DIFFUSION MODELS ARE REAL-TIME GAME ENGINES

Dani Valevski, Yaniv Leviathan, Moab Arar, Shlomi Fruchter

ICLR 2025: The Thirteenth International Conference on Learning Representations

AIM

Demonstrate that a neural model running in real-time can simulate a complex game at high quality.

INTERACTIVE WORLD SIMULATION

An Interactive Environment $\mathcal{E}$ consists of:

S: a space of latent states

O: a space of observations of the latent space

$V : S \rightarrow O$: a partial projection function

A: a set of actions

$p(s|a, s')$ *such that* $s, s' \in S$, $a \in A$: a transition probability function

INTERACTIVE WORLD SIMULATION

E: an input interactive environment

$s_0 \in S$: an initial state

$q(o_n | o_{<n}, a_{<n}), o_i \in O, a_i \in A$

INTERACTIVE WORLD SIMULATION

$D : O \times O \rightarrow R$: a distance metric between observations

$\pi(a_n | o_{<n}, a_{<n})$: a policy as human gameplay

S_0: a distribution on initial states

N_0 : a distribution on episode lengths

$E(D(o_q^i, o_p^i))$ where $n \sim N_0$, $0 \leq i \leq n$, and $o_q^i \sim q$,

$o_p^i \sim V(p)$

GAMENGEN

GameNGen is a generative diffusion model that learns to simulate the game under the settings of Interactive World Simulation.

Agent records all its actions and observations with teacher forcing.

Generative trained by Agent collected data.

GAMENGEN

DATA COLLECTION VIA AGENT PLAY

1. Player hit: -100 points.

2. Player death: -5,000 points.

3. Enemy hit: 300 points.

4. Enemy kill: 1,000 points.

5. Item/weapon pick up: 100 points.

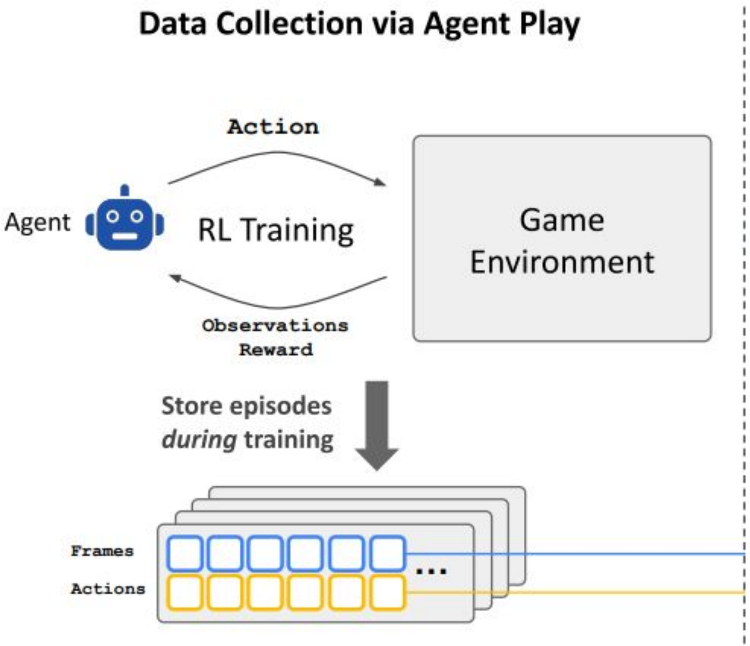
6. Secret found: 500 points.

