

Angry Birds Level Generation Using Walkthrough Descriptions

Sukai Huang

A thesis submitted for the degree of
Bachelor of Advanced Computing (Honours)
The Australian National University

June 2021

© Sukai Huang 2021

Except where otherwise indicated, this thesis is my own original work.

Sukai Huang

9 June 2021

To my grandparents.

Acknowledgments

First and foremost, I want to express my sincerest gratitude to my supervisor, Professor Jochen Renz for his support and patience over the whole honour year. The weekly research meeting really helped me to broaden my perspective on Artificial Intelligence (AI). Also, thank you Professor Jochen Renz again for providing me the opportunity to participate in IEEE AIBIRDS COG 2020 Level Generation Competition. The experience also helped me to form concrete idea about my honours project.

I would also like to thank my seniors Cheng, Peng, Ruiqi, Hua, Ekaterina, Chathura and Vimukthini. Your guidance helped me a lot. Thank you Cheng again for recommending me lots of reading materials to extend my background knowledge.

I must thank my girlfriend Siyang for taking care of me. I won't let you down!

Lastly, thanks all the modern science and technology. I feel lucky living in 21st century where I can still continue my research journey even under the COVID-19 pandemic.

Abstract

Angry Birds is a famous environment for agents to learn physical reasoning. However, the deep reinforcement learning agents often underperform due to a lack of training set of game levels. To address the issue, procedural level generation is used to synthesise new Angry Birds game levels. However, the current rule-based Angry Birds procedural level generator [Stephenson and Renz, 2016b, 2019] is incapable of generating game levels that aid agents in learning physical reasoning, as it cannot guarantee the level of physical reasoning required in order to solve the generated game levels.

Hence, in a new approach, we use walkthrough descriptions to generate Angry Birds game levels and train the Generative Adversarial Networks (GANs) based procedural level generator by imitating the high-quality handcrafted levels. Unlike the conventional imitation approach, the proposed one is able to control the style of the generated game levels and also enhance the diversity of the game level dataset via manipulating the input walkthrough descriptions. Both qualitative and quantitative evaluations are conducted to demonstrate that the generated game levels using this method demand high level of physical reasoning to solve, just like the handcrafted game levels.

Besides that, we developed a new Angry Birds walkthrough dataset called AbVat. It is a valuable dataset capable of facilitating a variety of meaningful research tasks in the domain of spatial-temporal understanding and reasoning.

Keywords— Procedural Level Generation, Generative Adversarial Network (GANs)

Contents

| | |
|--------------------------------------------------------|------------|
| Acknowledgments | vii |
| Abstract | ix |
| 1 Introduction | 1 |
| 1.1 Background | 1 |
| 1.1.1 Physics-based simulation game (PBSG) | 1 |
| 1.1.2 Procedural Level Generation (PLG) | 2 |
| 1.2 Motivation | 2 |
| 1.3 Contribution | 3 |
| 1.4 Thesis Outline | 4 |
| 2 Review of the Literature | 5 |
| 2.1 Rule-based generation | 5 |
| 2.2 Imitation of handcrafted levels | 6 |
| 2.3 Review and Summary | 7 |
| 3 Theoretical Background and Related Work | 11 |
| 3.1 Natural Language Processing (NLP) | 11 |
| 3.1.1 NLP and Reinforcement Learning Agent | 12 |
| 3.2 Multimodal training | 12 |
| 3.3 Generative Adversarial Network (GANs) | 13 |
| 3.3.1 Original GANs and the Adversarial Idea | 13 |
| 3.3.2 GANs and Density Estimation | 14 |
| 3.3.3 Variations of GANs model | 15 |
| 3.4 Summary | 18 |
| 4 AbVat Dataset | 19 |
| 4.1 Properties of AbVat | 19 |
| 4.1.1 Level representation | 21 |
| 4.1.2 Walkthrough data | 21 |

| | | |
|----------|--------------------------------------------------------------------------------------|-----------|
| 4.1.3 | Diversity | 24 |
| 4.2 | Constructing AbVat | 25 |
| 4.2.1 | Data collection | 25 |
| 4.2.2 | Data cleaning and preprocessing | 25 |
| 4.3 | Benchmark dataset for various tasks | 27 |
| 4.4 | summary | 28 |
| 5 | Preliminary attempts | 29 |
| 5.1 | Preliminary Attempt 1: contrastive learning on text and image pair | 29 |
| 5.1.1 | Method | 30 |
| 5.1.2 | Experiment and result | 32 |
| 5.1.3 | Discussion | 34 |
| 5.2 | Preliminary Attempt 2: transfer learning and "divide and conquer" strategy | 38 |
| 5.2.1 | Method | 39 |
| 5.2.2 | Experiment and result | 41 |
| 5.2.3 | Discussion | 42 |
| 5.3 | Summary | 47 |
| 6 | Design and Implementation | 49 |
| 6.1 | Architecture Overview | 49 |
| 6.2 | Module 1: level generation by multi-scale learning | 49 |
| 6.3 | Module 2: inverse mapping from game level to its latent representation | 51 |
| 6.4 | Module 3: mapping from text to latent representation | 53 |
| 6.5 | Data augmentation: Image | 54 |
| 6.6 | Summary | 56 |
| 7 | Experimental settings | 57 |
| 7.1 | Programming environment | 57 |
| 7.2 | Dataset | 57 |
| 7.3 | Baselines | 59 |
| 7.4 | Training details | 61 |
| 7.4.1 | Module 1: Level Generation model | 61 |
| 7.4.2 | Module 2: Reverse mapping level encoder | 61 |
| 7.4.3 | Module 3: Text to latent space mapper | 61 |
| 7.5 | Evaluation Metrics | 63 |
| 7.5.1 | Evaluation of novel level generation | 64 |

| | | |
|----------|------------------------------------------------------------------------|-----------|
| 8 | Results and Discussion | 65 |
| 8.1 | Quantitative Results | 65 |
| 8.1.1 | Level Generator’s performance | 65 |
| 8.1.2 | Level Encoder’s performance | 67 |
| 8.1.3 | Text Mapper’s performance | 67 |
| 8.2 | Qualitative Results | 68 |
| 8.2.1 | Overview of the generator’s performance | 68 |
| 8.2.2 | Reconstruction using Level Encoder | 70 |
| 8.2.3 | Level generation from walkthrough descriptions | 70 |
| 8.2.4 | Evaluation of the level of physical reasoning | 72 |
| 8.3 | Summary | 72 |
| 9 | Conclusion | 75 |
| 9.1 | Future Work | 75 |
| 9.1.1 | Improvement of AbVat dataset | 76 |
| 9.1.2 | Level stability testing and training | 76 |
| 9.1.3 | Better quantitative evaluation involving Reinforcement learning agents | 77 |
| | Appendix | 89 |
| 9.2 | Appendix 1: Generated game level samples | 89 |

List of Figures

| | | |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 2.1 | Comparison between machine generated level and handcrafted level | 8 |
| 3.1 | Saturating Loss problem illustration Copyright and adapted from Nips 2016 tutorial: Generative adversarial networks[Goodfellow, 2016] | 14 |
| 4.1 | Demonstration of AbVat data | 20 |
| 4.2 | Comparison between pixel-based and XML object representation | 22 |
| 4.3 | Novel object in Angry Birds Seasons Ham Dunk 2-2 - Kings episode | 22 |
| 4.4 | Example of the text segmentations of a walkthrough description based on critical states | 23 |
| 4.5 | Demonstration of irrelevant contents in walkthrough video | 26 |
| 4.6 | Example of noisy contents in a text walkthrough | 26 |
| 5.1 | The architecture of the cGANs based model of the first attempt. | 30 |
| 5.2 | A diagram for contrastive learning and negative samples | 32 |
| 5.3 | Angry Birds game level image samples generated in the first preliminary attempt | 34 |
| 5.4 | Loss plot for GANs model | 35 |
| 5.5 | Demonstration of the sparsity problem as no. of dimension increases | 37 |
| 5.6 | Illustration of the architecture for the second preliminary attempt | 38 |
| 5.7 | Generated samples from the second attempt using BiGAN | 43 |
| 5.8 | Reconstruction samples from the second attempt using BiGAN | 44 |
| 5.9 | Demonstration shows that there is not much visual differences between the successive frames, therefore the unsupervised pre-training video frames dataset is information-poor | 45 |
| 6.1 | Overview architecture of the proposed design for generating Angry Birds game levels from walkthrough description | 50 |

| | | |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 6.2 | Illustration of the Multi-scale learning in MSG-GANs model. The generator produces image at different resolutions and concatenate to the corresponding layer of the discriminator. Thus the discriminator is able to provide additional gradient information to guide the image synthesis at different scales Copyright and adapted from MSG-GAN paper [Karnewar and Wang, 2020] | 51 |
| 6.3 | Overview of the Swin-transformer architecture. Copyright and adapted from Swin-transformer paper [Liu et al., 2021] | 52 |
| 6.4 | An illustration of the window approach of self-attention. Copyright and adapted from Swin-transformer paper [Liu et al., 2021] | 52 |
| 6.5 | BERT model architecture Copyright and adapted from BERT paper [Devlin et al., 2018] | 54 |
| 6.6 | DistillBERT with a fully connected layer to perform regression task to get the latent representation of the associated game level image | 55 |
| 6.7 | Illustration of the data augmentation work in bCR-GAN and GAN-ada Copyright and adapted from GAN-ada paper [Karras et al., 2020a] | 56 |
| 7.1 | Total words distribution of the text walkthrough in the AbVat dataset | 62 |
| 8.1 | | 66 |
| 8.2 | The FID score of the GANs model for game level generation; the smaller the value, the better the performance. | 66 |
| 8.3 | Generated samples when the FID score rise suddenly. The sample shows a mode collapse happened | 66 |
| 8.4 | Loss plot of Swin-transformer level encoder | 67 |
| 8.5 | Loss plot for DistilBERT text mapper module | 68 |
| 8.6 | Samples from real game levels | 69 |
| 8.7 | Comparison of the pixel-based game level between the generator from the second preliminary attempt and the proposed generator | 69 |
| 8.8 | An example of typical generated game levels | 70 |
| 8.9 | Reconstruction of the game level, so as to testing the performance of level encoder in the system | 71 |
| 8.10 | Level generation using text walkthrough information | 71 |
| 8.11 | Demonstration of generating novel images by text walkthrough manipulation | 73 |
| 8.12 | Samples to examine if the procedural level generator is able to produce useful level in terms of physical reasoning | 73 |

| | | |
|-----|------------------------------------------------|----|
| 9.1 | Generated level sample 1 | 90 |
| 9.2 | Generated level sample 2 | 91 |
| 9.3 | Generated level sample 3 | 92 |
| 9.4 | 7×7 generated level samples | 93 |

List of Tables

| | | |
|-----|-----------------------------------------------------------------------------------------------------------|----|
| 2.1 | Comparison of level generation methods between two categories | 7 |
| 4.1 | Summary Stats about Angry Birds game included in AbVat | 21 |
| 5.1 | Nvidia GPU specs | 33 |
| 6.1 | Type of data augmentation and its applicable domains | 56 |
| 7.1 | Software environment specification description | 58 |
| 7.2 | Hardware environment specification description | 58 |
| 7.3 | Comparison between different approaches to game level generation using walk-through description | 60 |

Introduction

1.1 Background

In Section 1.1.1, I introduce the notion of "*Physics-based simulation game (PBSG)*" and the associated research challenges. Prerequisite knowledge of PBSG is useful because Angry Birds, the game on which this thesis is based, fits into this category. Section 1.1.2 will describe the "*Procedural Level Generation*" methodology (PLG). The works of this thesis, falls under this subject.

1.1.1 Physics-based simulation game (PBSG)

Physics-based simulation game (PBSG) is a category of video games in which the behaviour of the entities in the game world obeys Newtonian physics [Renz and Ge, 2015]. Of all different forms of PBSG, Puzzle PBSG have grown in popularity among Artificial Intelligence (AI) researchers because of the difficulty it possesses in the perspective of AI agents.

The Angry Birds game, a classic Puzzle PSBG, is considered to be challenging. Stephenson et al. [2020] demonstrated that the Angry Birds game is the first single player game that is EXPTIME-hard. The rationale falls into two aspects. Firstly, the game is not fully observable. (This type of game is often referred as "information incomplete game") AI agents have no access to the internal statistics, but rather, they can only collect game state information by perceiving the visual display of the game. Secondly, The action space for playing Angry Birds game is large and continuous. An agent can shoot a bird in 360 degrees with varying amount of strength. Therefore, even a minor difference in the input will produce a drastically different result. In addition, due to the problem of floating point imprecision, even the same operation may not yield the exact same result. Thus, it is not possible for an AI agent to come up with an optimal strategy by simply computing the exact outcome of each move.

1.1.2 Procedural Level Generation (PLG)

Procedural Level Generation (PLG) is a technique in the video game industry for automating the process of level creation by using algorithms rather than manually crafting. It significantly frees developers from a labor-intensive level creation process. This statement still applies in the research area of Artificial Intelligence [Shaker et al., 2016], because learning agents require enormous amount of training set of levels to obtain competitive strength. However, designing large scale meaningful levels that requires high level of physical reasoning to solve is almost inapplicable for most of researchers [Long et al., 2017].

As a result, academic interest in PLG, an algorithm capable of synthesising large-scale training sets of levels, has grown in recent years. Additionally, generating a wide range of diverse levels for training also benefits the learning agents in terms of generalisation ability.

1.2 Motivation

The performance of the deep reinforcement learning agents in Angry Birds game often suffer from a shortage of training sets of game levels. In the AIBirds's Angry Birds AI competition [Renz et al., 2015; Stephenson et al., 2018], only 21 game levels are provided for training, which is very insufficient. To address the issue, procedural level generation is used to synthesise new Angry Birds game levels.

The current Angry Birds game level generator, developed by Stephenson and Renz [2016a,b, 2019], generates Angry Birds game levels using a rule-based generation algorithm, with the probability of choosing and placing various building blocks affected by a pre-defined fitness function.

However, a severe disadvantage of rule-based method is that people cannot encode "abstract terms" into mathematical form and feed them to the rules' function. Thus, the Stephenson and Renz procedural level generator is incapable of introducing "physical reasoning" features into its fitness function. As a result, there is no guarantee that the synthesised game levels demand a high level of physical reasoning to solve. Therefore those synthesised levels are ineffective for agents to learn physical reasoning.

In contrast, a procedural level generation algorithm that generates game levels by imitating existing high-quality handcrafted ones is considered as a better approach. The generated levels will also demand a high level of physical reasoning to solve, as they imitate those high-quality handcrafted levels. In the perspective of statistics, imitation approach does not provide explicit parametric specification of the distribution, hence preventing quality degra-

dition due to an inappropriate choice of modelling.

However, the imitation approach raises two concerns: the first is how we can control the generation process to produce required game levels in the absence of explicit generation rules, and the second is how we can enhance the diversity of the game level dataset if they merely imitate the existing ones.

Therefore, we seek for an imitation-based procedural level generation approach to generate Angry Birds game levels that require high level of physical reasoning to solve, while also addressing the two concerns inherent in the imitation approach.

1.3 Contribution

Our approach uses walkthrough descriptions to generate Angry Birds game levels and trains the procedural level generator by imitating the high-quality handcrafted levels. We tested several Generative Adversarial Networks (GANs) variants as our backbone architecture to imitate handcrafted levels. Eventually by combining StackGAN2, the SOTA GAN model, with extra training tricks, we obtain a promising preliminary results.

The main contribution of the thesis work is shown as follows:

- In comparison to the Stephenson and Renz [2016a,b]’s rule-based level generator, the proposed approach can generate better game levels in terms of the level of physical reasoning required to solve.
- In comparison to conventional imitation approach, our proposed one achieves the following things:
 1. Our approach is able to control the style of the output game levels even in the absence of explicit generation rules, by providing associated input walkthrough descriptions
 2. Our approach is able to enhance the diversity of the game level dataset as it can generate novel levels by manipulating the input walkthrough descriptions via operations such as "concatenation," "replacement," and "removal"
- The work demonstrate the limitation of using Conditional GAN [Mirza and Osindero, 2014] and Bidirectional GAN models [Donahue et al., 2016] for generation tasks under training data shortage (Shown in Chapter 5). Following that, the work proves the effectiveness of applying latent space reduction, latent space mapping, transfer learning,

multi-scale learning and data augmentation to address data scarcity and sparsity problems and thus successfully train a deep learning model even with very limited amount of training data.

- Additionally, this thesis contributes to the development of AbVat, the Angry Birds game levels and walkthrough dataset (Shown in Chapter 4). AbVat is useful not just for training procedural level generation models, but also for a variety of other study fields, such as natural language processing and reinforcement learning.

1.4 Thesis Outline

The rest of this thesis is organized as follows.

- In Chapter 2, I will review the previous literature on procedural game level generations, and then discuss the contributions and limitations of the existing approaches.
- Chapter 3 provides a concise overview of the theoretical principles and prior knowledge required for comprehending and appreciating the work of this thesis.
- Chapter 4 introduces the AbVat dataset (i.e. the Angry Birds Visual and Text walkthrough dataset), which can be used in content generation task and many other AI related research areas.
- Chapter 5 describes the preliminary attempts for the game level generation.
- Chapter 6 shows the final proposed approach for the game level generation conditioned by walkthrough description using GANs model.
- Chapter 7 specifies the experimental settings and the evaluation metrics.
- In Chapter 8 we examine the performance of the proposed design using both qualitative and quantitative evaluation and eventually demonstrate the capability of generating useful game levels for learning physical reasoning.

Review of the Literature

I will review current approaches to procedural level generation in this Chapter. To our knowledge, There is no PLG literature that put emphasis on "walkthrough-oriented" level generation. In other words, no experiments have ever been conducted to enable generators to produce levels based on walkthrough descriptions. The scarcity of available game level and walkthrough training dataset may be the primary reason for this situation.

Nonetheless, prior works on procedural level generation can be classified into two broad categories:

1. Rule-based generation with the use of Evolutionary Algorithm (Shown in Section 2.1).
2. Imitation-based generation that learns from existing handcrafted game levels (Shown in Section 2.2).

However, each has its own set of advantages as well as serious limitations. Section 2.3 will review and summarise them.

2.1 Rule-based generation

Earlier studies on Angry Birds level generation concentrated on the diversity of architecture design. [Stephenson et al., 2018] Stephenson and Renz [2016a] developed a rule-based generator for building architecture. The rules are very straightforward:

1. A structure is constructed from the top to bottom.
2. Randomly select building blocks to form a symmetrical building component.
3. Expanded the structure by stacking the symmetrical building component to the bottom middle of the existing one.

A probability table is used to sample different building blocks so as to generate building components of various shapes. Thus, although the rules are simple, it can generate large amount of diverse game levels.

In his subsequent work [Stephenson and Renz, 2016b], an additional fitness function is introduced to evaluate the overall score of the generated structures. Later in Stephenson and Renz [2019]Kaidan et al. [2016], the fitness function is used under the procedure of Evolutionary Algorithm (EA) to control the style of the output generated game levels

However, quantifying and encoding "abstract terms" into mathematical forms become the severe issue of this approach. For instance, it is not applicable to encode "playability" and "enjoyment" concepts into mathematical forms and feed into the fitness function. Thus we cannot guarantee that the synthesised levels are enjoyable to play. (Abdullah et al. [2019] once suggested that an Angry Birds level would be more enjoyable if shooting a bird could trigger a chain of domino effects, though this definition is not precise and accurate.)

2.2 Imitation of handcrafted levels

While researchers strive to develop a suitable fitness function for their rule-based generators, other studies investigated the use of machine learning models to produce game levels by imitating the existing ones. As a result, an explicit modelling of the game levels is no longer required as the generator only needs to learn how to sample new levels that are close to the training data distribution.

Prior research on the use of Deep learning generative model to imitate existing game levels seems to be yielding promising results. Amongst of those experiments, Volz et al. [2018] trained a deep convolutional generative adversarial network (DCGAN) to generate levels for video game Super Mario Bros. Jain et al. [2016] uses Variational Autoencoder (VAE), an unsupervised learning model, to create levels for the platform game "Lode Runner". Summerville and Mateas [2016] used Long Short Term Memory (LSTMs) to generate Super Mario levels. The underlying principle of LSTMs model is that given a starting point (for example, the first segment of the game level), the model will produce a sequence of content segments and then concatenate them together to form a complete level. Whereas Snodgrass and Ontanón [2016] employs a Markov model to produce Super Mario levels. Compared to RNN model, Markov model is stochastic in nature, as outputs are sampled from a probability distribution.

| | Evolutionary Algorithm based Level Generation | Imitation learning based Level Generation |
|-----------------|--------------------------------------------------|----------------------------------------------|
| Controllability | High | Low |
| Diversity | High | TBC |
| Quality | Low | High |
| Training data | Not required | Required |

Table 2.1: Comparison of level generation methods between two categories

2.3 Review and Summary

In summary, although Angry Birds procedural level generation methods that use rule-based approach are capable of generating stable and diverse game levels, the levels produced using this approach are often useless for agents to learn physical reasoning, as it is difficult to hard-code abstract concepts into mathematical forms so as to constrain the generation process (Shown in Figure 2.1). In the perspective of human players, the machine generated levels often feel cluttered and uninteresting, and they can quickly tell the difference between machine generated levels and handcrafted ones.

