GANによるデータ拡張を用いた多様なステージ生成

高田 宗一郎[1,a)]　清 雄一[1]　田原 康之[1]　大須賀 昭彦[1]

概要：ビデオゲームにおけるステージの生成は，制作者の負担の軽減を目的として長年自動生成の研究が行われており，近年では深層学習を用いた手法による自動生成が研究されるようになってきている．深層学習によるステージ生成ではタイルベースのビデオゲームにおいて，GAN による手法が一定の成果をあげているが学習データの用意が課題となっている．本研究では少数のデータのみから GAN を学習し，多様なステージを生成可能なモデルを獲得する手法を検討する．GVGAI Framework の Zelda 環境において，提案する手法により生成したステージに対し定量的な評価を行い従来手法よりもプレイアビリティは低下したものの，多様なステージが生成できることが確認できた．また，提案する手法で学習したモデルでは，進化計算手法による入力変数の探索により従来手法よりも目的をより強く反映したステージ生成が可能であることを確認した．

Various Levels Generation with Data Augmentation Using GAN

Soichiro Takata[1,a)]　Yuichi Sei[1]　Yasuyuki Tahara[1]　Akihiko Ohsuga[1]

Abstract: The generation of levels in video games is intended to reduce the burden on the creator. Recently, deep learning methods have been studied in order to reduce the burden on the creator of the levels. In deep learning-based level generation, GAN-based methods have achieved some success in tile-based video games, but the preparation of training data has been a challenge. However, the preparation of training data has been an issue. In this study, we investigate a method to acquire a model that can generate various levels by learning a GAN from only a small amount of data. It was confirmed that a greater variety and a lower playability of levels can be generated than with conventional methods by quantitative evaluation of the levels generated by the proposed methods on GVGAI Framework's Zelda environment. In addition, the model learned by the proposed method The proposed method can generate levels that reflect the objectives more strongly than the conventional method by using CMA-ES to search for input variables in the model learned with the proposed method.

1. はじめに

Procedural Content Generation (PCG) とよばれるゲームコンテンツの自動作成技術の研究はビデオゲームが誕生して以降さかんに行われている．ゲームコンテンツの中でも，ステージはゲームの難易度や面白さに直結する部分であるが，人手による多くのステージの生成はコストが大きいため制作者の負担の軽減をモチベーションとして自動生成の研究が多く行われている．また近年ではゲーム AI を評価および学習する環境の整備のために PCG が用いられる事例もある [1]．近年では深層生成モデルの発展に伴い，PCG においても深層学習が用いられる事例が増えてきている [2]．タイルベースのビデオゲームにおいては深層生成モデルの 1 つである GAN によるステージ生成が成果をあげているが，GAN の学習に十分なデータの用意にコストがかかることや，生成データの多様性が課題となっている．ゲームステージの生成モデルにおいて，様々なステージが生成できることは，異なるゲーム体験を生むため，あるいは様々なステージからコンセプトに沿ったステージを選べるようになるために重要だと考えられる．AI の評価，学習の環境への適用を考えても，ステージが多様であることは AI の汎化性能の向上あるいは評価のために必要であ

[1] 電気通信大学 大学院情報理工学研究科
Graduate School of Informatics, The University of Electro-Communications

[a)] takata.soichiro@ohsuga.lab.uec.ac.jp

ると考えられる．

本研究では，少量のデータセットのみから多様なステージ生成が可能な GAN の学習手法について検討する．出力が多様な Generator を学習することで，様々なステージが生成できるようになるだけでなく，多様な生成が可能な Generator に対し先行研究 [3] のように入力の潜在変数を探索することで制作者の意図を反映したステージ生成ができるようになると考えられる．提案手法として GAN の生成するステージのうち制約を満たすステージを学習データとして加えるデータ拡張および学習時の正則化手法を提案し，実験と評価を行った．その結果，提案する手法により学習したモデルは，単純な GAN により学習したモデルや従来手法により学習したモデルと比較してプレイアビリティは低下したものの，多様なステージが生成できることが定量的に確認できた．また，多様なステージを生成することができるようになったことで，潜在空間を探索することによりプレイ可能かつある程度目的関数を反映したステージを生成することができることを確認した．

