2020

Mario AI PCG

--"automatic generation", "automatic testing" and "level selection"

S-IPA GROUP PROJECT GUIDE

(RISM SRBP IPA)

Zhang Zekun
Tao Xiyan
Xiao Yuchao

NUS National University of Singapore | ISS INSTITUTE OF SYSTEMS SCIENCE

https://github.com/ISA-has-two-projcet/MarioPCG

CONTENT

Executive Summary

Our system is a game development system that integrates "automatic generation", "automatic testing" and "level selection". It is committed to solving the intelligent map design of the game industry. It uses a Procedural Content Generation (PCG) technology, which is based on Generative Adversarial Networks (GANs), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), etc.

The specific game in this project is ***Super Mario Bros***, but the technique should generalize to any game for which an existing corpus of levels is available. Our GAN is trained on a single level from the original Super Mario Bros, available as part of the Video Game Level Corpus (VGLC). CMA-ES is then used to find ideal inputs to the GAN from within its latent vector space. During the evolution, the generated levels are evaluated using different fitness functions. This allows for the discovery of levels that exist between and beyond those sparse examples designed by human designers, and that also optimize additional goals. Our approach is capable of generating playable levels that meet various goals and is ready to be applied to level generation of other games, such as the games in the GVGAI framework. By training on only a single level, we are able to show that even with a very limited dataset, we can apply the presented approach successfully.

Product Plan

Product Description

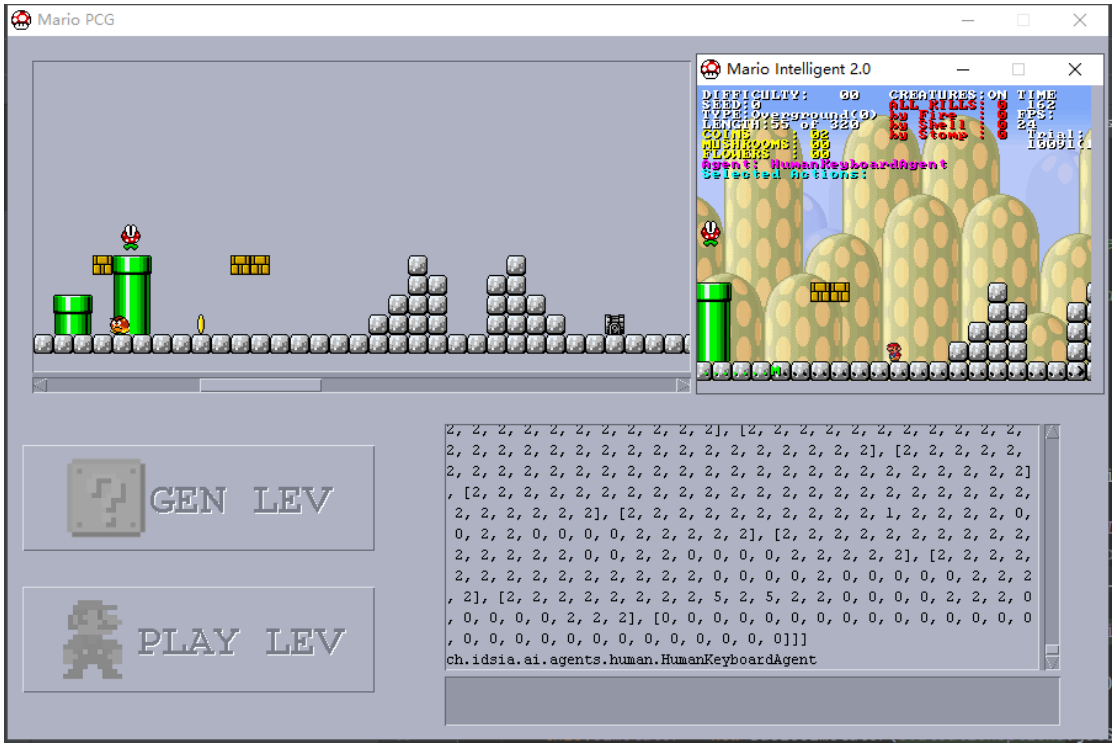


Figure 1. System interface display

The main function of our system is to be able to generate Mario game maps, and to provide automatic detection and trial play of new maps. This function is mainly realized through GAN & CMA-ES and other technologies. And our UI interface is designed based on java-swing. In the Techniques/Algorithms chapter of the project report, we discussed these technologies and algorithms in detail.

Strategy

Efficiently creating high-quality levels is a key challenge in the video game industry. Video game development can take thousands of man-hours to ensure that the player's experience is seamless and that the player may immerse themselves in the video game world. In today's video game market, where annual releases of video game are common, developers face significant time constraints that can limit their ability to build creative and complex levels.

In this report we present a framework to address the aforementioned issues

facing current PCGML (Procedural Content Generation Machine Learning) methods. Considered generically, our system assists with the procedural generation of aesthetically coherent levels which can be con- trolled for specific, predefined goals such as maximizing difficulty or number of solution paths. We accomplish this using a two-pronged approach: controllability is achieved with a fitness function and human evaluation of level quality, and coherence is reintroduced through the employment of a Long Short Term Memory Recurrent Neural Network (LSTM) (Hochreiter and Schmidhuber 1997). Our system focuses on Super Mario Bros (SMB1), which has been the subject of PCGML research for several years; in our work, we aimed to create levels that would prove to be challenging, even for expert level players.

We believe this is a very commercially valuable project, especially in today's game industry, whoever catches up with the development direction of PCGML will be more likely to become the leader in the future game industry.

Market Analysis

In 2015, Nintendo released Super Mario Manufacturing (Super Mario Maker) on the Wii U platform. In this game, you can use the editor to create your own Super Mario levels and upload them to the Internet to play with friends. You can use all the classic props and enemies in the game to create your own levels. This is like a Lego toy designed for adults. Twenty days after the release of "Made in Mario", Nintendo released the sales and operating data of this game: global sales reached 1 million copies, the number of self-made levels created by players reached 2.2 million, and the number of levels played by players worldwide exceeded 75 million.