7. New area: 20 * (1 + 0.5 * L1 distance) points.

8. Health delta: 10 * delta points.

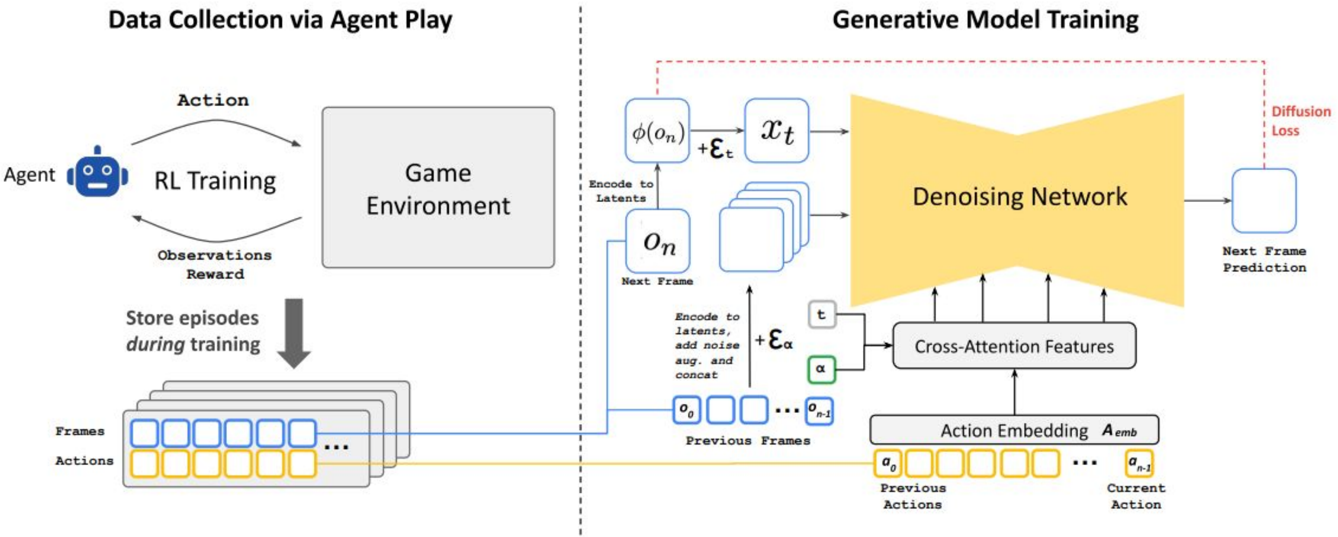
9. Armor delta: 10 * delta points.

10. Ammo delta: 10 * max(0, delta) + min(0, delta) points.



GAMENGEN

TRAINING THE GENERATIVE DIFFUSION MODEL



$$\mathcal{L} = \mathbb{E}_{t,\epsilon,T}\left[\|v(\epsilon, x_0, t) - v_{\theta'}(x_t, t, \{\phi(o_{i<n})\}, \{A_{emb}(a_{i<n})\})\|_2^2\right] \tag{1}$$

where $T = \{o_{i\leq n}, a_{i\leq n}\} \sim \mathcal{T}_{agent}$, $t \sim \mathcal{U}(0,1)$, $\epsilon \sim \mathcal{N}(0,\mathbf{I})$, $x_t = \sqrt{\bar\alpha_t}x_0 + \sqrt{1-\bar\alpha_t}\epsilon$, $x_0 = \phi(o_n)$, $v(\epsilon, x_0, t) = \sqrt{\bar\alpha_t}\epsilon - \sqrt{1-\bar\alpha_t}x_0$, and $v_{\theta'}$ is the v-prediction output of the model f_θ. The noise schedule $\bar\alpha_t$ is linear, similarly to Rombach et al. (2022).

GAMENGEN

TRAINING THE GENERATIVE DIFFUSION MODEL

Figure 4: **Auto-regressive drift.** Top: we present every 10th frame of a simple trajectory with 50 frames in which the player is not moving. Quality degrades fast after 20-30 steps. Bottom: the same trajectory with noise augmentation does not suffer from quality degradation.

GAMENGEN

INFERENCE

Setup:

Used DDIM Sampling.

Employed Classifier-Free Guidance only for the past observations.

Did not find guidance for the past actions to improve quality.

The weight of Classifier-Free Guidance was 1.5.

GAMENGEN

INFERENCE

Table 1: **Generation with Varying Sampling Steps.** We evaluate the generation quality of a GameNGen model with an increasing number of steps using PSNR and LPIPS metrics. "D" marks a 1-step distilled model. See Appendix A.6 for more details.

Steps	PSNR $\uparrow$	LPIPS $\downarrow$
D	31.10 ± 0.098	0.208 ± 0.002
1	25.47 ± 0.098	0.255 ± 0.002
2	31.91 ± 0.104	0.205 ± 0.002
4	32.58 ± 0.108	0.198 ± 0.002
8	32.55 ± 0.110	0.196 ± 0.002
16	32.44 ± 0.110	0.196 ± 0.002
32	32.32 ± 0.110	0.196 ± 0.002
64	32.19 ± 0.110	0.197 ± 0.002

EXPERIMENTAL SETUP

AGENT TRAINING

- trained using PPO with a simple CNN
- trained on CPU using the Stable Baselines 3 infrastructure
- provided with downscaled versions of the frame images and in-game map, each at resolution 160x120
- also had access to the last 32 actions it performed
- the feature network computes a representation of size 512 for each image
- PPO's actor and critic are 2-layer MLP heads on top of a concatenation of the outputs of the image feature network and the sequence of past actions
- trained the agent to play the game using the ViZDoom environment
- run 8 games in parallel, each with a replay buffer size of 512, a discount factor γ = 0.99, and an entropy coefficient of 0.1

