情報処理学会論文誌 Vol.65 No.1 69–82 (Jan. 2024)

少数サンプルから多様なゲームステージを生成するGANの学習手法の提案

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

受付日 2023年3月14日, 採録日 2023年10月3日

概要：ビデオゲームにおけるステージの生成は，ゲームの楽しさの向上や制作コスト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による手法が成果をあげているが，その学習データの用意は課題である．そこで，本研究では少数のサンプルのみから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提案する．本研究では，GANによるステージ生成の先行研究で用いられていた手法を改善した手法およびGANの学習時の損失関数に多様性を向上させる正則化項を加えて学習を行う手法を提案する．3つのタイルベースの2Dゲーム環境において，提案する手法により生成したステージ群に対し，その多様性を評価する評価指標を用意し，それにより定量的な評価を行った．その結果，従来手法によるモデルよりも制約を満たすステージの生成率は低下したものの，多様なステージが生成できることが確認できた．

キーワード：Procedural Content Generation, Tile-based Level Generation, Generative Adversarial Networks, Deep Learning

Diverse Level Generation for Tile-based Video Game Using Generative Adversarial Networks from Few Samples

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

Received: March 14, 2023, Accepted: October 3, 2023

Abstract: Automatic level generation for video games has been studied for many years in order to improve game enjoyment and reduce production costs. Although GAN-based methods have been successful in deep learning level generation for tile-based video games, the preparation of training data is a issue. In this study, we propose a method for learning GANs from a small number of samples to obtain a model that can generate a variety of levels. We propose a method that improves on the method used in previous studies of level generation using GANs, and a method that adds a regularization term to the loss function during GAN training to improve diversity. We evaluated the diversity of the levels generated by the proposed method in 3 tile-based 2D game environments using a quantitative evaluation metric. The results showed that the proposed method was able to generate more diverse levels than the existing method, although the rate of levels satisfying the constraints was lower than that of the existing model.

Keywords: procedural content generation, tile-based level generation, generative adversarial networks, deep learning

[1] 電気通信大学大学院情報理工学研究科情報学専攻
University of Electro-Communications Graduate School of Informatics and Engineering Department of Informatics, Chofu, Tokyo 182–8585, Japan
[a)] takata.soichiro@ohsuga.lab.uec.ac.jp
[b)] seiuny@uec.ac.jp
[c)] tahara@uec.ac.jp
[d)] ohsuga@uec.ac.jp

1. はじめに

1.1 概要

Procedural Content Generation（PCG）と呼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，多量のステージの人手による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．またゲーム AI を評価および学習する環境として 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ともない，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 Generative Adversarial Networks（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が課題となっている．一般に GAN による学習では，少量のデータによる学習から多様なデータを生成するモデルを獲得することは困難であるが，一方でゲームステージの生成モデルが多様なステージを生成できることは多様なゲーム体験を生むため，多様なステージから良いステージを選べるようになるために重要である．AI の評価，学習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る [1].

そこで本研究では，少量のデータ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，多様なステージを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ステージの難易度などの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，従来手法を改善したデータ拡張手法と学習データの選択法，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，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[*1]は低下したものの，ステージ群の多様性を測る評価指標により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，提案手法によるモデルは，潜在空間を探索することで，従来手法によるモデルよりも幅広い難易度のステージを生成できることを確認した．

本論文の構成は以下のとおりである．1 章で研究の概要を述べ，2 章で先行研究について述べる．3 章では提案手法のアプローチを従来手法の説明を交えながら説明する．4 章では実験の概要とその結果および考察を示し，5 章で本論文のまとめと今後の展望を述べる．

この論文は国際会議 ICAART で発表した内容を拡張したものである[*2].

[*1] モデルが生成したステージのうち，ステージとしての制約を満たすステージの割合

[*2] 情報処理学会のポリシとして国際会議論文は途中経過報告と見なされるため，二重投稿にならないことを確認している．

2. 先行研究

機械学習技術の発展にともなって，ゲームのコンテンツ生成に対しデータセットに基づくモデルを学習する機械学習の利用が増加している．このような機械学習技術により既存のゲームコンテンツで学習したモデルによるゲームコンテンツの生成は Procedural Content Generation via Machine Learning（PCGML）[4] と呼ばれ，研究が行われている．特に深層学習による生成モデルは，サンプルデータからモデルを学習する能力が高く，PCG においては GAN を用いたステージ生成の研究が成果をあげている．

GAN を用いてステージ生成を行った研究として，Volz らの MarioGAN [3] が有名である．Volz らは DCGAN [5] を用いて Super Mario Bros. のステージ生成を行い，従来の PCG で用いられていた探索ベースの手法と GAN によるステージ生成を組み合わせ，GAN の潜在空間を進化的手法で探索することで効率的にステージの探索を行えることを示した．具体的には，既存のステージのデータセットから学習された 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）[6] 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．ヒューリスティックやエージェントによるプレイの結果を評価関数として用いることで評価関数を最小化するステージを生成でき，ある程度の生成ステージの制御が可能であることを示している．また，熊谷ら [7] は，この MarioGAN のモデルを利用して進化計算によるステージ生成と準乱数，疑似乱数によるステージ生成を並行して行うことで，多様なステージを生成する手法を提案している．この研究は，多様なステージを生成するという点で本研究と類似しているが，この研究では，モデルに対する潜在変数をうまく選択して多様なステージを生成することを目的とするのに対し，本研究では多様なステージを生成可能なモデルを学習するという点が異なっている．MarioGAN 以外にも，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おり [8], [9], [10], [11], [12]，ゲーム以外にも同様の技術を用いている研究も行われている [13].

本研究同様に少数のデータから生成モデルを学習する研究には以下の 2 つの研究 [14], [15] がある．

Torrado ら [14] は，Generator と Discriminator に畳み込みによるモデルではなく，Self-Attention [16] をベースとしたモデルを用いて GVGAI Framework [17] 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ステージの制約条件をより遵守したプレイアビリティの高いステージ生成が可能な Conditional Embedding

Self Attention GAN（CESAGAN）を提案した．加えて，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 Bootstrapping と呼ばれるデータ拡張の手法を用い，データセットを補強することでごく少数のデータから GAN を学習する手法を提案している．これにより 5 つ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．また，Zakaria ら [15] は，2D パズルゲームである Sokoban（倉庫番）の深層学習によるステージ生成を行っており，GAN や VAE などの深層生成モデルや，深層強化学習を用いた手法などによる実験を行い，それらの定量的な比較を行っている．この研究では，GAN や Variational Auto Encoder などの深層生成モデルによりデータセットのステージデータからステージ生成モデルを学習するうえで，Bootstrapping によりデータセットを補強しつつ，生成ステージの解法（Solution）によりデータのクラスタリングを行う Diversity Sampling が提案されている．Diversity Sampling は，CESAGAN [14] で提案された Bootstrapping が，似た Solution のステージばかりを生成してしまうという欠点を改善する手法として提案され，実験では生成ステージの Solution の偏りの問題が改善されたことが報告されている．本研究では，これらの研究で提案された Bootstrapping および Diversity Sampling を用いた手法を従来手法として，比較対象として用いることにする．

3. 提案手法

3.1 概要

学習サンプルであるステージのデータからステージの特徴を学習し，新たなステージを生成する手法として，これまでの研究では深層生成モデルである GAN が多く用いられてきた．一般に GAN の学習には多くの学習データを必要とするが，ゲームのステージを大量に用意するのはコストがかかってしまい困難である．そこで，少数のデータからでも，ステージの特徴を学習し新規性のあるステージを生成できる手法が必要となる．少量のデータからステージ生成を行う GAN を学習する研究は行われている [14], [15] が，本研究では，従来の研究よりも多様なステージを生成できる Generator を獲得することを目的とする．提案手法の全体像を**図 1** に示す．

提案手法では，学習データを拡張する Filtered Bootstrapping，偏りなく学習データを選択する Balanced Diversity Sampling，生成データ間の距離を最大化する正則化を用いてネットワークの更新を行う Mode Seeking Regularization の 3 つの手法を提案し，これらを組み合わせてモデルの学習を行う．