In 2019, the sequel to "Super Mario Maker 2" was released on the Switch platform on June 28 this year. Super Mario Maker 2 is a gift for the 30th anniversary of the Mario series. In the past 30 years, Nintendo has launched more than 300 games with Super Mario as the protagonist, covering hundreds of millions of audiences. It is no exaggeration to say that Super Mario is a common memory of three generations. The game has a single-player story mode. In story mode, players need to challenge over 100 built-in levels to help Princess Peach rebuild the castle. In addition, players can create levels by themselves or with friends, and share levels online. Japanese media Famitsu announced the game's sales ranking last week. "Super Mario Maker 2" topped the list with 196,000 sets in the first week, 50,000 sets higher than the previous game's first week sales, and also better than in N3DS. There are about 30,000 more sets of "Super Mario Maker for N3DS" released on the Internet. In addition, the sales of the Nintendo Switch console doubled this week to 59,184 units.

According to the sales of games made by Mario, it is not difficult to see the endurance of Mario IP and the appeal of map design to people. Well, the wonderful and rich maps used to require manual design, which cost a lot of manpower, financial resources and time. If we can design a Mario game system that can intelligently generate levels, this also means that players can try almost massive Mario game map. If it can be achieved using machine learning technology, this is very valuable for game developers and players!

Game Industry Analysis

Maps have always played a key role in video games. Sometimes they are the main user interface, and sometimes they are more like a reference tool for players. Although maps are becoming more and more important for many games, it is still a relatively niche topic, and not many people know how to draw maps for games.

In March 2016, Spanish architects Enrique Parra and Manuel Saga published a blog post titled "The Cartography of Virtual Space: The Power of Maps in Video Games". They wrote: "Sometimes the map represents the real location. Sometimes they are just fictional places, but they all contain a graphic language specially designed to fit the overall tone of the game."

Whether it is a traditional two-dimensional environment or a modern 3D world, in most video games, players' perception of space is very important. With the rapid development of the gaming industry since the 1970s, players also need to keep up with the times and understand the environment they simulate.

In the early days of the industry, many people would draw their own maps in order to find their way in the game. Although designers have also begun to add basic maps, players will still do it. For example, VGMaps.com's *"Electronic Game Atlas"* contains approximately 40,000 maps created by players for more than 2,000 games.

Most experts believe that the map is a milestone in the early stage of the industry's development. In addition, the birth of new categories will also push developers to change the way they make maps, such as early role-playing games in the 1990s. "The earliest *Pirates!* designed by Sid Meier, is very interesting because it comes with a physical map. You have to use it to measure in the game to figure out where you are." 40-year-old Konstantinos Dimopoulos said. Dimo Paulos claims to be a game city planner and designer, and recently completed the manuscript of the book *Virtual Cities*: An Atlas & Exploration of Video Game Cities. Demo Pauls lives in Athens, Greece, and has a PhD in urban planning and geography, and a master's degree in urban and regional

planning, so he has a lot of unique insights into map design in video games. In the view of Demo Pauls, many classic game works have had a revolutionary influence in the field of map design.

For example, the RPG "Final Fantasy" released in 1987 was the first game to use a map similar to the real world; Deus Ex, 2000, and other games added real world geographic locations. The latest work in the series was worked by Eidos Montreal Room production, each level has a detailed flat map. In sandbox games like the "GTA" series, players can move freely in a huge 3D world and use GPS-like navigation to determine location and plan routes.

In general, game map design or level design is very complex and highly professional. First of all, you have to determine what the game mode is, such as tower defense, RPG, SLG, or others. Because the details of each game type are different. Next, you have to determine how big the map you want. Then, let the game planner determine what style or story the map is. For example, a map that connects the past and the next to two levels. Or a map with a certain scene story. Then, the difficulty of this map must be determined according to different game types. Generally, it is divided according to the difficulty that each team is accustomed to. Use the macro table in the excel table to output the number of monsters or the number of obstacles. Then, start to output the elements of the map. Generally, you can simulate a map by numerical planning. Put the parameters of each monster and the value of the characters you need to simulate into the excel macro table for simulation. Here you can also draw the map. Route prototype.

Okay, after we complete the above steps that belong to game planning, we will hand over the task to the artist. After the determination of the above series of values, the maximum and minimum values of obstacles or monsters in the map can be determined. The artist only needs to refine the map according to understanding. The above steps are more general. In reality, the game opening process requires different choices for different maps.

Financial Analysis

The US market research company NPD announced data on the US game console consumer market in the early morning of July 19, Beijing time. Nintendo's hardware sales continued to lead in June and the first half of 2019, while PS4 and Xbox One hardware sales continued to decline.

In June 2019, the overall sales of the game industry in the US market fell by 13% year-on-year to US$959 million (approximately RMB 6.617 billion). According to NPD statistics, the consumption of all sub-categories (hardware,

software, peripherals, and game cards) in the game has declined. Compared with the beginning of the year, consumption in June 2019 also dropped by 4%.

Game hardware consumption in 19 years dropped by 33% year-on-year to 235 million US dollars. Sales of all consoles except Nintendo Switch have declined. Sales of gaming peripherals and cards fell 7% to $338 million. According to information posted on Twitter by NPD analyst Mat Piscatella, the popularity of "Battle Royale" games only has short-term rather than long-term effects. "In 2018, the enthusiasm of "Fortress Night" promoted the growth of hardware and peripheral sales, but this year both sales have declined."

Nintendo also took the lead in software sales in June 2019. Although not counting digital sales, Nintendo's "Super Mario Creator 2 (aka "Super Mario Maker 2") still tops the sales charts. This game is also the highest-selling game in the "Mario Creator" series. "Super Mario Maker 2" will prepare some levels that have been designed in advance, but players can also use game materials and use their imagination to design new levels. This is also a great fun of the game. In addition, players can share their own designed levels through the Internet, or download levels shared by other players through the community to fully experience the fun of this game.

