Neural Computing and Applications (2021) 33:19–37
https://doi.org/10.1007/s00521-020-05383-8

S. I : NEURAL NETWORKS IN ART, SOUND AND DESIGN

Deep learning for procedural content generation

Jialin Liu[1] · Sam Snodgrass[2] · Ahmed Khalifa[3] · Sebastian Risi[2,4] · Georgios N. Yannakakis[2,5,6] · Julian Togelius[2,3]

Received: 14 May 2020 / Accepted: 23 September 2020 / Published online: 8 October 2020

Abstract
Procedural content generation in video games has a long history. Existing procedural content generation methods, such as search-based, solver-based, rule-based and grammar-based methods have been applied to various content types such as levels, maps, character models, and textures. A research field centered on content generation in games has existed for more than a decade. More recently, deep learning has powered a remarkable range of inventions in content production, which are applicable to games. While some cutting-edge deep learning methods are applied on their own, others are applied in combination with more traditional methods, or in an interactive setting. This article surveys the various deep learning methods that have been applied to generate game content directly or indirectly, discusses deep learning methods that could be used for content generation purposes but are rarely used today, and envisages some limitations and potential future directions of deep learning for procedural content generation.

Keywords Procedural content generation · Game design · Deep learning · Machine learning · Computational and artificial intelligence

1 Introduction

Deep learning has powered a remarkable range of inventions in content production in recent years, including new methods for generating audio, images, 3D objects, network layouts, and other content types across a range of domains. It stands to reason that many of these inventions would be applicable to games. In particular, modern video games require large quantities of high-definition media, which could potentially be generated through deep learning approaches. For example, promising recent methods for generating photo-realistic faces could be used for character creation in games.

At the same time, video games have a long tradition of procedural content generation (PCG) [132], where some forms of game content have been generated algorithmically for a long time; the history of digital PCG in games stretches back four decades. In the last decade and a half, we have additionally seen a research community spring up around challenges posed by game content generation [16, 93, 112, 129, 133, 134, 148]. This research community has applied methods from core computer science, such as grammar expansion [22]; AI, such as constraint solving [115] and evolutionary computation [7, 133]; and graphics,

✉ Julian Togelius
julian.togelius@nyu.edu; julian@togelius.com

Jialin Liu
liujl@sustech.edu.cn

Sam Snodgrass
sam@modl.ai

Ahmed Khalifa
ahmed.khalifa@nyu.edu

Sebastian Risi
sebr@itu.dk

Georgios N. Yannakakis
georgios.yannakakis@um.edu.mt

1 Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China
2 Modl.ai, Copenhagen, Denmark
3 New York University, New York, USA
4 IT University of Copenhagen, Copenhagen, Denmark
5 Institute of Digital Games, University of Malta, Msida, Malta
6 Technical University of Crete, Chania, Greece

such as fractal noise [24]. But only in the last few years have we seen a real effort to bring the tools of deep learning to game content generation.

Deep learning brings new opportunities and leads to exciting advances in PCG, such as generative adversarial networks (GANs) [32], deep variational autoencoders (VAEs) [63] and long short-term memory (LSTM) [34, 45]. However, those methods for other generative or creative purposes are not always applicable to games and need certain adaptations due to the functionality criteria of different game content. Methods for generating images (e.g., generative networks) can be used to generate image-like game content (e.g., level maps, landscapes, and sprites). However, the generated levels should be playable and require specific gameplay skill-depth. The generated sprites should imply specific character or emotion, as well as coherence within the game. Training reliable models requires a necessary amount and quality of data, while the available data of content and playing experience for most games are limited. Careful consideration and sophisticated design of adaptation techniques are requisites for applying deep learning methods to generate game content.

It is important to note that content generation has uses outside of designing and developing games for humans to experience. In addition to creating content in games meant for humans to play, content generation can also play a crucial role in creating generalizable game-based and game-like benchmarks for reinforcement learning and other forms of AI [26, 136].

This article surveys the various approaches that have been taken to generate game content with deep learning and also discusses methods proposed from within deep learning research that could be used for PCG purposes. First, we give an overview of types of game content that could conceivably be generated by deep learning, including the particular constraints and affordances of each content type and examples of such applications (if they exist), followed by an overview of applicable deep learning methods.