3.2 Filtered Bootstrapping

まず，従来手法である Bootstrapping [14] について述べる．文献 [14] では，少数のデータからの GAN の学習を可

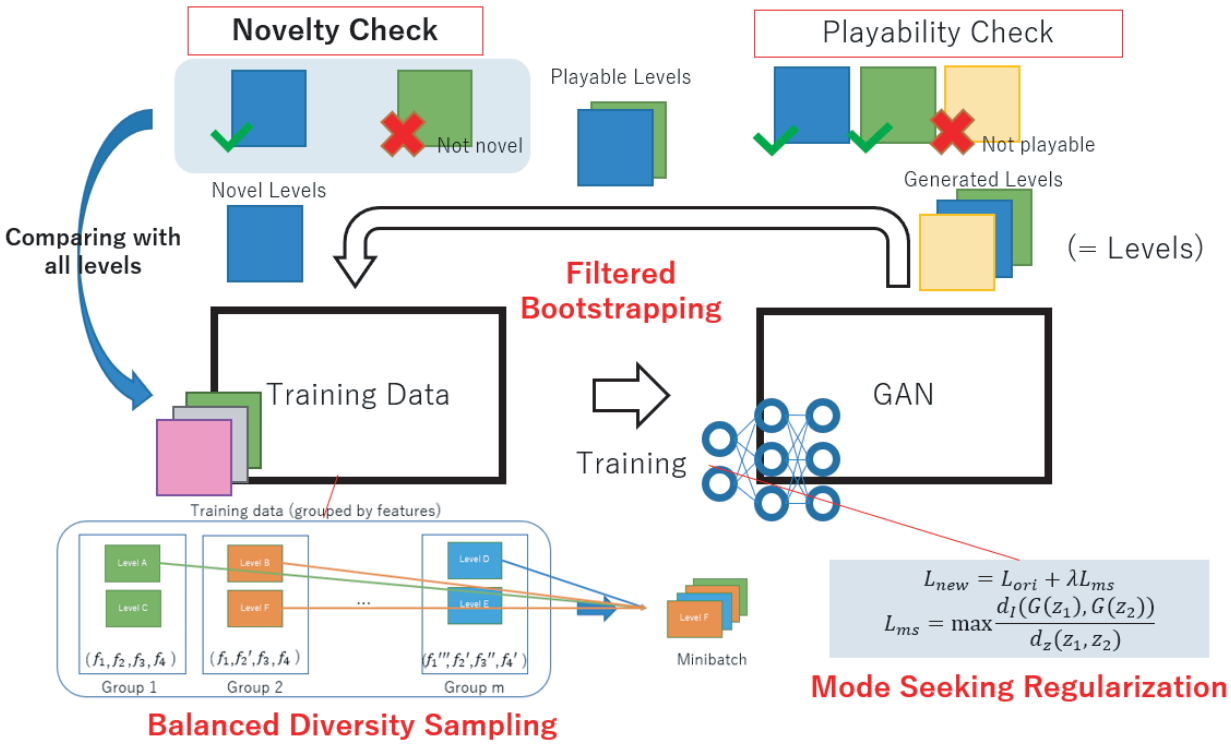


図 1 提案手法の全体像
Fig. 1 An overview of proposed method.

能にするために，Bootstrapping と呼ばれるデータ拡張手法が提案されている．Bootstrapping は，GAN の学習の過程において，Generator が生成したステージのうち，ステージの制約を満たしたプレイ可能なステージの一部を学習データセットに追加しながら GAN の学習を行う手法である．データセットに追加するステージは，2 次元の主成分分析により得られる 2 次元の特徴量に対して Elbow 法，k-means 法を用いて分類される最適な k 個のクラスタ重心に最も近い k ステージが選択される．つまり，生成した valid なステージのうち，2 次元の特徴量から互いに類似していないと判断される一部のステージを選択しデータセットに追加する．

この従来の Bootstrapping では，生成ステージのデータセットへの追加によりデータセットに偏りが生じ，それにより同じようなステージばかりを生成する Generator を学習してしまうという問題があった．データセットの偏りを生じないようにするために，生成ステージのうちデータセットにないような新規性の高いステージのみを追加することが有効であると考えられる．本研究では，そのような手法として Filtered Bootstrapping を提案する．Filtered Bootstrapping の疑似コードを Algorithm 1 に示す．Filtered Bootstrapping 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生成した Valid なステージを比較し，ハミング距離[*3]が一定の閾値以上のもののみを選択する（Algorithm 1，5–13 行目）．また，ハミング距離の基準により選択されたステージのうち，後述する Diversity Sampling の特徴量がデータセット内に少ないステージを 2 個[*4]まで優先的に選択し，データセットに追加する（Algorithm 1，17–21 行目）．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，データセット内の学習データの偏りの問題に対処できると考えられる．

Algorithm 1 Filtered Bootstrapping のアルゴリズム

Require: $\boldsymbol{l_g}$: generated levels from generator G, $\boldsymbol{l_d}$: levels in dataset, f_h: threshold of hamming distance, $\boldsymbol{C}$: constraints for valid levels.

1: **function** FILTERED_BOOTSTRAPPING($\boldsymbol{l_g}$)
2: $filtered\_levels \leftarrow []$
3: **for all** $l \in \boldsymbol{l_g}$ **do**
4: **if** l satisfies constraints $c \in \boldsymbol{C}$ all. **then**
5: $is\_novel \leftarrow true$
6: **for all** $l_d \in \boldsymbol{l_d}$ **do**
7: **if** dist(l,l_d) $\leq f_h$ **then**
8: $is\_novel \leftarrow false$
9: **end if**
10: **end for**
11: **if** $is\_novel = true$ **then**
12: $filtered\_levels \leftarrow filtered\_levels \cup \{l\}$.
13: **end if**
14: **end if**
15: **end for**
16: $selected\_levels \leftarrow []$
17: **for** $i = 0...max(2, size(filtered\_levels))$ **do**
18: Select l has fewest features in $\boldsymbol{l_d}$ from $filtered\_levels$.
19: $selected\_levels \leftarrow selected\_levels \cup \{l\}$
20: $filtered\_levels \leftarrow filtered\_levels - \{l\}$
21: **end for**
22: **return** $filtered\_levels$
23: **end function**

3.3 Balanced Diversity Sampling

まず，先行研究 [15], [18] で提案された Diversity Sampling について述べる．Diversity Sampling は，GAN の学習においてミニバッチの生成時にデータセット内の学習データを均等に使用するのではなく，ステージの特徴量ごとにサンプリングを行い学習データの多様化を図る手法である．ステージの特徴量は，ドメイン知識を用いてステージからいくつかの値（整数値）を抽出しペアにしたものを用いる．データセット内のステージをこの特徴量ごとにクラスタリングし，学習に用いるステージをサンプリングする際には，一様にクラスタを選択し，次にそのクラスタの中から一様にステージを選択する．これにより，従来の Bootstrapping を用いた学習で課題であった，似たようなステージばかりが生成されることが起こりにくくなり，設定した特徴量について多様なステージが生成できる．

Diversity Sampling では，データセット内のすべてのクラスタの中から一様にクラスタを選択する．特徴量はいくつかの要素の組であるが，全体からランダムに選択する手法では，特徴量の一部の要素が多様化しない可能性が考えられる．提案する Balanced Dievrsity Sampling では，特徴量のすべての要素について着目し，多様化させることを目指す．Balanced Diversity Sampling の疑似コードを Algorithm 2 に示す．Balanced Diversity Sampling では，まずデータセット内に存在する特徴量の要素を選択し，その要素を持つクラスタを選択する（Algorithm 2，5–7 行目）．これにより，データセット内に存在するすべての要素がそれぞれほぼ均等に学習データとして利用され，学習データを特徴量のすべての要素について多様化でき，Diversity Sampling により学習するよりも多様なステージを生成できるモデルが学習できると考えられる．

3.4 Mode seeking regularization

GAN の学習過程でより様々なステージを生成させるため，GAN の損失関数に Generator の生成データを多様化するための工夫を導入し学習を行うことを提案する．文献 [19] では，Conditional GAN [20] においてモード崩壊に

[*3] ここで，2 つのステージ間のハミング距離とは，2 つのステージを比較したとき，同じ位置に配置されている異なる種類のタイルの組の数とする．

[*4] 従来の Bootstrapping における PCA と Elbow 法および k-means 法による追加ステージの選択ではほとんどの場合 2 つのステージの選択が行われたため．

