Transformer-Based Diffusion for Game Generation

Betty Li Hou* Jack Lu* Georgy Savva*

Abstract

We explore the application of transformer-based diffusion models for game environment generation, investigating multiple approaches for conditioning on past frames and actions: Video Generation (VG) to predict the whole video sequence conditioned on past actions, and Single Frame Generation (SFG) to predict a single frame conditioned on past frames and actions, with both concatenation and cross-attention mechanisms. Using the ViZDoom My Way Home environment as our test environment, we demonstrate that while SFG models achieve superior performance in teacher-forcing scenarios with PSNR values up to 32.21, VG models show better stability in autoregressive generation, suggesting important tradeoffs between model architecture and performance.

1. Introduction

Computer games are carefully designed software systems that operate on the following game loop: Collect user inputs, update the game state, and render the visuals on screen. This loop, running at high frame rates, gives players the impression of an interactive virtual environment. The process of updating the game state and rendering is governed by a set of manually programmed rules unique to each game, which is carefully crafted and requires intense scrutiny.

The use of generative models as game engines offers a promising new direction for game design. Rather than manual programming, the models could generate the environment, as well as add new levels or change game mechanics. Moreover, having a robust learned simulation of a game can serve as a world model for future agent training. Recent years have seen several attempts to make this work; however, only the most recent attempt, (Valevski et al., 2024) has been able to achieve sufficiently high rendering results on a fairly complex game, DOOM. The leap in rendering quality and consistency was achieved by using the state-of-the-art image generation model, Stable Diffusion (Rombach et al., 2022). Stable Diffusion has become the standard in media generation as it can produce high-quality images conditioned on multi-modal inputs, such as text and images, and uses the U-Net CNN architecture as the diffusion backbone. Recent work has shown that the transformer architecture can be used as the diffusion backbone in Stable Diffusion, referred to as Diffusion Transformers (DiTs) (Peebles & Xie, 2022), with stronger performance and scalability. As such, we seek to apply DiTs for game generation, experimenting with different methods of implementation and conditioning.

*Equal contribution .

2. Related Works

2.1. Game Generation

World models have been introduced as a way to simulate game environments through a combination of a variational auto-encoder (VAE) to learn over input frames (vision capabilities) and a recurrent neural network (RNN) to predict future distributions given past information (memory capabilities). This has been tested and shown to be effective in the VizDoom environment (Ha & Schmidhuber, 2018). Generative models are able to simulate games as demonstrated by GameGAN (Kim et al., 2020), which uses a combination of LSTM and GAN. It has been proposed that world models can be learned in latent space, since compact representations of the game state may improve the efficiency of planning and simulation tasks (Hafner et al., 2019). Recent work has demonstrated the effectiveness of Stable Diffusion models in generating realistic game states to serve as world models (Alonso et al., 2024).

2.2. Diffusion Transformers

Advancements in diffusion models have introduced techniques for replacing traditional networks with transformer-based models. Vision Transformers (ViT) can capture long-range dependencies, which makes them promising for use in place of the U-net to improve stability and coherence (Peebles & Xie, 2022). Diffusion Transformers (DiTs) are based on the ViT architecture to operate on sequences of image patches and have been demonstrated as a scalable diffusion architecture. In addition to being more powerful, it allows more flexibility to condition on longer sequences, which is important for game generation.

3. Diffusion Game Simulation

We select ViZDooM My Way Home (Kempka et al., 2016) as the game environment to generate and evaluate our new methods on. In this game scenario, the agent must navigate a labyrinth to find a vest which finishes the game. The map is always the same and the vest remains in the same room, and rooms are of different colors, interconnected with corridors. The player is spawned in a randomly chosen room facing a random direction. There are 5 actions available: turn left, turn right, move forward, move left, move right.

Our method is as follows:

1. Generate a dataset of gameplay data: We generate episodes of gameplay using a random policy with action repeat (for smoother trajectories). An episode consists of frame and action pairs.

2. Fine-tune a pre-trained VAE decoder on single game frames in order to reconstruct the visual elements of ViZDoom.

3. Train a DiT-based diffusion model in VAE latent space: We randomly sample segments of a fixed length n, ensuring that the segment lies within the episode.

4. Evaluation: We take an episode from the test dataset and generate a trajectory based on the actions from it using our models both auto-regressively and with teacher forcing. In each case, the first generated frame has the index $n-1$ to ensure that the generated trajectory is seeded with the first frames of the ground truth episode. We measure the performance of the models using PSNR.