2. 研究背景

2.1 Procedural Content Generation(PCG)

ビデオゲームのコンテンツの自動生成は，Procedural Content Generation と呼ばれ長年研究が行われており，様々なゲームコンテンツがアルゴリズムによって生成されてきた．PCG の分野の中で，ゲームステージの生成は最もポピュラーなものであり，特に 2D のアクションゲームである Super Mario Bros. のような横スクロールのアクションゲームやローグライクゲームと呼ばれる探索型のアクションゲームなどが多く研究の対象として用いられてきた [4]．近年では画像生成や文章生成等で深層学習が著しい成果をあげていることから，PCG の分野においてもその有用性が期待され，ゲームコンテンツの生成に深層生成モデルや深層強化学習を用いる研究が盛んに行われるようになってきている [2], [5], [6], [7].

2.2 Generative Adversarial Networks(GAN)

深層学習による生成モデルの 1 つに，敵対的生成ネットワーク（GAN）がある．GAN は Goodfellow ら [8] が 2014 年に提案して以降，主に画像の生成モデルとして多くの研究が行われており，画像生成においては著しい成果をあげている．最も基本的な GAN では，データの生成を行うニューラルネットワークである Generator に加え，Generator により生成されたデータと実データを判別するネットワークである Discriminator を交互に学習を行うことで，Generator の生成データの分布を実データ分布に近づける．具体的には，式 (1) の形で定式化された価値関数 $V(D,G)$ によるミニマックスゲームを通して Generator G が学習される．

$$\min_D \max_G V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log D(G(z))] \quad (1)$$

2.3 GAN による PCG

近年では GAN を用いたステージ生成の研究が成果をあげている．Volz ら [3] は DCGAN[9] を用いて Super Mario Bros. のステージ生成を行い，GAN の潜在変数を進化計算によるブラックボックス最適化手法である Covariance Matrix Adaption Evolution Strategy(CMA-ES)[10] により最適化することで指定した目的を満たすステージを生成する手法を提案した．Torrado ら [11] は，Generator と Discriminator に畳み込みによるモデルではなく，Self-Attention[12] をベースとしたモデルを用いて GVGAI Framework[13] の Zelda 環境でステージ生成を行っている．Self-Attention Map にステージ上の各タイルの数の情報を結合して条件つき生成を行うことで，従来の畳み込みをベースとしたモデルによる生成よりもプレイアビリティの高いステージ生成が可能となり，重複率も低くなることを示した．また，生成データを学習途中で GAN の正解データとして組み込むことで，5 つというごく少量のデータからの学習でも 47%のプレイアビリティと 60.3%の重複率を達成したことが報告されている．その他，様々なタイルベースのビデオゲームにおいて GAN によるステージ生成手法の研究が行われている [14], [15], [16], [17], [18].

3. 提案手法

本研究では，タイルベースのビデオゲームにおいて少数の学習データのみから GAN により多様なステージを生成することを目指す．提案手法として Torrado らの研究 [11] で用いられたような GAN の生成データによる学習データの拡張とモデル学習時の損失関数に着目した手法を提案する．

3.1 データの拡張

本研究では Torrado らの Bootstrap 手法のように GAN が学習時に生成するデータを学習データセットに追加することで学習データの多様性を増し，Generator がより多様なステージを生成するようになることを目指す．本研究では，データの拡張手法として以下の 2 つの手法を提案する．