Algorithm 2 Balanced Diversity Sampling のアルゴリズム

Require: b: the size of minibatch, $\boldsymbol{l}_d$: levels in dataset, c: a feature value of a level, F_j: a set of feature numbers on the index j in C which is set of feature values in $\boldsymbol{l}_d$, $\boldsymbol{C_f}$: a set of feature values have feature number f. $\boldsymbol{l}_c$: a set of levels ($\subset \boldsymbol{l}_d$) have a feature value c,

function BALANCED_DIVERSITY_SAMPLING
  $minibatch \leftarrow []$
  for $i = 0, ..., b$ **do**
    Select $j \in \{0, ..., n\}$ randomly.
    Select $f \in F_j$ randomly.
    Select $c \in \boldsymbol{C_f}$ randomly.
    Select $l \in \boldsymbol{l}_c$ randomly.
    $minibatch \leftarrow minibatch \cup \{l\}$.
  end for
  return $minibatch$
end function

対処するための正則化項を GAN の損失関数に追加することで，より多様な画像の生成を可能にした．具体的には，潜在変数間の距離に対する生成画像間の距離の比率を最大化する mode seeking 正則化項を損失関数に追加し，正則化を行っている．mode seeking 正則化項 L_{ms} は，距離関数 $d_*(\cdot)$，潜在変数 $\boldsymbol{z_1}$，$\boldsymbol{z_2}$ として，以下の式 (1) で表される．

$$L_{ms} = \max_G \left(\frac{d_{\boldsymbol{I}}(G(\boldsymbol{z_1}), G(\boldsymbol{z_2}))}{d_{\boldsymbol{z}}(\boldsymbol{z_1}, \boldsymbol{z_2})} \right) \tag{1}$$

また，GAN の学習に用いる目的関数 L_{new} は，もとの目的関数 L_{ori} と係数 λ_{ms} により，以下の式 (2) で表される．

$$L_{new} = L_{ori} + \lambda_{ms} L_{ms} \tag{2}$$

本研究では，ステージの生成においても，この mode seeking 正則化項を導入して学習を行い，より多様なステージを生成することを目指す．この正則化項により，GAN の学習過程において，学習セットのステージと異なるステージの探索を容易にすることが期待される．

3.5 全体のアルゴリズム

提案手法を用いた GAN の学習アルゴリズムの全体を，疑似コードとして Algorithm 3 に示す．

なお，以下では提案手法の Filtered Bootstrapping を FB，Balanced Diversity Sampling を BDS，mode seeking regularization を MS と表すことがある．

4. 実験

4.1 対象とする環境

本研究では，ステージ生成を行う環境として，General Video Game AI（GVGAI）Framework [17] および，Video Game Level Corpus（VGLC）[21] から，Zelda（GVGAI），Boulderdash（GVGAI），Mario（VGLC）を用いた．GVGAI Framework は，ゲーム AI や PCG 研究，開発を行うためのフレームワークであり，専用の記述言語により記述された様々な種類のゲーム環境が用意されている．VGLC は，いくつかのメジャーな 2D ゲームのステージデータのコーパスであり，PCG 研究のために作成されたものである．Bootstrapping 手法が提案された先行研究 [14] では，多くのゲームにおいては数十程度のステージを用意することは現実的でないとして 5 つのステージのみを学習サンプルとして実験が行われている．そのため，本研究もその実験設定にならい同程度の学習サンプルから学習を行った．Zelda，Boulderdash は GVGAI 環境に用意された 5 ステージ，Mario では先行研究 [3] において学習サンプルとして用いられたステージの中から，ランダムに選択した 10 ステージを使用した．学習サンプルとして使用したステージを付録の**図 A·1**，**図 A·2**，**図 A·3** に示している．学習サンプルとして使用されるステージは，それぞれ攻略時の進路が異なり，また Diversity Sampling における特徴量について Zelda と Boulderdash では各ステージがそれぞれ異なる特徴量を持ち，Mario については 10 個のステージのうち 8 種類の特徴量を持っており，多様である．その他，各ゲームのルールや制約条件などの詳細については付録に記している．

Algorithm 3 提案手法の全体アルゴリズム

Require: n_{steps}: the training steps, n_{critic}: the number of iterations of the critic per generator iteration, m: the minibatch size, c: the clipping parameter, α: the learning rate

for $s = 0, ..., n_{steps} - 1$ **do**
  for $t = 0, ..., n_{critic} - 1$ **do**
    $\boldsymbol{x} \leftarrow$ BALANCED_DIVERSITY_SAMPLING($\boldsymbol{l_d}$).
    Sample $\boldsymbol{z} \sim p(\boldsymbol{z})$ a batch of gaussian distributions.
    $g_w \leftarrow \nabla_w [\frac{1}{m} \sum_{i=1}^{m} \min(0, -1 - f_w(\boldsymbol{x}^{(i)}))$
      $-\frac{1}{m} \sum_{i=1}^{m} \min(0, -1 + f_w(g_\theta(\boldsymbol{z}^{(i)})))]$
    $w \leftarrow w + \alpha \cdot RMSProp(w, g_w)$
    $w \leftarrow clip(w, -c, c)$
  end for
  Sample $\boldsymbol{z} \sim p(\boldsymbol{z})$ a batch of gaussian distributions.
  $g_\theta \leftarrow -\nabla_\theta \left[\frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(\boldsymbol{z}^{(i)})) \right.$
    $\left. +\lambda_{ms} \frac{1}{\frac{1}{\frac{m}{2}} \sum_{t=1}^{\frac{m}{2}} \frac{|g_\theta(\boldsymbol{z}^{(t)}) - g_\theta(\boldsymbol{z}^{(t+\frac{m}{2})})|}{|\boldsymbol{z}^{(t)} - \boldsymbol{z}^{(t+\frac{m}{2})}|}} \right]$
  $\theta \leftarrow \theta - \alpha \cdot RMSProp(\theta, g_\theta)$
  if $s \% n_{bs} = 0$ **then**
    $\boldsymbol{l_{bs}} \leftarrow []$
    Sample $\boldsymbol{z} \sim p(\boldsymbol{z})$ a batch of gaussian distributions.
    Decode $g_\theta(\boldsymbol{z})$ to levels $\boldsymbol{l}$.
    $\boldsymbol{l_{bs}} \leftarrow$ FILTERED_BOOTSTRAPPING($\boldsymbol{l}$)
    $\boldsymbol{l_d} \leftarrow \boldsymbol{l_d} \cup \boldsymbol{l_{bs}}$.
  end if
end for

4.2 実験設定

4.2.1 モデルの構成

ニューラルネットワークは，文献 [3] と同様に畳み込み

表 1 Generator のモデル構成．C はゲームごとのタイルの種類数
Table 1 The model structure of generator. C is the number of types of tiles.

Layer	Shape
Input	(32,)
Deconvolution	(256,4,4)
Batch Normalization	(256,4,4)
ReLU	(256,4,4)
Deconvolution	(128,8,8)
Batch Normalization	(128,8,8)
ReLU	(128,8,8)
Deconvolution	(64,16,16)
Batch Normalization	(64,16,16)
ReLU	(64,16,16)
Deconvolution	(C,32,32)
Softmax	(C,32,32)
Output	(C,32,32)

表 2 Discriminator のモデル構成．C はゲームごとのタイルの種類数
Table 2 The model structure of discriminator. C is the number of types of tiles.