On the other hand, the imitation-based procedural level generation methods aim to generate high-quality levels by imitating high-quality handcrafted ones. Procedural level generators in this manner are able to produce levels containing abstract level features without the need of designing a super sophisticated rules.

However, imitation learning models lack control over the styles of the generated levels due to the absence of explicit level modelling. In contrast, rule-based generators are able to constrain the generated game levels by changing the parameters of the fitness function. (e.g. Skinny buildings can be generated by penalising high aspect ratio value in the fitness function) All the current imitation-based procedural level generator does not address this issue. Another concerns for the imitation-based model is that whether it can produce diverse game levels since it only mimic the existing ones. We will examine the diversity of the imitation-based PLG model later in Chapter 8. The comparison between two types of level generation method is given in Table 2.1.

In conclusion, the results of the existing literature indicate that only feasible solution for generating levels that is useful for agents to learn physical reasoning falls into imitation-based approach. existing handcrafted levels. However, the imitation approach raises two concerns: the first is how we can control the generation process to produce required game levels in the absence of explicit generation rules, and the second is how we can enhance the diversity of the game level dataset if they merely imitate the existing ones. The work of this

this thesis aims to solve those two concerns. Before going into the details of our approach, we will discuss the theoretical background in the next chapter, which is helpful for comprehending and appreciating the work of this thesis.

Theoretical Background and Related Work

This chapter provides a concise overview of the theoretical principles and prior knowledge which is helpful for comprehending and appreciating the work of this thesis. Since our work makes use of walkthrough descriptions to condition the produced levels, readers should familiarise themselves with the research in *Natural Language Processing (NLP)* so as to understand the work of our study. (Shown in Section 3.1). The concept "*Multimodal training*" will be discussed in Section 3.2 because our model's training process is to utilise both walkthrough descriptions and handcrafted level images. Hence the work falls into this domain. In Section 3.3, we will introduce the *Generative Adversarial Networks (GANs)* model, which serve as the backbone generative model of the proposed procedural level generator.

3.1 Natural Language Processing (NLP)

Natural Language Processing (NLP) is a set of theoretically inspired statistical techniques for the automatic evaluation and representation of human language. Since the late 1990s, statistical NLP has been the dominant field of NLP analysis [Cambria and White, 2014]. Since then, researchers increasingly shifted their attention away from syntactic and symbolic representation of texts towards semantic representation associated with the context. Words that were previously symbolised by one-hot representation are now encoded into real-valued vectors. In 2013, Mikolov et al. [2013b] published his well-known "Word2Vec" technique for generating embedding vectors of words. The theoretical grounding of two proposed model architecture CBOW and Skip-gram [Mikolov et al., 2013a] date all the way back to Harris' distributional hypothesis in 1954 [Harris, 1954]: "words that are used and occur in the same contexts tend to purport similar meanings". Later in 1957, Firth further elaborated

and clarified the distributional hypothesis: "you shall know a word by the company it keeps" [Firth, 1957]. By applying Mikolov's method, people find that words with similar semantic meaning will stay close (often in cosine distance) in the vector space. Deep Learning (DL) models that use Word2Vec or other word embedding techniques are capable of tackling more complex NLP tasks that involve semantic comprehension of the context, such as Machine Translation and Question Answering.

The architecture of the NLP models has also evolved rapidly in recent years, from RNN [Rumelhart et al., 1985], LSTM [Hochreiter and Schmidhuber, 1997], GRU [Chung et al., 2014], to Transformers [Vaswani et al., 2017]. Now, models with multi-head, bidirectional and attention mechanism, such as BERT [Devlin et al., 2018] and GPT [Radford et al., 2018] have already been able to solve several basic Natural Language Understanding (NLU) challenges.

However, these state of the art (SOTA) models still underperform in complex tasks. For instance, GPT-2 model [Radford et al., 2019] that is trained for text/story generation often generates stories that are syntactically correct but semantically and logically wrong during the testing session.[Ha, 2019]

3.1.1 NLP and Reinforcement Learning Agent

Reinforcement learning (RL) agents that are trained by using natural language often have greater generalisation performance compared to models that are trained by other modality of data. Recently, big tech companies and research laboratories such as Microsoft [Côté et al., 2018] and Deepmind [Hill et al., 2020; Abramson et al., 2020] have put a premium on text-based learning agents. For instance, OpenAI asserts that their current Visual-Language model is capable of achieving SOTA success in one-shot Visual Question Answering (VQA) classification tasks [Radford et al., 2021]. Thus, having a generative level generator that generates game levels based on guide descriptions is really beneficial. By implementing the model, text walkthroughs can serve as built-in linguistic cues for the game levels, allowing for the training of a language-based agent capable of doing physical reasoning.

3.2 Multimodal training

In the real world environment, data often exists in various modality. As such, integrating different types of data into one single agent model (also known as Multimodal model) seems to be the future target to accomplish. There is so much more to discover in terms of the architecture design, training process design, loss function design to name a few.

In order to put different modality of data in to one model, two approaches have emerged:

1. Use visual information as value, and then text information as query and key to perform attention mechanism [Yu et al., 2019].
2. Encode both text and visual information into latent vectors. After that, concatenate them into one joint latent vector with higher dimension and feed them into the model [Li et al., 2019; Su et al., 2019; Hill et al., 2020].

Although there has been no discussion about the performance differences between the two models architectures so far, the preliminary experiment of our work shows that the "concatenating latent vectors" approach fails the training process under the shortage of training data (Details are in Chapter 5).

3.3 Generative Adversarial Network (GANs)

3.3.1 Original GANs and the Adversarial Idea

The adversarial training is essentially a minimax problem, where two modules of the system compete against each other and improve their performance by turns, creating a virtuous cycle of self-improving. The adversarial concept was commonly used in Reinforcement Learning (e.g. Actor-Critic Algorithm [Konda and Tsitsiklis, 2000] and AlphaZero's Self-play cycle [Silver et al., 2017]) and now it has been applied successfully to generative model.

The original GANs model [Goodfellow et al., 2014] contains two components: a generator and a discriminator. The generator G_θ is takes input from a latent variable z and produce a synthesised samples x_g . The discriminator D_ϕ is a function that denotes the probability that a sample comes from the real dataset than the generator. Therefore, the discriminator aims to maximise the its ability to give the correct label to both real samples and generated samples while the generator aims to make fake sample so good that the chance of correctly recognising fake samples is minimised. Therefore, the objective function of GANs is shown below:

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))] \quad (3.1)$$

The formula derives from the cross-entropy between the real and generated distribution. However, during the real practice, the Equation 3.1 cannot provide sufficient gradient for generator to train. It is because during early training stage, the quality of generated data is poor and thus they can be easily recognised by the discriminator (i.e. $D(G(z))$ tends

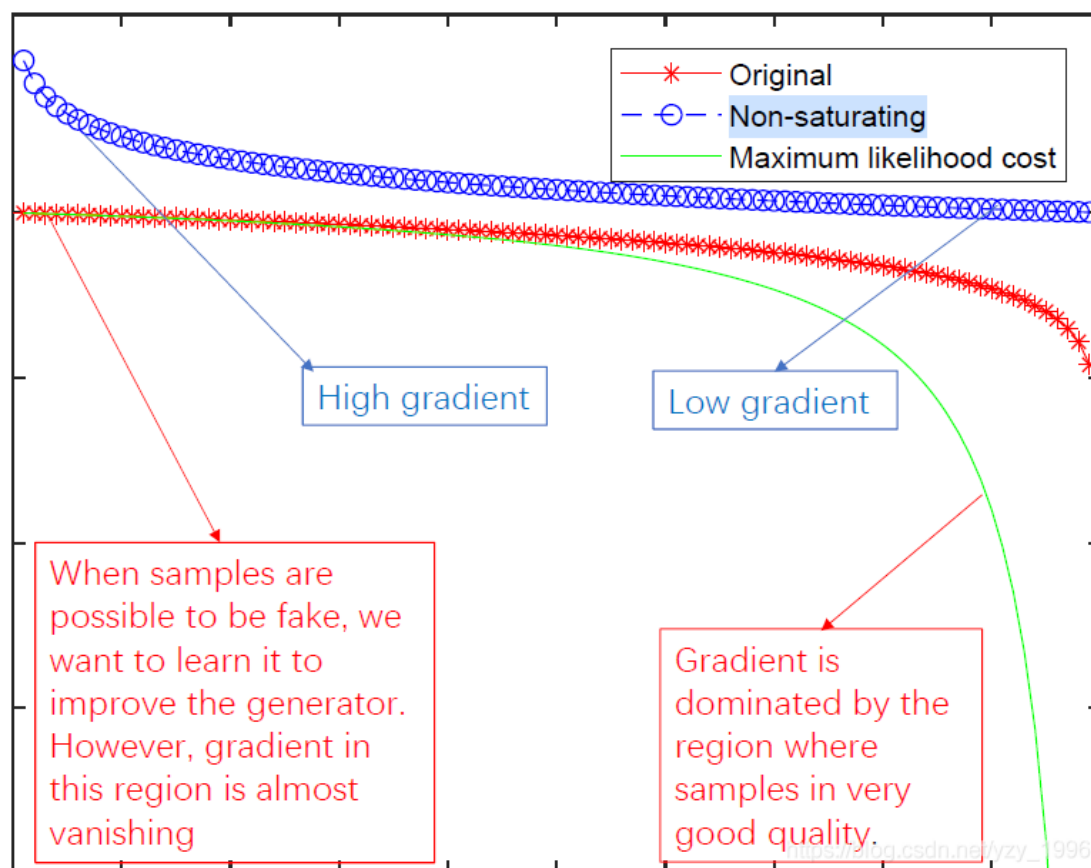


Figure 3.1: Saturating Loss problem illustration

Copyright and adapted from Nips 2016 tutorial: Generative adversarial networks[Goodfellow, 2016]

to be 0). In this case, $\log(1 - D(G(z)))$ does not provide sufficient gradient to train the generator. As such, the author suggests to change objective from $\min_G \log(1 - D(G(z)))$ to $\min_G -\log(D(G(z)))$ to provide much larger gradient during the early stage. This is also known as "saturating loss problem". (See Figure 3.1)

3.3.2 GANs and Density Estimation

The following well-known definitions can be found in many statistics textbooks.

Definition 3.1 (Density Estimation) *Given a finite set of observations of a random variable x_1, \dots, x_n , we then want to model the distribution of the population of random variable x . This task is known as density estimation.*

Since the main task of GANs model is to generated samples $\sim p_g(x)$ that look like

real data samples $\sim p_{data}(x)$, GANs model belongs to the category of density estimation (DE). Goodfellow [2016] classifies generative models into two broad categories: explicit and implicit density models. The distinction between the two is that explicit density models explicitly define and solve the estimated distribution $p_g(x)$ while implicit density model generate samples without explicitly defining $p_g(x)$. The following example illustrates the difference:

1. The well-known explicit density generative model Variational Autoencoder (VAE) [Kingma and Welling, 2013] is able to infer latent representation of the real data samples $p_g(z|x)$.
2. However, we are not able to do so with the GANs model because the density function $p_g(x;\theta)$ is not explicitly defined.

Researchers [Goodfellow, 2016; Lotter et al., 2015; Huang et al., 2017] discovered that GANs frequently yield higher-quality samples than explicit density models such as VAE. It is because we are not able to provide explicit density models an appropriate prior distribution or posterior distribution in real world problem (just like how people cannot appropriately encode "abstract features" into rule-based procedural level generator). Thus, the estimated density distribution generated through explicit modelling may not closely match the real data distribution. In contrast, the GANs model is not required to define a prior distribution. As a result, the generated samples will not be negatively affected by the unsuitable prior. This is also the same reason that people prefer to use deep learning instead of explicitly hardcoding features.

3.3.3 Variations of GANs model

We will introduce GANs' representative variants in the following section.

3.3.3.1 DCGAN

Deep Convolutional Generative Adversarial Network (DCGAN) [Radford et al., 2015] is a significant milestone because it is the first time that convolutional neural networks (CNN) and GAN are successfully combined and generate high quality images. The major innovations of the model are shown as follows:

1. Using "stride" instead of "pooling" for down/up sampling to avoid information loss.
2. Using batch normalisation for stabilising the learning process.
3. Unlike VGG network [Simonyan and Zisserman, 2014], no fully connected layers are added after the convolutional layer.

4. Using ReLU and LeakyReLU activation functions instead of Tanh to avoid saturation of the gradient.

Since then, the vast majority of GAN architectures used in Computer Vision (CV) have been built on the DCGAN architecture.

3.3.3.2 Conditional GAN

Mirza and Osindero [2014] proved that when the model is trained by using a longer joint latent representation obtained by concatenating embedding vectors of additional information and the original latent variable z , the GANs model can generate synthesised images conditioned by the given information. This discovery paves the way for future research on text to image generation using a GANs model.

3.3.3.3 Wassertrin GAN

Arjovsky et al. provide extensive theoretical work on the reason why original GANs model often encounters gradient vanishing and mode collapse [Arjovsky et al., 2017; Arjovsky and Bottou, 2017]. Arjovsky et al. proves the following, which appears as theorem 2.5 of Arjovsky and Bottou [2017]

$$G^* = \arg \min_G E_{x \sim p_{model}(x)} [-\log D^*(x)] = \arg \min_G \{ \text{KL}(p_{model} || p_{data}) - 2\text{JS}(p_{model} || p_{data}) \} \quad (3.2)$$

The Equation 3.2 above shows the objective function of the generator of GANs model. Such a objective function will cause the following problems.

1. Gradient instability: When $p_{model} \cap p_{data} \neq \emptyset$ and the distribution of both real data and synthesised data get closer, $\text{KL}(p_{model} || p_{data})$ and $-\text{JS}(p_{model} || p_{data})$ will contradict each other because both KL and JS distance are measuring the difference between two distributions. Thus, it will cause the gradient unstable.
2. gradient vanishing: When $p_{model} \cap p_{data} = \emptyset$, the function will give static value and thus the gradient become 0.
3. Mode collapse:
 - When $p_{data}(x) > p_{model}(x), p_{data}(x) > 0, p_{model}(x) \approx 0$, the KL divergences tends to be 0.
 - When $p_{model}(x) > p_{data}(x), p_{model}(x) > 0, p_{data}(x) \approx 0$, the KL divergences tends to be ∞ .

It turns out that the generator will face a far higher penalty if it generates alien samples. As a result, the generator will choose to produce repetitive but safe pictures rather than risk making more diversified ones. Such a problem is called "Mode collapse". In fact, every model that uses reverse KL divergences as its optimisation metric may face the aforementioned issue of mode collapse.

3.3.3.4 Progressive GAN

Generating a required high quality synthesised image is a challenging task. Therefore, Karras et al. [2017] proposed a novel training methodology for GANs. They started from generating low resolution images and then gradually add new layers to both the generator and discriminator so as to imitate more details progressively throughout training. This methodology both accelerates and stabilises the training process. For the first time, images with exceptional high resolution can be generated using GANs.

3.3.3.5 StackGAN

While CGAN is able to generate images that are conditioned by the added text information, the resulting images are frequently of poor quality. As such, Zhang et al. [2017b] divided the work into two sub problems, the first of which focuses on the generation of low resolution images and the second of which focuses on adding fine details on the top of the previous generated ones so as to form high resolution images. This "divide and conquer" training concept is very similar to that of Progressive GAN. Additionally, this study introduced a novel technique called "Conditioning Augmentation", which is to encode text information as a distribution over the latent space instead of encoding it into a single point. This concept is very similar to how VAE model produce the latent distribution for its real data samples. This technique prevents the encoder from generating discontinuous latent data manifold, hence stabilising the learning process.

3.3.3.6 BiGAN

GANs are capable of learning models that map from basic latent distributions to arbitrarily complex data distributions. However, due to the fact that the original GANs is an implicit density estimation model, it is incapable of learning the inverse mapping (i.e. the process of projecting data back into the latent space). BiGANs, proposed by Donahue et al. [2016], may be beneficial for auxiliary supervised discriminating tasks. The primary novelty in this approach is the addition of an encoder that converts the real data to latent representations and

instructs the discriminator to discriminate not only the samples but also the latent vectors. The extra encoder can do inverse mapping after training.

3.3.3.7 BigGAN

Brock et al. [2018] demonstrated that increasing the number of training samples, adding more channels (i.e. the number of parameters), and increasing the batchsize significantly improves the quality of the synthesised images from GANs models. Additionally, they demonstrated that the Truncation Trick [Marchesi, 2017], a latent sampling procedure in which latent vectors are sampled exclusively from a truncated normal distribution, aids in the generation of high-fidelity images.

3.4 Summary

Readers are expected to have a basic understanding about NLP, Multimodal training and GANs model after this Chapter. These concepts serve as the foundation for the thesis' work.

Imitation-based procedural level generation requires handcrafted game levels to as training data. As such, Chapter 4 will introduce a new dataset called "AbVat", that serves as the training dataset of the thesis' work. In what follows, the data collecting, cleaning, and preparation processes will be discussed in detail.

AbVat Dataset

Training dataset is the backbone of all learning-based models. Any deep learning models will fail to operate in the absence of a training dataset.

In order to train our machine learning model that generates game levels based on walk-through information, we require a training dataset that contains both game level structures and walkthrough descriptions. As such, this chapter introduces a new dataset named "AbVat", a "Angry *birds* Visual *and* Text walkthrough" dataset that consists of 1299 Angry birds level structures and their associated walkthroughs. (See Figure 4.1)

The properties of AbVat will be discussed in Section 4.1. Following that, we demonstrate how AbVat is constructed using an open-source web crawling technology. Additionally, in Section 4.2, we will discuss the data cleaning and preprocessing steps necessary to eliminate inconsistent and noisy data. We conclude the chapter by addressing all pertinent research tasks for which AbVat might be used as a training and benchmark dataset.

The reader is expected to comprehend the importance of the AbVat dataset and the ways in which it can be used as a training and testing resource for agents to learn physical reasoning.

4.1 Properties of AbVat

The current version of AbVat is built upon the official Angry Birds game series from Rovio Entertainment Corporation. This dataset will continue to evolve over time as we incorporate additional handcrafted levels and walkthrough information pairings. The current version of AbVat has the limitation of being unable to acquire the internal representation of the level structures owing to copyright concerns. At the moment, only pixel-based representations of levels are provided. In the future, we intend to migrate the pixel-based levels into accurate numeric representation using XML format. By doing this, we can directly reconstruct the game level in the open source version of Angry Birds: Science Birds.

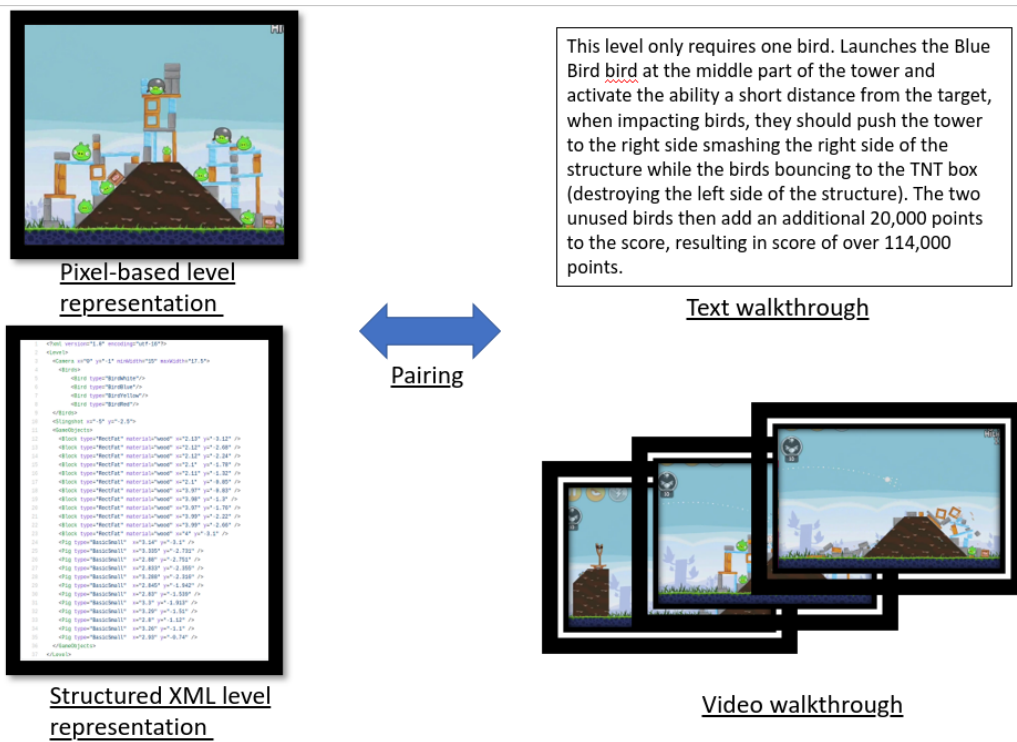


Figure 4.1: Demonstration of AbVat data

| Version. | No. of game levels | Avg. duration of video walkthrough /sec | Avg word count of text walkthrough |
|----------------------|--------------------|-----------------------------------------|------------------------------------|
| Original Angry Birds | 460 | 35.91 | 48 |
| Angry Birds Space | 393 | 35.72 | 47 |
| Angry Birds Star War | 209 | 30.75 | 45 |
| Angry Birds Seasons | 237 | 38.95 | 52 |

Table 4.1: Summary Stats about Angry Birds game included in AbVat

AbVat's goal is to collect a massive volume of high-quality game level and walkthrough data pairs through the course of its development. At the time of writing, AbVat has 1299 high-quality Angry Birds game levels and their associated walkthroughs. The data comes from the original Angry Birds game and its several variants, including "Angry Birds Star Wars", "Angry Birds Season" and "Angry Birds Space". (See Table 4.1)

4.1.1 Level representation

There are currently two ways to represent an Angry Birds level in AbVat. The first is a pixel based depiction of levels. It is done by capturing the overall structure of the game level manually on the display. The second is to represent through the a structured format. At the moment, we used the same XML format as Science Birds, which comprises four attributes for each game object (See Figure 4.2).