3.1.1 手法 1: 改良した Bootstrap 手法

この手法の概略図を図 1 に示す．

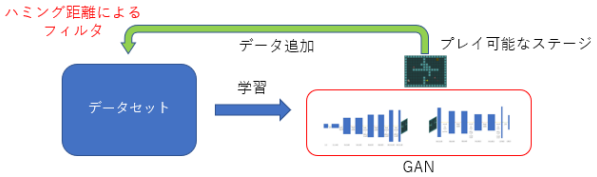

図 1 データ拡張手法 1 の概略図

Torrado らの研究における Bootstrap 手法は，GAN の学習途中に Generator の生成データのうちプレイ可能なものをデータセットに追加することで学習データの数を増やす手法であった．この手法では、追加するデータとデータセット内のデータとの比較を行っていないため、データセット内のステージが似たステージばかりになってしまう可能性がある．このようにデータセット内のデータが偏ってしまう問題を防ぐため，本手法ではデータの追加時にデータセット内のすべてのステージと追加するデータを比較し，ハミング距離（2 つのステージを比較したとき，同じ位置で異なる種類のタイルが配置されている数）が一定以上のもののみを追加するようにする．これにより，データの追加時にはより新規性の高いステージが追加されると考えられ，データの偏りの問題に対処できると考えられる．別のより良い基準を用いることで，よりデータの偏りが生じにくくなると考えられるが，それは今後の課題である．本研究ではゲームの種類によらず適用可能である単純なハミング距離による基準を用いた．

3.1.2 手法 2: データ拡張と学習を分離した手法

この手法の概略図を図 2 に示す．

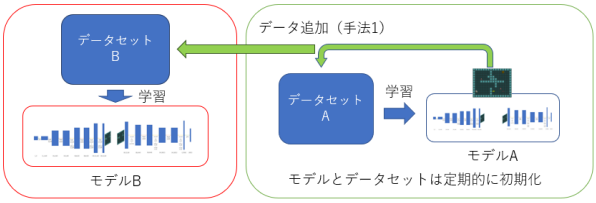

図 2 データ拡張手法 2 の概略図

Bootstrap 手法では，手法 1 のような工夫をしたとしても Generator が生成するデータ次第ではデータセット内のデータの分布が次第に偏り，生成するデータも偏ってしまうことが考えられる．そこで，Park らの研究 [15] のように学習データの拡張と GAN の学習部分を分離することで，より偏りの少ないデータセットを作成し学習する手法を提案する．

[15] ではデータ拡張のために用いる GAN を元データから学習し，その GAN により生成したプレイ可能なステージを用いて新しい GAN モデルの学習を行うという手法がとられている．本手法でも同様にデータ拡張するモデルと拡張したデータから学習するモデルの 2 つの GAN モデルを用意して学習を行いつつ，生成データの多様性を向上させるためのいくつかの工夫を加える．

本手法ではデータ拡張を行う GAN モデル A と拡張したデータにより学習を行う GAN モデル B の 2 種類のモデルを用意する．モデル A はデータセット A を用いて手法 1 によりデータ拡張を行いつつ学習し，モデル A が Bootstrap するデータをデータセット B にも追加していく．その後，データセット B のサイズが十分大きくなったところで，データセット B からモデル B の学習を行う．

モデル A の学習が進行すると，データセット A に偏りが生じてしまう可能性がある．そこで，定期的に学習したモデルの重みの初期化，データセット A の初期化を行う．

この手法では，データセットの準備に時間がかかってしまうものの，定期的にモデルの重みとデータセットを初期化しながらデータ拡張を行うため，元データ内の一部のデータへの偏りが起きにくいと考えられる．

3.2 GAN の学習

非常に少数の学習データによる GAN の学習では，Generator が学習データ内の特定のデータに類似したデータのみを生成するようになってしまう問題がある．この問題に対処し，より多様なステージを生成させるため，学習時の損失関数に，ミニバッチ内の生成データ間の L1 距離を最大化する正則化項を追加して学習を行う．GAN の損失関数として敵対的損失である Hinge Adversarial Loss[19] と合わせて，全体として以下の式 (2),(3) を用いる．ここで N はミニバッチのサイズである．

$$L_D = -\mathbb{E}_{x\sim p_{data}}[\min(0, -1 + D(x))] - \mathbb{E}_{z\sim p_z}[\min(0, -1 - D(G(z)))] \quad (2)$$

$$L_G = -\mathbb{E}_{z\sim p(z)}[D(G(z))] - \lambda_{div}\sum_{i=0}^{N-1}|G(z_i) - G(z_{i+1})| \quad (3)$$

4. 実験と結果

2D タイルベースのビデオゲームを対象として，提案する手法により少数データから GAN を学習しステージ生成の能力やその多様性について定量的および視覚的に評価，考察を行った．また提案手法により得られたモデルに対し CMA-ES による潜在変数の最適化を行い，目的に沿ったステージの生成が可能かを検証した．