Layer	Shape
Input	(C,32,32)
Convolution	(64,16,16)
Leaky ReLU	(64,16,16)
Convolution	(128,8,8)
Leaky ReLU	(128,8,8)
Convolution	(256,4,4)
Leaky ReLU	(256,4,4)
Convolution	(1,)
Output	(1,)

によるニューラルネットワークを用いた[*5]．

GAN のモデルを構成する Generator および Discriminator のモデル構造を，**表 1**，**表 2** に示す．また，このモデルの概略図を**図 2** に示す．本研究では，図 2 のとおり，シンプルな畳み込みによるネットワークを用いている．

Discriminator の各畳み込み層に対しては，Spectral Normalization [22] を適用しており，Discriminator の更新のたびに重みパラメータを一定の範囲にクリップする Parameter Clipping [23] を同時に用いている．これは，経験的に Parameter Clipping と Spectral Normalization を併用することで，少数データでのステージの学習が安定化したためである．

4.2.2 ステージの表現方法

高さ h 幅 w のステージをニューラルネットワークで扱える形式にするために，各タイルについてそのゲームにおけるタイルの種類数を c としてチャネル方向に one-hot 化し，サイズ (c,h,w) の 3 次元テンソルに変換する．ネットワークへの入力は高さおよび幅は同じ大きさであり，入力および出力のサイズを合わせるため，テンソルへのエンコード時に不足した高さおよび幅については決められた適当な種類のタイルでパディングする．

[*5] 後述する Zelda のステージ生成においては，ステージが小さいため畳み込み層および転置畳み込み層を 1 層減らし，出力，入力サイズを (C,16,16) としたモデルを使用している．

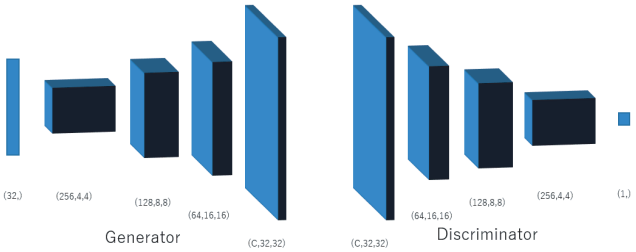


図 2 GAN のネットワークの概略図
Fig. 2 GAN model used for experiments.

表 3 学習における共通設定
Table 3 Parameters for training models. These parameters are common to each game.

潜在変数のサイズ	32
ミニバッチサイズ	32
学習率 (G)	1.0×10^{-4}
学習率 (D)	1.0×10^{-4}
学習ステップ数	5,000
Bootstrapping の間隔ステップ数 n_{bs}	10
Bootstrapping において一度に生成するステージ数	128

表 4 学習におけるゲームごとの設定
Table 4 Parameters for training models. These parameters are set to different values for each game.

	各ゲームにおける値 (Zelda/Mario/Boulderdash)
FB における閾値	92.5/95.0/85.0
MS の重み λ_{ms}	0.01/0.05/0.05

4.2.3 学習設定

実験では，それぞれの環境および手法についてモデルの学習を行う．学習は，敵対的損失として hinge adversarial loss [24] を用い，RMSProp [25] によりネットワークの重みを更新する．学習時のパラメータの設定を**表 3** および **表 4** に示す．

4.3 実験 1：生成ステージの多様性の定量評価

4.3.1 概要

この実験では，従来手法（Bootstrapping + Diversity Sampling），FB（と Diversity Sampling）を利用した手法，FB と BDS を利用した手法，FB と BDS と MS を利用した手法によりそれぞれ GAN の学習を行い，それぞれ学習したモデルに対して，以下で紹介するいくつかの指標により定量的に比較を行う．実験結果には，各手法ごとに 3 回ずつモデルの学習を行い，3 つのモデルの結果の平均値を

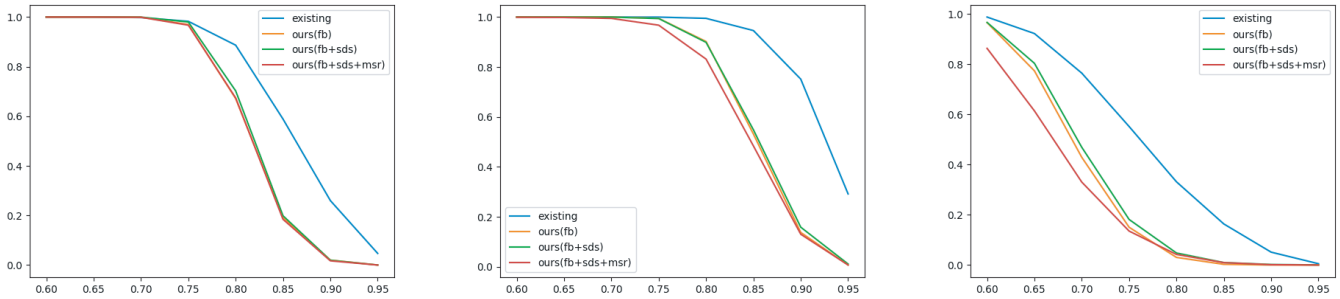


図 3 X%類似度のグラフ．左から Zelda，Mario，Boulderdash の順．横軸を X として縦軸が X%類似度．青線が従来手法（Bootstrapping + Diversity Sampling），黄線が FB を用いた場合，緑線が FB と BDS を用いた場合，赤線が FB，BDS，MS を用いた場合

Fig. 3 X% similarities for each method and each game. These graphs show results of Zelda, Mario, Boulderdash in a row. The horizontal line shows X and the horizontal one shows X% similarity. The blue line, yellow line, green line, red line shows existing method (Bootstrapping + Diversity Sampling), FB, FB + BDS, FB + BDS + MS respectively.

使用する．実験では，各手法および各モデルにおいてプレイアビリティの評価のためにちょうど 15,000 ステージを生成し，プレイアビリティの評価に用いる．加えて，それとは別に多様性の評価のためにプレイ可能なステージを 15,000 個生成するまでステージの生成を行い，プレイ可能な 15,000 ステージを多様性の評価に用いる．

4.3.2 評価指標

学習したモデルがステージとして制約を満たしたものを生成できているか，またその生成したステージの多様性を評価する指標として，以下の指標を提案し使用する．

プレイアビリティ（PA） 生成したステージのうち，どれだけの割合のステージがステージとしての制約を満たしているかを表す指標．

重複率（DR） 生成したステージのうち，どれだけの割合のステージが生成した他のステージと重複しているかを表す指標．

平均ハミング距離（AHD） 生成したステージのすべてのペアについてのハミング距離を平均した値．

生成特徴量数（FC） 生成したステージ群に存在する Diversity Sampling で用いる特徴量の種類の数．

X%類似度 生成したステージのすべてのペアについて，ハミング距離による類似度（1 − (ハミング距離)/(総タイル数)）が X%以上のものの割合．

プレイアビリティについては生成した 15,000 ステージから算出した．プレイ可能性の判定は，付録 A.1 に記した各ゲームの制約条件を満たすようにプログラムにより判定を行った．この制約を満たしている場合でもプレイアブルでないようなステージが存在する可能性はある[*6]が，プレイアビリティを完全に判定することは困難であるため，本研究ではこれらの制約条件をすべて満たす場合，そのステージはプレイ可能であると定義した．また，重複率は生成したステージのうちプレイアビリティの制約を満たした 15,000 ステージから算出した．平均ハミング距離および生成特徴量数，X%類似度の算出には，このプレイ可能な 15,000 ステージから 1,000 ステージをランダムに選択し，この 1,000 ステージから算出した．

本研究で用いる多様性の評価指標は，先行研究 [14] でも用いられている重複率（DR）と平均ハミング距離（AHD），また新たに提案した X%類似度はステージを単純に画像として見た際の多様性を評価し，FC はプレイヤの進路に大きく影響すると考えられるゲーム固有の特徴を反映してゲームプレイの多様性を評価する．ステージの多様性を評価する定量指標は先行研究において確立された手法は存在せず，生成ステージの真の多様性を評価する指標の確立は研究分野全体の課題であるが，本研究では，これらの複数の評価指標を用いることで生成ステージの多様性を定量的に評価する．

4.3.3 結果

それぞれのゲーム環境において従来手法および提案手法によって学習を行ったモデルで生成したステージに対して評価を行った結果を**表 5**，**表 6**，**表 7** に示す．また，生成したステージの例を Web 上に公開している[*7]．従来手法では，全体的に生成したステージの重複率が高く，平均ハミング距離が小さい．Bootstrapping と Diveristy Sampling を用いて学習を行うことで，データ拡張を行わずに GAN を学習するよりも多様性は増しているが，オリジナルのステージとかなり類似したステージを多く生成してしまう．提案手法では，Filtered Bootstrapping により追加するステージを新規性の高いステージのみに制限することで，平均ハミング距離，重複率，生成特徴量数といった指標が大幅に向上した．また，Balanced Diversity Sampling

[*6] 各ゲームについて 100 個の生成結果を目視により確認した結果，そのようなステージは確認されなかった．

[*7] https://github.com/Takata1128/GameLevelGAN_Images

表 5 Zelda におけるプレイアビリティと多様性の定量評価の結果
Table 5 Playability and diversity statistics of each method for Zelda.

