Game Level Generation with Agent-Discriminator GAN

Team 23

20190688 Joowon Han

1. Introduction

The video game industry is expanding rapidly, as well as the technologies related to it. Artificial intelligence for video games is one of such branches; one major application is game-playing AI, which ranges from decision making as seen in the renown AlphaGo [1] to the complex behavior in non-playing characters.

In 2020, NVIDIA presented a generative adversarial network (GAN) that learns from gameplay videos to generate actual, playable *Pac-Man* levels [2]. Inpired from this breakthrough, this project aims to find an effective method to generate a realistic video game level based on a playing agent as well as a traditional image discriminator.

2. Method

The baseline model for the project was the Generative Playing Network proposed by Bontrager and Togelius on 2021 [1].

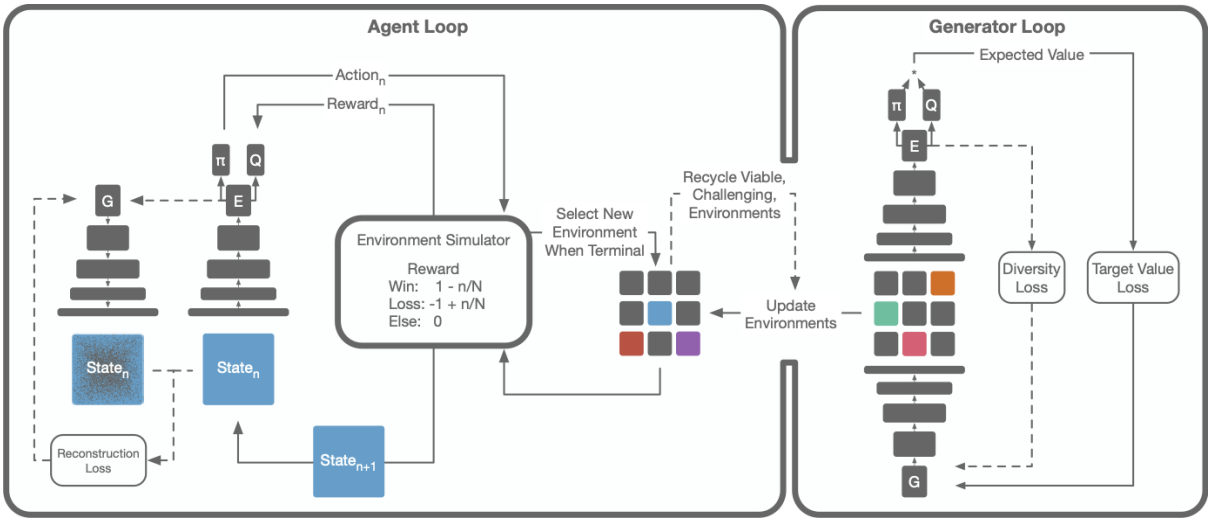

Figure 1. Generative Playing Networks (Image credit to corresponding author)

This network is a variation of GAN, where the discriminator position is replaced with a game-playing agent. The agent is rewarded or penalized based on its actions while playing through a given level. The project seeks improvement on this model by adding another network to this structure. Specifically, our model attempts to additionally optimize the generator by combining the traditional image-wise discriminator to the network.