Combining the above data, it is not difficult to see that games with good quality, rich content, and massive maps can more easily stand out in the current competitive game market. Although the game map is not the whole of the game, it is not the only factor that determines the quality of a game. But at present, intelligently designing game maps or levels through machine learning is an important development direction of the current game industry.

System Design

System Architecture

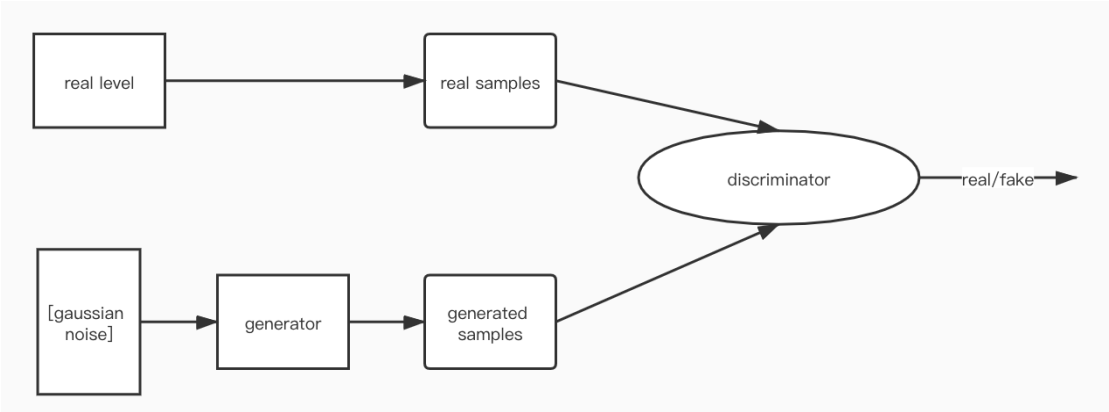


Figure 2. GAN training process

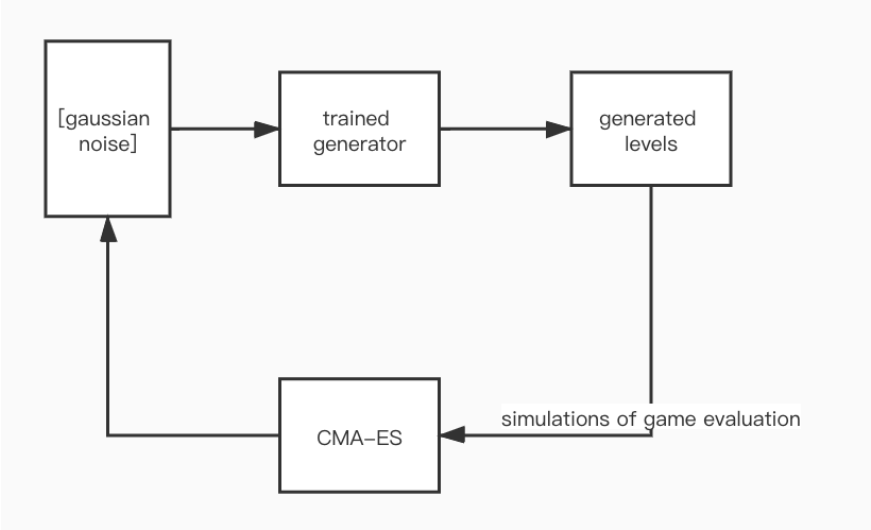


Figure 3. Evolution process

We refer to a paper, *Evolving Mario Levels in the Latent Space of a Deep Convolutional Generative Adversarial Network*[5]. The approach presented in this paper is to create new game levels that emulate those designed by experts using a variant of a Generative Adversarial Network (GAN). GANs are deep neural networks trained in an unsupervised way that have shown exceptional promise in reproducing aspects of images from a training set. Additionally, the space of levels encoded by the GAN is further searched using the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES), in order to discover levels with particular attributes. The idea of latent variable evolution (LVE) was Recently introduced in the context of interactive evolution of images and fingerprint generation but so far has not been applied to PCG of video game

levels. In the system design, we can see the system design of the project. Later we will introduce, Procedural Content Generation for games is discussed, followed descriptions of technical tools applied in this paper: GANs, latent variable evolution, and CMA-ES.

On the basis of the original system, we have also made many improvements and optimizations, such as adjusting the network parameters, and the fine structure, and changing the original fitness function. After optimization, our system is more stable, generates fewer broken items, and the level of the level is more reasonable.

System Development & Implementation

Tools

Python

Python is a widely used interpreted, high-level programming, and general-purpose programming language. Python supports multiple programming paradigms, including object-oriented, structured, imperative, functional and reflective programming. It has a dynamic type system and garbage collection function, can automatically manage memory usage, and it has a huge and extensive standard library.

Pytorch

Pytorch is a python-first deep learning framework. It is a very low-level framework like tensorflow, Caffe, and MXnet. Its predecessor is torch, and its main language interface is Lua. There are 9 of the top 10 machine learning projects on github. This is the era of python, not many people have used it, and it is relatively small. This system uses python's pytorch to build the network structure in training GAN.

Java

Java is a widely used computer programming language with the characteristics of cross-platform, object-oriented, and generic programming. It is widely used

in enterprise-level Web application development and mobile application development.

Swing

Swing is a GUI toolkit designed for Java, which is part of the basic Java classes. Swing includes graphical user interface functions, and its components include: text boxes, text fields, buttons, tables, lists, etc. Swing provides many better screen display elements than AWT. They are written in pure Java, so they can run across platforms like Java itself, unlike AWT. They are part of JFC.

