–22575 (2023)
8. Brooks, T., Peebles, B., Holmes, C., DePue, W., Guo, Y., Jing, L., Schnurr, D., Taylor, J., Luhman, T., Luhman, E., Ng, C., Wang, R., Ramesh, A.: Video generation models as world simulators (2024), https://openai.com/research/video-generation-models-as-world-simulators, accessed: 2026-06-29
9. Che, H., He, X., Liu, Q., Jin, C., Chen, H.: Gamegen-x: Interactive open-world game video generation. arXiv preprint arXiv:2411.00769 (2024)
10. Chen, B., Martí Monsó, D., Du, Y., Simchowitz, M., Tedrake, R., Sitzmann, V.: Diffusion forcing: Next-token prediction meets full-sequence diffusion. Advances in Neural Information Processing Systems **37**, 24081–24125 (2024)
11. Chen, H., Zhang, Y., Cun, X., Xia, M., Wang, X., Weng, C., Shan, Y.: Videocrafter2: Overcoming data limitations for high-quality video diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7310–7320 (2024)
12. Chen, T., Hu, X., Ding, Z., Jin, C.: Learning world models for interactive video generation (2025), https://arxiv.org/abs/2505.21996

13. Cui, J., Wu, J., Li, M., Yang, T., Li, X., Wang, R., Bai, A., Ban, Y., Hsieh, C.J.: Self-forcing++: Towards minute-scale high-quality video generation. arXiv preprint arXiv:2510.02283 (2025)
14. Dalal, K., Koceja, D., Xu, J., Zhao, Y., Han, S., Cheung, K.C., Kautz, J., Choi, Y., Sun, Y., Wang, X.: One-minute video generation with test-time training. In: 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 17702–17711 (2025). https://doi.org/10.1109/CVPR52734.2025.01649
15. Dao, T., Gu, A.: Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In: International Conference on Machine Learning (ICML) (2024)
16. Decart, Etched, Quevedo, J., McIntyre, Q., Campbell, S., Chen, X., Wachen, R.: Oasis: A universe in a transformer (2024), https://oasis-model.github.io/, accessed: 2026-06-29
17. Gu, A., Dao, T.: Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752 (2023)
18. Guo, J., Ye, Y., He, T., Wu, H., Jiang, Y., Pearce, T., Bian, J.: Mineworld: a real-time and open-source interactive world model on minecraft. arXiv preprint arXiv:2504.08388 (2025)
19. Guss, W.H., Houghton, B., Topin, N., Wang, P., Codel, C., Veloso, M., Salakhutdinov, R.: Minerl: A large-scale dataset of minecraft demonstrations. arXiv preprint arXiv:1907.13440 (2019)
20. Ha, D., Schmidhuber, J.: Recurrent world models facilitate policy evolution. Advances in neural information processing systems **31** (2018)
21. Hafner, D., Lillicrap, T., Norouzi, M., Ba, J.: Mastering atari with discrete world models. arXiv preprint arXiv:2010.02193 (2020)
22. Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., Fleet, D.J.: Video diffusion models. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems. vol. 35, pp. 8633–8646. Curran Associates, Inc. (2022), https://proceedings.neurips.cc/paper_files/paper/2022/file/39235c56aef13fb05a6adc95eb9d8d66-Paper-Conference.pdf
23. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
24. Hong, Y., Wei, J., Liu, X., Wang, X., Bai, Y., Li, H., Zhang, M., Xu, H.: Cogvideo: Large-scale pretraining for text-to-video generation with transformers. arXiv preprint arXiv:2205.15868 (2022)
25. Hu, A., Russell, L., Yeo, H., Murez, Z., Fedoseev, G., Kendall, A., Shotton, J., Corrado, G.: Gaia-1: A generative world model for autonomous driving. arXiv preprint arXiv:2309.17080 (2023)
26. Huang, X., Li, Z., He, G., Zhou, M., Shechtman, E.: Self forcing: Bridging the train-test gap in autoregressive video diffusion (2025), https://arxiv.org/abs/2506.08009