The usage of structural XML format has a number of advantages, one of which is that it enables developers to swiftly reconstruct the level right within in the Science Bird open-source framework, whereas a pixel-based level representation requires human labour to judge the content so as to retrieve the state of each game object. One may argue that we can use object detector to automate the process. For instance, Wang [2013] of Australian National University created the most widely used object extractor for Angry Birds game. However, because Wang's extractor's primary methodology is based on hardcoded rules, it is incapable of detecting new objects such as a basketball or a space ship from new version of Angry Bird games (See Figure 4.3).

4.1.2 Walkthrough data

There are two types of walkthrough data: walkthrough descriptions and walkthrough videos. The walkthrough descriptions do not provide specific instructions for players to follow (e.g.

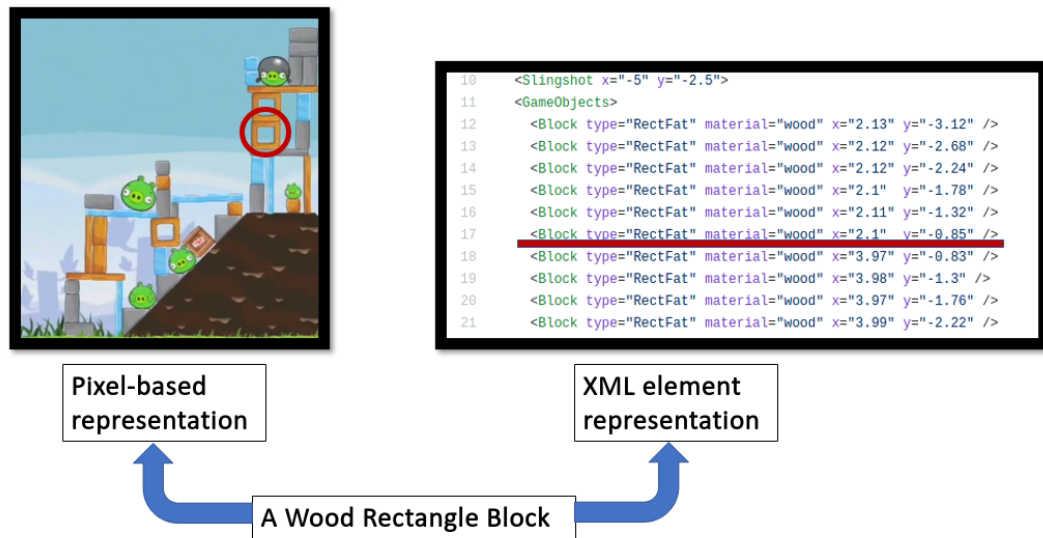


Figure 4.2: Comparison between pixel-based and XML object representation



Figure 4.3: Novel object in Angry Birds Seasons Ham Dunk 2-2 - Kings episode

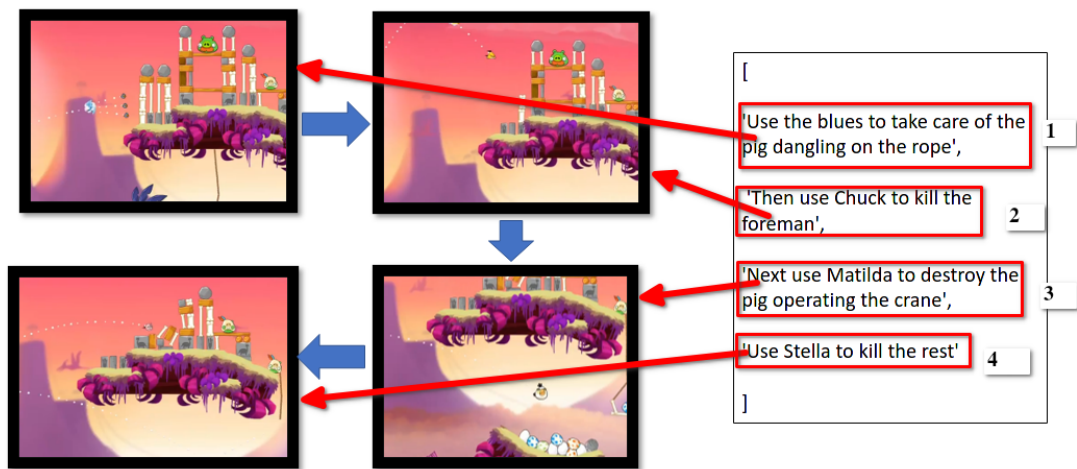


Figure 4.4: Example of the text segmentations of a walkthrough description based on critical states

Shoot Yellow birds at a 45-degree angle with 0.7 force), but rather they are general strategies. Notice that some levels contains more than one set of walkthroughs since they can be solved in a various ways.

The text version of the walkthrough is manually divided into chunks based on the critical states (See Figure 4.4), with the following definition:

Definition 4.1 (Critical state) *The term "critical state" refers to the moment when an active object (e.g. a bird) accomplishes an action that directly alters the environment of the game level or contributes to a chain of actions that lead to significant changes of the level environment.*

Compared to just supplying raw walkthrough descriptions, the segmentation of the texts provides auxiliary spatial and temporal information. This extra work allows us to train the model more efficiently or to conduct other meaningful research tasks in the future (Section 4.3 will go into further detail).

Not every text segment can be matched with a portion of the video walkthrough. It is because the text description may contain backup plans in case the previous shot fails to accomplish the objective. However, the video walkthroughs neither record the failed attempts nor backup plans. Therefore, it is a quite interesting research task that asks agents to distinguish between real and assumptive strategies. Indeed, the capacity to comprehend hypothetical situations is a critical component of AI's causal reasoning capability.

4.1.3 Diversity

AbVat is designed with the purpose of including a diverse range of handcrafted Angry Birds game levels. Diversity is a very essential criteria for training datasets. Models that are trained with diverse training dataset often have higher generalisation ability, which is essential in order to address the novelty problem. Following is the well-known definition:

Definition 4.2 (Novelty problem) *The phrase "novelty problem" refers to the circumstance in which AI encounters novel scenarios. The novelty hierarchy may be divided into four levels:*

1. *Novelty level 0: Objects or entities that have never been observed before.*
2. *Novelty level 1: Class of things or phenomena that have never been observed before.*
3. *Novelty level 2: Modification of an object's characteristics, such as its colour or shape, that was previously irrelevant and will affect the categorisation of objects.*
4. *Novelty level 3: A change in the way entities or features are specified, as a result of a dimension or coordinate system changes.*

The capacity to solve novelty problem is critical for real world AI systems. In other disciplines of research, this issue is also refereed as "Zero Shot Learning" (ZSL). The definitions can be found in many machine learning textbooks.

Definition 4.3 (Zero-shot learning) *In machine learning, zero-shot learning (ZSL) is a problem scenario in which a learner examines samples from classes that were not seen during training and must predict the class to which they belong. In general, zero-shot algorithms function by linking normal and novel classes with some type of auxiliary information that encodes unseen objects into a known domain.*

AbVat dataset enhanced its diversity by including many variants of the Angry Birds game. The current version of AbVat has four distinct Angry Birds variants and each variant has its own set of unique game objects, including new birds, new building blocks and even new background.

Additionally, the inherent physics simulation also varies between different variants, including item mass, friction coefficient, and even gravitational field. For instance, the gravitational force is directed downward in the original Angry Birds game. In Angry Birds Space, gravity is directed towards centre of the planet. As a result, birds may even rotate like satellites around the planet. Since these varying physical parameters are not observable by the players, agents must have a grasp of Newtonian physics in order to defeat all versions of the

Angry Birds game rather than simply memorise the trajectories. Hence, the diversity of the game levels make all the associated tasks challenging.

4.2 Constructing AbVat

The procedure of constructing AbVat dataset is described as follows.

4.2.1 Data collection

The initial stage of AbVat's construction is to collect high quality game level walkthroughs from the Internet. The current version is primarily based on data gathered from a well-known wiki service and forum website called Fandom. It bills itself as the largest fan platform for wiki hosting services in the world. The majority of the official Angry Birds game walkthroughs are available at the Fandom website's "Angry Birds Wiki".

In order to accelerate the data collecting process, we utilise the Scrapy framework to crawl the contents from the website. Scrapy is a Python web-crawling framework that is completely free and open-source. It has a thriving community with 36.3k stars and 8.4k forks on GitHub, and 14.7k related questions on StackOverflow. We queried different key terms when crawling the web to get as many level walkthroughs and level information as feasible. For example, we employ terms such as "Three Star Walkthrough," "Three Star Strategy(ies)," and "Walkthrough material" to search the wiki system for text descriptions of game level walkthroughs.

4.2.2 Data cleaning and preprocessing

To ensure that the information and walkthroughs collected are of highest possible quality, Each candidate of the data is manually cleaned and pre-processed. We excluded contents that are irrelevant to the gameplay in the following way:

1. In terms of video walkthroughs, the game menu screen at the start of each episode, the score counting scene at the end of each game episode and occasionally unrelated video advertisement scenes are removed (See Figure 4.5). Also, the video clips has been cut such that the first frame provides an overview of the whole level structures. It allows agents to retrieve level information more conveniently.
2. In terms of the text walkthroughs, sentences that are not related to the gameplay are removed (e.g. the discussion about the final score of each action). We also delete

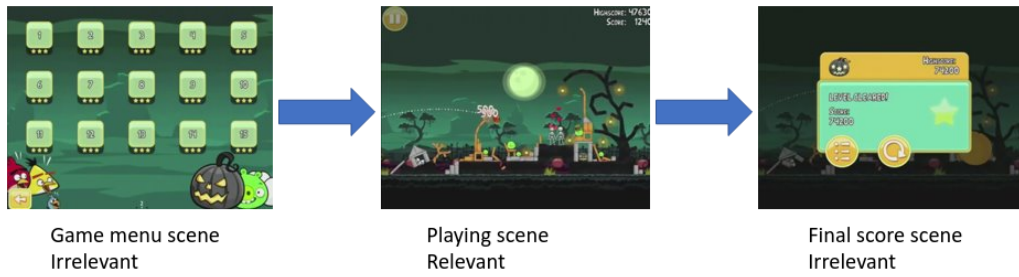


Figure 4.5: Demonstration of irrelevant contents in walkthrough video

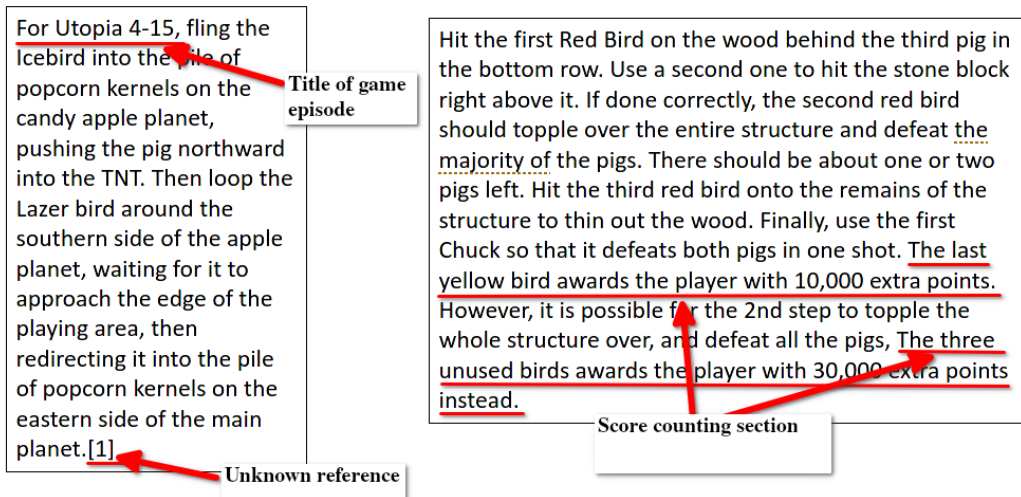


Figure 4.6: Example of noisy contents in a text walkthrough

sentences that contain the titles of Angry Birds game levels. This can decrease the likelihood that agents will merely memorise the exact successful actions associated to that specific game level. We also remove sentences that require agents to refer to other game levels. It is because this instruction requires agents to have external memory in order to store the knowledge of previous game levels and use that to generate solutions for the current one, which is not the primary interest of this dataset (See Figure 4.6).

As previously mentioned, each candidate for the text walkthrough is segmented based on "critical states". In comparison to delivering raw text paragraph, segmenting the text can offer supplementary spatial-temporal information about the game level, which helps models to train more effectively.

Because of the dataset’s current scale, there is only one human labeller. While data cleaning and preprocessing are carried out, I has to admit that the human labeller’s judgement on the text walkthrough’s segmentation may be biased. The remedy to this issue of bias is to have different human labellers clean and pre-process the text walkthrough data independently.

4.3 Benchmark dataset for various tasks

To our knowledge this is the only available game level and walkthrough dataset for the Puzzle Physics Based Simulation game (PBSG). Learning physical reasoning is challenging for Artificial Intelligence. Therefore, AbVat can be used as a benchmark dataset for a variety of meaningful research tasks, including the following:

- Category 1: in the presence of a interactive gaming environment
 1. Imitation learning (a.k.a. Inverse reinforcement learning): Use walkthrough descriptions as additional information, train agents’ behaviours by having them imitate the player moves in the walkthrough video.
 2. Meta learning: During reinforcement learning, use text walkthroughs as meta data and then examine the learning efficiency of employing text walkthroughs as auxiliary information.
 3. Robustness testing: Compare several descriptions and determine which one allows for faster learning for agents.
 4. Decremental learning: Remove text tokens from the walkthrough description gradually until the agent becomes unable to extract sufficient information to complete the level.
 5. State analysis and prediction: Check if the game’s current state matches the expected stage indicated in the text walkthrough and then determine if the game is still solvable in its current state.
- Category 2: in the absence of a interactive gaming environment
 1. Level Generation: Generate game levels based on the walkthrough descriptions of the game episode. Generate novel levels by combining segments of description from different game levels.

2. Walkthrough generation: Generate game walkthrough description by observing the level structure.
 3. Level Clustering: group similar levels based on the description or/and the level information.
 4. Parsing task: match sentences in the description to the associated interval of the video content. It requires agents to possess temporal understanding and reasoning.
 5. Slot filling task: Interpret the walkthrough description by filling in the pre-defined slot.
 6. Relationship Extraction/Prediction: Extract/predict the relationships between different entities in the game and how they change over time. The relationship between entities may not be explicitly clarified in the text walkthrough.
- Category 3: with additional labelling
 1. Matching task: Match objects and events mentioned in the text walkthrough with those shown in the video. It requires additional object and event labelling.
 2. Advanced Visual Question Answering (VQA): Provide queries to agents that demand spatial-temporal understanding and reasoning.

As shown above, AbVat can serve as a potential benchmark test for a variety of meaningful research tasks, particularly those involving physical reasoning and understanding.

4.4 summary

In summary, AbVat is a valuable dataset that can provide a variety of meaningful research tasks in the domain of physical reasoning and understanding. In this thesis, we only use it as training dataset for our procedural level generator.

In the next chapter we will discuss the preliminary attempts of our work.

Preliminary attempts

The aim of this thesis is to use walkthrough descriptions to generate Angry Birds game levels and train the procedural level generator by imitating the high-quality handcrafted levels and we decided to use GANs, the SOTA generative model, as the backbone architecture for our approach.

Section 5.1 discusses the initial attempt which tried to train an end-to-end cGAN model directly.

Section 5.2 discussed the second attempt, in which we used BiGAN model and divided the tasks into two sub problems: the first is just a normal level generation from latent space to game level content using BiGAN; the second is a mapping task from the text description to the latent representation of the corresponding game level.

Unfortunately, both of them failed because of some critical oversights, such as the sparsity issue in the high-dimensional latent space, and the data inefficiency issue during transfer learning. Nonetheless, we identified the root causes of each attempt's failures and devised remedial strategies in our final approach.

5.1 Preliminary Attempt 1: contrastive learning on text and image pair

The initial attempt is based on a very simple intuition. By having the text information into the latent value of the GANs generator, we can generate levels that are conditioned by the provided text. This is precisely the Conditional GAN idea from [Mirza and Osindero, 2014].

In addition to that, we included a training trick, called Contrastive Training, so as to improve the text's and levels' semantic coherence. Contrastive training has been a popular and effective strategy for classification tasks.

The following sections will cover the detailed methodology and experimental results,

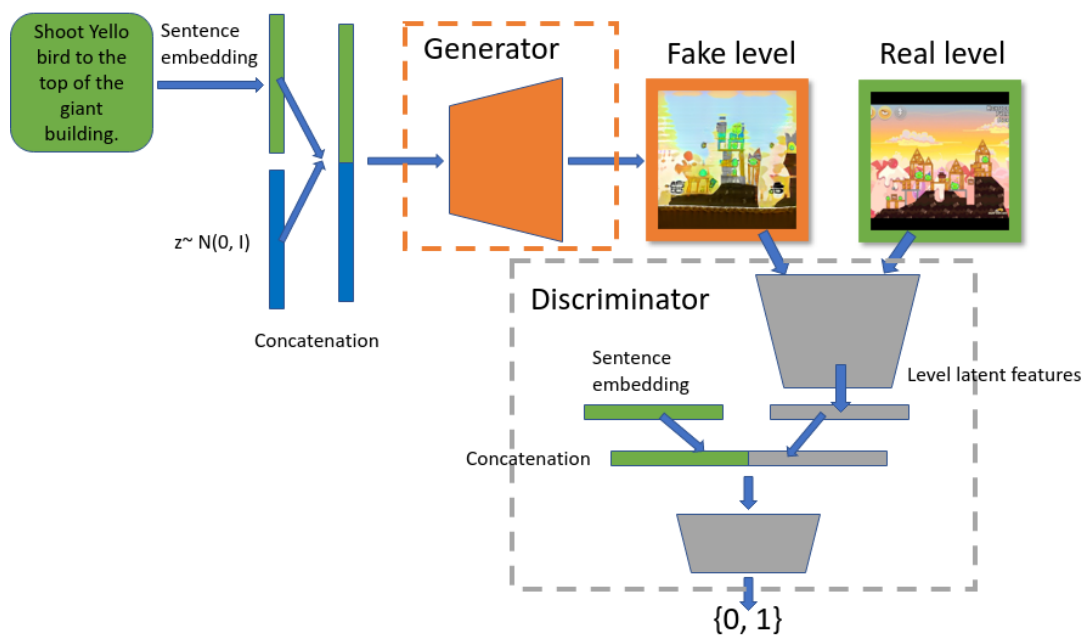


Figure 5.1: The architecture of the cGANs based model of the first attempt.

followed by a summary of the findings and suggestions for improvement.

5.1.1 Method

5.1.1.1 Overview

The initial attempt was based primarily on the cGAN architecture [Mirza and Osindero, 2014]. The walkthrough sentence embeddings were concatenated with the latent variable z to create a combined latent representation. It was then sent into the generator, which generates synthesised pixel-based level representations. Following that, both the actual and false samples were again combined with the sentence embeddings and sent to the discriminator. The discriminator will attempt to classify the incoming samples correctly. The loss value associated with the discriminator's performance will provide gradient for back-propagation method, training both the discriminator and the model generator (See Figure 5.1).

5.1.1.2 Sentence embedding

In order to concatenate walkthrough descriptions information to the original latent vector z , the walkthrough descriptions of the game level must be translated into embedding vectors with the semantic meaning preserved. We employed the off-the-shelf sentence encoder "Sen-

tence BERT" from Reimers and Gurevych [2019] in this experiment. In comparison to vanilla BERT, Sentence BERT incorporates new pooling and concatenation strategies on top of the basic BERT architecture, significantly improving the efficiency of the sentence embedding process. The off-the-shelf "Sentence BERT" transformer from the huggingface open-source library encodes a sentence into a 768-dimensional embedding vector.

5.1.1.3 Latent representation z

In GANs model, the generator takes a point from latent space z as input and generates synthesised samples. For cGAN to work, the generator must concatenate the original latent vector z with the conditioned information y and form joint latent representation. As such, the ratio of text embedding space to original latent space may have an effect on the final result. Typically, researchers construct GANs models using a 100-dimensional hypersphere with each variable derived from a Gaussian distribution with a mean of zero and a standard deviation of one. The well-known text-to-image GANs models, StackGAN, employed size 128 and 100 respectively for text embedding space and latent space. However, so far there has been no official discussions or agreements on the best ratio settings. In our experiment, we followed the conventional 100-dimensional latent space for z . However, further trials and analysis are required to determine the ideal parameters.