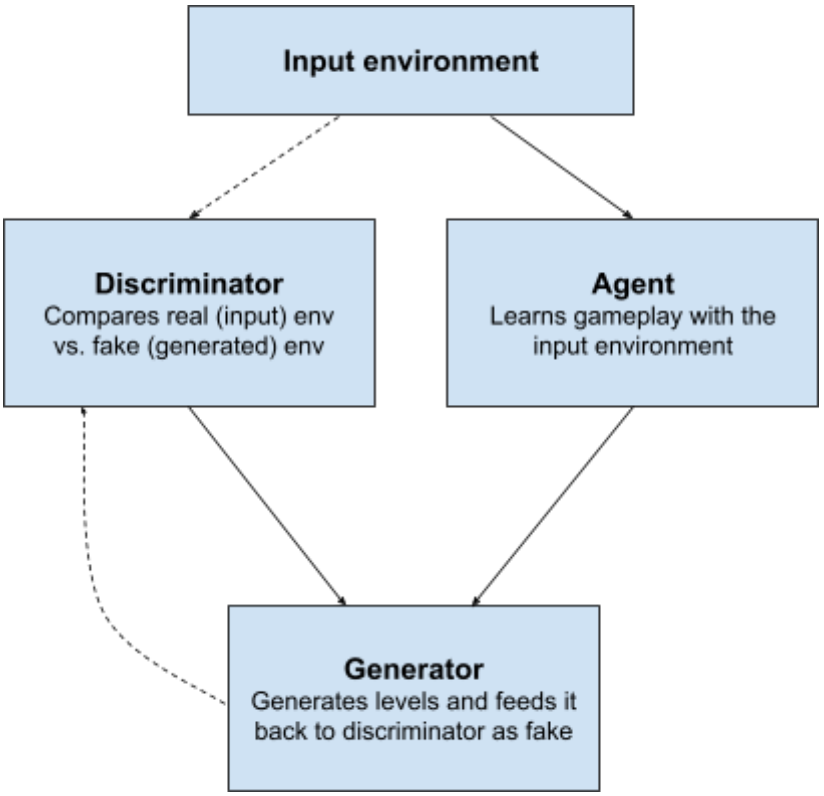

Figure 2. Brief structure of A-D GAN

A similar ‘three-player game’ architecture has been proposed in several researches, such as the Dual-Discriminator GAN (DDGAN) and Multi-Discriminator GAN (MD-GAN). However, these two models are irrelevant with our objective because DDGAN uses two discriminators with similar structure [2], and MD-GAN’s objective is to train a network over spread datasets [3]. Thus, we name the new strucrue A-D GAN, which is an abbreviation for Agent-Discriminator GAN.

We used the GVGAI dataset, a set of video game environments created for training networks. Out of the several available games, we chose Zelda (1983) for the following reasons:

1) It was used in the baseline model (GPN), so we need less data adjustment.
2) The map is fixed in size, so it is easier for the discriminator to learn.

3. Results and Analysis

The run for these results had the following parameters.

- number of learning steps (rl_batch) = 5e3
- number of steps in agent pretraining (pretrain) = 1e5

a. Without Agent-Discriminator structure (Agent only)

The generation loss was 0.00585, and the entropy had a value of 1.79114. The generated levels, from start to end, are as follows:

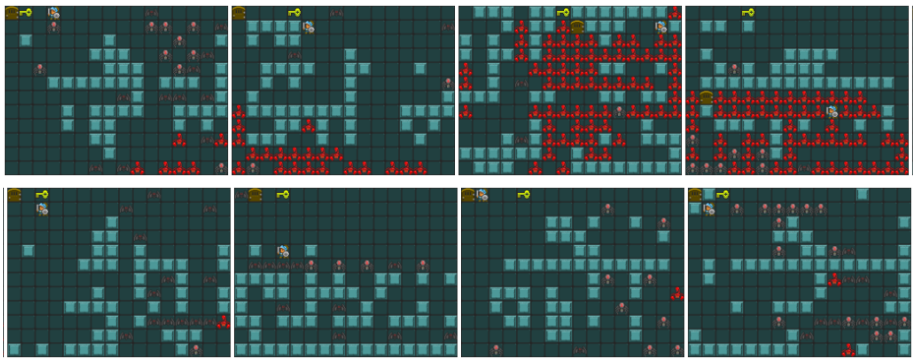

Figure 3-1. Levels generated on step 1 (agent only)

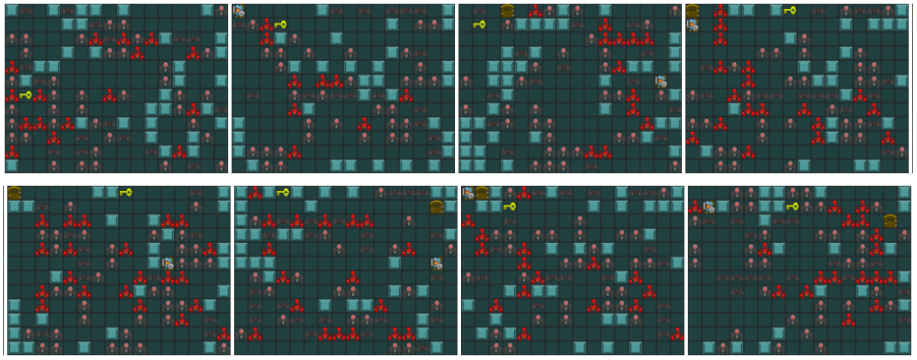

Figure 3-2. Levels generated on step 100 (agent only)

b. With Agent-Discriminator structure

The result had a generation loss of 0.00607 and entropy of 1.79073.

