運用 Diffusion Models 實現即時 Atari 遊戲畫面生成

Real-Time Atari Game Simulation with Diffusion Models

指導教授：朱威達
專題成員：李逵安
開發工具：Python, Cursor, PyTorch, HuggingFace Diffusers, Stable Baselines 3
測試環境：Ubuntu 24.04 LTS with GeForce RTX 4090 graphics card

一、簡介：

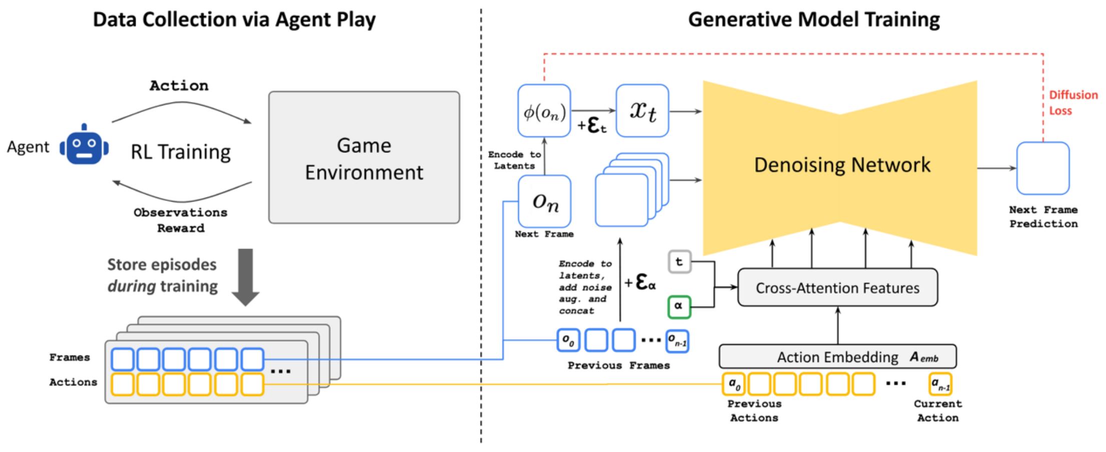

Figure 1: GameNGen method overview.

This project explores the application of diffusion models for real-time video game simulation, inspired by *Diffusion Models Are Real-Time Game Engines* (Valevski et al., 2024). We reimplement the core system, GameNGen, a neural game engine that replaces traditional rendering and game logic with a generative diffusion model conditioned on past gameplay frames and player actions. Unlike the original work, which demonstrates high-fidelity simulation of DOOM on TPU hardware, our implementation targets computationally simpler environments using classic Atari 2600 games such as Pong and Breakout to accommodate limited hardware resources. We collect (frame, action) trajectories via reinforcement learning agents and train a modified version of Stable Diffusion to predict the next frame autoregressively. To ensure stable long-horizon generation, we integrate techniques such as noise-augmented conditioning and truncated context lengths. Our results demonstrate the feasibility of simulating interactive game environments using diffusion models even under strict resource constraints, highlighting the generalizability and scalability of GameNGen-style neural engines to broader gaming domains.

二、測試結果：

Our implementation was evaluated on a GeForce RTX 4090 GPU using a Pong game simulation, trained over 628k steps with a batch size of 16.

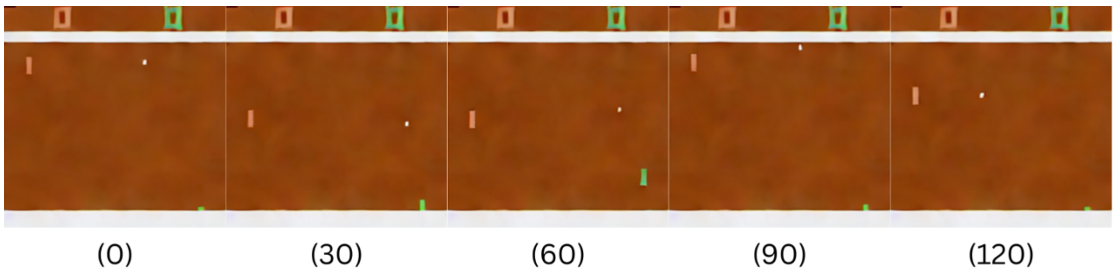

Figure 2: A trajectory of the Pong gameplay, with every 30th frame displayed across a total of 120 frames.

To assess visual fidelity, we measured PSNR and LPIPS scores by sampling initial states from 2,048 held-out gameplay sequences and predicting the next frame based on previous observations and actions. Our model achieved PSNR scores of **33.59** and **34.65**, and LPIPS scores of **0.13** and **0.04**, when using 4 and 8 denoising steps, respectively.

	Ours(4 steps)	Ours(8 steps)	JPEG Compression
PSNR ↑	33.59 ± 0.115	34.65 ± 0.133	**35.63 ± 0.074**
LPIPS ↓	0.128 ± 0.006	**0.047 ± 0.005**	0.07 ± 0.003

Table 1: PSNR and LPIPS scores of our model under different denoising steps compared with lossy JPEG compression (quality ≈ 20).

Overall, our model produces image quality on par with JPEG compression at a quality level of approximately 20, consistent with findings reported in the original paper.