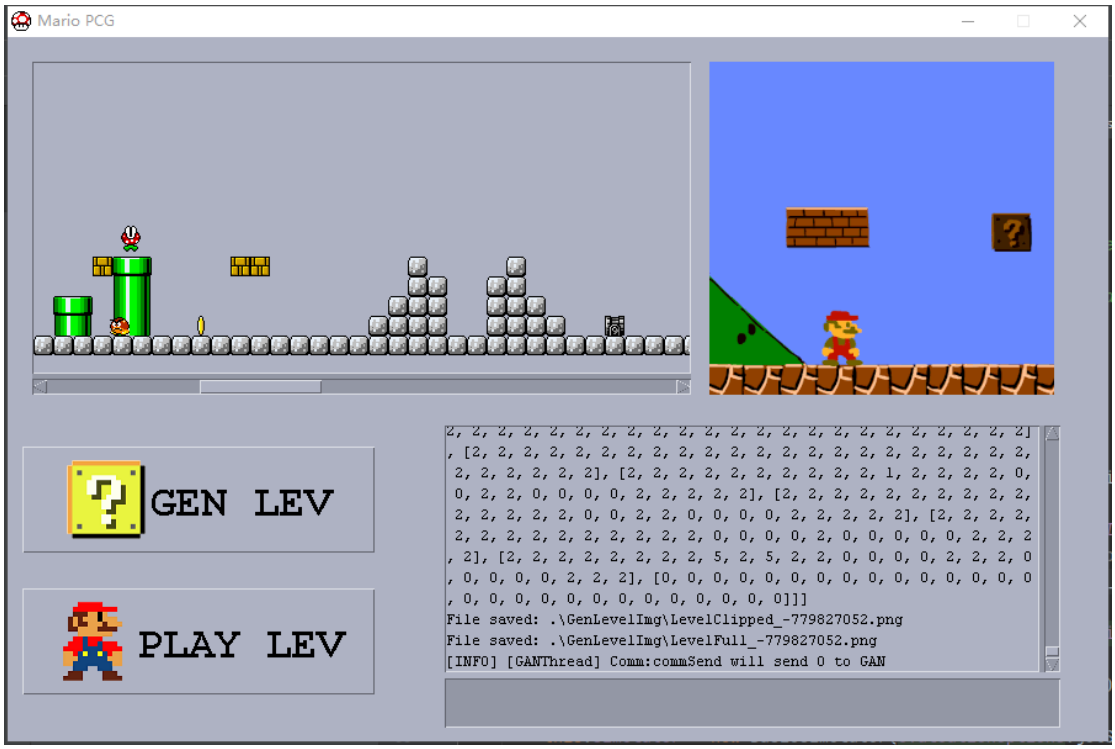


Figure 4. The UI interface of our system which uses swing design

A Java graphical interface is composed of various types of "elements", such as windows, menu bars, dialog boxes, labels, buttons, text boxes, etc. These "elements" are collectively called components.

According to different functions, components can be divided into top-level container, intermediate container, and basic components. The composition of a simple window is shown in the following hierarchical structure:

- Top container
  - Menu Bar
  - Intermediate container

- Basic components
- Basic components
- …

Techniques/Algorithms

Generative Adversarial Networks (GANs)

Introduction

In recent years, supervised learning with convolutional networks (CNNs) has seen huge adoption in computer vision applications. Comparatively, unsupervised learning with CNNs has received less attention. From the paper, *unsupervised representation learning with deep convolutional generative adversarial network*, it bridges the gap between the success of CNNs for supervised learning and unsupervised learning. it introduces a class of CNNs called deep convolutional generative adversarial networks (DCGANs), that have certain architectural constraints, and demonstrate that they are a strong candidate for unsupervised learning. Training on various image datasets, it shows convincing evidence that our deep convolutional adversarial pair learns a hierarchy of representations from object parts to scenes in both the generator and discriminator. Additionally, it uses the learned features for novel tasks - demonstrating their applicability as general image representations.[1]

Generative Adversarial Networks (GANs) were first introduced by Goodfellow et al. in 2014. Their training process can be seen as a two-player adversarial game in which a generator G (faking samples decoded from a random noise vector) and a discriminator D (distinguishing real/fake samples and outputting 0 or 1) are trained at the same time by playing against each other. The discriminator D aims at minimizing the probability of misjudgment, while the generator G aims at maximizing that probability. Thus, the generator is trained to deceive the discriminator by generating samples that are good enough to be classified as genuine. Training ideally reaches a steady state where G reliably generates realistic examples and D is no more accurate than a coin flip.

DCGAN

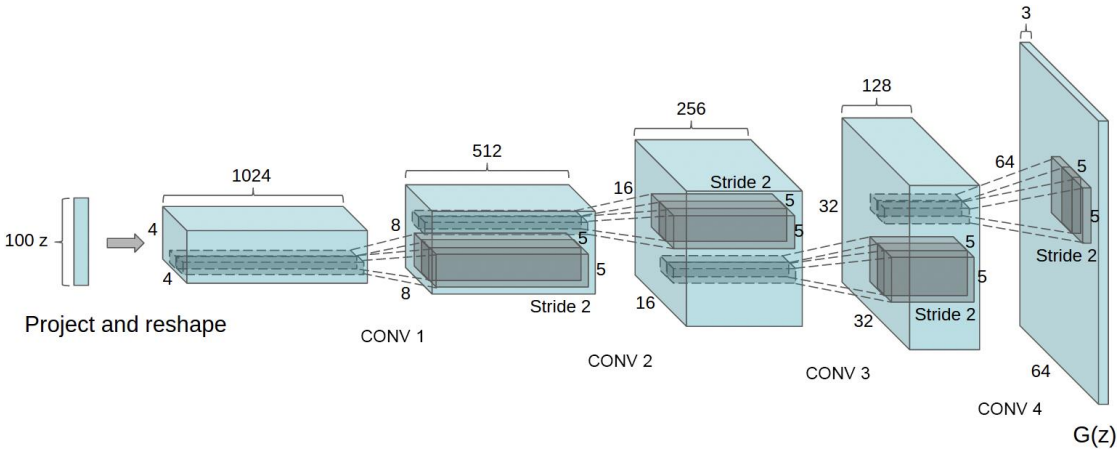


Figure 5. DCGAN generator (example, A 100-dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high-level representation into a 64 × 64 pixel image. Notably, no fully connected or pooling layers are used.)