4.1 実験設定

4.1.1 対象とするゲーム

本研究では，ステージ生成を行う環境として GVGAI Framework の Zelda 環境を用いた．この環境ではプレイヤーは高さ 12, 幅 16 のグリッド状のステージ上を動き，鍵を取得してゴールのマスに到達することでクリアとなる．ステージ上には敵が存在することがあり，避けるか攻撃によって倒さなければならない．ステージは壁のマス，床のマス，敵のマスなど全 8 種のマスから構成されているが，仕様上プレイ可能なステージであるためには以下のような制約が守られる必要がある．本研究では GAN の学習データとして，人手により作成された 5 種類の Zelda 環境のステージを用いた (図 3).

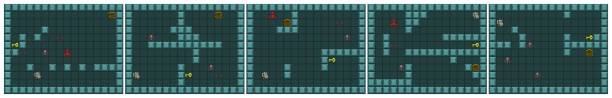

図 3 GAN の学習データとして用いるステージ

- プレイヤー，鍵，ゴールマスはただ一つ存在する．
- プレイヤーマスから鍵マスおよびゴールマスに到達可能である．
- ステージの上下左右は壁マスに囲まれている．

4.1.2 モデルアーキテクチャ

ステージは全 8 種類のマスをチャンネル方向に割り振り，サイズ (8,12,16) の 3 次元テンソルとして表現する．Generator の出力をステージに変換する際は，チャンネル方向に Softmax 関数を適用し各マスごとに最も値の大きいチャンネルに対応するタイルを選択する．

GAN のネットワークは Generator,Discriminator ともに 3 層の畳み込みによるモデルをベースに，先行研究 [11], [20] で有効性が示されている Self-Attention 層を畳み込み層の後に導入したモデルを用いた．詳細なモデル構造は付録の表 A·1,A·2 に示す．

4.1.3 実験条件

GAN の学習時の各パラメータの設定を表 1 に示す．なお，Bootstrap 手法はミニバッチサイズを 32 とする関係上，$5 \times 7 = 35$ 個の初期データセットから学習をはじめ，10Epoch ごとに制約を満たすデータをデータセットに追加するという実装を行った．

表 1 パラメータ設定

設定項目	
潜在変数の次元	32
バッチサイズ	32
最適化手法	RMSProp
Generator 学習率	0.00005
Discriminator 学習率	0.00005
λ_{div} の値	50.0
学習ステップ数	10000
初期データセットサイズ	35
生成データの追加間隔	1/10Epoch
提案手法におけるデータ追加の ハミング距離の閾値 (%)	10
手法 2 で拡張したデータ数	200

4.2 プレイアビリティおよび多様性の評価

4.2.1 評価指標

学習したモデルが生成するステージの定量的な評価を行うため，以下のような指標を用いて評価を行う．

プレイアビリティ 生成したステージが制約を満たしプレイ可能であるかどうか．実装上はプログラムを用いてすべての制約を満たすかどうかのチェックを行った．

平均ハミング距離 生成したステージ群のすべてのペアについてのハミング距離を平均したもの．

重複率 一定数生成したステージのうちユニークでないものがどれだけ存在するか．

各マス数の平均，分散 生成したステージ群のすべてのステージの各マップタイルの数の平均と分散．

重要マスの一致率 生成したステージ群のすべてのペアについて，Zelda における鍵，扉，プレイヤーといったただ 1 つ存在するマスの位置が一致しているペアの割合．

ステージ生成モデルは実際にプレイ可能なステージを生成しなければならないため，プレイアビリティの評価は必要である．また，生成したステージの多様性を定量的に測るため本研究では Generator が複数のステージを生成した際により重複せずに異なるステージを生成できるほど，また生成したステージ間のハミング距離がより大きいほど多様なステージ生成ができていると考え，重複率，平均ハミング距離，各マス数の分散といった指標を定量的な評価に用いた．加えて，Zelda においては鍵，ゴール，プレイヤーといったマスは各 1 つしか存在せず，このゲームにおいて重要なオブジェクトであると考え，ゲーム体験として異なるステージを生成できているかを評価するためにこれらのオブジェクトの位置の一致率を評価指標として用いた．プレイアビリティは 10000 ステージ生成したうちのプレイ可能ステージの割合を算出し，重複率はプレイ可能なステージを 10000 ステージ生成し重複しているステージの割合を算出した．ハミング距離と各マス数の平均，分散はプレイ可能な 1000 ステージ間で算出した．