2 Scope of the review

This article discusses the use of deep learning (DL) methods, here defined as neural networks with at least two layers and some nonlinearity [33], for game content generation. We take an inclusive view of games as any games a human would conceivably play, including board games, card games, and any type of video games, such as arcade games, role-playing games, first-person shooters, puzzle games, and many others. Several other surveys and overviews of PCG in games already exist. Here, we delineate the scope of our article by comparing it to existing books and surveys in Sects. 2.1 and 2.2. Section 2.3 describes our paper selection methodology.

2.1 Related work

A number of books and surveys of PCG with different focuses and aims have been published in the past two decade [16, 93, 112, 129, 133, 134, 148]. The two text-books for PCG [112] and Game AI [148] cover the search-based methods, solver-based methods, constructive generation methods (such as cellular automata and grammar-based methods), fractals, noise, and ad-hoc methods for generating diverse game content. De Kegel and Haahr [16] reviewed the PCG methods for eleven categories of puzzles, but few work based on deep learning has been reported. The article by Togelius et al. reviews the search-based PCG methods, defined as using meta-heuristics to search in a predefined content space, not necessarily represented by the same format of the content itself, and automatically generate new content [133]. The search is led by a fitness or evaluation function which measures the quality or playability of the generated content. The experience-driven PCG framework [147] largely adopts a search-based approach and reviews ways in which algorithms can generate content for adjusting the player experience. Most of the reviewed search-based methods in both survey papers rely on evolutionary algorithms. In this article, we also cover some search-based methods which cooperated with deep learning methods for generating content. The most famous example may be latent variable evolution [5]. Risi and Togelius [93] focuses on PCG for applications in Reinforcement Learning (RL), while the work based on RL methods reviewed in this article mainly used RL agents to play the generated levels, which indirectly served as content evaluators. Khalifa et al. [62] models the level generation as an iterative process that one needs to edit the levels to meet certain requirements or achieve some specific goals. RL agents need to learn to generate levels through this iterative process. The study of Summerville et al. [129], published in 2018, reviews the PCG via Machine Learning (PCGML) methods, building on e.g. Markov chains (e.g., [118–120, 131, 152]), n-grams (e.g., [14]), and Bayes nets (e.g.,[37]), whereas we will focus exclusively on deep learning in this article.

2.2 Novelty of the review

The differences between the current article and the PCGML survey [129] is that (i) our article focuses on DL-based methods, defined at the beginning of Sect. 2 (although other techniques will be mentioned for contrast); (ii) our article surveys more types of game content, such as narrative text and graphical textures; (iii) we also discuss applications of deep learning to support PCG, such as for content quality prediction; and (iv) our survey is written more than three years after the PCGML survey was first submitted and two years after it was published, during which time an avalanche of new work in the field has appeared.

During the two years after the publication of [129], PCG via deep learning has been growing quickly and a significant number of papers and articles have been published. The trend was mainly set by latent variable evolution [5] in 2018. A review of the state-of-the-art and the latest applications of deep learning to PCG is needed.

2.3 Paper collection methodology

To collect the related papers published or online since 2018, till end of August 2020, we have searched with Google Scholar and Web of Science using the following search terms ("*game*") *AND* ("*design*") and ("*game*") *AND* ("*procedural content generation*" *OR* "*pcg*"), separately. We systematically went through the returned papers, most of which were publications in the IEEE Transactions on Computational Intelligence and AI in Games (T-CIAIG), the IEEE Transactions on Games (ToG), in the proceedings of the IEEE Conference on Computational Intelligence and Games (CIG) series, the IEEE Conference on Games (CoG) series, the International Conference on the Foundations of Digital Games (FDG) series, the Artificial Intelligence for Interactive Digital Entertainment (AIIDE) Conference series and their related workshops, as well as special sessions at other conferences, such as the IEEE Congress on Evolutionary Computation (IEEE CEC). We also went through the papers that have been recently accepted in 2020 by the conferences mentioned above. Only work that involves direct or indirect use of DL-based methods for generating game content or evaluating content or content generators are reviewed in this article, while the ones being returned due to citations with the search terms but are out of our scope are not included.