4. Method

This section discusses various diffusion model architectures for conditionally predicting new frames on past frames and past actions[1]. We denote batch size with B, hidden embedding dimension with M, and the channel, height, and width dimensions of VAE latent as C, H, W.

4.1. Video Generation

In this approach, the diffusion model is trained to predict n frames (the whole video sequence) conditioned on the corresponding actions of the first $n-1$ frames but not the frames themselves.

During training, we take a segment of an episode of length n, patchify the VAE latents of each frame into tokens, add forward diffusion noise to all tokens of all n frames, add learnable frame embeddings of dimension M that specify the timestep of each frame within the sequence, and add learnable action embeddings of dimension M that specify the action of each frame. Figure 1 shows how each image is processed. All concatenated tokens across frames are fed into unmodified DiT blocks. Figure 2 shows the overall Video Generation diffusion model architecture.

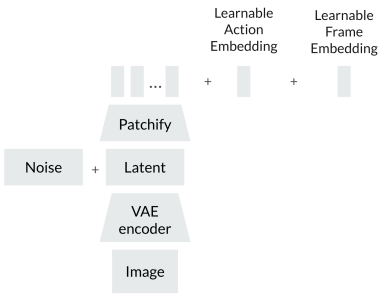

Figure 1. Before feeding into the DiT, each image is processed by patchifying the VAE latents into tokens, adding diffusion noise, and learnable action embeddings.

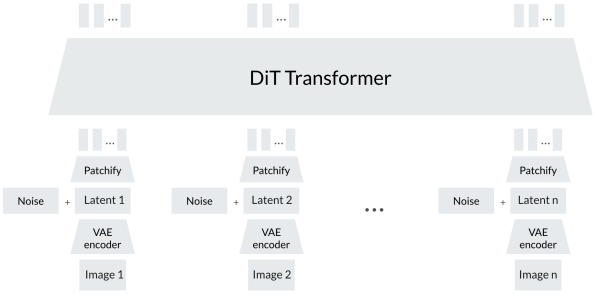

Figure 2. Overall architecture of the Video Generation model.

During inference, we condition our model on the past $n-1$ frames with the RePaint method with $r=1$ and $j=1$. We mask out the tokens corresponding to the last frame and perform video completion based on the past $n-1$ frames. Specifically, at each reverse diffusion step, we add forward diffusion noise to the $n-1$ frames' clean tokens and only keep the output tokens of the final frame.

4.2. Single Frame Generation

In this approach, the diffusion model is trained to predict a single frame at a time. During training, we take a segment of an episode of length n, add forward diffusion noise to the final image in the segment, and optimize the model to denoise the final image while conditioning on the past $n-1$ frames and corresponding actions. During inference, we keep a rolling window of $n-1$ past frames and corresponding actions to conditionally sample the next frame from the trained model. We try two different conditioning mecha-

[1]GitHub repo with implementations here: https://github.com/georgysavva/diffusion-for-simulation.

nisms for the $n-1$ past frames and actions—concatenation conditioning and cross-attention conditioning—described below. Figure 3 illustrates these two architectures.

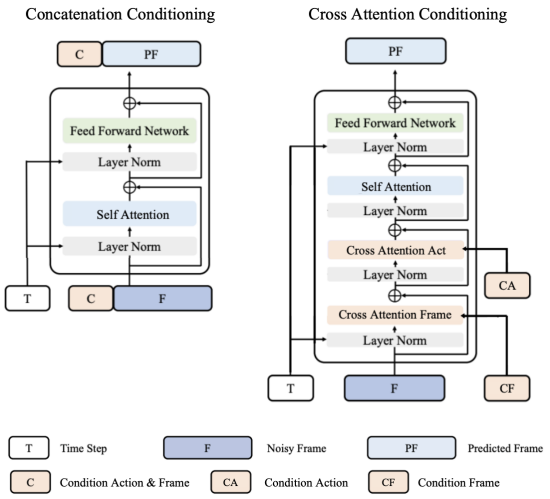

Figure 3. Two methods of single frame generation: concatenation conditioning and cross attention conditioning

4.2.1. Concatenation Conditioning

We concatenate the $n-1$ past frames' VAE latents with the new frame's latent to form a single tensor of shape B, n, C, H, W. We patchify each frame's latent into tokens of hidden dimension M, add forward diffusion noise to the tokens of the final frame, add corresponding action embeddings to tokens from the first $n-1$ frames, and concatenate all tokens of all frames into a pool of tokens. Finally, we input all tokens into unmodified DiT blocks with self-attention, discard all output tokens corresponding to the first $n-1$ frames because their inputs were clean tokens, and compute loss on the token noise predictions for the final frame. Since DiT blocks use the self-attention mechanism, this method's runtime scales quadratically to n.