4.2.2 結果と考察

学習した各モデルにおいて評価を行った結果を表 2 に示す．また，各手法により生成したステージの例を図 4，5，A·1，A·2 に示す．比較手法として，[11] における Bootstrap のみを用いた手法と，ベースラインとして単純に 5 つのステージデータから GAN の学習を行った手法とを比較している．なお比較手法である従来手法 [11] および提案手法における Bootstrap は，学習 10Epoch ごとに制約を満たすステージをデータセットに 1 つ追加するという実装を行っている．

提案手法のデータ拡張手法の手法 1 と手法 2 について比較すると，手法 1 は生成データ間のハミング距離が大きくなっている一方で，生成データのタイルの数の平均値が元データと大きく離れてしまっている．これは，生成データを学習データとして追加しながら学習するにつれ，データセット内のデータの分布が徐々に偏ってしまったためと考えられる．手法 2 ではこの問題は改善され，データの偏りは起こりにくくなっている一方でプレイアビリティは低下してしまっている．

表 2 定量評価の結果

	プレイアビリティ (%)↑	平均ハミング距離 ↑	重複率 (%)↓	重要マスの一致率 (%)↓	壁マス数の平均/分散	床マス数の平均/分散	敵マス数の平均/分散
提案手法 (手法 1)	47.3	**43.7**	**0.0**	22.1	86.0/41.0	95.7/63.7	7.3/11.3
提案手法 (手法 2)	10.2	34.2	**0.0**	**10.2**	74.0/42.0	113.3/35.9	1.6/2.7
従来手法 [11]	**98.4**	8.8	45.5	100.0	68.9/7.1	117.2/7.6	2.9/1.2
ベースライン	94.2	17.9	97.7	49.2	66.0/0.38	120.1/0.62	3.0/0.062
元データ	100.0	34.8	0.0	0.0	71.4/67.8	114.4/65.5	3.2/0.15

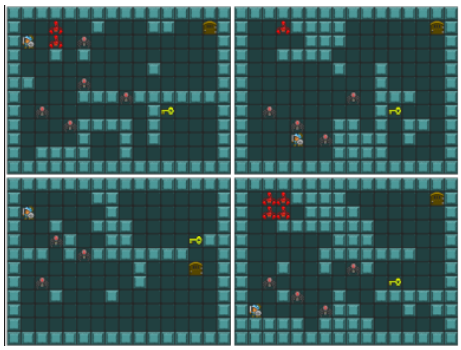

図 4 提案手法（手法 1）により生成されたステージの例

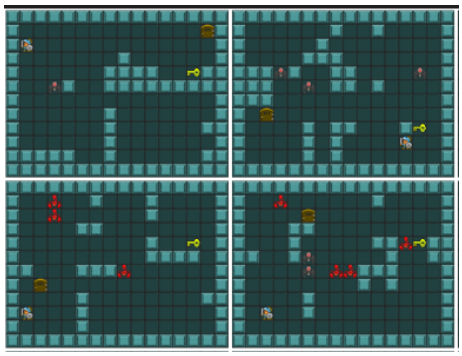

図 5 提案手法（手法 2）により生成されたステージの例

提案手法と従来手法を比較すると，多様性を示す指標である平均ハミング距離，重複率，重要マスの一致率，各マス数の分散といった値において 2 つの提案手法がベースライン手法や従来手法である Bootstrap のみの手法を上回っており，提案手法により多様なステージを生成することができたといえる．単純な Bootstrap を行う従来手法は，今回の実験に用いたモデルでは学習初期に特定のデータに近いデータばかり生成してしまうため，Bootstrap により類似したデータのみが学習データとして追加され続けることで重要マスの一致率が 100%となってしまい，類似したステージしか生成できなかった．5 つのステージから単純に GAN の学習を行うベースライン手法では生成データの偏りが生じ，学習データセットとほぼ同じステージしか生成できなかった．