5.1.1.4 Contrastive training

Theoretically, training using negative examples enables the discriminator to develop a more consistent semantic relationship between the conditional text information and the samples. Other works, such as Tao et al. [2020], advocated for the inclusion of negative samples in GANs models to guarantee that the matched sample-text data points are located at the lowest point on the discriminator loss function surface, resulting in an efficient minimax process (See Figure 5.2).

As a result, the loss function of the cGAN architecture combined with contrastive learning is as follows:

$$\begin{aligned}
 L_D = & - \mathbf{E}_{(x,t) \sim p_{data}} [\log D(x, e_t)] \\
 & - \mathbf{E}_{z \sim p_z, t \sim p_{data}} [\log(1 - D(G(z, e_t), e_t))] \\
 & + \lambda \mathbf{E}_{(x, t_{mis}) \sim p_{mismatch_data}} [\log D(x, e_{t_{mis}})]
 \end{aligned} \tag{5.1}$$

$$L_G = \mathbf{E}_{z \sim p_z, t \sim p_{data}} [\log(1 - D(G(z, e_t), e_t))] \tag{5.2}$$

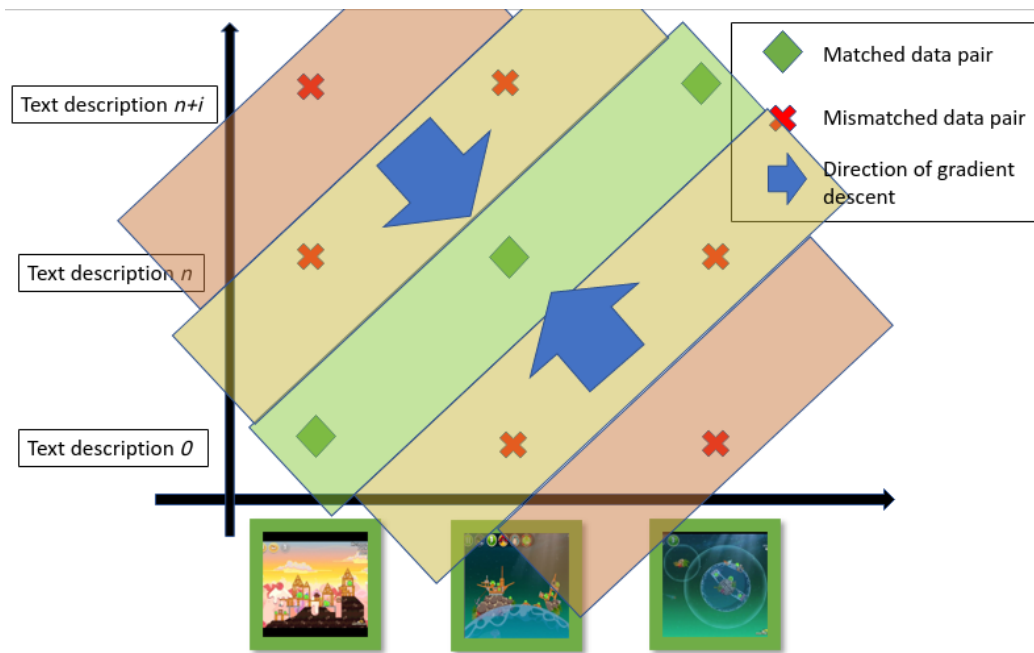


Figure 5.2: A diagram for contrastive learning and negative samples

Equation 5.1 represents the loss function for the discriminator in the GANs model, whereas Equation 5.2 represents the loss function for the generator in the GANs model. x denotes actual game level samples, whereas t denotes the corresponding text walkthrough. e_t is the embedding vector and λ is the contrastive training term's coefficient.

5.1.2 Experiment and result

5.1.2.1 Hardware and software settings

The model was built using Pytorch framework [Paszke et al., 2019] with Nvidia GeForce RTX 3090 GPU (See Table 5.1)

5.1.2.2 Dataset

AbVat dataset is our only choice. AbVat was discussed in length in Chapter 4.

| GPU | TITAN RTX | GeForce RTX 3090 |
|------------------|-------------------------|-------------------------|
| SMs | 72 | 82 |
| CUDA Cores | 4608 | 10496 |
| Tensor Cores | 576 (2nd Generation) | 328 (3rd Generation) |
| RT Cores | 72 | 82 |
| ROPs | 96 | 112 |
| GPU Boost Clock | 1770 MHz | 1695 MHz |
| Memory Clock | 7000 MHz | 9750 MHz |
| Total Memory | 24GB GDDR6 | 24GB GDDR6X |
| Memory Interface | 384-bit | 384-bit |
| Memory Bandwidth | 672 GB/s | 936 GB/s |
| TGP | 280W | 350W |

Table 5.1: Nvidia GPU specs

5.1.2.3 Training settings

The suggested network was optimised using the Adam algorithm [Kingma and Ba, 2014] with $\beta_1 = 0.0$ and $\beta_2 = 0.9$. According to the Two Timescale Update Rule (TTUR) [Heusel et al., 2017], the learning rate for the generator is set to 0.0001 and 0.0004 for the discriminator. The contrastive learning term’s coefficient is set at 0.5. The model was trained over a period of 200 epochs.

The loss function of the training process was WGAN-GP [Gulrajani et al., 2017]. As mentioned in Section 3.3.3.3, WGAN-GP assists in resolving mode collapse and gradient vanishing problems for GANs model by applying the following steps:

1. Convert to a regression task by removing the log in the loss function.
2. Second, optimise using Wasserstein distance rather than JS divergence.
3. Thirdly, employ the gradient penalty to establish Lipschitz continuity (a term that refers to a strong form of uniform continuity for functions).

5.1.2.4 Evaluation settings

A qualitative inspection and a loss plot of the model will be given. Because of time constraints, we were unable to run benchmark tests using other state-of-the-art approaches. Nonetheless, the results of the second attempt as well as the final approach may be utilised to conduct parallel comparison between different approaches.

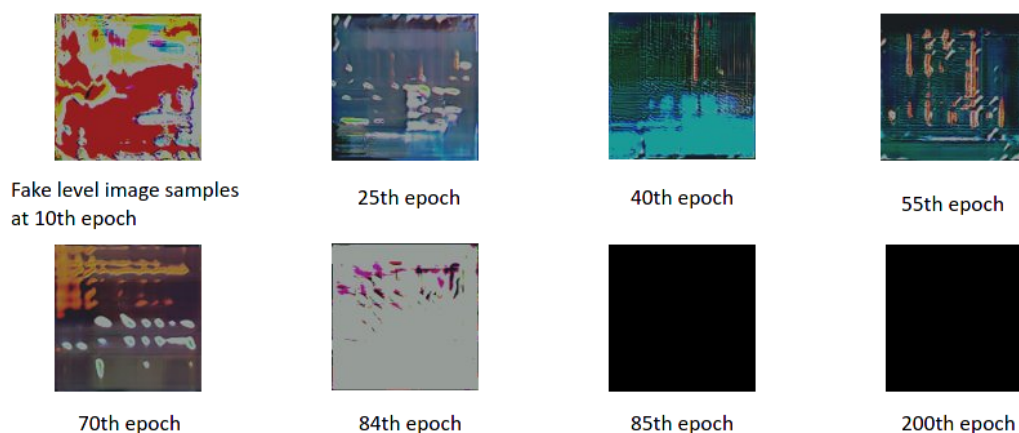


Figure 5.3: Angry Birds game level image samples generated in the first preliminary attempt

5.1.2.5 Result

After 85 epochs, the synthesised samples were entirely black, which indicated the mode collapse of the model. Additionally, materials synthesised prior than 85 epochs were incapable of forming consistent and recognisable structures. Moreover, there is no discernible improvement in image quality during the course of the training (See Figure 5.3).

According to Figure 5.4, the loss value of the discriminator dropped to near zero very early in the training procedure. In contrast, the generator's loss value oscillated more strongly during the training duration. Additionally, the generator's average loss value increased through the training time, showing that it does not converge to a stable and performable stage.

5.1.3 Discussion

The results indicate that cGAN failed to generate meaningful pixel-based level representations via walkthrough descriptions.

Following a thorough examination of the method and theory, I conclude that the following two aspects are the primary causes to the failure:

1. Discriminator overpowered the generator because of the additional contrastive training procedure, resulting in very little gradients provided for back-propagation. Hence the generator cannot progress.

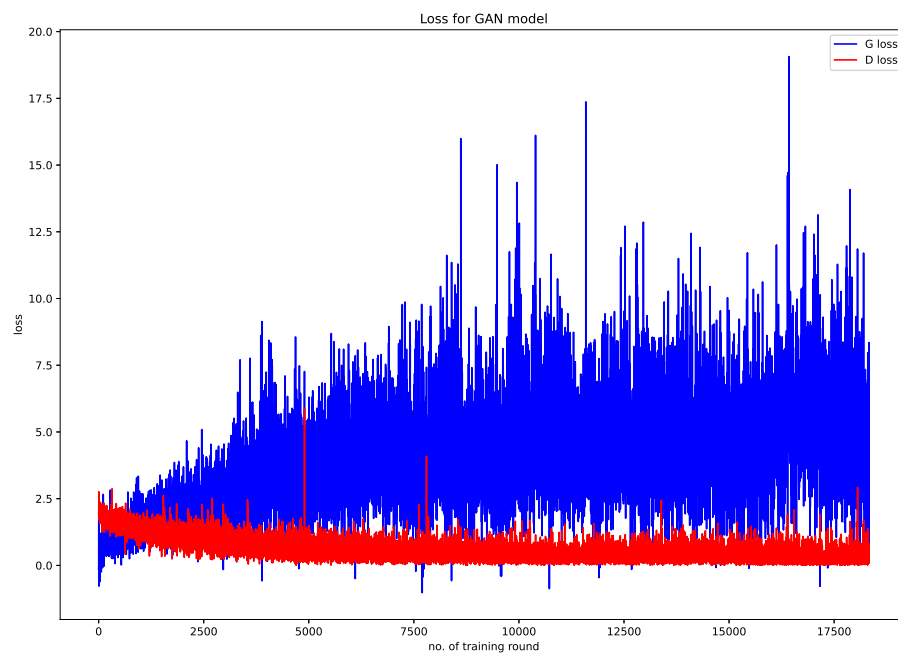


Figure 5.4: Loss plot for GANs model

2. The concatenation of the latent vector and the text embedding resulted in a sparse and discontinuous latent data manifold, which makes interpolation difficult for machine learning models.

5.1.3.1 "Precocious" discriminator due to contrastive learning

According to Equation 5.2, a well-performing D produced insufficient gradient for generator to make progress, resulting in the generator being trapped at local minimum and producing low quality samples. Recalling the GANs model's loss function (Equations 5.1 and 5.2), we can see that the discriminator's loss function is constrained by an additional contrastive training term. According to Figure 5.4, by training with extra mismatched data pairs (i.e. negative sample), the discriminator converges significantly quicker than the generator.

Some may suggest to form an balanced and effective minimax adversarial training process by adding contrastive training loss into the generator's loss function. However, it is unhelpful. Recall that the objective for the original generator is $\min \mathbf{E}_z[\log(1 - D(G(z)))]$, which has the intuitive connotation that the generator wishes to create synthesised samples that are extremely similar to genuine samples, such that the discriminator recognises the synthesised sample as real. However, applying additional contrastive learning cost makes no sense, and the generator will eventually output low-quality samples in order to lower the objective. The following is a demonstration:

$$\begin{aligned}
\min L_G &= \min \mathbf{E}_{z \sim p_z, t \sim p_{data}} [\log(1 - D(G(z, e_t), e_t))] && \text{[original loss]} \\
&\quad - \mathbf{E}_{z \sim p_z, t_{mis} \sim p_{mismatch_data}} [\log(1 - D(G(z, e_{t_{mis}}), e_{t_{mis}}))] && \text{[contrastive loss]} \\
&= \min \mathbf{E}_{z \sim p_z, t \sim p_{data}} [\log(1 - D(G(z, e_t), e_t))] \\
&\quad + \mathbf{E}_{z \sim p_z, t_{mis} \sim p_{mismatch_data}} [\log D(G(z, e_{t_{mis}}), e_{t_{mis}})] \\
&\implies \min \mathbf{E}_{z \sim p_z, t_{mis} \sim p_{mismatch_data}} [\log D(G(z, e_{t_{mis}}), e_{t_{mis}})]
\end{aligned}$$

The loss function eventually tends to generate a sample $G(z, e_{t_{mis}})$ of low quality because it is the simplest way of lowering this generator's objective.

In conclusion, while contrastive training is beneficial for training classifiers in Machine Learning tasks, it is ineffective for developing GAN models. It is because the contrastive training will unbalance the training pace between the discriminator and generator, causing the discriminator to converge much more quickly and subsequently provide the generator with insufficient loss information to advance. The solution of it is to just avoid the use

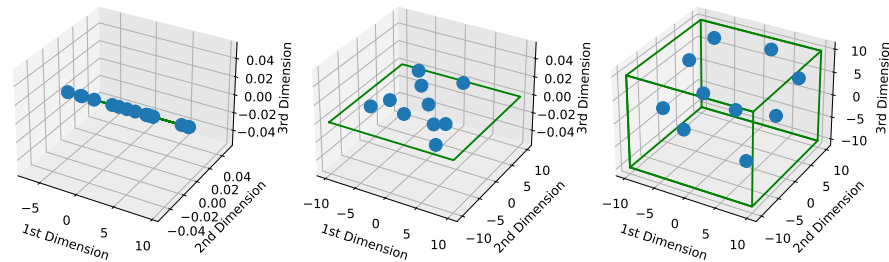


Figure 5.5: Demonstration of the sparsity problem as no. of dimension increases

of contrastive training metric in the discriminator's cost function. The under-trained discriminator's loss value generates a more stochastic gradient for the generator, increasing the likelihood of leaping out of the local minimum.

5.1.3.2 Sparse and discontinuous latent data manifold

Concatenating the text embeddings to create a combined latent representation resulted in a more sparse and discontinuous manifold of latent data. Our text embedding is a 768-dimensional vector and the original latent variable z is a 100-dimensional vector, the joint latent vector (z, e_t) is then laying in an 868-dimensional latent space. As the number of dimensions rises, the data becomes exponentially more sparse, making interpolation task extremely difficult for the machine learning model to handle. This is frequently referred to as "the curse of dimensionality".

As seen in Figure 5.5, as the number of dimensions rises, the number of possible ways for data points to differ from one another rises as well. It implies that the distance between the next neighbouring data point would also rise exponentially, making interpolation more difficult.

The scale of the AbVat dataset is quite tiny. The proposed cGAN model was required to solve interpolation over the 868-dimensional latent space using just 1299 data points. As such, it is extremely difficult for the GANs generator to conduct the right mapping between latent space and real game level distribution.

There is currently no consensus about the size of latent space dimensions on the basis of limited size of the training dataset. Some papers only ablate the choice of latent space dimensionality in the presence of a sufficiently large dataset. For example, Brock et al. [2018]

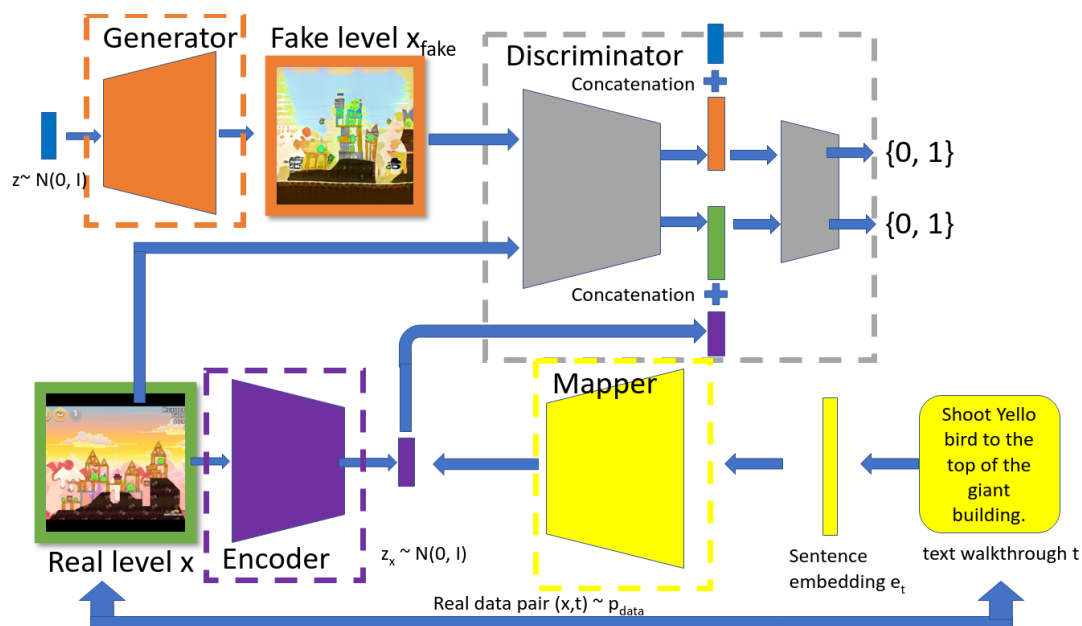


Figure 5.6: Illustration of the architecture for the second preliminary attempt

demonstrated their BigGAN model can be effectively trained with latent dimensions ranging from $z \in \mathbb{R}^8$ to $z \in \mathbb{R}^{128}$.

In conclusion, the high-dimensional latent variable created by concatenating text embedding vectors increases the sparsity of the data, making interpolation more challenging for the GANs model.

5.2 Preliminary Attempt 2: transfer learning and "divide and conquer" strategy

The first preliminary attempt showed that the model cannot function correctly because of discriminator's additional contrastive training and generator's the high dimensional latent space. To solve those issues, we split the task into two sub-problems: firstly, we only sample high quality level images using GANs model; secondly, we map the text walkthrough to the latent space of the generator (See Figure 5.6). This results in a low size of latent space dimensions for the generator, which alleviates the sparsity problem. Transfer learning is also used in this attempt to compensate for the scarcity of training data.

5.2.1 Method

5.2.1.1 "Mapping rather than concatenating" and Bidirectional GANs model

The first attempt suggested that the high-dimensional joint latent representation makes interpolation difficult for cGAN model. Therefore, instead of concatenating two different modalities of data in cGAN, we should "map" the text information to latent space. In order to do that, we divide the task into two sub-tasks.

For this time, the GANs model focuses exclusively on generating data samples using a standard latent variable z . After that, we require an additional text mapper to map the text walkthrough to the associated level data's latent representation.

Since the mapper is trained by supervised regression method, it requires the latent representation of the real level data. However, the latent representation of the real data $p(z|x)$ is not accessible because GANs is an implicit density estimation model (See Section 3.3. Rather of explicitly defining the distribution, a GANs model can only be a decent data sampler.

As a result, an extra encoder must be constructed so that it can project the data sample back onto the latent space. Additionally, this encoder can be trained directly using back-propagation of the loss gradient from the GANs' discriminator. The following are the steps in the training procedure:

1. Normal GANs model part
 - (a) Sample a latent value z from a standard Gaussian distribution $\mathcal{N}(\mu = 0, \sigma = 1)$, it is fed into the generator to produced $G(z) \sim p_G$.
 - (b) At this time, both synthesised data $G(z)$ and the latent representation z are fed into the discriminator ($D(G(z), z)$).
 - (c) The loss information generated from the discriminator will back-propagate and train the generator module.
2. Additional Encoder module
 - (a) The real data sample $x \sim p_{data}$ will be concatenated with its latent representation produced by the extra encoder $E(x) \sim p_E$. After that both of them are fed into the discriminator D .
 - (b) the loss gradient from $D(x, E(x))$ will also back-propagate back to the encoder module. The encoder will then learn to mimic the real latent representation, to the point where the synthesised latent value for real data is indistinguishable from the real latent value from the generator's latent space p_z .

3. Text walkthrough mapper

- (a) After we obtain the projection value $E(x)$ from the real data distribution p_{data} to the latent space p_z , we apply an external text walkthrough mapper to map the associated text information t , where $(x, t) \sim p_{data}$, to the latent representation of the matched game level ($E(x) = M(e_t)$).
- (b) The mapper is trained by performing regression supervised learning method.

Therefore, the game level generation procedure is as follows:

1. Having a walkthrough description $t, t \sim p_{text}$, the associated embedding e_t is fed into the "Mapper" module to produce the corresponding latent representation $M(e_t) \sim p_z$.
2. The latent representation is fed into the generator to produce associated synthesised game level $G(M(e_t)) \sim p_{generator}$.
3. Therefore we obtain a $(t, G(M(e_t)))$ walkthrough and synthesised level pair.

This type of GANs model, which conducts reverse mapping from data sample x to its latent representation z , is referred as Bidirectional GAN (BiGAN). This architecture was first introduced by Donahue et al. [2016].