GANs quickly became popular in some sub-fields of computer vision, such as image generation. As the example in Figure 5, however, training GANs is not trivial and often results in unstable models. Many extensions have been proposed, such as Deep Convolutional Generative Adversarial Networks (DCGANs), a class of Convolutional Neural Networks (CNNs). A particularly interesting variation are Wasserstein GANs (WGANs). WGANs minimize the approximated Earth-Mover (EM) distance (also called Wasserstein metric), which is used to measure how different the trained model distribution and the real distribution are. WGANs have been demonstrated to achieve more stable training than standard GANs.[1] In our system we use DCGANs. It means that we use a set of constraints on the architectural topology of Convolutional GANs that make them stable to train in most settings.

Procedural Content Generation

Introduction

Procedural Content Generation (PCG) is a technology that can follow a certain design pattern to automatically generate new levels, maps and even monsters and other game content as the game progresses. This technology uses a random seed and a series of parameters adjusted by designers and players to

arrange and combine some ready-made game content to generate "new" content.

In most cases, PCG technology is used as a low-cost and efficient way to increase game content and length. From the complex and changeable terrain in Minecraft and Terraria, to the unique scenes and dungeons in various roguelikes, to the very simple combination of objects in Temple Run and Flappy bird, the use of PCG technology can expand the substantial length of these games Being "infinite" can also make each game experience slightly different. PCG is generally used for several purposes:

1. Increase the uncertainty of the game. Example: Civilization series. If this type of game uses a fixed map for each round, it is easy to solidify and form a well-known optimal solution.

2. Speed up the production efficiency of game content. Example: Assassin's Creed series. This type of game requires large-scale generation of scene objects with a high degree of repetition. If a designer is used to manually place them, a lot of time and cost will be wasted, but they are only used in the game development process and are not executed on a large scale in the game process.

3. Extend the length of the game. Example: Spore. This kind of game hopes that the process generation technology will increase the time required for the exhaustion of the game content, especially in the late game, giving players almost infinite exploration space

The first academic Procedural Content Generation competition was the 2010 Mario AI Championship, in which the participants were required to submit a level generator which implements a provided Java interface and returns a new level within 60 seconds. The competition framework was implemented based on Infinite Mario Bros, a public clone of Super Mario Bros.

The availability and popularity of the Mario AI framework has led to several approaches for generating levels for Super Mario Bros. Shaker et al. evolved Mario levels using Grammatical Evolution (GE). In 2016, Summerville and Mateas applied Long Short- Term Memory Recurrent Neural Networks (LSTMs) to generate game levels trained on existing Mario levels, and then improved the generated levels by incorporating player path information. This approach inspired a novel approach to level generation, in which new levels are generated automatically from a sketch of some de- sired path drawn by a human designer. Another approach that was trained using existing Mario levels is that of Jain et al., which trained auto-encoders to generate new levels using a binary encoding where empty (accessible) spaces are represented by 0 and

the others (e.g., terrain, enemy, tunnel, etc.) by 1. Though this approach could generate interesting levels, the use of random noise inputs into the trained auto-encoder sometimes led to problematic levels. Additionally, because of the binary encoding, no distinction was made between various possible types of tiles.[6]

Latent Variable Evolution

Introduction

The first latent variable evolution (LVE) approach was introduced by Bontrager et al. [3]In their work the authors train a GAN on a set of real fingerprint images and then apply evolutionary search to find a latent vector that matches with as many subjects in the dataset as possible.

In another paper Bontrager et al. [2]present an interactive evolutionary system, in which users can evolve the latent vectors for a GAN trained on different classes of objects (e.g., faces or shoes). Because the GAN is trained on a specific target domain, it becomes a compact and robust genotype-to-phenotype mapping (i.e., most produced phenotypes do resemble valid domain artifacts) and users were able to guide evolution towards images that closely resembled given target images. Such target-based evolution has been shown to be challenging with other indirect encodings.

Because of the promising previous LVE approaches, in this paper we investigate how latent GAN vectors can be evolved through a fitness-based approach in the context of level generation.

Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

Introduction

Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [4] is a powerful and widely used evolutionary algorithm that is well suited for evolving vectors of real numbers. The CMA-ES is a second-order method using the covariance matrix estimated iteratively by finite differences. It has been demonstrated to be efficient for optimizing non-linear non-convex problems in the continuous

domain without a-priori domain knowledge, and it does not rely on the assumption of a smooth fitness landscape.

We applied CMA-ES to evolve the latent vector and applied several fitness functions on the generated levels. Fitness functions can be based on purely static properties of the generated levels, or on the results of game simulations using artificial agents.

Improvement and optimization

On the basis of the original system, we have also made many improvements and optimizations, such as adjusting the network parameters, and the fine structure, and changing the original fitness function. After optimization, our system is more stable, generates fewer broken items, and the level of the level is more reasonable.

Besides, we design a UI for “Mario AI PCG” system. In this way, we can show our system vividly and directly.

Robust of the generator

Figure 6. Broken pipes

Figure 7. Enemy trapped

According to the paper, *Evolving Mario Levels in the Latent Space of a Deep Convolutional Generative Adversaria*l [6], we initially realized the “Mario AI PCG function”, but we also found the shortcomings of the previous system design. The original version of “Mario AI PCG” may generate some "broken titles", as shown in Figure 6, or maps with individual preferences (such as maximizing or minimizing the number of players’ jumps or "enemy trapped”, as shown in Figure 7), which is not in line with human preferences. We think that according to the paper we used for reference, it was not well designed in the final design of the fitness function. We have changed the original function. At present, it seems that the effect of changing the fitness function is very good.

And, after we optimized the fitness function, we not only solved the "enemy trapped" and "broken pipes" phenomenon, but also solved the problem of combining the two testing methods to select the parent. According to the evolutional strategy in the reference paper, it is impossible to pass both Representation-based testing and Agent-based testing at the same time. The combination of the two testing methods determines the parent, and only one of them can be selected.

