DiffusionArcade: Diffusion-based framework for real-time simulation of Atari games

Damian Kopp (324944), Colin Smyth (376857), Adriana Orellana (376792), Tim Arni (274586)
CS-503 Project Proposal

***Abstract*—Traditional game engines rely on hand-coded logic to simulate game states, requiring significant manual effort and domain expertise. In this project, we explore an alternative approach by training a diffusion model to generate future game frames conditioned on past frames and user actions. To create training data, a reinforcement learning agent will first learn to play Atari games like *Pong*, generating diverse gameplay trajectories. The trained diffusion model will then serve as a neural game engine, with performance evaluated using PSNR, LPIPS, and human ratings to assess visual quality and playability.**

I. Introduction

A. What is the problem you want to solve?

Traditional game engines follow a structured process. First, they collect user inputs. Based on these inputs, the game state is updated through predefined logic, and the updated state is then rendered on the screen, creating an interactive experience for the player. Such computer games are mostly rule-based and involve physics simulations, predefined animations, and rigid game logic.

Recent progress in generative models, particularly in diffusion models, has shown that they can produce high-quality images and videos [1], and can be easily conditioned using multi-modal inputs such as text or images. Based on this, we aim to explore whether diffusion models can be used as real-time game engines by predicting the next game frame based on a sequence of previous frames and user actions.

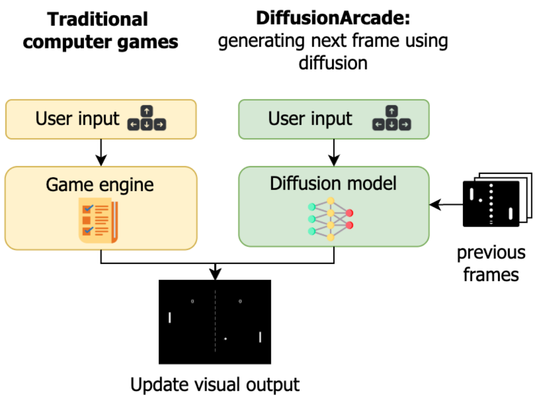

Figure 1. Comparison between traditional computer game rendering and the proposed solution (DiffusionArcade)

Potential Research Questions

1) How does using a diffusion model for real-time game simulation impact the visual quality of game frames?
2) How does the number of previous frames affect the quality of generated frames in simple 2D games, and can accurate predictions be achieved with only a short temporal history?
3) To what extent can a diffusion model trained on one Atari game generalize to simulate environments in different Atari games with similar dynamics?

B. Why is it important that this problem be solved?

Using a neural network that can autonomously update the game state and render the next frame introduces a more flexible approach to designing interactive environments. This approach not only reduces the amount of manual work needed to build games but also marks a significant paradigm shift: games become sets of learned parameters rather than lines of code. Additionally, this project offers an excellent learning opportunity to deepen our understanding of world models and diffusion models, as well as learning how to train a reinforcement learning agent. As discussed in [2], world models provide a viable path toward more efficient and safer agent training. Basing these models on diffusion aligns with current trends in the field, as indicated in [1].

II. Method and Deliveries

A. How do you solve the problem?

In this project, we will train a diffusion model to generate the next frame in simple Atari-style games, such as Pong or Breakout. To achieve this, we need a dataset that captures how players interact with the game over time. To create this, we will first train a reinforcement learning (RL) agent to play the game, using PLE [1] or ALE [2]. The advantage of using an RL agent for data generation is that it will enable us to potentially generalize to other games with minimal changes to the setup and also to solve the problem that collecting large-scale human gameplay data is often very expensive. Moreover, to ensure the diffusion-based game engine is engaging and playable by humans, the agent's trajectories should also approximate human-like gameplay. Consequently, we will design a reward function that promotes exploration while still encouraging the agent to win the game. As the RL agent trains under this reward scheme, we will record the frames and actions from each session. Then, we will use this data to train a diffusion model (using

[1]https://pygame-learning-environment.readthedocs.io/
[2]https://github.com/Farama-Foundation/Arcade-Learning-Environment

Hugging Face's Diffusers package[3]) that predicts frames based on past frames and actions.