27. Huang, Z., He, Y., Yu, J., Zhang, F., Si, C., Jiang, Y., Zhang, Y., Wu, T., Jin, Q., Chanpaisit, N., Wang, Y., Chen, X., Wang, L., Lin, D., Qiao, Y., Liu, Z.: VBench: Comprehensive benchmark suite for video generative models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2024)
28. Kang, B., Yue, Y., Lu, R., Lin, Z., Zhao, Y., Wang, K., Huang, G., Feng, J.: How far is video generation from world model: A physical law perspective. arXiv preprint arXiv:2411.02385 (2024)
29. Kingma, D.P., Welling, M.: Auto-encoding variational bayes (2022), https://arxiv.org/abs/1312.6114
30. Liao, Y., Zhou, P., Huang, S., Yang, D., Chen, S., Jiang, Y., Hu, Y., Cai, J., Liu, S., Luo, J., Chen, L., Yan, S., Yao, M., Ren, G.: Genie envisioner: A unified world foundation platform for robotic manipulation. arXiv preprint arXiv:2508.05635 (2025)

31. NVIDIA, :, Agarwal, N., Ali, A., Bala, M., Balaji, Y., Barker, E., Cai, T., Chattopadhyay, P., Chen, Y., Cui, Y., Ding, Y., Dworakowski, D., Fan, J., Fenzi, M., Ferroni, F., Fidler, S., Fox, D., Ge, S., Ge, Y., Gu, J., Gururani, S., He, E., Huang, J., Huffman, J., Jannaty, P., Jin, J., Kim, S.W., Klár, G., Lam, G., Lan, S., Leal-Taixe, L., Li, A., Li, Z., Lin, C.H., Lin, T.Y., Ling, H., Liu, M.Y., Liu, X., Luo, A., Ma, Q., Mao, H., Mo, K., Mousavian, A., Nah, S., Niverty, S., Page, D., Paschalidou, D., Patel, Z., Pavao, L., Ramezanali, M., Reda, F., Ren, X., Sabavat, V.R.N., Schmerling, E., Shi, S., Stefaniak, B., Tang, S., Tchapmi, L., Tredak, P., Tseng, W.C., Varghese, J., Wang, H., Wang, H., Wang, H., Wang, T.C., Wei, F., Wei, X., Wu, J.Z., Xu, J., Yang, W., Yen-Chen, L., Zeng, X., Zeng, Y., Zhang, J., Zhang, Q., Zhang, Y., Zhao, Q., Zolkowski, A.: Cosmos world foundation model platform for physical ai (2025), https://arxiv.org/abs/2501.03575
32. Parker-Holder, J., Ball, P., Bruce, J., Dasagi, V., Holsheimer, K., Kaplanis, C., Moufarek, A., Scully, G., Shar, J., Shi, J., Spencer, S., Yung, J., Dennis, M., Kenjeyev, S., Long, S., Mnih, V., Chan, H., Gazeau, M., Li, B., Pardo, F., Wang, L., Zhang, L., Besse, F., Harley, T., Mitenkova, A., Wang, J., Clune, J., Hassabis, D., Hadsell, R., Bolton, A., Singh, S., Rocktäschel, T.: Genie 2: A large-scale foundation world model (2024), https://deepmind.google/discover/blog/genie-2-a-large-scale-foundation-world-model/, accessed: 2026-06-29
33. Pasukonis, J., Lillicrap, T., Hafner, D.: Evaluating long-term memory in 3d mazes. arXiv preprint arXiv:2210.13383 (2022)
34. Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 4195–4205 (2023)
35. Peng, X., Zheng, Z., Shen, C., Young, T., Guo, X., Wang, B., Xu, H., Liu, H., Jiang, M., Li, W., Wang, Y., Ye, A., Ren, G., Ma, Q., Liang, W., Lian, X., Wu, X., Zhong, Y., Li, Z., Gong, C., Lei, G., Cheng, L., Zhang, L., Li, M., Zhang, R., Hu, S., Huang, S., Wang, X., Zhao, Y., Wang, Y., Wei, Z., You, Y.: Open-sora 2.0: Training a commercial-level video generation model in $200k. arXiv preprint arXiv:2503.09642 (2025)
36. Po, R., Nitzan, Y., Zhang, R., Chen, B., Dao, T., Shechtman, E., Wetzstein, G., Huang, X.: Long-context state-space video world models (2025), https://arxiv.org/abs/2505.20171
37. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents (2022), https://arxiv.org/abs/2204.06125