提案手法は，多様なステージを生成できた一方でプレイアビリティについては大きく低下している．ステージとして正しく生成できないほとんどのケースは鍵，扉，プレイヤーマスの個数制約を満たしていないケースであり，これらのマスの生成位置がほぼ固定化されているベースライン手法や Bootstrap のみの手法ではプレイアビリティが高くなり，逆にこれらの位置が多様化している提案手法ではプレイアビリティが低下している．プレイアビリティの低さに関しては，ある程度の値があれば生成数を増やすことや後述する入力の潜在変数を探索することにより対応可能ではあるものの，CESAGAN のようにモデルに工夫を加えることで改善できる可能性があり，今後の課題である．

また，鍵，扉，プレイヤーマスは提案手法でもその位置が固定化されがちであり，これらのマスの位置に関してはあまり多様化ができていない．これは学習データが少なすぎることに加え，提案手法の工夫を用いたとしてもデータ拡張時にデータに偏りが生じてしまうためであると考えられ，今後アルゴリズムやモデルを工夫することで改善していきたい．

4.3 潜在変数の探索による入力の最適化

提案手法により多様な出力が可能な Generator を獲得できたことで，潜在変数の探索により目的に沿ったステージを生成する手法を有効に活用できると考えられる．Volz らの研究 [3] では，入力の潜在変数を目的関数に沿って CMA-ES により最適化することで制作者の意図をある程度反映させたステージ生成が可能であることが示されている．CMA-ES は進化計算によるブラックボックス最適化アルゴリズムであり，多次元ベクトルに対し目的関数値を最小化する解を進化計算により探索する．Generator の入力の潜在変数を CMA-ES により最適化することで，目的関数に沿ったステージを生成する潜在変数を発見することができる．

ここでは以下の 2 つの目的関数 F_1, F_2 について潜在変数の最適化を行い，どの程度目的関数を反映した生成が可能かを検証する．ここで，P はステージがプレイ可能であるとき 1, そうでないとき 0 の値をとる変数である．

$$F_1 = 100P - \#walls - 5 \times \#enemies \tag{4}$$

$$F_2 = 100P + \#walls + \#enemies \tag{5}$$

F_1 はプレイ可能なステージを生成しつつ，壁の数と敵の数を最大化すること，F_2 はプレイ可能なステージを生成しつつ，壁の数と敵の数を最小化することを目的として設計した目的関数である．

表 3 CMA-ES により潜在変数の最適化を行った後に得られたステージのプレイアビリティおよび目的関数値．目的関数の平均値はプレイ可能なステージを生成したときの値のみ平均している．

	F_1		F_2	
	プレイアビリティ (%)↑	平均値 ↓	プレイアビリティ (%)↑	平均値 ↓
提案手法 (手法 2)	100	-113.5	98	56.4
従来手法 [11]	100	-102.7	100	64.0

実装上は python の CMA-ES のライブラリである pycma[*1]を用いて，初期集団数 14，標準偏差の値は 0.5 に初期化し，150 イテレーション固定で最適化を行った．データ拡張手法 2 を用いた提案手法により獲得されたモデルについて適当にサンプリングした z を初期値として F_1, F_2 の目的関数により最適化を行った結果の例を図 6,7 に示す．

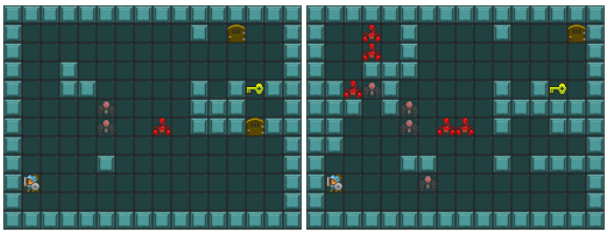

図 6 提案手法 (手法 2) で学習したモデルにおける F_1 による最適化の結果．左側が初期値より生成したステージ，右側が発見した最適解より生成したステージ．