4.2.2. Cross Attention Conditioning

Rather than feeding all clean tokens of the past $n-1$ frames into the DiT block, we only feed the noisy tokens of the final frame into the DiT block and condition on the past frame's clean tokens with the cross-attention mechanism. We add two cross-attention blocks to each DiT block to attend to the $n-1$ past frames and their corresponding actions separately: the query vectors from the final frame and the key, and value vectors from either the past frame clean tokens or action embeddings. Since the number of query tokens is fixed to be the number of tokens from the last frame, this method's runtime scales linearly to n.

5. Experiments & Results

We collect 2000 episodes for training and 200 for testing. Episodes contain up to 2100 frames, and shorter if the agent finishes the game earlier (this occurrs in 12.5% of episodes). Due to resource constraints, we base our experiments on a relatively short number of past frames, up to 7. We run experiments by training DiT-B/4 from scratch or fine-tuning the publicly available pre-trained DiT XL-2. We keep the frame resolution to be 256 by 256. Table 5 shows the PSNR values for all methods, between Video Generation and Single Frame Generation, B/4 trained from scratch and XL-2 pretrained, and teacher forcing and auto-regressive models.

Method	Teacher Forcing	Auto-Regressive
VG B/4	24.53	15.09
VG XL-2	21.19	14.24
SFG / Concatenation B/4	32.21	14.10
SFG / Concatenation XL-2	26.55	12.66
SFG / Cross-Attention B/4	29.45	14.54
SFG / Cross-Attention XL-2	24.82	13.64

Table 1. PSNR values for Video Generation (VG) and Single Frame Generation (SFG), with either Concatenation or Cross-Attention conditioning, under Teacher Forcing and Auto Regressive settings.

We observe that the video generation model performs much better at autoregressive inference than the models trained to predict a single frame. However, the latter is better at teacher-forcing trajectory generation and produces trajectories with consistency, almost indistinguishable from the ground-truth trajectory.

Another trend we observe is that the B/4 models we train from scratch achieve better performance than the larger pre-trained models with less training time, therefore using a pre-trained model did not provide significant advantage.

Comparing concatenation and cross-attention conditioning for single frame generation, we observe that the concatenation technique is more computationally expensive and requires 4x more GPU memory. Moreover, at least for this small number of past frames, cross-attention conditioning attains a better performance than concatenation conditioning.

We include generated images and videos in Section 6, along with the ground truth to compare. Additional experiment details and results are shown in Appendix A.

6. Generated Results

The following shows 10 generated frames under each method, with Figure 4 as the ground truth frames.

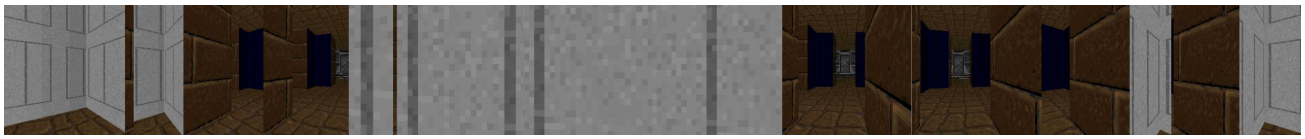

Figure 4. Ground truth. Click for video.

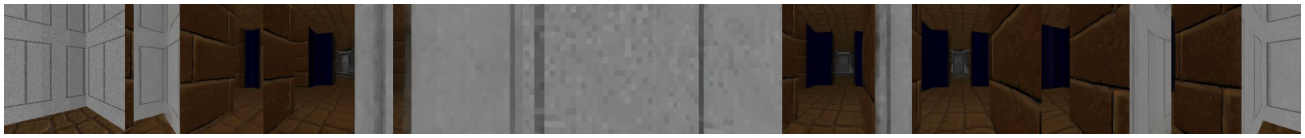

Figure 5. Video Generation B/4 teacher forcing. Click for video.

Figure 6. Video Generation B/4 autoregressive. Click for video.

Figure 7. Single Frame Generation Concatenation B/4 teacher forcing. Click for video.

Figure 8. Single Frame Generation Concatenation B/4 autoregressive. Click for video.

Figure 9. Single Frame Generation Cross-Attention B/4 teacher forcing. Click for video.