Since we do not need to concatenate the text-walkthrough embedding to the z , the size of the latent space dimensions of the GANs' generator can be as small as the conventional $z \in \mathbb{R}^{100}$. This way, we reduce sparsity of the training data and therefore aids the generator in doing more accurate interpolation.

5.2.1.2 Why not reduce the dimensionality of the text walkthrough while maintaining the cGAN architecture?

There is an alternative remedy for the cGAN model, which is to shrink the text walkthrough's dimensions. As a result, we will have a shorter joint latent vector for the generator, which will ease the training data manifold's sparsity problem.

However, by reducing the dimensionality of the text embeddings, we may lose information about the text walkthrough. Consequently, it may reduce the quality and diversity of the synthesised level image. This is a trade-off between training stability and the fidelity of the output.

5.2.1.3 Transfer learning

We also attempted to use transfer learning techniques in the second attempt. To boost generalisation performance, we initially trained the GANs model using the COCO dataset [Lin et al., 2014]. However, because the COCO dataset is not relevant to the Angry Birds Game, we continued training the GANs model using frames collected from the AbVat video walkthrough.

We do not reuse the pre-trained discriminator throughout the actual training procedure. This is because a properly trained discriminator will generate insufficient loss information to allow the generator and encoder to advance. To be precise, a well-trained discriminator does not allow the generators and encoders for sufficient exploration of the unknown space, but instead forces them to exploit inside the fixed local domain.

5.2.2 Experiment and result

Most of the experimental settings are the same as the previous attempt, except the followings:

5.2.2.1 Dataset

Again, the training process makes use of the walkthrough descriptions and the game level data pairs from the AbVat dataset. We employed both the COCO dataset and video frames retrieved from the AbVat video walkthrough for the transfer learning procedure. The COCO dataset aims to train the basic visual ability, while the AbVat video frames are used to improve generalisation performance of the model directly in the context of the Angry Birds game.

We extract frames every three seconds from the AbVat dataset's video walkthroughs. As such, we accumulated a total of 24736 Angry Birds gameplay images.

5.2.2.2 Training settings

Similar to the previous attempt, we used Adam optimiser with $\beta_1 = 0$, $\beta_2 = 0.999$. A conventional 0.0004 and 0.00005 learning rate were used for the discriminator and both of generator and encoder. Because of a shortage of data, we raised the number of epochs to 1600 to allow the model to converge. The batch size was set to 32 in order to fully use the GPU's memory capacity. Setting the batch size to the maximum available was recommended by Brock et al. [2018]. The objective function was the AGES-ALL proposed by Shen et al. [2020]. It claimed that AGES-ALL is locally equivalent to simultaneously minimizing all the f -divergence with

strongly convex f with scaling clipping method. This hence prevented from vanishing or exploding gradient problem better than the original f -divergence employed in f -GAN model [Nowozin et al., 2016].

5.2.2.3 Result

The Bidirectional GANs model did not yield high-quality level pictures. The generator was unable to generate meaningful and recognisable building components. The produced images appear fuzzy, despite the fact that they include the right colours from the genuine data samples. Mode collapse occurred as repeated samples were generated by different random latent vectors (See Figure 5.7).

Because of the fact that this is a bidirectional GAN model, we may evaluate the model's capability by "reconstructing" the original data sample. Specifically, the encoder will be fed the original data sample in order to generate the associated latent representation of the real data $z_x = E(x), z_x \sim p_E$. The latent vector is then sent to the generator, which reconstructs the data $x_{recon} = G(E(x))$. From Figure 5.8, we can find that most of the rebuilt image is inconsistent with the original image (i.e. incorrect reconstruction). Another form of failure is mode collapse. Nevertheless, there are some "relatively" good reconstruction samples that attempted to replicate the original data's characteristics.

5.2.3 Discussion

While the general quality of the synthesised data generated by BiGAN in the second attempt is far higher than that produced by the cGAN model in the first attempt, three factors continue to have a significant influence on performance:

1. The pre-trained generator does not effectively take the advantages of the data from the previous settings to extract information that may be useful when learning the current task.
2. The reverse mapping in the BiGAN model results in the explicit density model characteristics, and thereby reducing its generation capacity due to poor design of prior distribution (See Section 3.3).
3. Finally, we cannot ignore the impact of a dearth of training data. Given that the vast majority of accessible large-scale training datasets do not pertain to the Angry Birds domain, we cannot rely on transfer learning to resolve the data shortage issue.

Details will be discussed in what follows.



Figure 5.7: Generated samples from the second attempt using BiGAN

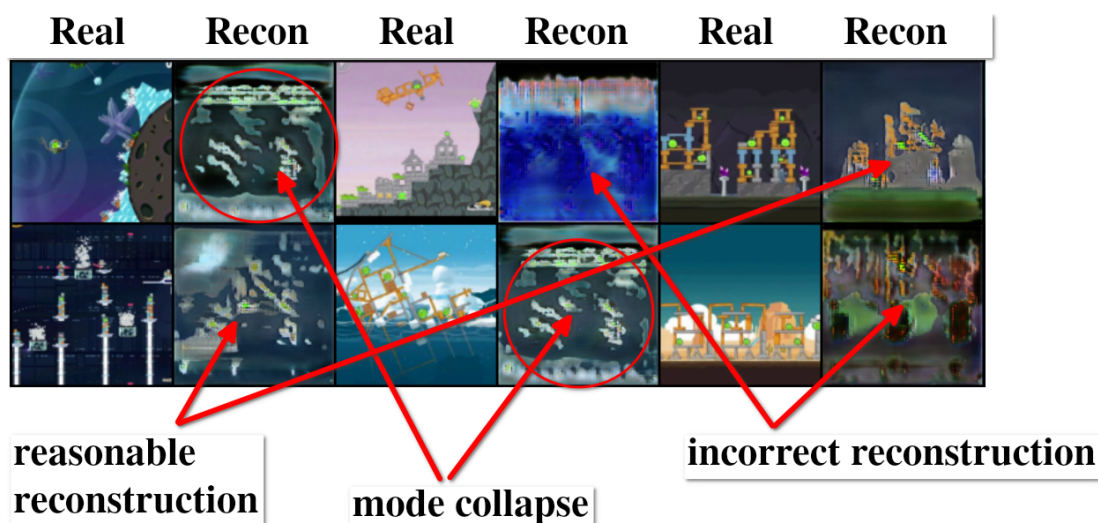


Figure 5.8: Reconstruction samples from the second attempt using BiGAN

5.2.3.1 Ineffective transfer learning

It is shown that COCO dataset and Angry Birds video walkthrough frames did not improve the model's performance sufficiently. The COCO dataset is ineffective since it bears little resemblance to the Angry Birds game domain. However, ideally, the Angry Birds video walkthrough frames should presumably improve the generator's performance, given that these pre-training datasets are in the Angry Birds game domain.

The truth is, the Angry Birds gameplay video dataset is a poor candidate for unsupervised pre-training. This is because the different frames are not sufficiently varied. As shown in Figure 5.9, the successive frames are visually similar to their neighbours, and thus the model cannot extract diverse and informative knowledge from them. In other words, the Angry Birds gameplay video collection is information-poor, which means that even though the size of the dataset can reach 24736, the total knowledge that can be extracted from this dataset is scarce.

The subsequent frames appear identical owing to the fact that the background information occupies the majority of the image area. These shapes and colours in the background will remain consistent throughout the gaming episode. Worse still, this background information does not help to the development of level content because it neither contributes to the gameplay nor interacts with the players.

However, the consecutive frames are in fact distinct in terms of their states if examining closely. As seen by the illustration, the structure of the building progressively collapses



Figure 5.9: Demonstration shows that there is not much visual differences between the successive frames, therefore the unsupervised pre-training video frames dataset is information-poor

throughout the frames. However, because the building structures only comprise a small percentage of the overall screen, it is extremely difficult to extract state information from the model.

One solution is to change the pixel-based level representation to XML-format level representation. By doing this, we can filter out the gameplay-irrelevant information. Another solution is to have additional Area-of-Interest (AoI) labelling so as to guide the model to concentrate on the significant area. However it requires additional labouring force to do the labelling task.

5.2.3.2 Reverse mapping essentially means explicit density model

There were many existing works that tried to explain the relationship between Variational Autoencoder (VAE) and GANs model. Nowozin et al. [2016]; Arjovsky and Bottou [2017] gave theoretical proofs that the objective of the original GAN is equivalent to minimising JS Divergence, which is just a special instance of f -divergence family. Works from Donahue et al. [2016]; Shen et al. [2020] showed that VAE and GAN are special case for BiGAN, whose the objective is also equivalent to minimize a instance (i.e. KL divergence) of f -divergence family. We recall the following equation due to Shen et al. [2020]'s paper that demonstrates

the objective for VAE is a special case for BiGAN's objective:

Given that general objective for BiGAN is

$$L(\theta, \phi) = D_f(p_e(x, z), p_g(x, z))$$

where θ refers to the parameters of the generator

and ϕ refers to the parameters of the encoder

$p_e(x, z)$ is the joint encoder distribution $p_{data}(x)p_e(z|x)$

$p_g(x, z)$ is the joint generator distribution $p_z(z)p_g(x|z)$

D_f function measures the f -divergence

(5.3)

Then, the objective for VAE is

$$\begin{aligned} L_{VAE} &= -\mathbf{E}_{x \sim p_{data}} [\mathbf{E}_{z \sim p_e(z|x)} [\log p_g(x|z)]] - D_{KL}(p_e(z|x), p_z(z)) \\ &= D_{KL}(p_e(x, z), p_g(x, z)) - \mathbf{E}_{p_{data}(x)} [\log p_{data}(x)] \\ &= L(\theta, \phi) \quad \text{where } f\text{-divergence is KL divergence} \end{aligned}$$

More details can be found in their papers, but in summary, Bidirectional GAN can be regarded as a combination model of VAE and GANs. Therefore, BiGAN is a partially explicit density model. The generation performance is worse than the vanilla one-directional GANs model due to inappropriate choice of the prior distribution p_z of the model (Usually for simplicity, the prior distribution is always set to standard Gaussian distribution). The performance further decreases with the lack of training dataset (See Section 3.3 for more details).

5.2.3.3 Lack of training data

Lack of training data is the main issue behind all the phenomenon. All the remedy approaches we came up with so far, from "decreasing latent space dimensionality" to "transfer learning", aim to mitigate the influence of the shortage of training data. All GANs architecture should perform effectively given abundant and diverse training data. However, the total amount of data entries in our model for training level generation is just 1299, which is considered insufficient for deep learning models. Typically, deep learning models are trained using a large diversified dataset, such as ImageNet [Deng et al., 2009], which comprises 15 million high-resolution photos classified into 22,000 groups.

We cannot rely on transfer learning since we lack sufficient in-domain datasets for pre-

training. Rather than that, we will focus on how to improve the training procedure's data efficiency so that the model may be trained successfully even with a limited quantity of data.

5.3 Summary

The first preliminary attempt is the first trial of generating level contents conditioned by text walkthrough using cGAN architecture. However because of the unbalanced training efficiency between the discriminator and the generator, as well as the sparsity issue caused by high dimensional latent space, the first attempt failed to function.

The second preliminary attempt tried to reduce the dimensionality of the latent space of the generator using BiGAN and an additional text mapper. We also tried to apply transfer learning tricks to alleviate the shortage of training dataset. However, because of information-scarcity in the pre-training dataset and the explicit density modelling nature of BiGAN, the second model's performance did not meet expectations either.

There are two possible remedies for the second preliminary attempt. The first is to avoid using the partially explicit density model BiGAN, which may perform worse than the original GAN. The second approach is to develop a method for training a model with great data efficiency, such that our level generation model may continue to perform effectively with just about 1,000 training examples.

This brings us to our final implementation. Chapter 6 will describe the methodology used to address the remaining issues from the second preliminary attempt, as well as the experimental result, which demonstrated a promising performance of level generation conditioned by text walkthrough information.

Design and Implementation

Because of a lack of training data, both models from the early attempts, notably cGAN and BiGAN, failed to work. Nonetheless, we conducted a failure analysis and developed an improved design, which will be presented in this chapter. Keep in mind that our goal is not just to imitate existing levels, but to build novel levels that are not seen in the training dataset, so that this model may be used as a valuable procedural level generator for generating large scale training levels for AI agents' development and testing. Chapter 7 will discuss the experiments and the outcomes in depth.

6.1 Architecture Overview

The proposed architecture is in fact an "incremental" upgrade of the previous designs in the preliminary attempts. According to the failure analysis for the second preliminary attempt, it is better to avoid the use of BiGAN model because of the generation performance drop caused by its explicit density modelling feature. Also we need to increase the data efficiency of the training process. Therefore, we came up with the architecture shown in Figure 6.1.

There are two significant modifications between this design and the previous one (See Figure 5.6). The first is the decoupling of the discriminator and encoder modules so as to avoid explicit modelling of the distribution $p(x, z)$. The second is the addition of a data augmentation module to increase the data training efficiency. The next sections will address various aspects of the design.

6.2 Module 1: level generation by multi-scale learning

According to the earlier research [Zhang et al., 2017b; Karras et al., 2017, 2019, 2020b; Karnewar and Wang, 2020], multi-scale learning can significantly improve the training stability of one-directional GANs models. Multi-scale learning can be regarded as a variant of "divide and

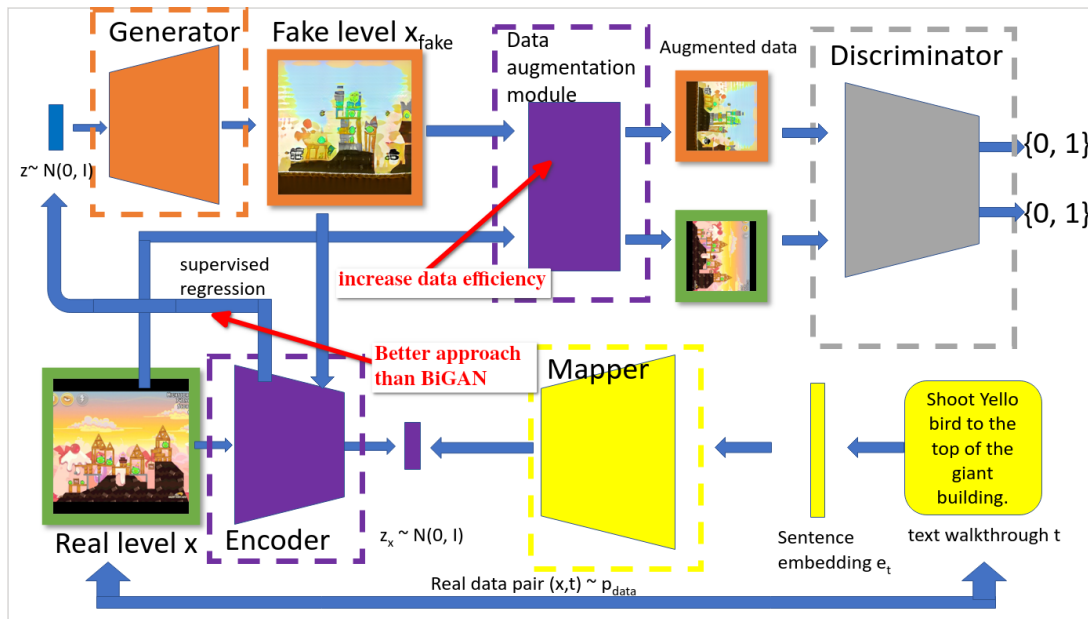


Figure 6.1: Overview architecture of the proposed design for generating Angry Birds game levels from walkthrough description

conquer" strategy, in which the the generator synthesises the data at different scales (e.g. In terms of image data, the generator synthesises the image at different resolutions). Then, the set of synthesised data at different levels is fed separately to the the discriminator. As a result, the produced gradient is able to guide the data generation process at each hierarchical levels.

Multi-Scale Gradient GAN (MSG-GAN) is a GANs model that utilises the concept of "multi-scale learning". As shown in Figure 6.2, it started with a minuscule layer that generates images at very low resolution (e.g. 4×4). The resulting data is subsequently fed to the corresponding layer of the discriminator through a "skip" connection. By doing this, the discriminator needs to distinguish the sample depending on its representation at various scales. As a result, it provides additional control to the generation process, starting from a general view to finer details, thereby stabilising the training process.

In our work, we employed the StyleGAN2 model [Karras et al., 2020b] to generate game levels. StyleGAN2 uses MSG-GAN as the generator's backbone architecture and ResNet [He et al., 2016] as the discriminator's. The residual connection in the ResNet framework also adheres to the idea of "multi-scale learning".

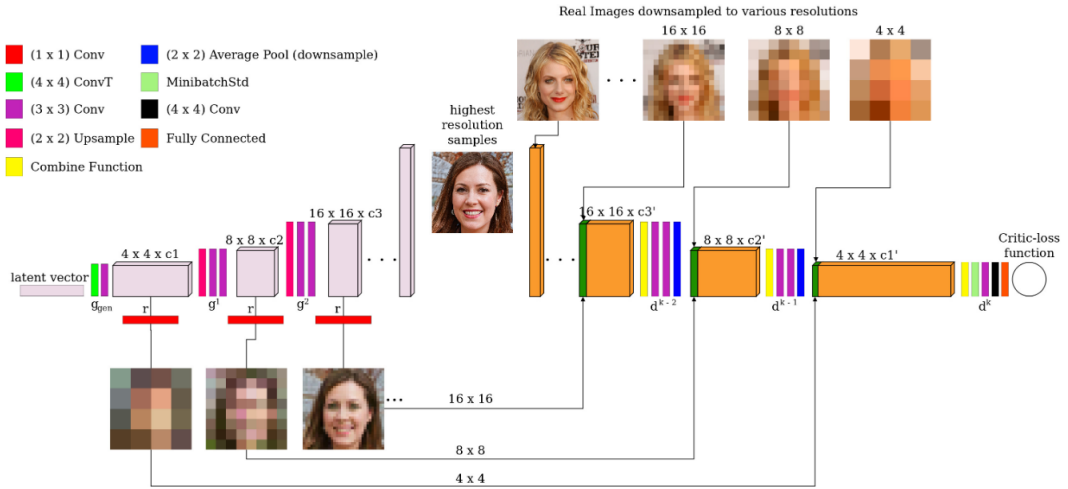


Figure 6.2: Illustration of the Multi-scale learning in MSG-GANs model. The generator produces image at different resolutions and concatenate to the corresponding layer of the discriminator. Thus the discriminator is able to provide additional gradient information to guide the image synthesis at different scales
 Copyright and adapted from MSG-GAN paper [Karnewar and Wang, 2020]

6.3 Module 2: inverse mapping from game level to its latent representation

An encoder that maps the game level images back to their latent value is required in our approach, because the latent values of the real data images are required to train the text mapper (Module 3). Our encoder module is built around the Swin-transformer [Liu et al., 2021] (See Figure 6.3). This is a vision backbone network that processes visual input using the Transformer architecture [Vaswani et al., 2017], which is well-known for its capacity to model long-range relationships in data via the self-attention mechanism. In comparison to earlier research on vision transformers (e.g. [Dosovitskiy et al., 2020]), Swin-transformer significantly reduces the computational complexity from quadratic to linear with respect to the size of input images.