B. How will you validate your solution?

To validate the performance of our model, we will use two quantitative metrics: the Peak Signal-to-Noise Ratio (PSNR), which measures the quality of a reconstructed image compared to the original one, and the Learned Perceptual Image Patch Similarity (LPIPS), which evaluates the perceptual similarity between images. These metrics will help us assess the image quality of our model and how well the predicted frames match what we expect. We will also conduct a qualitative evaluation with 5 human raters who will assess (1) the visual quality of the generated frames and (2) the overall playability of the game.

C. Milestone 1

The fact that our project consists of two models allows us to focus initially on building a working RL agent that learns to play the game in an exploratory fashion. At this point, our goal is to explore and compare numerous reward functions that encourage exploration while also guiding the agent to win the game. We will experiment with a simplified version of the reward function described in [3], applied to our specific problem. We will also explore the ideas outlined in [4], finding a balance between extrinsic and intrinsic rewards.

We will begin training the RL agent on a simple game like *Pong*. This represents an interesting case since the diffusion model should learn to reproduce the ball's behavior (e.g. bouncing when it hits a wall). While this involves implicit physics, the model is expected to capture it directly from the training data.

By the end of this milestone, we expect to have a diverse dataset of game frames and the corresponding actions taken by the agent.

D. Milestone 2

The second part of the project will consist of training a generative diffusion model using the data collected in the first part. The model will learn to generate the next frame given a fixed number of previous frames and associated actions. To generate the latent vectors for the game frames, we will use a pre-trained Variational autoencoder (VAE) and explore whether fine-tuning the decoder with the observations collected with the RL agent improves the quality of the generated frames. In this milestone, we will also test different numbers of past frames (context length) and analyze how this affects the generation quality. Finally, we will evaluate the model's generalization capabilities by testing it on another game with similar dynamics.

[3]https://huggingface.co/docs/diffusers

III. Related Work

Diffusion models have shown strong capabilities in generating high-quality images and videos. For example, Sora [1] creates realistic video sequences from text prompts, demonstrating the potential of such models for frame generation in interactive environments like games.

GameNGen [3] proposes the methodology we aim to implement: an RL agent is trained to generate gameplay data, which is then used to train a diffusion model for real-time simulation. The authors successfully applied this pipeline to recreate Doom using a neural game engine.

DIAMOND [2] shows that diffusion models can serve as accurate world models for training RL agents. Although our approach reverses this order by training the agent first, we will adopt DIAMOND's insights on stabilizing diffusion-based generation to ensure high-quality, consistent frame predictions in our real-time Atari setup.

IV. Discussion

A. Implications and Broader Impact

This concept could extend to higher-resolution and 3D games, pushing forward the development of neural network-based game engines. Training RL agents entirely "in imagination" reduces the need for hand-coded simulations and aligns with the trend of using diffusion-based world models for more efficient and safer RL agent training. Replacing code with learned parameters in game creation enables novel user experiences and lower development costs, but also raises ethical concerns, such as the potential for violent or discriminatory content and the displacement of game development jobs.

B. Potential Risks and Shortcomings

If the initial RL agent fails to generate a representative dataset, progress on milestone 2 will be hindered, as the diffusion model depends on high-quality, diverse data to ensure robustness, even in rare gameplay scenarios. While models like Stable Diffusion v1.4 should run on a single A100 GPU and be sufficient in quality for *Pong*, scaling to more complex games will increase compute demands. Finally, although generalization to multiple Atari games is a goal, the project may end up focused solely on *Pong* due to time constraints.

References

[1] Y. Liu, K. Zhang, Y. Li, Z. Yan, C. Gao, R. Chen, Z. Yuan, Y. Huang, H. Sun, J. Gao, L. He, and L. Sun, "Sora: A review on background, technology, limitations, and opportunities of large vision models," 2024. [Online]. Available: https://arxiv.org/abs/2402.17177

[2] E. Alonso, A. Jelley, V. Micheli, A. Kanervisto, A. Storkey, T. Pearce, and F. Fleuret, "Diffusion for world modeling: Visual details matter in atari," 2024. [Online]. Available: https://arxiv.org/abs/2405.12399

[3] D. Valevski, Y. Leviathan, M. Arar, and S. Fruchter, “Diffusion models are real-time game engines,” 2024. [Online]. Available: https://arxiv.org/abs/2408.14837

[4] S. Singh, A. Barto, and N. Chentanez, “Intrinsically motivated reinforcement learning.” 01 2004.