The specific changes on fitness function are as follows.

Optimized Fitness function

When creating our fitness function, we needed to take two separate components of difficulty into account, the local difficulty of specific actions, and the global difficulty of completing a level as a whole. The notion of difficulty described above aggregates the local difficulty of a level but fails to account for the global difficulty. To compensate for this, we also used a global measure of level difficulty based on the fraction of times that an AI agent was successfully

able to complete the level. The A* agent used in our implementation was non-deterministic, and in practice was able to complete difficult levels less than 30% of the time and easy ones 80% or more of the time. As a result, we were able to run this A* agent on a level multiple times and treat the fraction of successful runs as a measure of the global difficulty of the level. However, if an AI agent with high variance is not available for some game, one can substitute multiple AI agents with varying degrees of ability to achieve the same result by using the fraction of agents that successfully complete a level as a measure of the global difficulty.

One problem with attempting to maximize the difficulty of generated levels is that infeasible levels (i.e., those which are impossible to complete), despite being the most difficult levels, are typically unwanted as they are not interesting to play and would lead to player frustration. To account for this, we penalize such levels by assigning them a negative fitness value corresponding to the length of the level that can be completed. More formally, let f be the maximum fraction of the level that any AI agent has traversed. If $f < 1$, which typically only happens when the level cannot be completed, the level is assigned a fitness of $f - 1$, which rewards levels that have a larger playable area while also ensuring that in- completable levels are always less fit than completable levels.

We used hyperparameters, $k1 \ldots k3 \in R$ as weights for the various components of our fitness function. The fitness value F of a level L with an aggregated local difficulty of ℓ and a global difficulty of g, and where f is the greatest fraction of the level that was completed by any AI agent:

$$F = \begin{cases} k1 \cdot \ell + k2 \cdot g & if\ f = 1.0 \\ k3 \cdot (f - 1) & if\ f < 1.0 \end{cases}$$

Because the CMA-ES implementation we used was designed to find the minima of the given fitness function, in practice we negate the result of F when calculating fitness.

Reference

[1] Alec Radford & Luke Metz, Soumith Chintala, unsupervised representation learning with deep convolutional generative adversarial network, Under review as a conference paper at ICLR 2016

[2] Philip Bontrager, Wending Lin, Julian Togelius, and Sebastian Risi. 2018. Deep Interactive Evolution. European Conference on the Applications of Evolutionary Computation (EvoApplications).

[3] Philip Bontrager, Julian Togelius, and Nasir Memon. 2017. DeepMasterPrint: Generating Fingerprints for Presentation Attacks. arXiv preprint arXiv:1705.07386 (2017).

[4] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. 2003. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). Evolutionary Computation 11, 1 (2003), 1–18.

[5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. In Proceedings of the 34nd International Conference on Machine Learning, ICML.

[6] Vanessa Volz, Jacob Schrum, Jialin Liu, Simon M. Lucas, Evolving Mario Levels in the Latent Space of a Deep Convolutional Generative Adversarial Network, arXiv:1805.00728v1 [cs.AI] 2 May 2018.

Appendix

Installation and User Guide

1. Clone the project from: https://github.com/ISA-has-two-projcet/MarioPCG

2. Install JDK8 & JRE8 from:
https://docs.oracle.com/javase/8/docs/technotes/guides/install/install_overview.html

3. Install PyTorch from:
https://pytorch.org/get-started/locally/

4. Enter root directory of the project ("…/MarioPCG/"), change "**my_python_path.txt**" to your own python env.

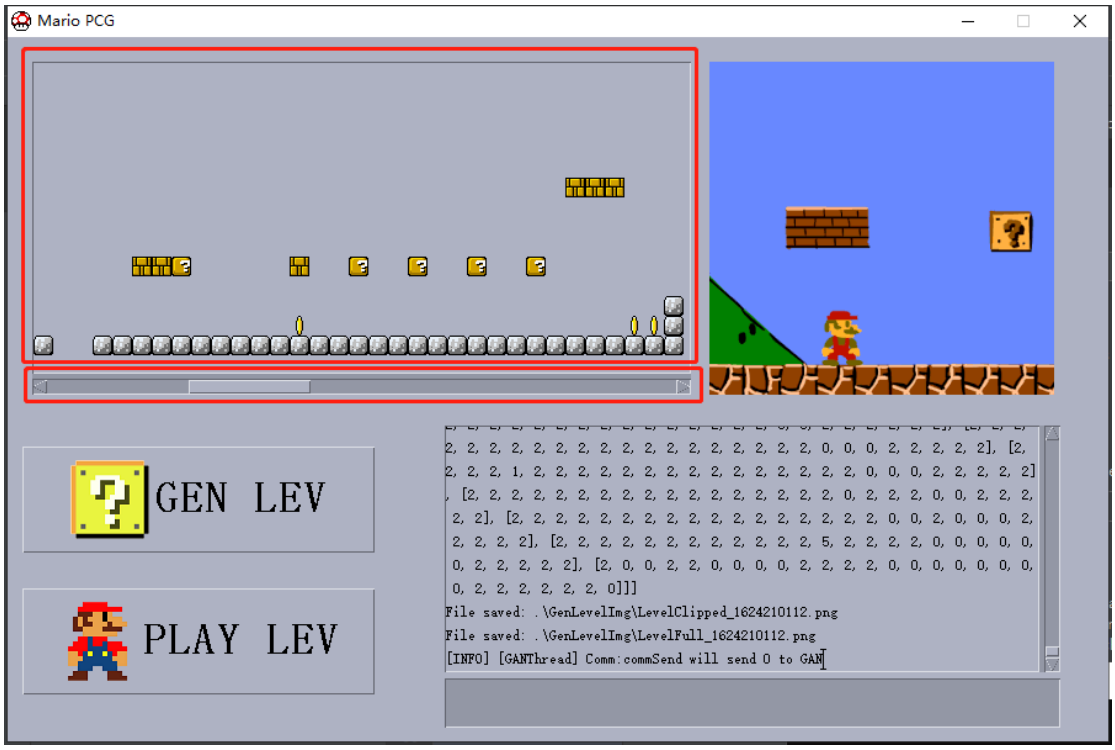
5. In same root directory, you can find a jar file: "**marioaiPCG.jar**"