Specifically, Swin-transformer reduces computing costs by dividing the global receptive field into smaller windows, allowing the model to calculate just the local self-attention. As seen in Figure 6.4, the computational cost of global self-attention is $C_{global} = (8 \times 8)^2 P = 4096 \times P$ where P is the cost of one patch performing the attention mechanism with another patch. When the global receptive field is partitioned into four smaller windows, the

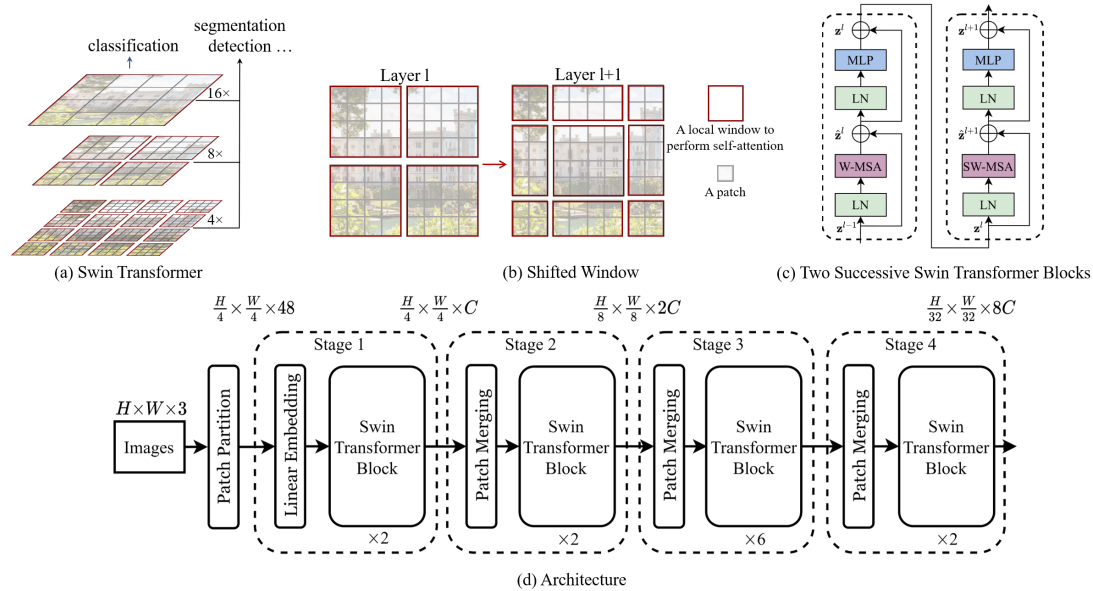


Figure 6.3: Overview of the Swin-transformer architecture.
 Copyright and adapted from Swin-transformer paper [Liu et al., 2021]

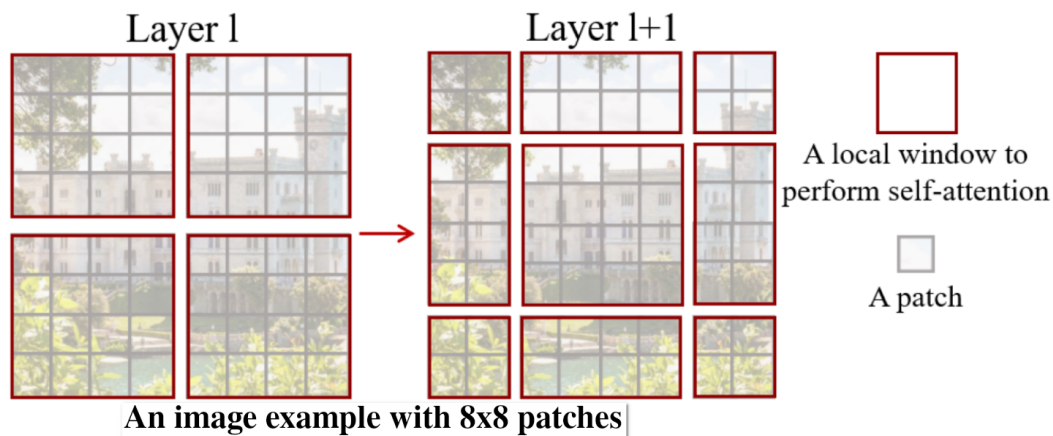


Figure 6.4: An illustration of the window approach of self-attention.
 Copyright and adapted from Swin-transformer paper [Liu et al., 2021]

computational cost drops to $C_{\text{window}} = 4 \times \left(\frac{8}{2} \times \frac{8}{2}\right)^2 P = 1024 \times P$.

However, because separate local windows are isolated, the self-attention module lacks connectivity across the distinct windows areas. As a result of the absence of global dependency modelling, performance may suffer. To resolve this, the author shifted the window partitioning in the following layer so as to link areas that were previously disjoint. Such a shifting behaviour is quite similar to how a convolutional neural network (CNN) kernel convolves around the input volume.

In terms of training, since we changed the level generation model back to one-directional GANs, it is not required to explicitly specify the model distribution $p(x, z) = p(x|z)p(z)$. As a result, we may prevent performance degradation caused by an inappropriate choice of the prior. In the new approach, we applied supervised regression training on the encoder using the synthesised picture samples as training inputs and the related latent values as labels. We employed the traditional L2 loss function for the regression task, which is also known as Mean Square Error loss (MSE). In comparison to other regression loss functions, L2 loss does not accept outliers, which is necessary in our case since an outlier value in the latent space produces significantly different game level images.

6.4 Module 3: mapping from text to latent representation

By using "Module 2", we are able to obtain the latent representation of the real handcrafted game level. For the final step, we need a language model that maps the language text to the latent representation of the associated game level image. By doing so, we are able to complete the whole generation process, as mentioned in the previous section 5.2.1.1.

The BERT [Devlin et al., 2018] model has served as the mainstream backbone architecture for the deep learning language model since 2018. It uses multi-head, multi-layer Transformer blocks. (See Figure 6.5). Besides the use of "Self-attention" mechanism in the modelling process, BERT is most known for two of its unique pre-training strategies, namely "*Masked language modelling*" and "*Next sentence prediction*". The BERT network allows for a large scale unsupervised pre-training process. After doing that, the pre-trained model may be fine-tuned and applied to its downstream task. Our work of mapping text to latent space is an example of the downstream task.

In practise, we used pre-trained DistilBERT [Sanh et al., 2019] from the huggingface library. DistilBERT is a lightweight pre-trained version of BERT that is easily deployable on a personal computer. To achieve the mapping task, the model was fined-tuned by adding an extra fully connected layer at the end of the model with the output dimension equal to

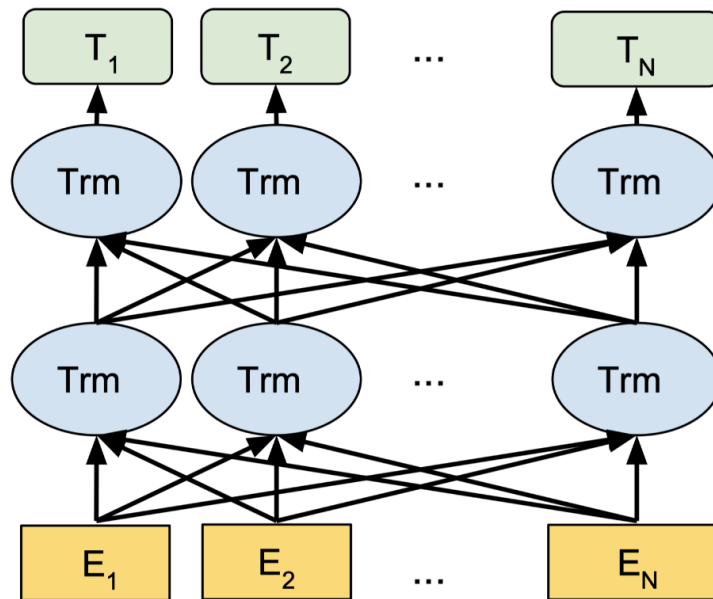


Figure 6.5: BERT model architecture
Copyright and adapted from BERT paper [Devlin et al., 2018]

the latent space dimension (See Figure 6.6). This way, we are able to conduct supervised regression training using the walkthrough sentences as training data and the latent value of the corresponding game level image as the training labels.

6.5 Data augmentation: Image

In general, deep learning models are data hungry. As seen in our earlier attempts, the well-known GAN models cGAN and BiGAN both failed to work due to a lack of training data. As such, data augmentation is introduced so as to enable the training process to be more data efficient, allowing the model to converge even with limited data.

Data augmentation technique has already been widely used in the computer vision domain as a "regularisation" method to avoid the over-fitting problem. The theoretical justification for why it can enhance performance on CV tasks like "classification", "detection" and "segmentation" is as follows:

1. When an image sample is projected into high-dimensional feature space, the data manifold is discontinuous and sparse, making it difficult for the model to map.
2. The data augmentation techniques such as "Flip," "Rotation," and "Resize" do not alter the meaning of the image; rather, they helps the image to span out in the feature space

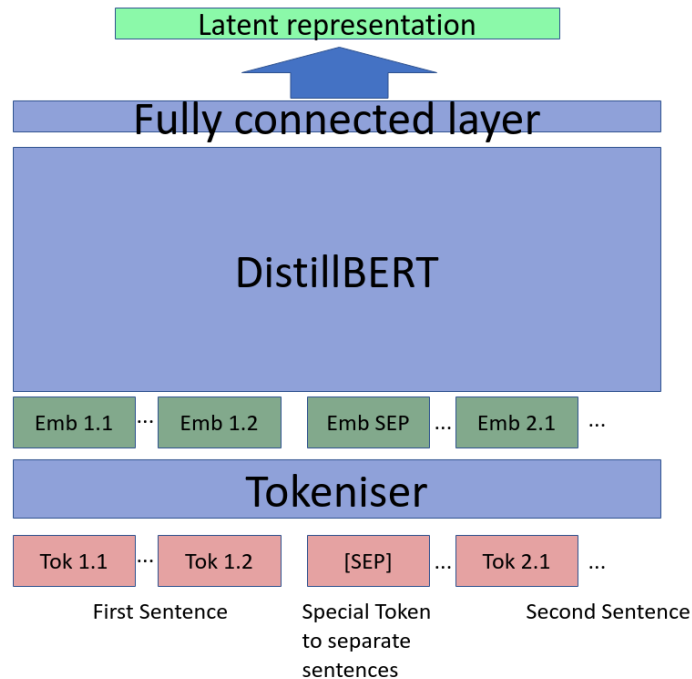


Figure 6.6: DistillBERT with a fully connected layer to perform regression task to get the latent representation of the associated game level image

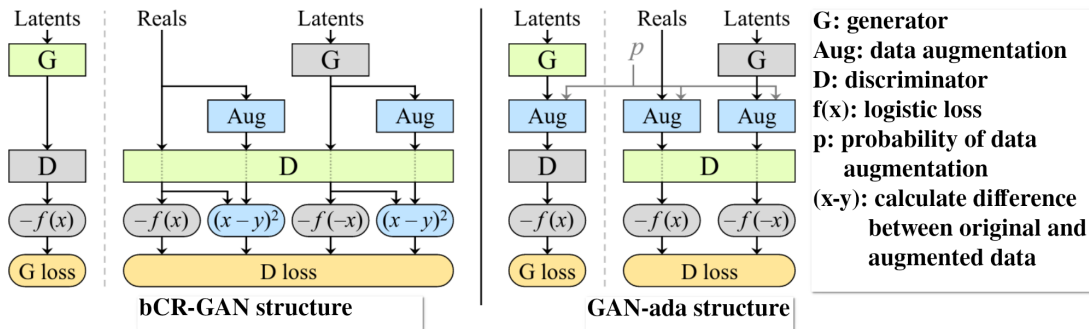
to form a continuous domain, which hence solve the sparsity problem.

3. As indicated by [Zhang et al., 2017a; Yun et al., 2019], "Mixup" and "Cutmix" act as a regularisation method that causes the deep learning networks to favour the "linear behaviour" in the feature space. As a result, the resilience against adversarial examples is increased, and the training process is stabilised.

Work from [Karras et al., 2020a] proved that by having data augmentation layer in the GANs model, the generator can function well even with very limited data. The reason is because data augmentation increases the quantity as well as the diversity of data samples observed by the discriminator. According to the loss function for the GANs model (See Equation 3.1), the generator updates itself by back-propagating the gradient coming from the discriminator. So, by having the data augmentation module, the discriminator is able to offer the generator with more and diverse loss information, allowing it to progress more effectively. The paper also spot the mistake from the previous study on data augmentation for GANs model. The bCR-GAN model from [Zhao et al., 2020] applies the data augmentation technique exclusively during the discriminator training phase. This results in a mismatch between the distribution that the generator wishes to match and the distribution that the

| Data augmentation method | Applicable domain |
|-----------------------------|-----------------------------------------|
| Flip | Classification, Detection, Segmentation |
| Rotation | Classification, Segmentation |
| Resize | Classification, Detection, Segmentation |
| Cropping | Classification, Detection, Segmentation |
| Noise | Classification, Detection, Segmentation |
| Color distortions | Classification, Detection, Segmentation |
| Geometric distortions | Classification, Segmentation |
| Random erase, CutOut | Classification, Detection |
| Hide-and-seek | Classification, Detection |
| Mixup [Zhang et al., 2017a] | Classification, Detection |
| CutMix [Yun et al., 2019] | Classification, Detection |

Table 6.1: Type of data augmentation and its applicable domains

Figure 6.7: Illustration of the data augmentation work in bCR-GAN and GAN-ada
Copyright and adapted from GAN-ada paper [Karras et al., 2020a]

discriminator frequently works at, thereby preventing them from forming stable adversarial training (See Figure 6.7).

6.6 Summary

In this Chapter I describe and justify the designs and methods I used for constructing the proposed level generation model. The designs and methods aim to resolve the issues from previous models in the preliminary attempts. At this point, a promising system that generates Angry Birds game levels by walkthrough description is presented. Chapter 7 will present the experimental settings for the proposed procedural level generation model.

Experimental settings

The next sections detail the experimental conditions and results of the suggested approach. Section 7.1 specifies the programming environment in which the experiment was done. The dataset utilised in the experiment is described in Section 7.2. Section 7.3 describes the baseline models against the proposed model. Section 7.4 discusses the proposed model’s hyperparameter parameters. Finally, Section 7.5 details the evaluation metrics we utilised to assess the method’s performance.

7.1 Programming environment

The experiment was done in the Anaconda Python environment, with Pytorch as the deep learning framework. Deep learning algorithm was accelerated using NVIDIA CUDA and the cuDNN library. The experiment was conducted using an NVIDIA GeForce RTX 3090 GPU with up to 10496 CUDA Cores and up to 24 GB GDDR6X RAM. Table 7.1 and Table 7.2 show the software and hardware specifications.

7.2 Dataset

We used the AbVat dataset, which was discussed in Chapter 4, to train the entire system. It includes 1299 pairs of walkthrough descriptions and level images for the handcrafted Angry Birds game episode. First of all, we trained the system’s level generation module with only the AbVat’s game level images. The training images are downscaled to 256×256 to accelerate and also stabilise the GANs model training process.

The level encoder module receives its training data from the well-trained level generation module. In total, we generated a total 50000 synthesised training data. The training labels are the random latent values generated by numpy random seeds ranging from 0 to 50000, while the training inputs are the associated synthesised image samples from StackGAN2 generator.

| Software environment | Specification |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Operating System | 64-bit Arch Linux Kernel version 5.11.16 |
| Programming Environment | Anaconda python Pytorch framework |
| NVIDIA Config | Driver Version: 465.27 CUDA version: 11.3 |
| Library packages | imageio, imageio-ffmpeg nltk 3.6.2 numpy 1.20.1 pandas 1.2.4 python 3.8.8 pytorch 1.9.0 pytorch-nightly tqdm 4.59.0 torchvision 0.10.0 transformers 4.5.1 |

Table 7.1: Software environment specification description

| Hardware environment | Specification |
|----------------------|-------------------------------------------|
| CPU | AMD Ryzen 5 2600 6-Core Socket AM4 3.4GHz |
| Memory | 16GB (2x8GB) 3200Mhz DDR4 RAM |
| Hard Drive | 512GB M.2 NVMe SSD |
| GPU | Nvidia GeForce RTX 3090 24G |

Table 7.2: Hardware environment specification description

To train the text description mapper module, we utilise the AbVat dataset’s text descriptions as training data and the associated latent representations from the well-trained level encoder module as the training labels. However, this indicates that the language model has only a total of 1299 available training data. As a result, we applied a transfer learning strategy in which we used the off-the-shelf pre-trained weights from the huggingface library as the language model’s starting configuration, alleviating the issue of data scarcity.

COCO and CelebA datasets were also utilised locally to pre-train the weights of the StackGAN2 model in the level generation module, as well as the weights in the Swin-transformer network in the level encoder module. Both datasets are scaled into 256×256 to conform to the AbVat dataset’s property. Although neither of the pre-training datasets is in the realm of the Angry Birds game, they are anticipated to train the model’s basic visual cognition, which is handled by the network’s bottom level layers.

7.3 **Baselines**

To evaluate the relative performance of the proposed framework, one can compare to previous two frameworks stated in the preliminary attempts in Chapter 5. A general comparison table 7.3 between three approaches is shown below.

| Framework | Level Generation model | Reverse mapping level encoder | Text to latent space mapper | Distinctive training strategy | Advantages |
|-----------------------|------------------------|-------------------------------|-----------------------------|-----------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| Preliminary Attempt 1 | conditional GAN | Nil | Nil | Contrastive learning | Easy to implement |
| Preliminary Attempt 2 | BiGAN | ResNet | DistilBERT | Bidirectional training Transfer learning | Reduce sparsity of high dimensional latent data |
| Proposed one | StyleGAN2 | Swin-transformer | DistilBERT | Data augmentation Multi-scale learning Self-attention mechanism | Avoid explicit density modelling and significantly solve sparsity and scarcity of training data. |

Table 7.3: Comparison between different approaches to game level generation using walkthrough description

7.4 Training details

7.4.1 Module 1: Level Generation model

The first module makes use of the StackGAN2 neural network. We input the model a 100-dimensional latent vector z and it produces a 256×256 synthesised game level image. For both the generator and discriminator, we utilise the Adam optimiser with $\beta_1 = 0$ and $\beta_2 = 0.99$. Both initial learning rate is set to 0.0025. We did not specify individual learning rates because Adam is capable of autonomously modifying the learning speed. The probability of style mixing is set to 0.6 (style mixing is a training trick that tries to decouple the feature into singular dimension of the latent space).

The data augmentation layer is activated with a probability of 0.6 and data augmentation options are "pixel blitting," "geometric transformation," and "colour transformation," as indicated in the study [Karras et al., 2020a].

7.4.2 Module 2: Reverse mapping level encoder

The Swin-transformer model used AdamW optimiser, with a base learning rate of 0.00125, a decay rate of 0.1 and a momentum of 0.9, $\beta_1 = 0.9$, $\beta_2 = 0.999$. The total number of training epochs is set at 300, with each epoch containing 50000 number of training data. The drop path rate is set to 0.5 during training, which aids in decoupling the functionality of each path. The loss function for the regression problem is set to MSE loss. The number of layer is set to 4. The default number of window self-attention blocks for each layer is [2, 2, 18, 2], whereas the default number of heads being for each layer is [4, 8, 16, 32].

7.4.3 Module 3: Text to latent space mapper

We used a pre-trained DistilBERT model called "distilbert-based-uncase" from the huggingface library. A fully connected layer was inserted on top of the pre-trained model, mapping the model's default output dimension of 768 to the latent space dimension of 100. AdamW optimizer was adopted, which has a learning rate of 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. MSE was used as loss function.

The BERTTweetTokeniser from huggingface library was used to tokenize the words in the text walkthrough. A specific token "[SEP]" was used to denote the need to separate distinct sentences. The transformer block's maximum input length is set at 128. This means that walkthrough descriptions will be trimmed if they exceeds the word count of 128. Nevertheless, the majority of the text walkthrough shall remain within the range, as seen in Figure 7.1.

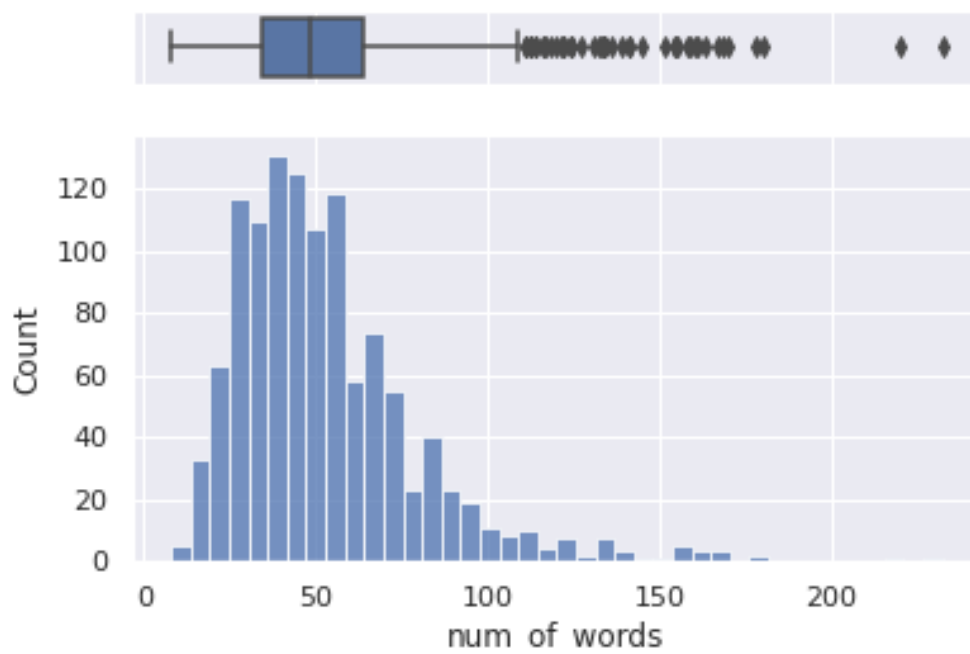


Figure 7.1: Total words distribution of the text walkthrough in the AbVat dataset

7.5 Evaluation Metrics

For Module 2 and Module 3, since they are doing the regression task, it is suitable to report the loss score of the training process to indicate the performance of the modules.

A variant of the Inception Score (IS) metric called "Fréchet Inception Distance" (FID), proposed by [Heusel et al., 2017], is commonly used to measure the performance of GANs model. The IS and FID formulae are shown below.

$$\begin{aligned} IS(G) &= \exp(\mathbf{E}_{x \sim p_g} D_{KL}(p(y|x)||p(y))) \\ FID(G) &= \|\mu_r - \mu_g\|^2 = \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2}) \end{aligned} \quad (7.1)$$

$x \sim p_g$ refers to the synthesised image from the generator. $p(y|x)$ refers to the probability distribution that a Inception V3 model classify the given synthesised image x . $p(y)$ refers to the marginal probability distribution of the classes that a synthesised image can belong to. D_{KL} is the KL divergence, which measures the distance between two distributions. μ and Σ refer to the mean and covariance of the distribution. So, generating images of high quality and variety results in larger $D_{KL}(p(y||)||p(y))$ and hence a higher Inceptions Score.