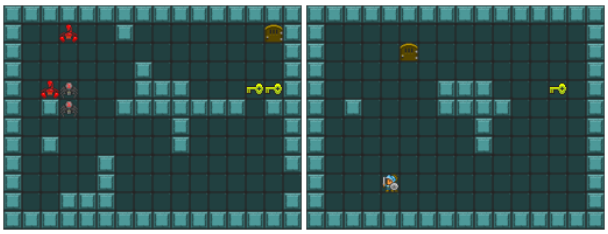

図 7 提案手法 (手法 2) で学習したモデルにおける F_2 による最適化の結果．左側が初期値より生成したステージ，右側が発見した最適解より生成したステージ．

F_1, F_2 どちらで最適化を行った場合も，敵や壁マスの数を狙い通りに変化させることができており，ある程度目的に沿ってステージを生成することができていることがわかる．

また，提案手法と従来手法について，50 個の異なる初期値からそれぞれ最適化を行った結果のステージのプレイアビリティとその平均の目的関数値を比較し，提案手法が従来手法と比べどれほど目的に沿った編集ができるようになっているか検証した．表 3 より，提案手法も従来手法も，潜在変数の最適化によりほとんどの場合でプレイ可能なステージを獲得することができており、提案手法の方がより強く目的関数値の最小化に成功している．この結果から，提案手法はより多様な出力が可能になったことにより，目的に沿ったステージを生成する能力が高くなっているといえる．

[*1] https://github.com/CMA-ES/pycma

5. おわりに

本研究では，GVGAI Framework の Zelda 環境において，生成データによる学習データの拡張とモデル学習時の正則化により GAN によって多様なステージが生成でき，また CMA-ES による最適化によりある程度目的に沿ったステージ生成が可能であることを確認した．Zelda は比較的小さいステージかつあまり複雑なステージ制約がなく，提案手法によりある程度多様なステージを生成できたものの，この程度のゲームであれば GAN を用いずともランダムに壁や敵マスなどを配置することで十分に多様なステージを生成可能である．提案手法はタイルベースのゲームであれば他のゲームにも適用可能であるため，今後はより大きなステージや複雑なゲームにおいて実験を行い，有効性を検証していく．

付　　録

A.1 GAN のモデル構造

表 A·1 Generator のモデル構成

Layer	Shape
Input	(32,)
Deconvolution	(512,3,4)
Batch Normalization	(512,3,4)
ReLU	(512,3,4)
Upsample	(512,6,8)
Convolution	(256,6,8)
Batch Normalization	(256,6,8)
ReLU	(256,6,8)
Self-Attention	(256,6,8)
Upsample	(256,12,16)
Convolution	(128,12,16)
Batch Normalization	(128,12,16)
ReLU	(128,12,16)
Self-Attention	(128,12,16)
Convolution	(8,12,16)
Softmax	(8,12,16)
Output	(8,12,16)

表 A·2　Discrimiantor のモデル構成

Layer	Shape
Input	(8,12,16)
Convolution	(128,12,16)
Leaky ReLU	(128,12,16)
Self-Attention	(128,12,16)
Convolution	(256,6,8)
Leaky ReLU	(256,6,8)
Self-Attention	(256,6,8)
Convolution	(512,3,4)
Leaky ReLU	(512,3,4)
Convolution	(1,)
Output	(1,)