	PA (%)↑	AHD↑	DR (%)↓	FC↑
Existing	**56.4**	26.3	23.1	95.3
Ours(FB)	35.3	35.0	**0.0**	265.7
Ours(FB+BDS)	25.7	34.5	**0.0**	408.3
Ours(FB+BDS+MS)	21.4	**35.1**	**0.0**	**506.7**

表 6 Mario におけるプレイアビリティと多様性の定量評価の結果
Table 6 Playability and diversity statistics of each method for Mario.

	PA (%)↑	AHD↑	DR (%)↓	FC↑
Existing	**73.0**	29.4	53.1	44.7
Ours(FB)	44.3	57.4	**0.2**	200.3
Ours(FB+BDS)	49.6	56.6	0.8	204.3
Ours(FB+BDS+MS)	44.2	**60.5**	0.4	**286.7**

表 7 Boulderdash におけるプレイアビリティと多様性の定量評価の結果
Table 7 Playability and diversity statistics of each method for Boulderdash.

	PA (%)↑	AHD↑	DR (%)↓	FC↑
Existing	**62.7**	79.8	1.0	84.3
Ours(FB)	30.0	104.0	**0.0**	255.3
Ours(FB+BDS)	25.4	101.9	0.2	337.0
Ours(FB+BDS+MS)	21.5	**110.5**	**0.0**	**338.7**

によって，特に Boulderdash と Zelda において Diversity Sampling を用いた場合よりも生成特徴量数が増加しており，設計した特徴量について多様なステージが生成できていることが分かる．加えて，mode seeking regularization を導入することで，さらに多様性の指標を全体的に向上させることができている．全体的に，提案手法により生成したステージのプレイアビリティは低下しているものの，生成ステージの多様性は向上しており多様なステージを生成できるモデルが獲得できたことが定量的に示されている．

4.4 実験 2：潜在変数の探索手法の有効性の評価

4.4.1 概要

先行研究 [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製作者の意図を反映したステージの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，ガウス分布を用いた多点探索を用いて進化計算により連続最適化を行う．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．本実験では，提案手法により学習したモデルが，ステージの難易度を易化する目的関数と難化する目的関数についてどれだけその目的を達成する生成ができているかを確認する．比較手法として，従来手法（Bootstrapping + Diversity Sampling）によるモデルを用いる．CMA-ES のパラメータについては初期集団数は 14 とし，標準偏差は 0.5 に初期化して 100 イテレーションの最適化を行った．

4.4.2 Zelda における目的関数

Zelda において，壁タイルの数および敵タイルの数が多いほど難しく，少ないほどやさしいステージであると定め，最小化により難易度を難化させる $F_{z_{hard}}$，難易度を易化させる $F_{z_{easy}}$ を設計した．ここで，$\#wall$ は壁タイルの数，$\#enemy$ は敵タイルの数を表している．

$$F_{z_{hard}} = \begin{cases} 1000 & \text{if not playable} \\ -\#wall - 3 \times \#enemy & \text{otherwise} \end{cases} \tag{3}$$

$$F_{z_{easy}} = \begin{cases} 1000 & \text{if not playable} \\ \#wall + 3 \times \#enemy & \text{otherwise} \end{cases} \tag{4}$$

4.4.3 Mario における目的関数

Mario において，下から 1 行目の空タイル（穴になっている部分）の数，敵タイルの数が多いほど難しく，少ないほどやさしいステージであると定め，最小化により難易度を難化させる $F_{m_{hard}}$，難易度を易化させる $F_{m_{easy}}$ を設計した．ここで，$\#hole$ は下から 1 行目の空タイル（穴になっている部分）の数，$\#enemy$ は敵タイルの数を表している．

$$F_{m_{hard}} = \begin{cases} 1000 & \text{if not playable} \\ -\#hole - \#enemy & \text{otherwise} \end{cases} \tag{5}$$

$$F_{m_{easy}} = \begin{cases} 1000 & \text{if not playable} \\ \#hole + \#enemy & \text{otherwise} \end{cases} \tag{6}$$

4.4.4 結果

Zelda，Mario それぞれに対して，各目的関数により潜在変数の最適化を行った結果を**表 8**，**表 9** に示し，提案手法で学習したモデルでの最適化の例を付録の**図 A·4**，**図 A·5**，**図 A·6**，**図 A·7** に示す．

表 8，9 より，提案手法も従来手法も，潜在変数の最適化によりプレイ可能なステージを獲得することができていることが分かる．また，生成したレベルの平均ハミング距離を比較すると提案手法によるモデルの方が様々なバリエーションのステージを生成できていることが分かる．加えて目的関数値を比較すると，Zelda と Mario ともに提案手法によるモデルは難易度のやさしいステージの生成は同程度もしくは少し劣る程度にとどまるものの，難しいステージの生成では既存手法によるモデルよりも大きく最適化でき

表 8 Zelda において，$F_{z_{hard}}$，$F_{z_{easy}}$ それぞれで最適化したときのスコアと平均ハミング距離

Table 8 Playability, average values of objective function and average hamming distances for Zelda levels generated from latent variables optimized by CMA-ES.

	$F_{z_{hard}}$			$F_{z_{easy}}$		
	PA (%)↑	Value ↓	AHD ↑	PA (%)↑	Value ↓	AHD ↑
Ours	100.0	**−109.4**	**37.2**	100.0	64.3	**22.0**
Existing	100.0	−92.4	28.5	100.0	**60.0**	13.8

表 9 Mario において，$F_{m_{hard}}$，$F_{m_{easy}}$ それぞれで最適化したときのスコアと平均ハミング距離

Table 9 Playability, average values of objective function and average hamming distances for Mario levels generated from latent variables optimized by CMA-ES.

	$F_{m_{hard}}$			$F_{m_{easy}}$		
	PA (%)↑	Value ↓	AHD ↑	PA (%)↑	Value ↓	AHD ↑
Ours	100.0	**−16.02**	**54.5**	100.0	0.4	**48.1**
Existing	100.0	−6.3	16.8	100.0	**0.16**	32.0

ており，従来手法によるモデルよりも難しいステージを生成できている．

また，Zelda と Mario において，提案手法および従来手法によるモデルで生成した 10,000 ステージの難易度について，その分布を**図 4**，**図 5**，**図 6**，**図 7** に示している．これらの結果から，提案手法によるモデルは，従来手法によるモデルと比較して全体的に難しいステージを生成するようになってはいるが，十分にやさしいステージの生成とより難しいステージの生成ができており，全体として，より幅広い難易度のステージを生成できるようになっていることが分かる．そのため，提案手法により学習したモデルは，やさしいステージや難しいステージを狙って生成することができるようになっており潜在変数の探索による目的関数の意図を反映したステージ生成手法を，従来手法によるモデルよりも有効に利用できると考えられる．

4.5 考察

実験結果より，提案する手法により学習を行ったモデルは，従来手法により学習を行ったモデルよりも多様な幅広い難易度のステージを生成できることを確認した．提案手法の中では特に Filtered Bootstrapping による影響が最も大きく，平均ハミング距離（AHD）や重複率（DR）を大きく向上させただけでなく，ハミング距離とは直接的に関係のない生成特徴量数の指標も改善されている．これより，ハミング距離によるフィルタリングがステージ上のオブジェクトの位置や数などの指標の多様化についても有効であり，ハミング距離による新規性の判断の有効性を示している．Balanced Diversity Sampling については，Zelda と Boulderdash について生成特徴量数を大きく向上させ，Mario においてもわずかながら増加がみられた．この有効性は特徴量として何を選択するかに依存すると考えられるが，今回の実験においては特にタイルの位置に関する特徴

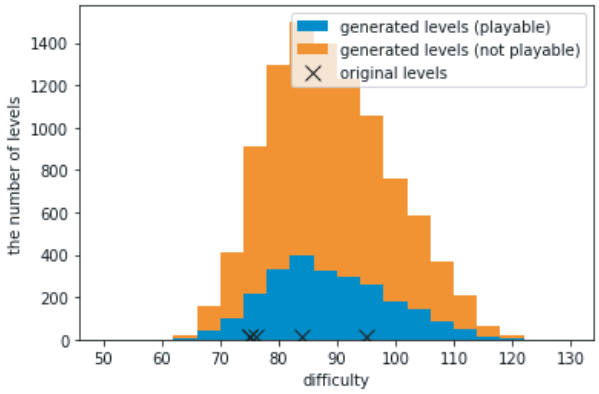


図 4 提案手法により学習したモデルで生成した Zelda のステージの難易度（$\#wall + \#enemy \times 3$）の分布．× 印は学習サンプルとして用いたステージ

Fig. 4 The distribution of difficulty (#wall + #enemy × 3) of Zelda levels generated by the modal learning by proposed method.