3 Content types

Generally, game content can be distinguished from the content meant for non-interactive media by various forms of functionality constraints. Video, images, and music all require coherence, and in general that aesthetic suffers when the coherence fails. For example, GANs can often create images that are locally convincing but globally incoherent, such as a side-view of a car where the front wheels have a different size and style to the back wheels. This may be annoying to the human viewer, but the image still unmistakably depicts a car; it doesn't turn into a blur of random pixels just because the wheels on the car don't match. In contrast, when generating a game level, if the final door has no matching key the level is unplayable; the level's utility as content is not just slightly diminished, but essentially zero (unless manually repaired). Making a neural network learn to produce only functional content is often a tall task, and is one of the core challenges of using deep learning for PCG. Not all types of game content have the same extent of functional constraints however, and some offer affordances that may make content generation relatively easier. Also, not all content is *necessary*; depending on the game's design, there might be artifacts that are allowed to be broken, as the user can simply discard them and select others. Weapons in *Borderlands* are a good example of optional content.

3.1 Game levels

The most common type of content to generate in games is levels. These are spaces in two or three dimensions that need to be traversed. Typically, these are necessary rather than optional and have strong functional constraints that require them to be playable. For example, there cannot be impassable geometry (such as gaps or walls) blocking traversal of the level, items needed to finish the levels must be present, and enemies cannot be unbeatable. 2D, side-scrolling platform games is a genre where procedural generation is particularly common, both in entertainment-focused games (in particular indie games) and in academic research. Among the former, the standout game Spelunky has defined a way of building 2D platform games around PCG; among the latter, the Mario AI Framework [135], built around an open-source clone of Super Mario Bros, has been used in so many research projects that it could be called the "drosophila of PCG research". Another type of commonly attempted 2D level is the rogue-like or dungeon-crawler level, where the objectives and constraints are similar to the platform game level, but which are viewed from the top down so physics works differently. Related to this are levels for first-person shooters. Another kind of 2D level is the battle map, used in strategy games such as StarCraft or player-versus-player modes of first-person shooters. While such maps also have "hard" constraints, such as sufficient room for the players' bases, there are also the softer constraints of balancing; many features contribute to the quality of battle maps, but balancing is paramount.

Levels for music games, such as Guitar Hero or Dance Dance revolution, can be seen as 2D levels as well. Here the player is automatically moved along the level and has to carry out certain actions in time with the music, as prompted by level features. Some interesting work has been done on learning to create such music game levels from existing music (e.g., [21, 139]).

3.2 Text

Almost all games include some form of text, and typically they use text to convey narrative. This text typically has very strong constraints, as it needs to be truthful with regard to what happens in the game. For example, if the text says that the King lives in Stockholm, this must actually be the case lest it misleads the player. Traditionally, generative text in games has not been very ambitious and used simple text substitution or grammar-based approaches. Outside of games, deep learning has made great strides with LSTM networks [34, 45] and, more recently, transformers able to generate coherent and stylistically relevant text. However, these methods are not easy to integrate into most games because of the lack of control over deep learning-based text generators. However, games such as AI Dungeon 2 have managed to build gameplay on top of almost uncontrollable text generation.

3.3 Character models

Faces and character models are examples where deep learning has advanced content creation capabilities radically in recent years, but these methods have generally not made their way into games. Datasets of thousands of real human faces, such as the Celeb-A dataset [75], have become standard benchmark for developing new GAN variations, leading to some impressive breakthroughs in face generation. While many games have a need for (human) faces in various roles, including for freshly generated NPCs, the character design feature of role-playing games is a standout application case for controllable PCG, where machine learning-based methods have yet to make their mark. Depending on the features of the game, these faces or models might need to be animatable, so that they can produce believable movements or facial expressions.

3.4 Textures

Textures are used in almost all 3D games, and is perhaps the type of content that has the fewest functionality constraints. Procedural methods such as Perlin Noise [24, 89] have been used for texture generation in games since the birth of commercial 3D games with *DOOM*. Deep learning methods for texture generation could provide a viable alternative in this case.