  In each iteration, the network is trained using a batch size of 64 for 10 epochs, with a learning rate of 1e-4. We perform a total of 50M environment steps

EXPERIMENTAL SETUP

GENERATIVE MODEL TRAINING

- trained all simulation models from a pretrained checkpoint of Stable Diffusion 1.4, unfreezing all U-Net parameters
- used a batch size of 128 and a constant learning rate of 2e-5, with the Adafactor optimizer without weight decay and gradient clipping of 1.0
- the context frames condition is dropped with probability 0.1 to allow CFG during inference
- trained using 128 TPU-v5e devices with data parallelization
- unless noted otherwise, all results in the paper are after 700,000 training steps
- for noise augmentation, used a maximal noise level of 0.7, with 10 embedding buckets
- used a batch size of 2,048 for optimizing the latent decoder, other training parameters are identical to those of the denoiser
- for training data, used a random subset of 70M examples from the recorded trajectories played by the agent during RL training and evaluation
- fll image frames are at a resolution of 320x240 padded to 320x256
- used a context length of 64

RESULTS

SIMULATION QUALITY

PSNR: 29.43, LPIPS: 0.249

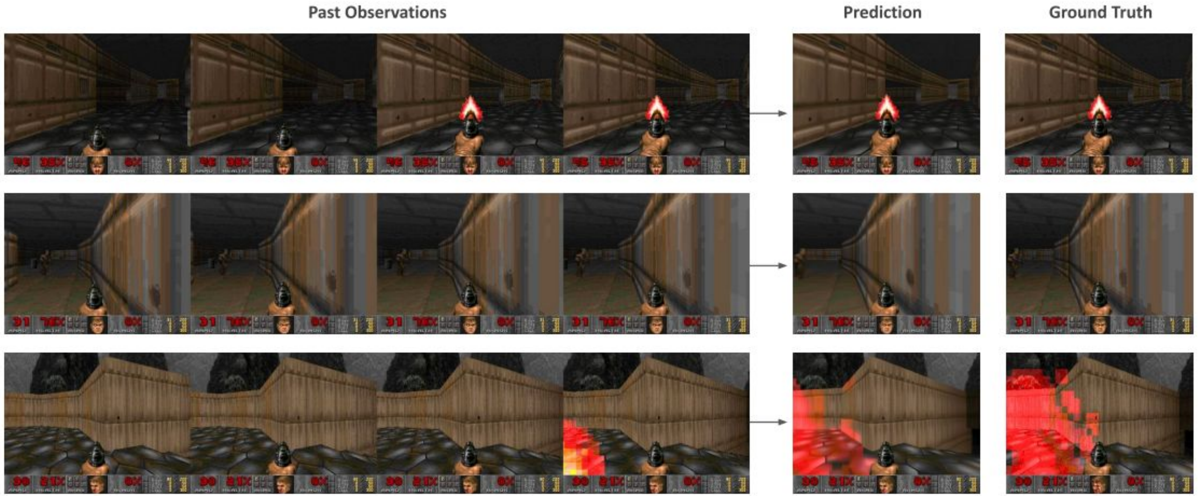


Figure 5: **Model predictions vs. ground truth.** Only the last 4 frames of the past observations context are shown.