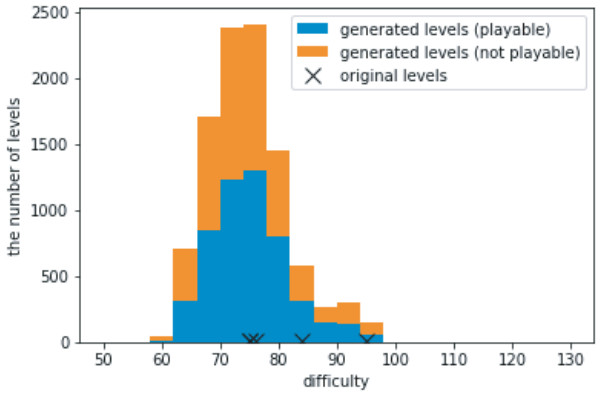


図 5 従来手法により学習したモデルで生成した Zelda のステージの難易度（$\#wall + \#enemy \times 3$）の分布．× 印は学習サンプルとして用いたステージ

Fig. 5 The distribution of difficulty (#wall + #enemy × 3) of Zelda levels generated by the modal learning by existing method.

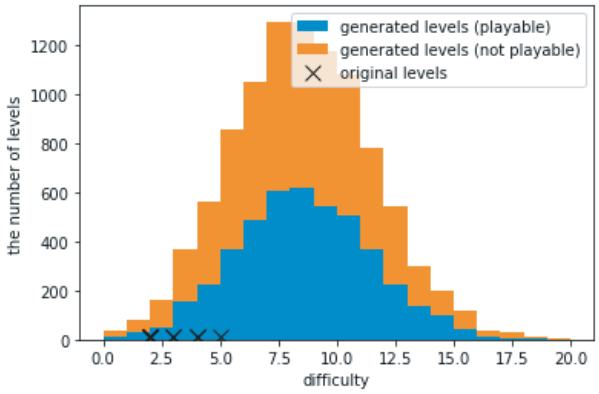


図 6 提案手法により学習したモデルで生成した Mario のステージの難易度（$\#enemy + \#hole$）の分布．× 印は学習サンプルとして用いたステージ

Fig. 6 The distribution of difficulty (#wall + #hole) of Mario levels generated by the modal learning by proposed method.

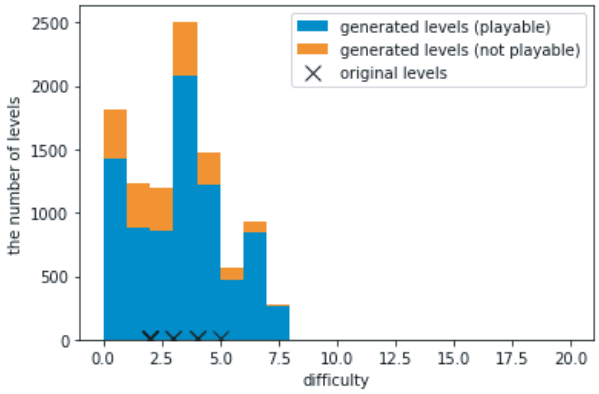


図 7 従来手法により学習したモデルで生成した Mario のステージの難易度（$\#enemy + \#hole$）の分布．× 印は学習サンプルとして用いたステージ

Fig. 7 The distribution of difficulty (#wall + #hole) of Mario levels generated by the modal learning by existing method.

の多様化に対し有効なようであった．

それぞれの提案手法において多様性を向上することができた一方で，生成ステージのプレイアビリティは低下している．Filtered Bootstrapping や Balanced Diversity Sampling を利用して学習を行うことで，より多様なステージを学習データとして用いるために，学習データが少ない場合や偏った場合よりもステージの局所的，大域的な特徴を学習することが難しくなり，プレイアビリティが低下してしまう．また mode seeking 正則化項は，学習データの分布の近似という目的を無視し，生成データ間の L1 距離を最大化する方向へネットワークを更新しようとする項であるためステージとして正しくないステージを生成する割合が高くなってしまったと考えられる．Zelda や Boulderdash のように特定のタイルの生成数に制限があるような環境では，特定のタイルを全体でただ 1 つ生成しなければならないという大域的な制約が最も困難な制約であり，プレイアビリティが低下する最も大きな要因となっていた．

提案手法により生成ステージのプレイアビリティは低下したが，本研究で用いたモデルは軽量で，並列計算により短時間で 1 度に大量のデータを生成することができる．実際に，Zelda のプレイ可能とは限らないステージを 128 個生成するのにかかる時間は約 0.2 秒である．このような場合，大量のデータを生成しその中からプレイ可能なステージを選択することができるため，プレイアビリティが多少低いモデルでも，多様な生成が可能であれば多様なプレイ可能ステージを多く生成することができる．実際に，Zelda では表 5 にあるとおり従来手法に対し提案手法はプレイアビリティは約 $\frac{2}{5}$ 程度に低下しているのに対し FC は約 5 倍となっていることから，同程度の FC を持つステージ群を生成するために従来手法によるモデルは提案手法によるモデルと比較して少なくとも $\frac{2}{5} \times 5 = 2$ 倍以上[*8]のステージを生成する必要がある．Mario，Boulderdash においても，FC の増加の割合がプレイアビリティの減少の割合を上回っており，同様である．また，4.4 節で示した CMA-ES による手法により，CPU 上で数秒の計算時間[*9]でプレイ可能かつ目的関数に沿ったステージを探索することも可能である．このような理由から，プレイアビリティの低下は大きな問題ではないと考えられる．

本研究では，比較的シンプルなモデル，環境を用いて少数のサンプルから学習し多様なステージを生成できるモデルの学習手法を提案したが，今後より複雑な制約を持つゲームや大規模なゲームに本手法を適用する場合や生成効率の向上のためには生成データのプレイアビリティを改善するための工夫は必要であると考えられる．ステージの制約は一般的に微分不可能であり，誤差逆伝播法により Generator に直接的にステージ制約を守るように最適化を行うことは困難である．そのため，文献 [14] のようにモデルアーキテクチャに工夫を施したり，文献 [18] のように GAN ではなく自己回帰モデルやフローモデルなどの他の生成モデルにより学習を行うことや，文献 [26] のような GA を用いた CNN の重みパラメータ最適化を生成モデルに適用し，微分不可能な目的関数でもモデルを最適化できるようになれば，この問題は改善される可能性があると考えられる．

また，本研究ではステージの生成のみに着目し実験を行ったが，実際のゲーム開発においては生成されたステージをテストプレイする必要があり，それらを含めた工程全体の評価を行う必要があると考えられる．近年では，これらのテストプレイにおいても自動化の研究が進められており [27], [28]，またゲーム自体の設計や開発を自動化する研

*8 実際には，従来手法のモデルは生成可能ステージの種類が提案手法によるモデルに対し少ないと考えられるため，ステージをいくら生成しても提案手法が達成する FC に届かない可能性がある．

*9 Zelda において，1 つのステージの最適化にかかった時間は平均して 4.95 秒であった．

究も進められている [29]．そのような技術と組み合わせることで，ステージ製作全体の工程を低いコストで達成できるのではないかと考えられるが，ステージ製作工程全体の評価については今後の課題である．