38. Ren, X., Lu, Y., Cao, T., Gao, R., Huang, S., Sabour, A., Shen, T., Pfaff, T., Wu, J.Z., Chen, R., Kim, S.W., Gao, J., Leal-Taixe, L., Chen, M., Fidler, S., Ling, H.: Cosmos-drive-dreams: Scalable synthetic driving data generation with world foundation models (2025), https://arxiv.org/abs/2506.09042
39. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10684–10695 (June 2022)
40. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation (2015), https://arxiv.org/abs/1505.04597
41. Song, K., Chen, B., Simchowitz, M., Du, Y., Tedrake, R., Sitzmann, V.: History-guided video diffusion (2025), https://arxiv.org/abs/2502.06764
42. Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., Liu, Y.: Roformer: Enhanced transformer with rotary position embedding. Neurocomputing **568**, 127063 (2024)
43. Sun, Y., Li, X., Dalal, K., Xu, J., Vikram, A., Zhang, G., Dubois, Y., Chen, X., Wang, X., Koyejo, S., Hashimoto, T., Guestrin, C.: Learning to (learn at test time): Rnns with expressive hidden states (2025), https://arxiv.org/abs/2407.04620
44. team, W.L.: Generating bigger and better worlds (2025), https://www.worldlabs.ai/blog/bigger-better-worlds, accessed: 2026-06-29
45. Valevski, D., Leviathan, Y., Arar, M., Fruchter, S.: Diffusion models are real-time game engines. arXiv preprint arXiv:2408.14837 (2024)

46. Wang, H., Ma, C.Y., Liu, Y.C., Hou, J., Xu, T., Wang, J., Juefei-Xu, F., Luo, Y., Zhang, P., Hou, T., et al.: Lingen: Towards high-resolution minute-length text-to-video generation with linear computational complexity. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 2578–2588 (2025)
47. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004)
48. Watter, M., Springenberg, J., Boedecker, J., Riedmiller, M.: Embed to control: A locally linear latent dynamics model for control from raw images. Advances in neural information processing systems **28** (2015)
49. Wu, J., Yin, S., Feng, N., He, X., Li, D., Hao, J., Long, M.: ivideogpt: Interactive videogpts are scalable world models. Advances in Neural Information Processing Systems **37**, 68082–68119 (2024)
50. Xiao, Z., Lan, Y., Zhou, Y., Ouyang, W., Yang, S., Zeng, Y., Pan, X.: Worldmem: Long-term consistent world simulation with memory (2025), https://arxiv.org/abs/2504.12369
51. Yang, Z., Teng, J., Zheng, W., Ding, M., Huang, S., Xu, J., Yang, Y., Hong, W., Zhang, X., Feng, G., et al.: Cogvideox: Text-to-video diffusion models with an expert transformer. arXiv preprint arXiv:2408.06072 (2024)
52. Yin, T., Zhang, Q., Zhang, R., Freeman, W.T., Durand, F., Shechtman, E., Huang, X.: From slow bidirectional to fast autoregressive video diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 22963–22974 (June 2025)
53. Yu, J., Bai, J., Qin, Y., Liu, Q., Wang, X., Wan, P., Zhang, D., Liu, X.: Context as memory: Scene-consistent interactive long video generation with memory retrieval. arXiv preprint arXiv:2506.03141 (2025)
54. Yu, J., Qin, Y., Wang, X., Wan, P., Zhang, D., Liu, X.: Gamefactory: Creating new games with generative interactive videos. arXiv preprint arXiv:2501.08325 (2025)
55. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)
56. Zhang, T., Bi, S., Hong, Y., Zhang, K., Luan, F., Yang, S., Sunkavalli, K., Freeman, W.T., Tan, H.: Test-time training done right (2025), https://arxiv.org/abs/2505.23884
57. Zheng, Z., Peng, X., Yang, T., Shen, C., Li, S., Liu, H., Zhou, Y., Li, T., You, Y.: Open-sora: Democratizing efficient video production for all. arXiv preprint arXiv:2412.20404 (2024)