3.5 Music and sound

Most games feature a soundtrack, often composed of both music and sound effects. The constraints on the soundtrack tend to be relatively soft compared to other types of content constraints; the sound effects should be appropriate to the actions in the game at any given moment, and the music to the emotional tone of the moment, but inappropriate sound does not necessarily break the game. Quite a few games involve some kind of procedural soundtrack, and some research projects have focused on music generation able to adapt to affective shifts in real-time [106]. At the same time, deep learning has made impressive strides in learning to generate music with some modes of controllability [18], but we have yet to see the use of deep learning methods for sound generation in games.

4 Training methods and neural architectures of DLPCG

Due to the different types and roles of content in games, diverse deep learning methods have been adapted for PCG. In this section, we present different ways to apply deep learning for PCG systems, the target content, and their generality. The approaches are categorized by the type of machine learning method used for training. Additionally, works combining evolutionary computation techniques to deep learning methods are also presented. The works reviewed in this section are summarized in Fig. 1, categorized by the content types and deep learning methods.

Generating different types of content often requires different types of neural architectures. In the use cases reviewed in Sects. 4.1 and 4.2, LSTMs are mostly used for time-dependent sequential data (e.g., action sequences, agent paths, charts for rhythm) and language models, while convolutional neural networks are often used for any type of image-like content. A very popular class of architecture for content generation are GANs [32]. A GAN consists of two networks, a generator and a discriminator that are trained iteratively to allow the generator to create more realistic content, while the discriminator is getting better at distinguish generated content from real data.

4.1 Supervised learning

Supervised learning (SL) methods have been used in a variety of ways for content generation. Often as a predictor, SL models predict the gameplay outcomes of games with the generated content, either for evaluating the quality of content or for meeting specific preferences (such as game

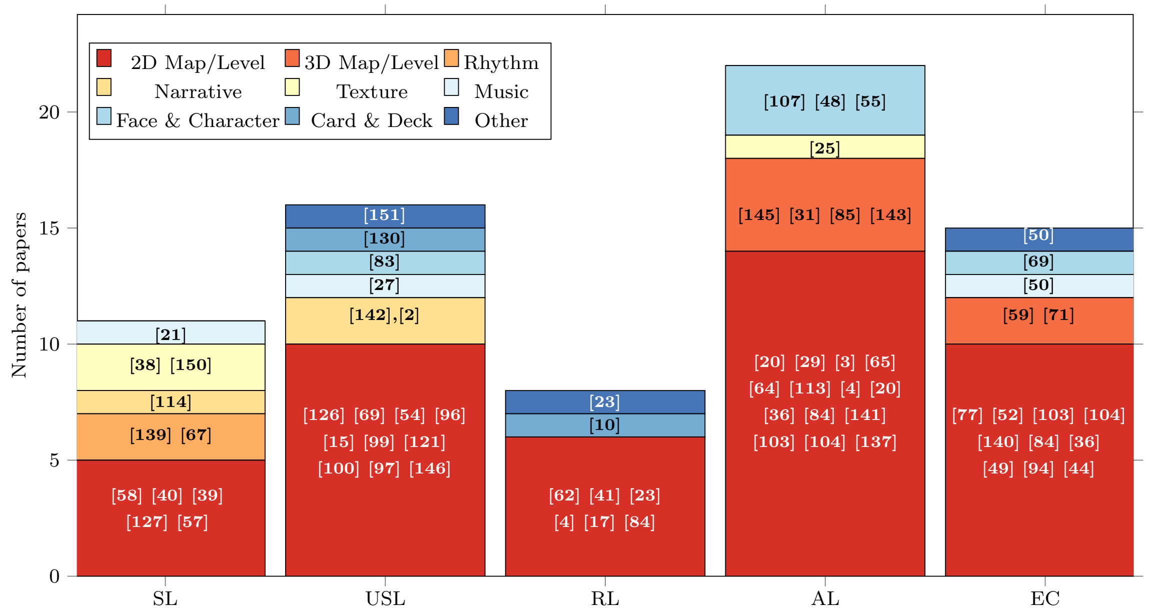

Fig. 1 This figure shows the distribution of research by methods and content types. We notice the disproportionately large amount of work on 2D level and map generation compared to all other content types

style, image style and color) or adapting the generated levels to desired skill-depth.