5. まとめ

本研究では，いくつかのゲーム環境において，5，10 個程度の少数のサンプルから GAN の学習を行い，ハミング距離によるフィルタ処理を用いた生成データによる学習データの拡張，Balanced Diversity Sampling による多様な学習データのサンプリングおよび GAN の学習時の正則化により，従来手法によるモデルよりも多様なステージが生成可能なモデルを獲得できることを示した．特に，ハミング距離によるフィルタ処理を用いたデータ拡張である Filtered Bootstrapping が生成データの多様性を最も大きく改善した．一方で，提案手法により多様性の指標が向上したが制約を満たしたステージを生成する割合は低下し，特に Mode seeking regularization を導入して学習した場合さらに低下してしまうことが分かった．提案手法では，生成ステージのプレイアビリティは低下するものの，CMA-ES による潜在変数の最適化により制約を満たしたステージを狙って生成することができる．加えて多様なステージが生成できるようになったことで，従来手法によるモデルよりも難易度の高いステージを生成でき，幅広い難易度の多様なステージを生成できることを確認した．

本研究では，生成ステージの多様性とプレイアビリティを両立できておらず，両者はトレードオフの関係にあり，多様なステージを生成するモデルでは生成ステージが制約を満たす割合は低下してしまう．今後は，より多様なステージを生成できつつプレイアビリティが高くなるように，またより複雑な制約を持つステージ生成を行えるように，学習手法や生成モデルを改善，再考していくべきである．

謝辞 本研究は JSPS 科研費 JP21H03496，JP22K12157 の助成を受けたものです．

参考文献

[1] Justesen, N., Torrado, R.R., Bontrager, P., Khalifa, A., Togelius, J. and Risi, S.: Procedural level generation improves generality of deep reinforcement learning, *CoRR*, Vol.abs/1806.10729 (2018).

[2] Liu, J., Snodgrass, S., Khalifa, A., Risi, S., Yannakakis, G.N. and Togelius, J.: Deep learning for procedural content generation, *CoRR*, Vol.abs/2010.04548 (2020).

[3] Volz, V., Schrum, J., Liu, J., Lucas, S.M., Smith, A.M. and Risi, S.: Evolving mario levels in the latent space of a deep convolutional generative adversarial network, *CoRR*, Vol.abs/1805.00728 (2018).

[4] Summerville, A., Snodgrass, S., Guzdial, M., Holmgård, C., Hoover, A.K., Isaksen, A., Nealen, A. and Togelius, J.: *Procedural content generation via machine learning* (*PCGML*) (2017).

[5] Radford, A., Metz, L. and Chintala, S.: *Unsupervised representation learning with deep convolutional generative adversarial networks* (2015).

[6] Hansen, N., Müller, S. and Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), *Evolutionary Computation*, Vol.11, pp.1–18 (2003).

[7] 熊谷 涼，高木智章，高玉圭樹，佐藤寛之：MarioGAN と進化計算による多様なステージ生成に関する検討，計測自動制御学会システム・情報部門学術講演会講演論文集，Vol.2020, ROMBUNNO.GS2-3-2 (2020).

[8] Awiszus, M., Schubert, F. and Rosenhahn, B.: *Toadgan: Coherent style level generation from a single example* (2020).

[9] Hald, A., Hansen, J.S., Kristensen, J.T. and Burelli, P.: Procedural content generation of puzzle games using conditional generative adversarial networks, *International Conference on the Foundations of Digital Games* (2020).

[10] Park, K., Mott, B., Min, W., Boyer, K., Wiebe, E. and Lester, J.: *Generating educational game levels with multistep deep convolutional generative adversarial networks*, pp.1–8 (08 2019).

[11] Steckel, K. and Schrum, J.: Illuminating the space of beatable lode runner levels produced by various generative adversarial networks, *CoRR*, Vol.abs/2101.07868 (2021).

[12] Kumaran, V., Mott, B.W. and Lester, J.C.: Generating game levels for multiple distinct games with a common latent space, *AIIDE* (2020).

[13] Chen, Z. and Lyu, D.: 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), available from ⟨https://onlinelibrary.wiley.com/doi/pdf/10.1002/cav.2063⟩.

[14] Torrado, R.R., Khalifa, A., Green, M.C., Justesen, N., Risi, S. and Togelius, J.: Bootstrapping conditional gans for video game level generation, *2020 IEEE Conference on Games* (*CoG*), pp.41–48 (2020).

[15] Zakaria, Y., Fayek, M. and Hadhoud, M.: Procedural level generation for sokoban via deep learning: An experimental study, *IEEE Trans. Games*, p.1 (2022).

[16] Zhang, H., Goodfellow, I., Metaxas, D. and Odena, A.: *Self-attention generative adversarial networks* (2018).

[17] Perez-Liebana, D., Liu, J., Khalifa, A., Gaina, R.D., Togelius, J. and Lucas, S.M.: *vglc* (2018).

[18] Zakaria, Y., Fayek, M. and Hadhoud, M.: *Start small: Training game level generators from nothing by learning at multiple sizes* (2022).

[19] Mao, Q., Lee, H.-Y., Tseng, H.-Y., Ma, S. and Yang, M.-H.: Mode seeking generative adversarial networks for diverse image synthesis, *CoRR*, Vol.abs/1903.05628 (2019).

[20] Mirza, M. and Osindero, S.: *Conditional generative adversarial nets* (2014).

[21] Summerville, A.J., Snodgrass, S., Mateas, M. and Villar, S.O.: The VGLC: The video game level corpus, *CoRR*, Vol.abs/1606.07487 (2016).

[22] Miyato, T., Kataoka, T., Koyama, M. and Yoshida, Y.: Spectral normalization for generative adversarial networks, *CoRR*, Vol.abs/1802.05957 (2018).

[23] Arjovsky, M., Chintala, S. and Bottou, L.: *Wasserstein GAN* (2017).

[24] Lim, J.H. and Ye, J.C.: *Geometric GAN* (2017).

[25] Hinton, G., Srivastava, N. and Swersky, K.: *Neural networks for machine learning lecture 6a overview of mini-*

batch gradient descent (2012).
[26] Esfahanian, P. and Akhavan, M.: *GACNN: Training deep convolutional neural networks with genetic algorithm* (2019).
[27] Paduraru, C., Paduraru, M. and Stefanescu, A.: Rivergame-a game testing tool using artificial intelligence, *2022 IEEE Conference on Software Testing, Verification and Validation* (*ICST*), pp.422–432, IEEE (2022).
[28] Albaghajati, A.M. and Ahmed, M.A.K.: Video game automated testing approaches: An assessment framework, *IEEE Trans. Games* (2020).
[29] Edwards, G., Subianto, N., Englund, D., Goh, J.W., Coughran, N., Milton, Z., Mirnateghi, N. and Shah, S.A.A.: The role of machine learning in game development domain-a review of current trends and future directions, *2021 Digital Image Computing: Techniques and Applications* (*DICTA*), pp.1–7 (2021).

付　　録

A.1　対象とする環境の詳細

ここでは，本論文でステージ生成の対象として用いた 3 つの環境の詳細について示す．

A.1.1　Zelda（GVGAI）

概要　Zelda は 2 次元の迷路型のゲームであり，プレイヤは高さ 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．ステージは壁のタイル，床のタイル，敵のタイルなど全 8 種のタイルから構成されている．

学習データ　学習データとして，図 A·1 の 5 種類のステージを用いた．

制約条件　制約条件は以下のとおり．

- プレイヤ，鍵，ゴールタイルはただ 1 つ存在する．
- プレイヤタイルから鍵タイルおよびゴールタイルに到達可能である．
- ステージの上下左右は壁タイルに囲まれている．

特徴量設計　Diversity Sampling で用いる特徴量として以下の数値の組を用いた．

- プレイヤタイルの位置
- ゴールタイルの位置
- 鍵タイルの位置
- 敵タイルの数/3

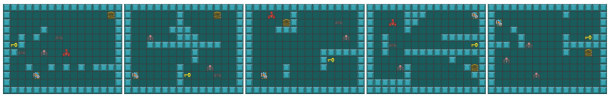


図 A·1　GAN の学習データとして用いる Zelda のステージ
Fig. A·1　Original Zelda levels used for training data.