RESULTS

SIMULATION QUALITY

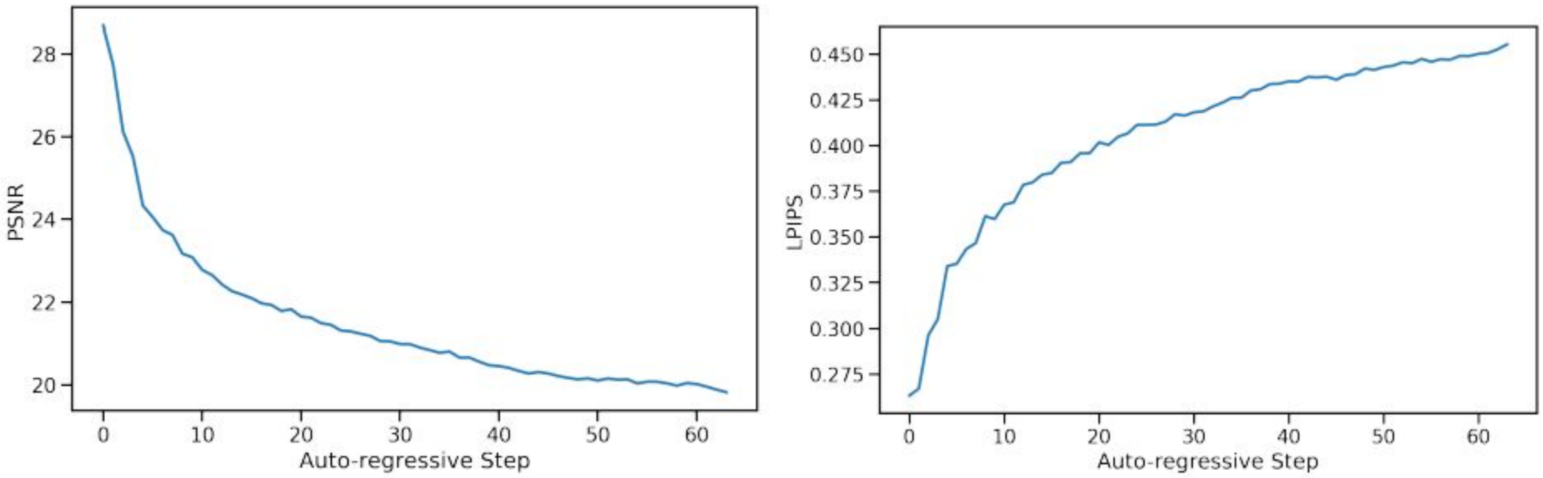


Figure 6: **Auto-regressive evaluation.** PSNR and LPIPS metrics over 64 auto-regressive steps.

RESULTS

SIMULATION QUALITY

1.6s → 58%, 3.2s → 60%, 3s after 5 or 10 minutes → 50%

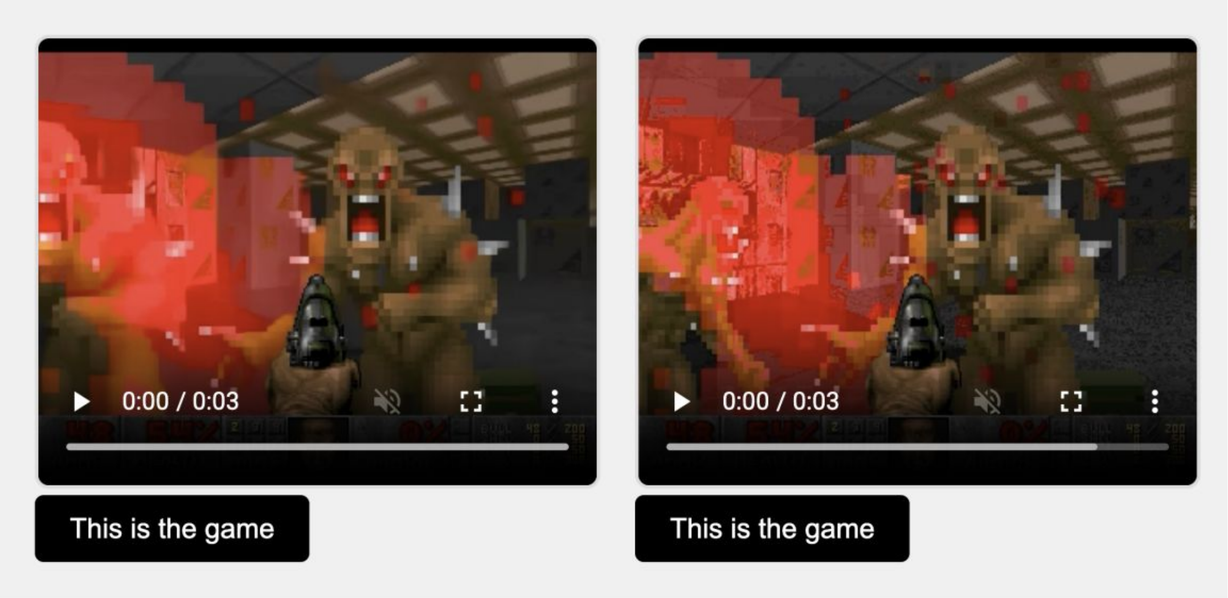


Figure 17: A screenshot of the tool used for human evaluations (see Section 5.1).

RESULTS

ABLATIONS

Table 2: **Number of history frames.** We ablate the number of history frames used as context using 8912 test-set examples from 5 levels. More frames generally improve both PSNR and LPIPS metrics.