The study of Summerville et al. [127] extracted player paths in Mario from gameplay videos and used them to annotate training levels. Then, separate LSTMs are trained on levels annotated with different players' paths in order to generate personalized levels based on the players' chosen paths [127]. Then, Guzdial et al. [40] trained a random forest on expert-labeled design patterns from Mario levels (i.e., small sections of levels given descriptive class labels) to classify level structures and an autoencoder with level structures and labels as input to generate new instances of those design patterns.

Karavolos et al. [57] trained a CNN to predict the outcomes of a simplified 3 versus 3 multiplayer deathmatch shooter game to evaluate and determine whether the levels, represented by maps and weapon parameters, are balanced or favor a team. Based on the outcome predictor from [57], Karavolos et al. [58] further designed a DL surrogate model for pairing levels and character classes for desired game outcomes.

Tsujino and Yamanishi [139] represented rhythm-based video game levels by charts and implemented Dance Dance Gradation (DDG), a system with LSTMs trained on levels of different degrees of difficulty to generate new levels. DDG can tune the difficulty degree of generated charts by changing the fractions of easy or hard charts used to compose the training dataset [139]. Liang et al. [67] used C-BLSTM [105] to generate levels of rhythm games, represented by actions and corresponding timing, of different difficulties, trained on the beatmaps collected from *OSU!*, a famous rhythm game.

Beside considering skill-depth required in game levels, the emotion sent by content has also been studied. Guzdial et al. [38] studied the emotion shown by the game visuals, such as abstract texture, color of game maps and scene, including the visual effects, and trained a CNN to generate textures for some given target emotion.

Soares and Bulitko [123] trained a VAE [63] to classify NPC behaviors to Leaders, Followers, and Random, in a simple artificial life environment. Sirota et al. [114] trained two RNNs, a speaker and a listener, by playing a referential game with concepts and human-generated annotations to design communication systems for NPCs in games.

4.2 Standard unsupervised learning

Most unsupervised learning (USL) techniques in PCG focus on learning a representation of all the content and then sample new content from this representation. For example, using autoencoders to learn to replicate game levels. Another direction usually taken is transforming the data into a sequence and use unsupervised learning to learn the relation between these elements similar to Markov Chains relations. For example, learning from a text corpus how to predict the next word based on the previous ones.

Summerville and Mateas [126] trained LSTMs on Mario levels annotated with agent paths by representing the 2D levels as one-dimensional strings of tiles. Jain et al. [54] trained autoencoders on sliding-window segments of Super Mario Bros levels, which were represented by 2D arrays, to generate and repair levels. Jain et al. [54] considered a tile as being empty or occupied, but has inspired many follow-up investigations. Blending has lead to new and creative game levels. Sarkar and Cooper [96] trained separate LSTMs on two different game domains (Mario and Kid Icarus) and generated new blended level sections with alternating generators. Sarkar et al.[99] further explored generating blended levels by training variational autoencoders and GANs on Mario and Kid Icarus, and generating new blended level sections that interpolate between the domains using the latent vectors. Snodgrass and Sarkar [121] also used VAEs to model and generate platformer level structures which was finished by using a search-based approach to blend details from several other games. Sarkar et al. [100] explored two variants of VAEs (linear are GRU) for blending platforming game levels and associated paths in those levels. Sarkar and Cooper [97] trained VAEs to learn a sequential model of level segment generation and a random forest classifier to determine the exact location of a newly generated segment to the previous segment (an ancestor). The resulted levels are not only more coherent [97], but also more creative [98] because of the changing altitude of platformer and various possible heading directions. Yang et al. [146] trained Gaussian Mixture VAE to learn relation between game level segments from various games (Super Mario Bros, Kid Icarus, and Megaman) and later be able to generate level segments that follow a certain distribution/style. Davoodi et al. [15] trained an autoencoder to repair manually designed levels for different games by re-iterating it over the decoder while using a trained discriminator from a GAN model to determine the stopping criteria. Besides levels, autoencoder has also been used to generate 3D shapes [151].