A.2　比較手法により生成されたステージ

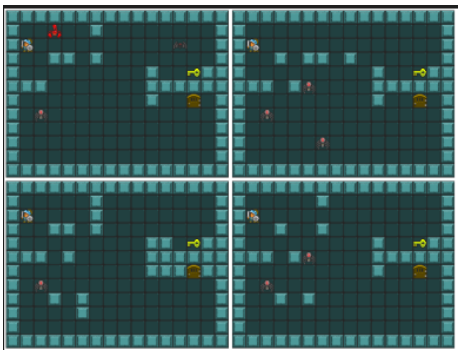

図 A·1　Bootstrap のみの手法で生成されたステージの例

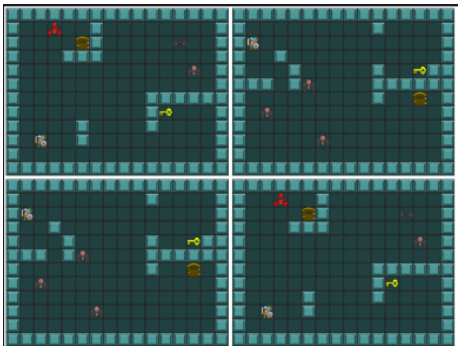

図 A·2　ベースライン手法で生成されたステージの例

参考文献

[1] Niels Justesen, Ruben Rodriguez Torrado, Philip Bontrager, Ahmed Khalifa, Julian Togelius, and Sebastian Risi. Procedural level generation improves generality of deep reinforcement learning. *CoRR*, Vol. abs/1806.10729, , 2018.

[2] Jialin Liu, Sam Snodgrass, Ahmed Khalifa, Sebastian Risi, Georgios N. Yannakakis, and Julian Togelius. Deep learning for procedural content generation. *CoRR*, Vol. abs/2010.04548, , 2020.

[3] Vanessa Volz, Jacob Schrum, Jialin Liu, Simon M. Lucas, Adam M. Smith, and Sebastian Risi. Evolving mario levels in the latent space of a deep convolutional generative adversarial network. *CoRR*, Vol. abs/1805.00728, , 2018.

[4] Noor ShakerJulian TogeliusMark J. Nelson. *Procedural Content Generation in Games*. SpringerLink, 2016.

[5] Linus Gisslén, Andy Eakins, Camilo Gordillo, Joakim Bergdahl, and Konrad Tollmar. Adversarial reinforcement learning for procedural content generation. *CoRR*, Vol. abs/2103.04847, , 2021.

[6] Philip Bontrager and Julian Togelius. Learning to Generate Levels From Nothing. *arXiv:2002.05259 [cs]*, August 2021. arXiv: 2002.05259.

[7] Ahmed Khalifa, Philip Bontrager, Sam Earle, and Julian Togelius. PCGRL: Procedural Content Generation via Reinforcement Learning. *arXiv:2001.09212 [cs, stat]*, August 2020. arXiv: 2001.09212.

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[9] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

[10] Nikolaus Hansen, Sibylle Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, Vol. 11, pp. 1–18, 02 2003.

[11] Ruben Rodriguez Torrado, Ahmed Khalifa, Michael Cerny Green, Niels Justesen, Sebastian Risi, and Julian Togelius. Bootstrapping conditional gans for video game level generation. *CoRR*, Vol. abs/1910.01603, , 2019.

[12] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks, 2018.

[13] Diego Perez-Liebana, Jialin Liu, Ahmed Khalifa, Raluca D. Gaina, Julian Togelius, and Simon M. Lucas. General video game ai: a multi-track framework for evaluating agents, games and content generation algorithms, 2018.

[14] Maren Awiszus, Frederik Schubert, and Bodo Rosenhahn. Toad-gan: Coherent style level generation from a single example, 2020.

[15] Andreas Hald, Jens Struckmann Hansen, Jeppe Theiss Kristensen, and Paolo Burelli. Procedural content generation of puzzle games using conditional generative adversarial networks. *International Conference on the Foundations of Digital Games*, 2020.

[16] Kyungjin Park, Bradford Mott, Wookhee Min, Kristy Boyer, Eric Wiebe, and James Lester. Generating educational game levels with multistep deep convolutional generative adversarial networks. pp. 1–8, 08 2019.

[17] Kirby Steckel and Jacob Schrum. Illuminating the space of beatable lode runner levels produced by various generative adversarial networks. *CoRR*, Vol. abs/2101.07868, , 2021.

[18] Vikram Kumaran, Bradford W. Mott, and James C. Lester. Generating game levels for multiple distinct games with a common latent space. In *AIIDE*, 2020.

[19] Jae Hyun Lim and Jong Chul Ye. Geometric gan, 2017.

[20] Ziwei Chen and Desheng Lyu. Procedural generation of virtual pavilions via a deep convolutional generative adversarial network. *Computer Animation and Virtual Worlds*, Vol. 33, No. 3-4, p. e2063, 2022. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cav.2063.