History Context Length	PSNR ↑	LPIPS ↓
64	22.36 ± 0.033	0.295 ± 0.001
32	22.31 ± 0.033	0.296 ± 0.001
16	22.28 ± 0.033	0.296 ± 0.001
8	22.26 ± 0.033	0.296 ± 0.001
4	22.26 ± 0.034	0.298 ± 0.001
2	22.03 ± 0.037	0.304 ± 0.001
1	20.94 ± 0.044	0.358 ± 0.001

RESULTS

ABLATIONS

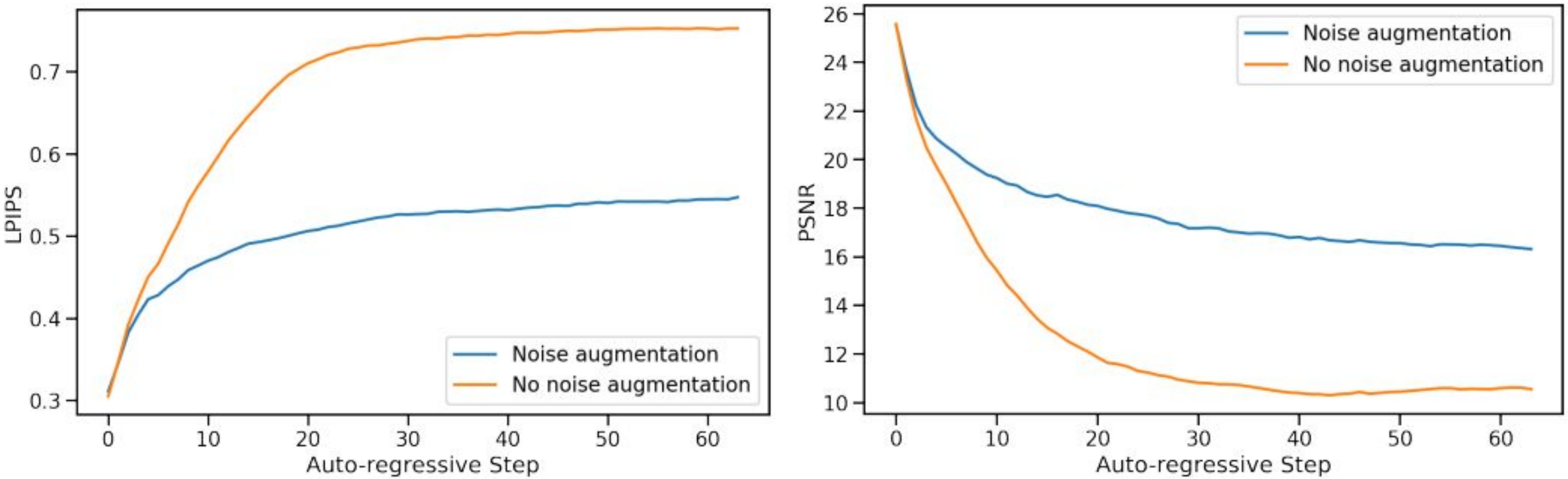


Figure 7: **Impact of Noise Augmentation.** The plots show average LPIPS (lower is better) and PSNR (higher is better) values for each auto-regressive step. When noise augmentation is not used quality degrades quickly after 10-20 frames. This is prevented by noise augmentation.

RESULTS

ABLATIONS

Table 3: **Performance on Different Difficulty Levels.** We compare the performance of models trained using Agent-generated and Random-generated data across easy, medium, and hard splits of the dataset. Easy and medium have 112 items, hard has 232 items. Metrics are computed for each trajectory on a single frame after 3 seconds.

Difficulty Level	Data Generation Policy	PSNR ↑	LPIPS ↓
Easy	Agent	20.94 ± 0.76	0.48 ± 0.01
	Random	20.20 ± 0.83	0.48 ± 0.01
Medium	Agent	20.21 ± 0.36	0.50 ± 0.01
	Random	16.50 ± 0.41	0.59 ± 0.01
Hard	Agent	17.51 ± 0.35	0.60 ± 0.01
	Random	15.39 ± 0.43	0.61 ± 0.00

DISCUSSION

Summary:

High-quality real-time gameplay at 20 frames per second is possible on a neural model. However, there are some limitations.

Limitations:

GameNGen suffers from a limited amount of memory.

There are differences between the agent’s behaviour and those of human players.

We are not able to easily produce new games with GameNGen.

Future Work:

Nothing in their technique is DOOM-specific except for the reward function for the RL agent.

Thank you for your attention.