6. Execute command "**java -jar marioaiPCG.jar**" in terminal to run this jar

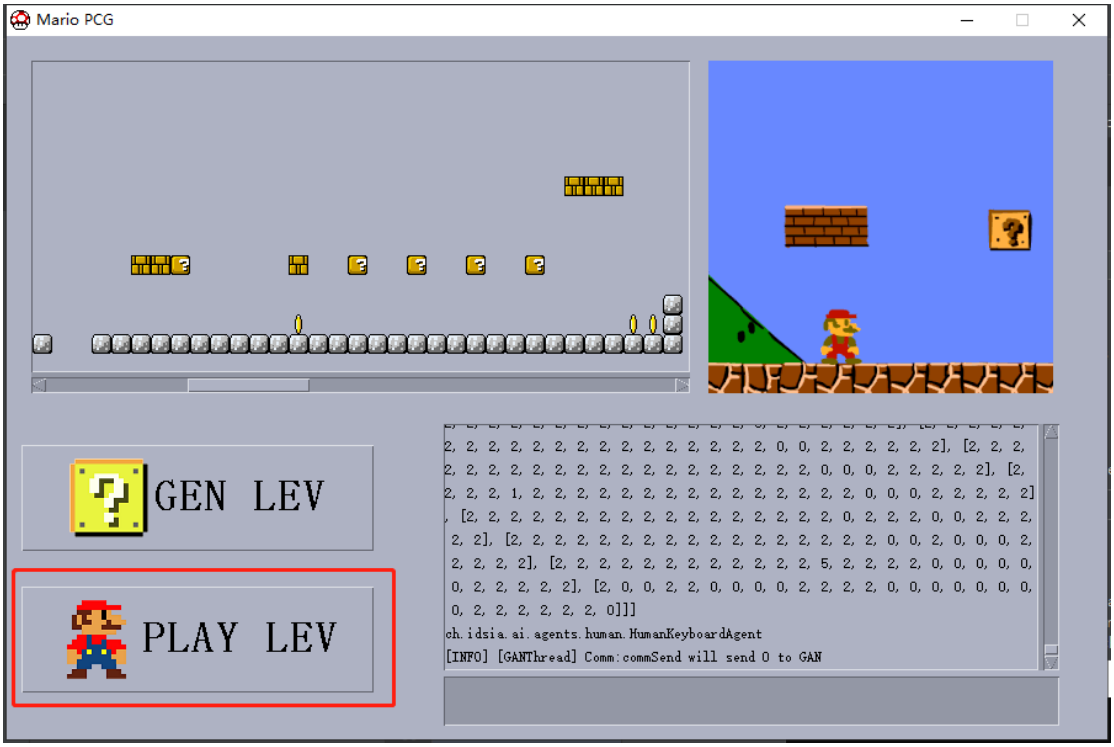
7. Click "**GEN LEV**" button to start evolution algorithm to generate a level:

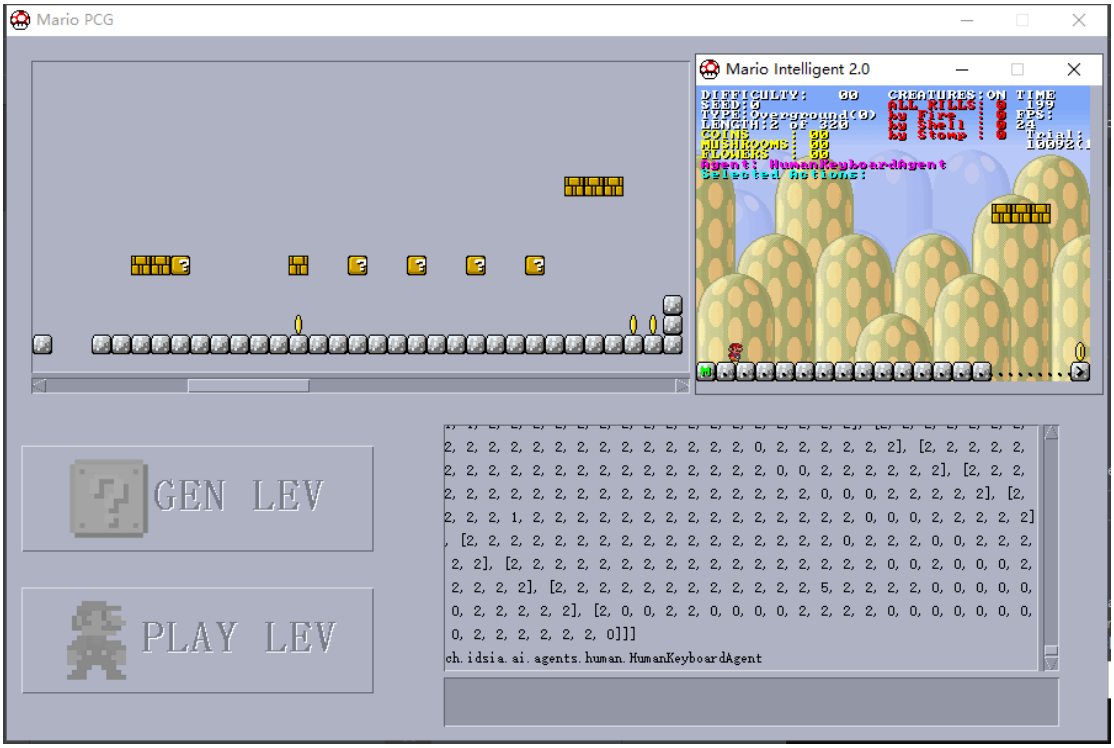
8.Wait for some minutes, you can see the generation progress and log here:

9. When finishing the level generation, you can preview new level here:



10.Click “**PLAY LEV**”, let’s play it now!