Moreover, USL methods for image generation have also been applied to generating sprites and characters in games. The recent work by Mordvintsev et al. [83] learned cellular automata (CA) to imitate the development of organism and generate images, represented by 2D grids of cells. A cell is similar to the tile considered in the MarioGan [140] (explained later in Sect. 4.5). A cell contains a cell state (e.g., a discrete value or a vector of RGB values), while a tile contains a discrete value which refers to an object type or part of it.

Applications of USL methods to content generation for card games and text adventure games have also been investigated. An example is [130]. Summerville and Mateas [130] trained encoding and decoding LSTMs on Magic: The Gathering cards, represented as sequences of tokens corresponding to the important information on the cards (e.g., mana cost, effect, power, etc.). The LSTMs were trained on corrupted versions of the cards, and encoded cards were used as input to the decoder at generation time. Another example is the endless text adventure game *AI Dungeon 2*[1] (earlier version as *AI Dungeon*). AI Dungeon 2 is built on OpenAI's GPT-2 model [92], a 1.5B parameter Transformer, and fine-tuned on some text adventures obtained from https://chooseyourstory.com, according to its developer Nick Walton [142]. In a game, a player can interact with the game by inputting text commands, and then the AI dungeon master will generate content of the game environment (updates in the game story) according to the commands and provide text feedback. By doing so, each player can build his/her own unique game story. Ammanabrolu et al. [2] focused on procedurally generating interactive fiction worlds and proposed AskBERT to construct knowledge graph. AskBERT extracts objective information in the game worlds, such as characters and objects, via question-answering model. Ferreira et al. [27] proposed *Bardo Composer*, a system that automatically composes music for tabletop role-playing games. In *Bardo Composer*, a BERT model cooperates with a stochastic bi-objective beam search model to identify music emotion, and then generate music pieces that reflects the identified emotion.

4.3 Reinforcement learning

Using reinforcement learning (RL) for PCG is a very recent proposition which is just beginning to be explored. Here, the generation task is transformed into a Markov decision process (MDP), where a model is trained to iteratively select the action that would maximize expected future content quality. This transformation is not an easy task and there is no standard way of handling it.

One of the early projects that uses RL is by Chen et al. [10]. They used a small network of one hidden layer to generate a hearthstone deck of cards that can beat a specific other deck given a certain player. The agent can modify the current deck by substituting any of its cards with a different one. The goal is to maximize the win rate of the playing agent using the current deck against a predefined deck.

Earle [23] used RL to play the game of SimCity (Maxis, 1989). They used a fractal network (convolutional network with structured skip connections) as their network architecture and optimized it toward increasing the city population. At each step, the agent can change any space on the map to any other type. This project is a borderline example of PCG. The aim of the project was to play the game of SimCity where the trained agent will learn to be a city planner/generator.

[1] https://github.com/AIDungeon/AIDungeon.

[2] https://github.com/amidos2006/gym-pcgrl.

As we can see, most of the RL PCG requires an adaptation for the input to be able to be used during generation. Khalifa et al. [62] introduced a framework[2] for 2D level generation using RL. The generation process is framed as an iterative process where at every step the generator modifies the level toward certain goals (based on the current generation problem). They proposed 3 main transformations: Narrow, Turtle, and Wide. These transformations focus on the different ways that the generator controls where it is modifying. Figure 2 shows examples of the generated levels over three different problems using trained agents in the PCGRL framework.

4.4 Adversarial learning

Adversarial learning (AL) models are perfect for generating content represented by pixel-based images or 2D array of tiles, such as levels as a map, landscapes and sprites. The most popular model among the reviewed works would be GAN [32] and its variants.