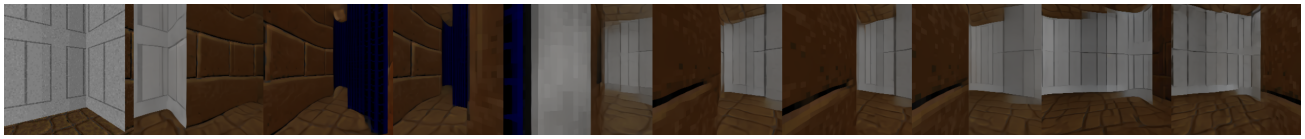

Figure 10. Single Frame Generation Cross-Attention B/4 autoregressive. Click for video.

7. Limitations & Future Work

Context length Due to time and GPU constraints, the current models do not incorporate much of the past information in the episode. In order to work with longer sequences, we will continue exploring cross attention conditioning as it scales linearly with respect to the number of past steps. We will also try compressing past frame representations into lower dimensions. Since DOOM images are not as complex as real world images, we believe further encoding past context into lower dimensions before conditioning the diffusion models with it may improve memory and training time efficiency. Another approach is to use more scalable attention mechanisms such as Infini Attention (Munkhdalai et al., 2024) and Ring Attention (Liu et al., 2023).

Autoregressive collapse When simulating DOOM with diffusion models, only the Video Generation models were able to produce meaningful results for autoregressive trajectory generation, while other models collapsed to repeating similar frames quickly. A possible resolution to this problem is once the model has reached a reasonable performance with the existing training scheme, we can train it autoregressively with some past frames generated from itself. This way, the model will learn to correct its errors when performing autoregressive generation at test time.

Combining video generation with past conditioning In Video Generation, we denoise a stack of frames in order to achieve autoregressive generation capabilities; however, this is very computationally expensive as we must treat the past frames and the current one exactly the same, and cannot use cross attention or dimensionality reduction techniques. Future work should explore a way to combine Video Generation techniques for the last m frames with a more efficient conditioning technique, such as cross attention or encoding for the n frames that were before ($m << n$). This approach would combine the robust autoregressive generation properties of video models together with the scalability and flexibility of various conditioning techniques.

8. Conclusion

We explore methods of transformer-based diffusion models for game environment generation, demonstrating that our Single Frame Generation method is stronger at teacher-forcing trajectory generation, while our Video Generation model has better autoregressive inference performance. For Single Frame Generation, conditioning by cross-attention achieves better performance than concatenation conditioning. In order to handle longer sequence lengths and address autoregressive collapse, future work should explore more scalable attention mechanisms, approaches combining video generation with efficient past conditioning, and techniques for improving autoregressive generation stability. These findings provide important directions for advancing the field of game environment simulation using generative models.

Contribution Statement

BLH, JL, and GS contributed equally to this work. JL led the Video Generation method and experiments, GS led the Single Frame Generation method and experiments, and BLH oversaw project organization. All team members contributed to ideation and discussion throughout.

References

Alonso, E., Jelley, A., Micheli, V., Kanervisto, A., Storkey, A., Pearce, T., and Fleuret, F. Diffusion for world modeling: Visual details matter in atari. *arXiv preprint arXiv:2405.12399*, 2024.

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

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

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. Vizdoom: A doom-based ai research platform for visual reinforcement learning, 2016. URL https://arxiv.org/abs/1605.02097.

Kim, S. W., Zhou, Y., Philion, J., Torralba, A., and Fidler, S. Learning to simulate dynamic environments with gamegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1231–1240, 2020.

Liu, H., Zaharia, M., and Abbeel, P. Ring attention with blockwise transformers for near-infinite context, 2023. URL https://arxiv.org/abs/2310.01889.

Munkhdalai, T., Faruqui, M., and Gopal, S. Leave no context behind: Efficient infinite context transformers with infini-attention, 2024. URL https://arxiv.org/abs/2404.07143.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752.

Valevski, D., Leviathan, Y., Arar, M., and Fruchter, S. Diffusion models are real-time game engines, 2024. URL https://arxiv.org/abs/2408.14837.

A. Experiment Details

METHOD	TEST LOSS	LEARNING RATE	BATCH SIZE	TRAINING STEPS	# CONDITIONING STEPS
VG B-4	0.058	1E-4	128	150,000	7
VG XL-2	0.03	1.25E-5	16	130,00	3
SFG / CONCATENATION B-4	0.13	1.0E-4	128	280,00	7
SFG / CONCATENATION XL-2	0.14	6.25E-6	8	66,000	7
SFG / CROSS-ATTENTION B-4	0.14	2.0E-4	256	94,000	7
SFG / CROSS-ATTENTION XL-2	0.14	1.75E-5	32	216,000	4

Table 2. Configuration details.