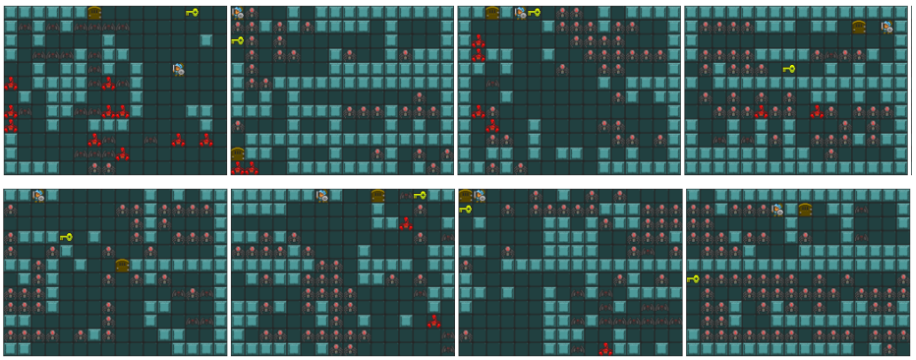

Figure 4-1. Levels generated on step 1 (agent-discriminator)

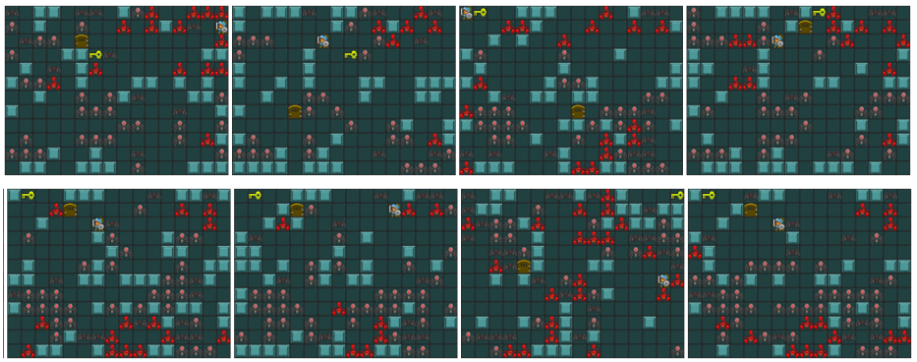

Figure 4-2. Levels generated on step 100 (agent-discriminator)

It was difficult to track the differences with the human eye, and most of the measured indices (reconstruction loss, diversity, mean reward etc.) had matching distributions. However, several indices showed noticeable differences.

The A-D model showed lower entropy, higher level reward, and slightly higher winning probability. This indicates that the model yields more predictable and possibly 'playable' levels for the agent.

In the following three graphs, the blue line denotes agent-only GPN and the orange line denotes A-D structured GAN.