A.1.2　Mario（VGLC）

概要　Mario は 2 次元の横スクロール型のアクションゲームである．プレイヤはステージの左端からスタートし，右端に到達することでゴールとなる．穴に落ちたり，敵に横から接触してしまうとゲームオーバとなる．本ステージは全体的には横長のステージとなっており，本研究では MarioGAN と同様に，分割した縦 14 マス，横 28 マスの部分を生成する．ステージは地面のタイル，空のタイル，敵のタイルなど全 10 種のタイルから構成されている．

学習データ　学習データとして，図 A·2 の 10 ステージを用いた．

制約条件　制約条件は以下のとおり．

- プレイヤ，鍵，ゴールマスはただ 1 つ存在する．
- ダイヤモンドマスは 10 個以上存在する．
- プレイヤ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．

特徴量設計　Diversity Sampling で用いる特徴量として以下の数値の組を用いた．

- 土管の数
- 下から 1 行目の空のタイル（穴になっている部分）の数 /3
- 敵タイルの数 /3
- 地面タイルの数 /10

A.1.3　Boulderdash（GVGAI）

概要　Boulderdash は 2 次元の迷路型のゲームであり，プレイヤは高さ 13, 幅 26 のグリッド状のステージ上を動き，ステージ上に存在するダイヤモンドを 10 個以上獲得してゴールのマスに到達することでクリアとなる．プレイヤは空マスは自由に移動することができるが，土のマス，岩のマス，壁のマスには移動することができない．土のマスはプレイヤが攻撃することにより破壊することができる．また，岩のマスは，下方向に空マスが発生すると下のマスに落下する性質がある．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．ステージは土のタイル，岩のタイル，敵のタイルなど全 10 種のタイルから構成されている．

学習データ　学習データとして，図 A·3 の 5 ステージを用いた．

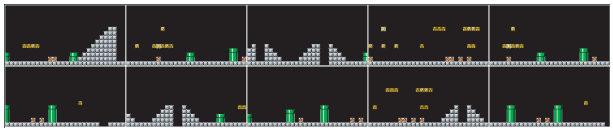


図 A·2　GAN の学習データとして用いる Mario のステージ
Fig. A·2　Original Mario levels used for training data.

図 A·3 GAN の学習データとして用いる Boulderdash のステージ
Fig. A·3 Original Boulderdash levels used for training data.

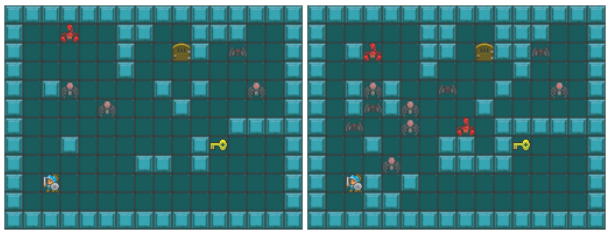


図 A·4 $F_{z_{hard}}$ で最適化を行った Zelda ステージの例．左側が最適化前，右側が最適化後
Fig. A·4 An Example of Zelda's level optimization with $F_{z_{hard}}$ for the model trained by the proposed method. The left level are generated from the initial values, and the right one are generated from the optimal values.

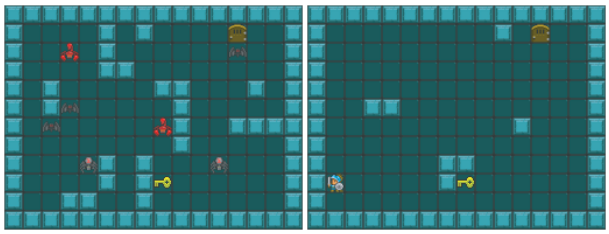


図 A·5 $F_{z_{easy}}$ で最適化を行った Zelda ステージの例．左側が最適化前，右側が最適化後
Fig. A·5 An Example of Zelda's level optimization with $F_{z_{easy}}$ for the model trained by the proposed method. The left level are generated from the initial values, and the right one are generated from the optimal values.

制約条件 制約条件は以下のとおり．

- 左端 4 列のいずれかの位置から右端 4 列のいずれかの位置まで到達可能である[*10]．
- 地面のタイルの下側に隣接するタイルは地面のタイルである．

特徴量設計 Diversity Sampling で用いる特徴量として以下の数値の組を用いた．

- プレイヤタイルの位置
- ゴールタイルの位置
- 敵タイルの数 /3
- ダイアモンドタイルの数 /5

A.2 潜在変数の最適化により獲得したステージの例

提案手法により学習したモデルに対し，各目的関数について CMA-ES による最適化を行った例の様子をそれぞれ図 A·4，図 A·5，図 A·6，図 A·7 に示す．図 A·4，A·5，A·6，A·7 より，提案手法により学習を行ったモデルでは，どちらの目的関数で最適化を行った場合も，敵や壁マスなどの数を狙いどおりに変化させることができており，目的に沿ったステージを生成することができている．

[*10] 文献 [3] では到達可能性は AI の Solver によるゲームプレイにより判定しているが，本研究ではプログラムによるヒューリスティックな判定を行っている．

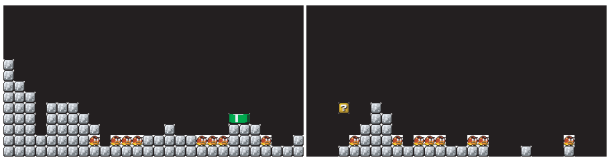


図 A·6 $F_{m_{hard}}$ で最適化を行った Mario ステージの例．左側が最適化前，右側が最適化後
Fig. A·6 An Example of Mario's level optimization with $F_{m_{hard}}$ for the model trained by the proposed method. The left level are generated from the initial values, and the right one are generated from the optimal values.

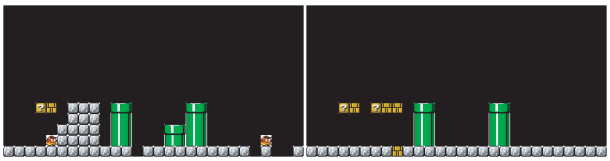
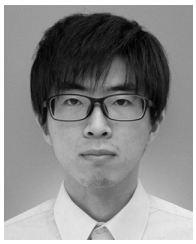


図 A·7 $F_{m_{easy}}$ で最適化を行った Mario ステージの例．左側が最適化前，右側が最適化後
Fig. A·7 An Example of Mario's level optimization with $F_{m_{easy}}$ for the model trained by the proposed method. The left level are generated from the initial values, and the right one are generated from the optimal values.



高田 宗一郎

1998 年生．2021 年電気通信大学情報理工学域卒業．同年電気通信大学大学院情報理工学研究科情報学専攻入学．主に深層学習によるゲームのステージ生成に関する研究に従事．

清 雄一 （正会員）

1981 年生．2009 年東京大学大学院情報理工学系研究科博士後期課程修了．同年（株）三菱総合研究所入社．2013 年より年電気通信大学．現在，同大学大学院情報理工学研究科准教授．博士（情報理工学）．エージェント，プライバシ保護技術等の研究に従事．2016 年度土木学会水工学論文賞，情報処理学会論文賞受賞．電子情報通信学会，日本ソフトウェア科学会，IEEE Computer Society 各会員．

田原 康之 （正会員）

1966 年生．1991 年東京大学大学院理学系研究科数学専攻修士課程修了．同年（株）東芝入社．1993〜1996 年情報処理振興事業協会に出向．1996〜1997 年英国 City 大学客員研究員．1997〜1998 年英国 Imperial College 客員研究員．2003 年国立情報学研究所着任．2008 年より電気通信大学准教授．博士（情報科学）（早稲田大学）．エージェント技術，およびソフトウェア工学等の研究に従事．日本ソフトウェア科学会会員．

大須賀 昭彦 （正会員）

1958 年生．1981 年上智大学理工学部数学科卒業．同年（株）東芝入社．同社研究開発センター，ソフトウェア技術センター等に所属．1985〜1989 年（財）新世代コンピュータ技術開発機構（ICOT）出向．2007 年より電気通信大学．現在，同大学大学院情報理工学研究科教授．2017 年より同大学大学院情報システム学研究科研究科長併任．2012 年より国立情報学研究所客員教授兼任．工学博士（早稲田大学）．ソフトウェア工学，エージェント，人工知能の研究に従事．1986 年度および 2016 年度情報処理学会論文賞，2013 年度人工知能学会研究会優秀賞，2014 年度同学会功労賞，2018 年度電子情報通信学会 ISS 活動功労賞受賞．IEEE Computer Society Japan Chapter Chair，人工知能学会理事，日本ソフトウェア科学会理事，同学会監事等を歴任．電子情報通信学会，人工知能学会，日本ソフトウェア科学会，電気学会，IEEE Computer Society 各会員．本会フェロー．