Mario PCG
GEN LEV
PLAY LEV
ch.idsia.ai.agents.human.HumanKeyboardAgent
[INFO] [GANThread] Comm:commSend will send 0 to GAN


Mario PCG
Mario Intelligent 2.0
GEN LEV
PLAY LEV
ch.idsia.ai.agents.human.HumanKeyboardAgent

Individual project report

Tao Xiyan individual project report

(1) personal contribution to group project

1. Participate in Project topic selection;
2. Participate in the overall development of the first version of the project;
3. Participate in Project schedule planning;
4. Complete optimization of the fitness function;
5. The overall design of the final version of the project;
6. Video split design, video script & narration writing;
7. Design the PPT for display our algorithms and system design and for the video;
8. Report writing (Business Case / Product Plan / Market Research/ Techniques/Algorithms/etc.).

(2) what learnt is most useful for you

1. How to design a better fitness function for individual problem. (Know what we want is very important for machine learning.)
2. This is my first enforcement-learning project.
3. How to use java-swing to realize the UI for our system.
4. How to look at the entire project (business value, business outlook, business goals) with leadership thinking.
5. How to design advertising video, write advertising video script, design advertising video storyboard.

Now, I have some experience in participating in the establishment of the entire project to the later publicity. After all, I was only involved in back-end development or algorithms in my previous work and study, and my vision was very cramped. This project brought me a different learning experience for computer students.

(3) how you can apply the knowledge and skills in other situations or your workplaces.

In this project, we referenced and reproduced the paper, and found many problems in it. Because our "Mario AI PCG" uses Procedural Content Generation and artificial intelligence technology to design maps that are closely related to gameplay, and it is a relatively new attempt in the game industry, so the previous papers may not be perfect, we discuss and analyze and improved the original fitness function. And the improved effect is very good. After optimizing the fitness function, we not only solved the "enemy trapped" and

"broken pipes" phenomenon, but also solved the problem of selecting the parent by combining the two testing methods. According to the evolutional strategy in the reference paper, it is impossible to pass both Representation-based testing and Agent-based testing at the same time. The combination of the two testing methods determines the parent, and only one of them can be selected. In the future, I will continue to improve this project and extend this technology to more games.

And, through this project, I learned to have the courage to try and make mistakes and explore. When faced with some emerging fields like PCGML, I need some bold attempts. Finally, the landing of deep learning algorithms has always been a problem in the industry. These experiences have provided me with valuable experience in my future work.

Xiao Yuchao individual project report

(1) Your personal contribution to the project.

In this project, I participated in the training of the DCGAN network and network parameter adjustment. I adjusted the hyperparameters and constraints of the network to make the network more stable for training. I adjusted the fine network structure and changed the original adaptive function. After optimization, our system is more stable, less damaged items are generated, and the levels are more reasonable. I also participated in the study of latent variable evolution (LVE) and studied how latent GAN vectors can be evolved through a fitness-based approach in the context of level generation.

(2) What you have learnt from the project.

In the project, I learned about the DCGAN network and how to use it. I also learned how to increase the training stability of the network by adding constraints to the network. In addition, I also learned the design ideas and usage methods of latent variable evolution (LVE) method. Most importantly, I learned the methods of intelligent generation, intelligent testing and intelligent optimization in game development. This has never been encountered in my previous studies.

(3) How you can apply this in future work-related projects.

In the future related work, the most practical thing I have learned is the development ideas of intelligent generation, intelligent optimization, and intelligent testing. Using GAN, GA and other methods, on the one hand, you can find the optimal solution strategy for the problem of determining the target

and direction, and on the other hand, you can generate the data needed to solve the problem and fill in some data missing. These methods allow us to obtain a functional MVP in a short time. Based on MVP, methods such as Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) can be used to help us optimize the generated MVP system. The use of these methods can shorten the development cycle and quickly obtain the model we expect on problems with clear goals but complex processes.

Zhang Zekun individual project report

(1) Your personal contribution to the project.

- Train the DCGAN with VGLC data to generate Super Mario Bros levels. Then tune and test the trained model.
- Experiment with different intelligent agents to play generated levels, choose one of them (AStar Agent), and extract useful behavior data after each turn of playing. Each turn, the collected behavior data with the level's statistic data will be sent to CMA-ES module for evolution.
- Improve the CMA-ES fitness function with additional agent's behavior data (to balance the difficulty), and levels' statistic feature (to fix some issues like broken pipes).
- Integrate different parts of the system and make a GUI with Java swing

(2) What you have learnt from the project.

- How to code with Java and Python simultaneously
- How DCGAN works, and how to train a DCGAN to generate game levels
- How CMA evolution strategy works and how to utilize it with intelligent game agents to evolve game levels.
- How to use Java swing to build desktop GUI applications
- How to manage threads in Java

(3) How you can apply this in future work-related projects.

- In this project, we tried to utilize AI techniques to handle PCG problems, which is a promising field in the game industry. Currently, more and more game studios are trying to introduce Ai techniques to their level design processes. And my hope is to integrate advanced Ai techniques with game development in my future job, and the experience of this project is very valuable.
- GAN is an interesting topic and gaining more and more attention now. For it's very potential in the content creation industry, which including game industry. Other than level generation, GAN has many successful use cases

in game art creation field, like automatic plant and animal generation. In this project, I learned techniques and knowledge about GAN, which can help me a lot in my future work-related projects.

- Java swing is a light-weighted and practical GUI library, I learned how to use it to make simple desktop GUI programs in this project. In my former job, as a game programmer, I usually needed to make some tiny widgets to help game artists and designers to do some tech works without coding. Most times, I would write a .bat or .sh file, but my colleagues always complained that it was hard to use and count-intuition. I think it's a good idea to use swing to improve these widgets' user experience in my future work.