As we can see, IS does not take the gap between the synthesised and actual data distributions into account. And the performance is largely dependent on the external Inception classifier. To address this issue, FID evaluates synthesised pictures based on how closely the distribution matches that of the real data.

As a matter of fact, qualitative evaluation plays an more important role in this project than quantitative evaluation. This is because critical components of a game's level, like "enjoyment", "entertainment," and "level of physical reasoning required" are difficult to quantify. It is also difficult to quantify how a synthesised game level matches the semantic meaning of the corresponding text walkthrough. This might be done by having an language-based game agent to test if it could solve the generated game level by analysing the walkthrough information. However, such a quantitative measurement is over complicated and outside the scope of this project.

We will examine the output level images qualitatively in this project to see whether the output is properly conditioned by the walkthrough description. Apart from the loss plot, we also inspect the quality of the reconstructed pictures (i.e. $G(E(G(z))) = G(z)$) to determine if the encoder effectively encoded the game level content's characteristics into the latent space.

7.5.1 Evaluation of novel level generation

Most importantly, the following evaluation processes will be used to examine whether the design is capable of generating novel game levels, hence increasing the number of useful game levels for the development and testing of AI agents.

1. By concatenating two walkthrough descriptions, determine whether the synthesised game level possesses both characteristics of the source game levels.
2. By swapping parts of two walkthrough descriptions, determine whether the two synthesised game levels share partial characteristics from each of the source game level.
3. By removing parts of a walkthrough description, determine whether the synthesised game level loses some characteristics from the source game levels.
4. By inputting noise, determine the effect on the synthesised game level.

Results and Discussion

8.1 Quantitative Results

8.1.1 Level Generator's performance

FID score measures the similarity between the synthesised data distribution and the real data distribution. As seen in Figure 8.1, the game level generator's performance continued to improve as it approached a lower FID score. However, at the midway period, the FID score suddenly increased dramatically. This indicates that the generator did not produce data that is close to the actual data distribution. To understand precisely what occurs, we must do a supplemental qualitative investigation.

As seen in Figure 8.3, when the FID score increases abruptly, the generated samples become completely twisted in their colours. Additionally, the majority of the structure appears to be identical. It implies that the generator faced mode collapse, a situation in which the generator gets trapped in local minima as it prefers to generate monotonous but safe images. The GANs model recovered in a relatively short period of time (between timestamp 300 to 350) due to the use of Wasserstein loss metric.

Interestingly, we can see that the generator's initial FID is actually lower than the mode collapse period's score. This is mostly due to the fact that we pre-trained the generator using the CelebA dataset. Although the CelebA dataset is not tied to Angry Birds, it contributes to the network's fundamental visual cognitive capacity and makes training more efficient.

We pause the training of the GANs model after 5 days due to the time constraint. As seen in the figure, the model may continue to improve with more training. Whether the model can get a lower FID score following mode collapse is yet to be proved.

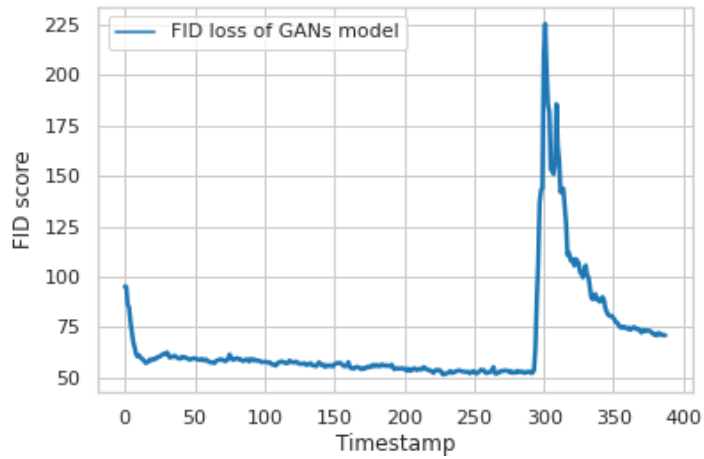


Figure 8.1:

Figure 8.2: The FID score of the GANs model for game level generation; the smaller the value, the better the performance.



Figure 8.3: Generated samples when the FID score rise suddenly. The sample shows a mode collapse happened

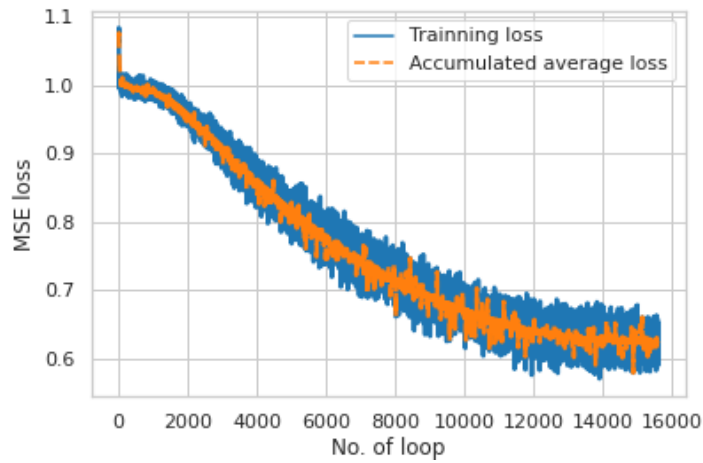


Figure 8.4: Loss plot of Swin-transformer level encoder

8.1.2 Level Encoder's performance

Figure 8.4 illustrates the MSE loss of the training procedure for the Swin-transformer level encoder. Clearly, encoder loss continues to decrease over time. However, the gradient of the loss plot indicates that the level encoder still has a lot of room to improve, but owing to time constraints, we halt the training process after two days.

Notice that the training time for level encoder is set to be shorter than that for the level generator. This is based on the fact that regression model is easier to train than the generative model.

The lowest MSE loss is around 0.6. Eventually, errors from various modules will aggregate and generate an undesired output. Our third model, the text language mapper, will be trained using the output of this level encoder. As a result, even if the text mapper is well-trained, the output may be erroneous owing to the under-trained level encoder.

8.1.3 Text Mapper's performance

As seen in Figure 8.5, the DistilBERT text mapper is well-trained. This is mostly because we only have 1299 training data for the language model. Nonetheless, the pre-trained DistilBERT weights are likely to improve the model's generalisation performance. However, qualitative checks are required to ascertain real performance.

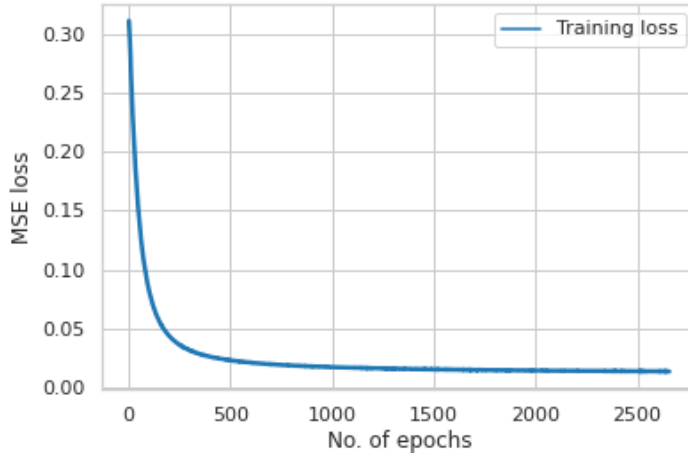


Figure 8.5: Loss plot for DistilBERT text mapper module

8.2 Qualitative Results

8.2.1 Overview of the generator's performance

As seen in Figure 8.7, our proposed generator achieves a large performance improvement. Although the game objects are a little bit skewed and some fine details are lost, they are clearly identifiable in the level image samples. We think that the fidelity of the generated output has met our expectation, as programmers can easily reconstruct the levels in the open-source Science Birds framework using the provided images.

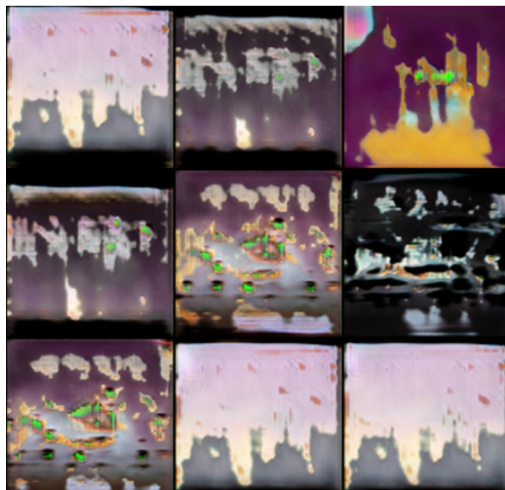
We are able to discover some quite interesting things by browsing through the produced samples. First of all, we can see that the generated level images retain the background information, despite that it does not contribute to the gameplay.

The generator is capable of generating complex but stable structures. As seen in Figure 8.8(a), even though the model contains no additional procedures to verify the structure's stability, the generator automatically learns the "good features" from the handcrafted training data.

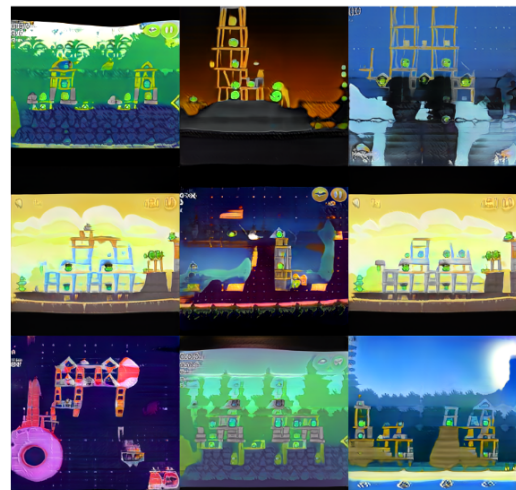
However, there are certain issues with the generated game levels. As seen in Figure 8.8(b), the generated level may be unsolvable because a group of pigs is completely covered with indestructible stones.



Figure 8.6: Samples from real game levels



Generated pixel-based level samples by BiGAN in preliminary attempt 2



Generated pixel-based level samples by the StackGAN2 with data augmentation technique

Figure 8.7: Comparison of the pixel-based game level between the generator from the second preliminary attempt and the proposed generator

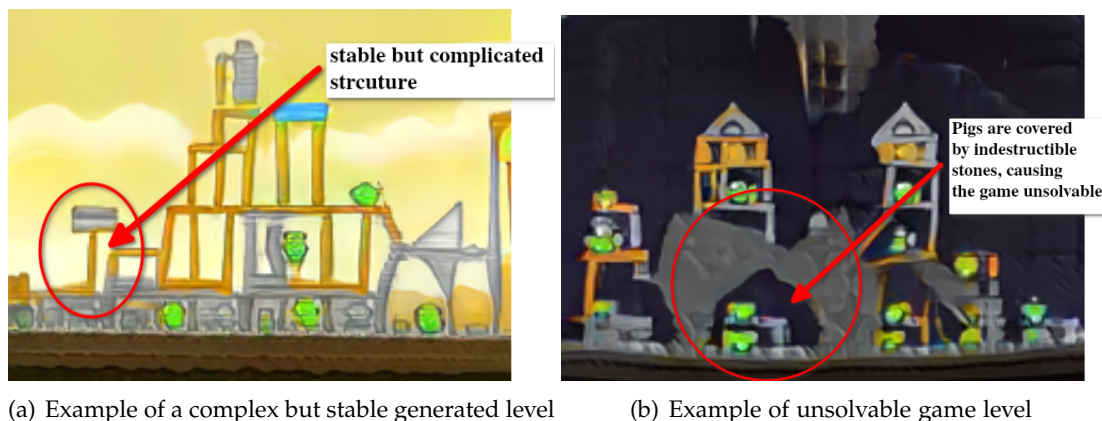


Figure 8.8: An example of typical generated game levels

8.2.2 Reconstruction using Level Encoder

To evaluate the level encoder’s performance, we can do a reconstruction task. This is accomplished by converting level images to latent vectors using the level encoder and then regenerating the image using the generator ($G(z) \approx G(E(G(z)))$). Figure 8.9 demonstrates that the majority of the structure and state of the game objects are kept, however some errors occur throughout the reconstruction process. For example, in the third image pairs, one pig is deleted after reconstruction. The loss plot in the previous section also confirms the existence of the mistakes.

8.2.3 Level generation from walkthrough descriptions

Figure 8.10 illustrates the generation of Angry Birds levels using walkthrough descriptions. The synthesised game level images exhibit similarities to the real levels, hence demonstrating that this model is capable of controlling the content of outputs using walkthrough information. This is a significant improvement over the existing imitation learning models as the previous models are unable to control the output due to implicit data modelling.

The fifth example demonstrated that, though the general feeling between the real and the synthesised levels are similar, the details between them are quite different. For instance, even though both of them utilised square blocks to build the structure, the material varies from stone to wood. This, however, actually further proved that our model is capable of generating novel game levels that are not included in the training dataset, and hence it highlights the capacity of creating a diverse game environment for the AI agents.

We further examine the capacity of producing novel game levels by manipulating the text



Figure 8.9: Reconstruction of the game level, so as to testing the performance of level encoder in the system

| | | | | | |
|--------------------------|----------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Walkthrough descriptions | Simply to fire the Red Bird at the tallest pillar so the level goes down in a domino effect. | Fire the Red bird through the first pig and into the stone tower without touching any wood. The towers will domino and, with some luck, the only remaining pigs should be in the open. Use the remaining birds to clear. | Blast the Yellow bird through all of the stalactites in the middle of the level. To pass through a stalactite, you must pass through the top of it. Fire the first Red bird head-on into the left structure, causing it to topple onto the TNT below. Fire the next Red bird into the top part of the last tower. The pig inside should pop, and the bird should push the large pig off the ledge. | Sling Stella between Foreman Pig's car and the chariot which have the glass horse carrying King Pig and a Small Pig with wings. Activate her power on impact to lift the front of the car and the behind of the chariot up and usually clip the right wing of the Small Pig. | Fire the first Big Brother through the middle of the ice tower, taking out the four pigs on the left side. Launch the next bird into the top hollow stone square, pushing all towers to the right. If necessary, use the last Big Brother to clean up. |
| Real level image | | | | | |
| Synthesised level image | | | | | |

Figure 8.10: Level generation using text walkthrough information

walkthrough. Figure 8.11 shows that, by performing manipulation of the text walkthrough input, the generator is able to produce novel levels that are unseen in the training data. But meanwhile, they are still constrained by the text information.

1. By concatenating two walkthrough descriptions, we can see that the number of game objects in the generated level increases. We believe that this is because a lengthy text tour frequently suggests a sophisticated game level with more game objects in the training dataset. Thus, by increasing the length of text walkthrough, the generator will tend to move to the feature space that has large amount of game objects.
2. By removing parts of a walkthrough description, we see a drop in the number of game objects. Yet, it still retains some features from the original game level.
3. By replacing parts of two walkthrough descriptions, the generated level looks quite different from the original one. This may be because "replacement" is not a linear arithmetic operation in the feature domain.

Surprisingly, when we feed the model irrelevant text, it still constructs an acceptable structure. This, we believe, is because when the text mapper maps unknown or unfamiliar text input to the latent value, it remains within the generator's working domain, and thus, it generates an acceptable level picture rather than random noises.

8.2.4 Evaluation of the level of physical reasoning

Our work also aim to produce meaningful levels that allow agents to learn physical reasoning. This can be evaluated by checking whether the generated levels are consistent to the physical reasoning information provided in the walkthrough description. As shown in Figure 8.12, the model is able to capture the physical information in the walkthrough description to some extent. For example in Figure 8.12(a) and Figure 8.12(b), since the descriptions mention "domino effect", the model tends to generate several thin structures close to each other so as to provide a good condition for such a "domino effect".

8.3 Summary

The experiment results suggest that the proposed generator is capable of controlling the output using text walkthrough. Additionally, the results demonstrated that the model is capable of increasing the "variability" of data as an imitation approach, as we can generate novel levels by manipulating the text inputs via "concatenation," "replacement," and "removal".





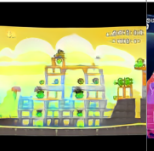

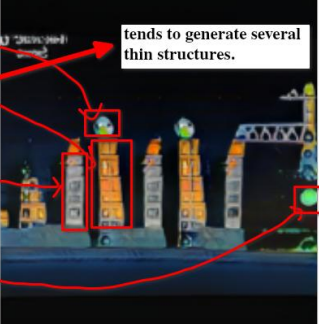

| | | | | | | |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| Walkthrough descriptions | loft a Black Bird just left of the pot o' gold wearing a large leprechaun's hat in the top-left corner of the level [SEP] This should clear everything on the left side [SEP] Loft another one at a similar trajectory to land it just left of the main structure [SEP] Finish off with a White Bird at the extreme right, if needed | Loft Hai over the structure [SEP] spinning him back through the horizontal ice block in the bottom-right. [SEP] With a precise shot, the bird will continue all the way to the well-protected pig within. | loft a Black Bird just left of the pot o' gold wearing a large leprechaun's hat in the top-left corner of the level [SEP] This should clear everything on the left side [SEP] Loft another one at a similar trajectory to land it just left of the main structure [SEP] Finish off with a White Bird at the extreme right, if needed Loft Hai over the structure [SEP] spinning him back through the horizontal ice block in the bottom-right. [SEP] With a precise shot, the bird will continue all the way to the well-protected pig within. | Finish off with a White Bird at the extreme right, if needed | loft a Black Bird just left of the pot o' gold wearing a large leprechaun's hat in the top-left corner of the level [SEP] With a precise shot, the bird will continue all the way to the well-protected pig within | I like to eat banana [SEP] I also like to eat apples [SEP] I do not like to eat oranges [SEP] |
| Generated level image |  |  |  |  |  |  |
| Type of manipulation | Original 1 | Original 2 | Concatenation | Removal | Replacement | Irrelevant |

Figure 8.11: Demonstration of generating novel images by text walkthrough manipulation

| | | | |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| Walkthrough descriptions | Synthesised level image | Walkthrough descriptions | Synthesised level image |
| Fire the Red bird through the first pig and into the stone tower without touching any wood. The towers will domino and, with some luck, the only remaining pigs should be in the open. Use the remaining birds to clear. |  tends to generate several thin structures. | Simply to fire the Red Bird at the tallest pillar so the level goes down in a domino effect. |  it tends to build multiple pillars to achieve domino effect |

(a) Generated level and walkthrough pair 1

(b) Generated level and walkthrough pair 2

Figure 8.12: Samples to examine if the procedural level generator is able to produce useful level in terms of physical reasoning

Also, the result showed that the system was able to capture the abstract concepts to some extent, and thus we saw a promising capability of generating useful levels for agents to learn physical reasoning. However, not all concepts can be captured accurately by the system. I attribute this phenomenon to a extremely lack of training data. A remedy can be asking different volunteers to come up with their own walkthrough descriptions for the same game level. By doing this, the system can form a more general understanding of the feature of the game levels rather than overfitted to one description.

The conclusion is presented in the next chapter.

Conclusion

This thesis explored several methods for generating Angry Birds game levels that are conditioned by walkthrough descriptions. With the use of high performance GPGPU and Pytorch deep learning framework, we came up with a promising Angry Birds game level generator by using Generative Adversarial Networks (GANs), the state-of-the-art generative model, as the backbone architecture. By applying techniques like "Multi-scale training", "Transfer learning" and so on, we effectively stabilise the learning process and alleviate the problem due to data sparsity and diversity.

By providing different walkthrough descriptions, the model tends to generate various game levels that are consistent to the physical reasoning concepts conveyed in the provided texts. This demonstrates a promising capability of providing high quality game levels that can be used for developing and testing physical reasoning agents and algorithms.

Additionally, the thesis work resolves the concerns in regard to imitation-based algorithm. Imitation approach would be considered incapable of control and output and increase the diversity. However, this thesis showed that we can generate conditioned levels by inputting specific walkthrough description. Moreover, it can generate novel game levels, simply by manipulating the walkthrough description inputs using methods like "concatenation", "replacement" and "removal".

Another contribution of this thesis is construction of the Angry Birds game level and walkthrough dataset, AbVat. This dataset not only helps to train procedural level generation model like in this thesis, it can also be used for many other research areas, including natural language processing and reinforcement learning.

9.1 Future Work

This thesis is just a preliminary work to the larger goal, which is to increase the variety and quantity of virtual environments available for the development and testing of physical

reasoning agents. Due to a lack of time and the heavy workload associated with the research subject, several areas remain to be improved.

9.1.1 Improvement of AbVat dataset

The current AbVat dataset comprises just 1299 training samples, which is insufficient for deep learning models that require a lot of data. Thus, the first step is to amass a wider range of handcrafted game levels. There are many Angry Birds game levels accessible; however, I did not collect them in this project due to the absence of a text walkthrough. So, we may collect all accessible data for game levels first, and then seek volunteers to generate the text walkthroughs for the new game levels.