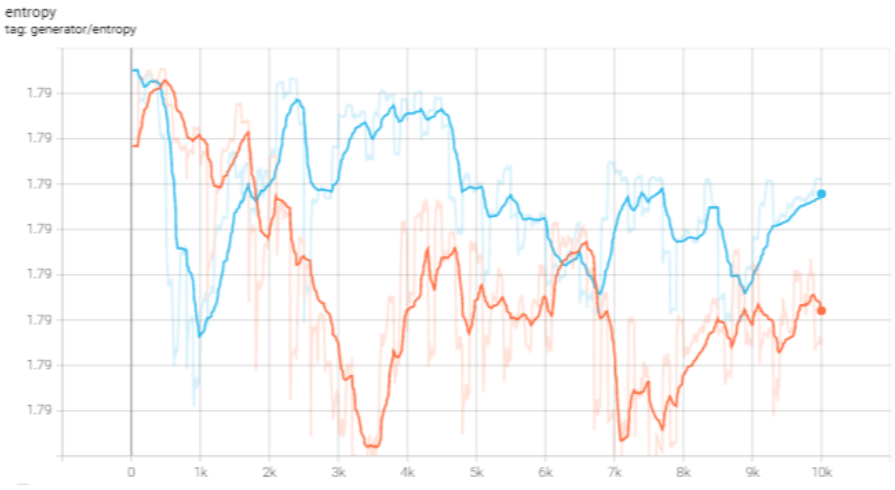

Figure 5-1. Entropy

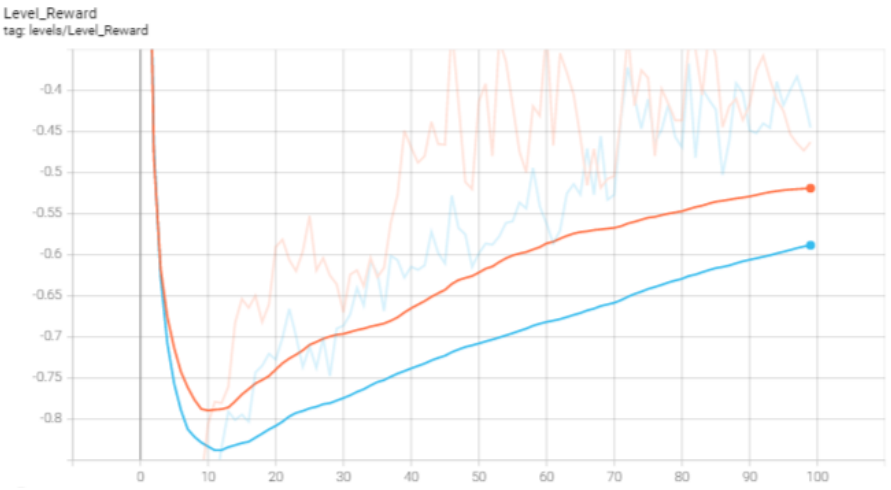

Figure 5-2. Reward for well-formed level generation

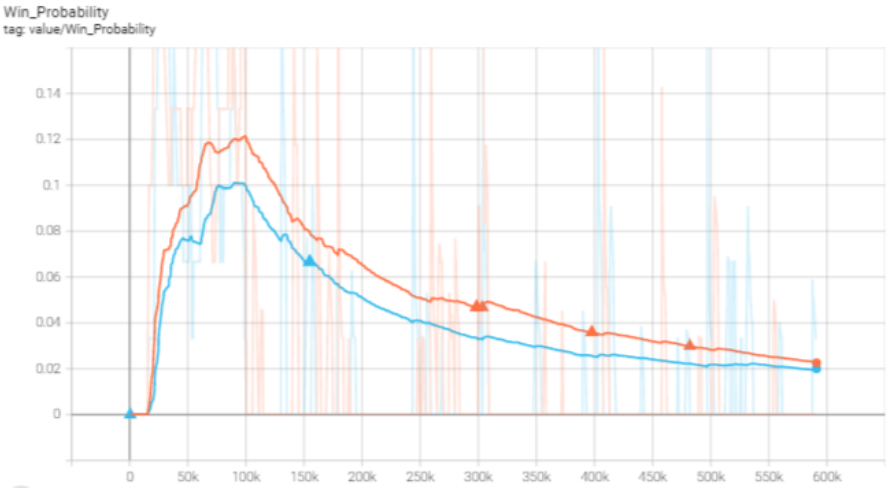

Figure 5-3. The agent's winning probability over the generated levels

Still, there are some limitations to this result, such as considerably small number of steps and that lower entropy might not necessarily indicate a well-made level. To further improve the model, we need a deeper analysis about the playability of a level. Also, note that the agent's winning probability first increases and then drops gradually; this seems desirable for an adversarial structure, but a winning probability too low can be misleading (e.g. unplayable levels instead of reasonably difficult levels).

4. Contributions

Since this project was conducted by a one-person team, this part is omitted.

5. Software and Data

The Github repository for the project can be found here.
https://github.com/JWHan717/CS492I-Project

6. Reference

[1] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search". *Nature*, 529, 484-489, 2016.

[2] S. Kim, Y. Zhou, J. Philion, A. Torralba, S. Fidler, "Learning to Simulate Dynamic Environments with GameGAN". *CVPR*, 2020.

[3] P. Bontrager, J. Togelius, "Learning to Generate Levels From Nothing". *arXiv:2002.05259v2*, 2021.

[4] T. Nguyen et al., "Dual Discrimimator Generative Adversarial Nets". *NIPS 2017*.

[5] C. Hardy, "MD-GAN: Multi-Discriminator Generative Adversarial Networks for Distributed Datasets". *arXiv:1811.03850,* 2019.

[6] I. Albuquerque et al., "Multi-objective training of Generative Adversarial Networks with multiple discriminators". *arXiv:1901.08680*, 2019.