Additionally, an appealing future work would be to change the existing pixel-based level representation to XML format, that would allow the level generation model to be trained directly on the XML format. This way, visual aesthetic and background information that is irrelevant to the gameplay can be filtered out. Furthermore, by utilising an XML-based level representation, we can directly construct the level within the Science Birds framework, allowing for the execution of further experiments.

9.1.2 Level stability testing and training

A fantastic feature of physics-based simulation games is that the objects in them obey to Newtonian physics. As a result, it is critical to maintain the stability of the generated game levels in the Angry Birds game. Unstable generated structures will simply collapse at the start of the game, leaving the game unplayable. Due to the fact that this project's level generation task is purely pixel-based, creating a level stability checker for images is extremely difficult: in order to gather their precise spatial information, we must extract the objects from the noisy background and re-model them, which is already a very complicated task. In contrast, the spatial information about the game objects can be easily collected from a XML format level representation and consequently a direct stability check can be executed.

Additionally, further study should be conducted to avoid unstable structures. Rather than repairing the unstable structure after each generation, we shall investigate neuro- symbolic techniques for constraining the generation process, such as incorporating the stability metric score in the training loss function.

9.1.3 Better quantitative evaluation involving Reinforcement learning agents

Our present work lacks quantitative metrics for determining if the created game level is semantically coherent with the associated text walkthrough. This might be accomplished by training a language-based agent to play the synthesised game level using the associated text walkthrough. This way, we can assess the semantic consistency of the generated game levels via the performance of the language-based agent. Assuming the text walkthrough is precise and instructive, if the agent is able to solve the generated level within fewer trials, it is deemed more semantically consistent, and vice versa.

However, developing a well-trained language agent is a difficult challenge in and of itself, which is why we launched this project: to improve the diversity and quantity of virtual environments in which reinforcement learning agents can advance.

Bibliography

- ABDULLAH, F.; PALIYAWAN, P.; THAWONMAS, R.; HARADA, T.; AND BACHTIAR, F. A., 2019. An angry birds level generator with rube goldberg machine mechanisms. In *2019 IEEE Conference on Games (CoG)*, 1–8. IEEE. (cited on page 6)
- ABRAMSON, J.; AHUJA, A.; BRUSSEE, A.; CARNEVALE, F.; CASSIN, M.; CLARK, S.; DUDZIK, A.; GEORGIEV, P.; GUY, A.; HARLEY, T.; ET AL., 2020. Imitating interactive intelligence. *arXiv preprint arXiv:2012.05672*, (2020). (cited on page 12)
- ALVAREZ, A.; DAHLKOG, S.; FONT, J.; HOLMBERG, J.; NOLASCO, C.; AND ÖSTERMAN, A., 2018. Fostering creativity in the mixed-initiative evolutionary dungeon designer. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*, 1–8.
- ALVAREZ, A.; DAHLKOG, S.; FONT, J.; AND TOGELIUS, J., 2019. Empowering quality diversity in dungeon design with interactive constrained map-elites. In *2019 IEEE Conference on Games (CoG)*, 1–8. IEEE.
- AMATO, A., 2017. Procedural content generation in the game industry. In *Game Dynamics*, 15–25. Springer.
- ARJOVSKY, M. AND BOTTOU, L., 2017. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, (2017). (cited on pages 16 and 45)
- ARJOVSKY, M.; CHINTALA, S.; AND BOTTOU, L., 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*, 214–223. PMLR. (cited on page 16)
- AWISZUS, M.; SCHUBERT, F.; AND ROSENHAHN, B., 2020. Toad-gan: coherent style level generation from a single example. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 16, 10–16.
- BAHDANAU, D.; CHO, K.; AND BENGIO, Y., 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, (2014).
- BROCK, A.; DONAHUE, J.; AND SIMONYAN, K., 2018. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, (2018). (cited on pages 18, 37, and 41)

- CAMBRIA, E. AND WHITE, B., 2014. Jumping nlp curves: A review of natural language processing research. *IEEE Computational intelligence magazine*, 9, 2 (2014), 48–57. (cited on page 11)
- CHENG, V., CHATHURA, 2021. Angry birds agent benchmark. In *2021 IEEE Conference on Games (CoG)*, unknown. IEEE.
- CHUNG, J.; GULCEHRE, C.; CHO, K.; AND BENGIO, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, (2014). (cited on page 12)
- CÔTÉ, M.-A.; KÁDÁR, Á.; YUAN, X.; KYBARTAS, B.; BARNES, T.; FINE, E.; MOORE, J.; HAUSKNECHT, M.; EL ASRI, L.; ADADA, M.; ET AL., 2018. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*, 41–75. Springer. (cited on page 12)
- DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; AND FEI-FEI, L., 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee. (cited on page 46)
- DEVLIN, J.; CHANG, M.-W.; LEE, K.; AND TOUTANOVA, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, (2018). (cited on pages xvi, 12, 53, and 54)
- DONAHUE, J.; KRÄHENBÜHL, P.; AND DARRELL, T., 2016. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, (2016). (cited on pages 3, 17, 40, and 45)
- DOSOVITSKIY, A.; BEYER, L.; KOLESNIKOV, A.; WEISSENBORN, D.; ZHAI, X.; UNTERTHINER, T.; DEHGHANI, M.; MINDERER, M.; HEIGOLD, G.; GELLY, S.; ET AL., 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, (2020). (cited on page 51)
- FIRTH, J. R., 1957. Papers in linguistics, 1934-1951. (1957). (cited on page 12)
- GIACOMELLO, E.; LANZI, P. L.; AND LOIACONO, D., 2018. Doom level generation using generative adversarial networks. In *2018 IEEE Games, Entertainment, Media Conference (GEM)*, 316–323. IEEE.
- GOODFELLOW, I., 2016. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, (2016). (cited on pages xv, 14, and 15)

-
- GOODFELLOW, I. J.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A.; AND BENGIO, Y., 2014. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, (2014). (cited on page 13)
- GUI, J.; SUN, Z.; WEN, Y.; TAO, D.; AND YE, J., 2020. A review on generative adversarial networks: Algorithms, theory, and applications. *arXiv preprint arXiv:2001.06937*, (2020).
- GULRAJANI, I.; AHMED, F.; ARJOVSKY, M.; DUMOULIN, V.; AND COURVILLE, A., 2017. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, (2017). (cited on page 33)
- HA, 2019. Gpt-2 trial conclusion and thoughts. zhihu.com [Online; posted 06-May-2019]. (cited on page 12)
- HARRIS, Z. S., 1954. Distributional structure. *Word*, 10, 2-3 (1954), 146–162. (cited on page 11)
- HE, K.; ZHANG, X.; REN, S.; AND SUN, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778. (cited on page 50)
- HEUSEL, M.; RAMSAUER, H.; UNTERTHINER, T.; NESSLER, B.; AND HOCHREITER, S., 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*, (2017). (cited on pages 33 and 63)
- HILBERT, D., 1935. Über die stetige abbildung einer linie auf ein flächenstück. In *Dritter Band: Analysis: Grundlagen der Mathematik: Physik Verschiedenes*, 1–2. Springer.
- HILL, F.; TIELEMAN, O.; VON GLEHN, T.; WONG, N.; MERZIC, H.; AND CLARK, S., 2020. Grounded language learning fast and slow. *arXiv preprint arXiv:2009.01719*, (2020). (cited on pages 12 and 13)
- HOCHREITER, S. AND SCHMIDHUBER, J., 1997. Long short-term memory. *Neural computation*, 9, 8 (1997), 1735–1780. (cited on page 12)
- HUANG, C.-W.; TOUATI, A.; DINH, L.; DROZDZAL, M.; HAVAEI, M.; CHARLIN, L.; AND COURVILLE, A., 2017. Learnable explicit density for continuous latent space and variational inference. *arXiv preprint arXiv:1710.02248*, (2017). (cited on page 15)
- JAIN, R.; ISAKSEN, A.; HOLMGÅRD, C.; AND TOGELIUS, J., 2016. Autoencoders for level generation, repair, and recognition. In *Proceedings of the ICCV Workshop on Computational Creativity and Games*, 9. (cited on page 6)

- JOHNSON, J.; HARIHARAN, B.; VAN DER MAATEN, L.; FEI-FEI, L.; LAWRENCE ZITNICK, C.; AND GIRSHICK, R., 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2901–2910.
- KAIDAN, M.; HARADA, T.; CHU, C. Y.; AND THAWONMAS, R., 2016. Procedural generation of angry birds levels with adjustable difficulty. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, 1311–1316. IEEE. (cited on page 6)
- KARNEWAR, A. AND WANG, O., 2020. Msg-gan: Multi-scale gradients for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7799–7808. (cited on pages xvi, 49, and 51)
- KARRAS, T.; AILA, T.; LAINE, S.; AND LEHTINEN, J., 2017. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, (2017). (cited on pages 17 and 49)
- KARRAS, T.; AITTALA, M.; HELLSTEN, J.; LAINE, S.; LEHTINEN, J.; AND AILA, T., 2020a. Training generative adversarial networks with limited data. *arXiv preprint arXiv:2006.06676*, (2020). (cited on pages xvi, 55, 56, and 61)
- KARRAS, T.; LAINE, S.; AND AILA, T., 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4401–4410. (cited on page 49)
- KARRAS, T.; LAINE, S.; AITTALA, M.; HELLSTEN, J.; LEHTINEN, J.; AND AILA, T., 2020b. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8110–8119. (cited on pages 49 and 50)
- KINGMA, D. P. AND BA, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, (2014). (cited on page 33)
- KINGMA, D. P. AND WELLING, M., 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, (2013). (cited on page 15)
- KONDA, V. R. AND TSITSIKLIS, J. N., 2000. Actor-critic algorithms. In *Advances in neural information processing systems*, 1008–1014. Citeseer. (cited on page 13)
- LI, L. H.; YATSKAR, M.; YIN, D.; HSIEH, C.-J.; AND CHANG, K.-W., 2019. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, (2019). (cited on page 13)

-
- LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; AND ZITNICK, C. L., 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, 740–755. Springer. (cited on page 41)
- LIU, Z.; LIN, Y.; CAO, Y.; HU, H.; WEI, Y.; ZHANG, Z.; LIN, S.; AND GUO, B., 2021. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, (2021). (cited on pages xvi, 51, and 52)
- LONG, Y.; LIU, L.; SHAO, L.; SHEN, F.; DING, G.; AND HAN, J., 2017. From zero-shot learning to conventional supervised classification: Unseen visual data synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1627–1636. (cited on page 2)
- LOTTER, W.; KREIMAN, G.; AND COX, D., 2015. Unsupervised learning of visual structure using predictive generative networks. *arXiv preprint arXiv:1511.06380*, (2015). (cited on page 15)
- MARCHESE, M., 2017. Megapixel size image creation using generative adversarial networks. *arXiv preprint arXiv:1706.00082*, (2017). (cited on page 18)
- MIKOLOV, T.; CHEN, K.; CORRADO, G.; AND DEAN, J., 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, (2013). (cited on page 11)
- MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G.; AND DEAN, J., 2013b. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, (2013). (cited on page 11)
- MIRZA, M. AND OSINDERO, S., 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, (2014). (cited on pages 3, 16, 29, and 30)
- MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; GRAVES, A.; ANTONOGLU, I.; WIERSTRA, D.; AND RIEDMILLER, M., 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, (2013).
- NOWOZIN, S.; CSEKE, B.; AND TOMIOKA, R., 2016. f-gan: Training generative neural samplers using variational divergence minimization. *arXiv preprint arXiv:1606.00709*, (2016). (cited on pages 42 and 45)
- PASZKE, A.; GROSS, S.; MASSA, F.; LERER, A.; BRADBURY, J.; CHANAN, G.; KILLEEN, T.; LIN, Z.; GIMELSHEIN, N.; ANTIGA, L.; ET AL., 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, (2019). (cited on page 32)

- PRENDINGER, H. AND SCHURZ, G., 1996. Reasoning about action and change. *Journal of logic, language and information*, 5, 2 (1996), 209–245.
- RADFORD, A.; KIM, J. W.; HALLACY, C.; RAMESH, A.; GOH, G.; AGARWAL, S.; SASTRY, G.; ASKELL, A.; MISHKIN, P.; CLARK, J.; ET AL., 2021. Learning transferable visual models from natural language supervision. *Image*, 2 (2021), T2. (cited on page 12)
- RADFORD, A.; METZ, L.; AND CHINTALA, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, (2015). (cited on page 15)
- RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; AND SUTSKEVER, I., 2018. Improving language understanding by generative pre-training. (2018). (cited on page 12)
- RADFORD, A.; WU, J.; CHILD, R.; LUAN, D.; AMODEI, D.; AND SUTSKEVER, I., 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1, 8 (2019), 9. (cited on page 12)
- REIMERS, N. AND GUREVYCH, I., 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, (2019). (cited on page 31)
- RENZ, J. AND GE, X., 2015. Physics simulation games. *Artificial Intelligence Group, Research School of Computer Science. Handbook of Digital Games and Entertainment Technologies, Physics Simulation Games*, (2015). (cited on page 1)
- RENZ, J.; GE, X.; GOULD, S.; AND ZHANG, P., 2015. The angry birds ai competition. *AI Magazine*, 36, 2 (2015), 85–87. (cited on page 2)
- RENZ, J.; GE, X.; VERMA, R.; AND ZHANG, P., 2016. Angry birds as a challenge for artificial intelligence. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30.
- RUMELHART, D. E.; HINTON, G. E.; AND WILLIAMS, R. J., 1985. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science. (cited on page 12)
- SANH, V.; DEBUT, L.; CHAUMOND, J.; AND WOLF, T., 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, (2019). (cited on page 53)
- SENATOR, M. T., 2019. Science of artificial intelligence and learning for open-world novelty (sail-on). *DARPA Executive Conference Center*, (2019).

-
- SHAKER, N.; SMITH, G.; AND YANNAKAKIS, G. N., 2016. Evaluating content generators. In *Procedural Content Generation in Games*, 215–224. Springer. (cited on page 2)
- SHEN, X.; ZHANG, T.; AND CHEN, K., 2020. Bidirectional generative modeling using adversarial gradient estimation. *arXiv preprint arXiv:2002.09161*, (2020). (cited on pages 41 and 45)
- SILVER, D.; HUANG, A.; MADDISON, C. J.; GUEZ, A.; SIFRE, L.; VAN DEN DRIESSCHE, G.; SCHRITTWIESER, J.; ANTONOGLU, I.; PANNEERSHELVAM, V.; LANCTOT, M.; ET AL., 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529, 7587 (2016), 484–489.
- SILVER, D.; HUBERT, T.; SCHRITTWIESER, J.; ANTONOGLU, I.; LAI, M.; GUEZ, A.; LANCTOT, M.; SIFRE, L.; KUMARAN, D.; GRAEPFEL, T.; ET AL., 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, (2017). (cited on page 13)
- SIMONYAN, K. AND ZISSERMAN, A., 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, (2014). (cited on page 15)
- SNODGRASS, S. AND ONTANÓN, S., 2016. Learning to generate video game maps using markov models. *IEEE transactions on computational intelligence and AI in games*, 9, 4 (2016), 410–422. (cited on page 6)
- STEPHENSON, M. AND RENZ, J., 2016a. Procedural generation of complex stable structures for angry birds levels. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8. IEEE. (cited on pages 2, 3, and 5)
- STEPHENSON, M. AND RENZ, J., 2016b. Procedural generation of levels for angry birds style physics games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 12. (cited on pages ix, 2, 3, and 6)
- STEPHENSON, M. AND RENZ, J., 2019. Agent-based adaptive level generation for dynamic difficulty adjustment in angry birds. *arXiv preprint arXiv:1902.02518*, (2019). (cited on pages ix, 2, and 6)
- STEPHENSON, M.; RENZ, J.; AND GE, X., 2020. The computational complexity of angry birds. *Artificial Intelligence*, 280 (2020), 103232. (cited on page 1)
- STEPHENSON, M.; RENZ, J.; GE, X.; FERREIRA, L.; TOGELIUS, J.; AND ZHANG, P., 2018. The 2017 aibirds level generation competition. *IEEE Transactions on Games*, 11, 3 (2018), 275–284. (cited on pages 2 and 5)

- SU, W.; ZHU, X.; CAO, Y.; LI, B.; LU, L.; WEI, F.; AND DAI, J., 2019. Vi-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, (2019). (cited on page 13)
- SUMMERVILLE, A. AND MATEAS, M., 2016. Super mario as a string: Platformer level generation via lstms. *arXiv preprint arXiv:1603.00930*, (2016). (cited on page 6)
- TAO, M.; TANG, H.; WU, S.; SEBE, N.; JING, X.-Y.; WU, F.; AND BAO, B., 2020. Df-gan: Deep fusion generative adversarial networks for text-to-image synthesis. *arXiv preprint arXiv:2008.05865*, (2020). (cited on page 31)
- VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; AND POLOSUKHIN, I., 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*, (2017). (cited on pages 12 and 51)
- VOLZ, V.; SCHRUM, J.; LIU, J.; LUCAS, S. M.; SMITH, A.; AND RISI, S., 2018. Evolving mario levels in the latent space of a deep convolutional generative adversarial network. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 221–228. (cited on page 6)
- WANG, A., 2013. Angry birds project (vision component). (2013). (cited on page 21)
- YI, K.; GAN, C.; LI, Y.; KOHLI, P.; WU, J.; TORRALBA, A.; AND TENENBAUM, J. B., 2019. Clevrer: Collision events for video representation and reasoning. *arXiv preprint arXiv:1910.01442*, (2019).
- YU, Z.; YU, J.; CUI, Y.; TAO, D.; AND TIAN, Q., 2019. Deep modular co-attention networks for visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6281–6290. (cited on page 13)
- YUN, S.; HAN, D.; OH, S. J.; CHUN, S.; CHOE, J.; AND YOO, Y., 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6023–6032. (cited on pages 55 and 56)
- ZHANG, H.; CISSE, M.; DAUPHIN, Y. N.; AND LOPEZ-PAZ, D., 2017a. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, (2017). (cited on pages 55 and 56)
- ZHANG, H.; XU, T.; LI, H.; ZHANG, S.; WANG, X.; HUANG, X.; AND METAXAS, D. N., 2017b. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, 5907–5915. (cited on pages 17 and 49)

ZHAO, Z.; SINGH, S.; LEE, H.; ZHANG, Z.; ODENA, A.; AND ZHANG, H., 2020. Improved consistency regularization for gans. *arXiv preprint arXiv:2002.04724*, (2020). (cited on page 55)

Appendix

9.2 Appendix 1: Generated game level samples



Figure 9.1: Generated level sample 1

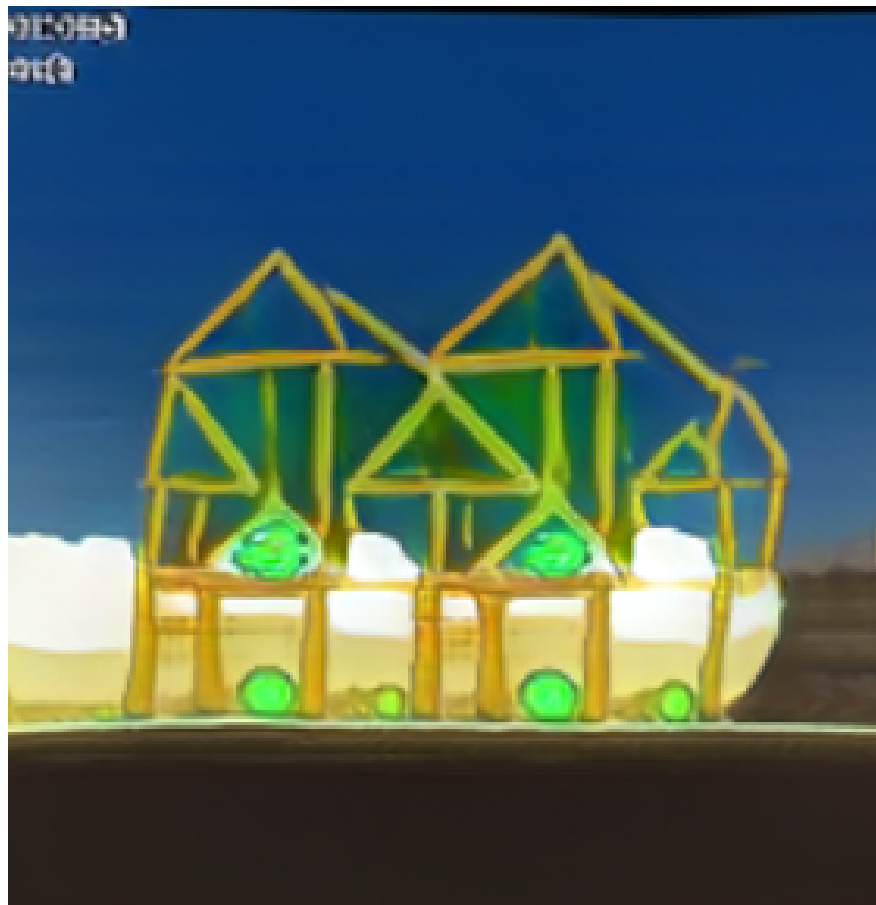


Figure 9.2: Generated level sample 2



Figure 9.3: Generated level sample 3

Figure 9.4: 7×7 generated level samples