2D levels of most arcade games can be simplified as 2D arrays of tiles, where each tile contains a type of object or part of an object. Examples include the levels designed using Video Game Description Language (VGDL) [101] in the General Video Game AI platform [87, 88], and the tile-based levels in the Mario AI framework [110]. As shown in the top-left of Fig. 4, each tile contains a type of object or part of it, such as ground, pipe, empty and enemy, represented either by a symbol or an integer. Kuang and Luo [64] implemented an interactive map designing system using different generative models to generate 2D maps, which can be further extended to 3D scenes. Torrado et al. [137] designed a new GAN architecture, Conditional Embedding Self-Attention GAN (CESAGAN), to tackle the low quality and diversity issue of generated 2D levels by traditional GANs, and increased the amount of training data to CESAGAN with a bootstrapping technique. They applied their technique to *Zelda*, a dungeon crawler game from GVGAI [87].

To facilitate the input form for generative models, such as GANs, 3D landscapes are often converted to 2D height map. Wulff-Jensen et al. [145] trained a deep convolutional GAN (DCGAN) on digital elevation maps sampled from the Alps dataset to generate 2D height maps as input to Unity for creating 3D landscapes for video games. Giacomello et al. [31] converted each 3D DOOM level to several 2D images, among which a *HeightMap* was used to indicate the 3D information and other were top-down images of the corresponding level. In [31], two GANs were trained on human-designed levels, one of which took plain 2D images as input and the other used both the images and

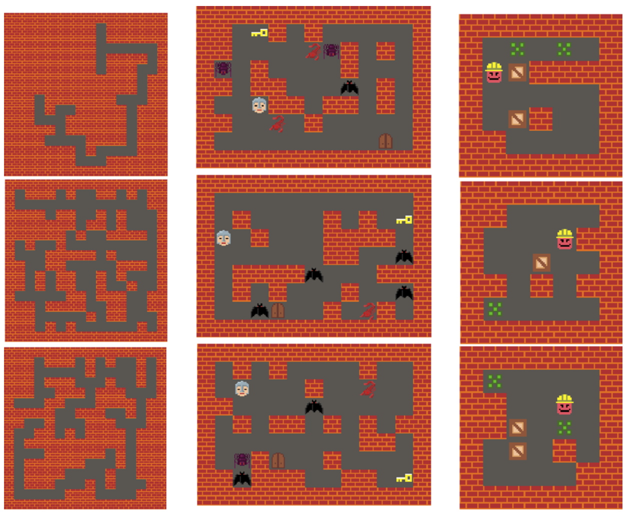

Fig. 2 Generated examples from three different problems using PCGRL environment introduced by Khalifa et al. [62]

some of the extracted features. Park et al. [85] trained a multistep DCGAN, adapted from [140], to generate levels of an educational game, ENGAGE. The levels were represented by a 2D array of tiles, from a top-down view, during training and creation, and then converted to 3D levels to be used in the game [85]. Volz et al. [141] explored the use of GANs in the context of match-3 levels, attempting to model the local and global structures of those levels. Awiszus et al. [3] proposed token-based oneshot arbitrary dimension generative adversarial network (TOAD-GAN), adapted from SinGan [108], trained on a single sample level, to generate tile-based levels. In the work using GANs for level generation that have been reviewed so far, game levels are tackled as image only during training, while the constraints for validating levels are not considered at all. Recently, Di Liello et al. [20] presented constrained adversarial networks (CANs) which encourages the generator to learn to generate valid levels by penalizing it due to invalid structures generated during training. But still, these methods generate individual segments of platformer levels separately and then combine them together randomly or according to some increasing level difficulty [140]. Different from above work, Fontaine et al. [29] proposed latent space illumination (LSI), which uses quality diversity algorithms, such as Covariance Matrix Adaptation MAP-Elites (CMA-ME) [28], to search the latent space of trained generators, aiming at increasing the diversity of generated levels. A recent work by Kumaran et al. focused on generating levels in multiple distinct games. Instead of training several GANs for these games separately, a novel GAN architecture, composed of a branched generator and multiple parallel discriminators, was proposed [65].

[2] https://github.com/amidos2006/gym-pcgrl.

Besides generating 2D and 3D levels represented as pixel-based or tile-based images, texture [25] and sprite generation [48] have also been investigated. Hong et al. [48] generated 2D image sprites using a multi-discriminator GAN, in which two encoders were used for bone graph, shape and color, without sharing parameters. Additionally, two discriminators, one for shape and the other for color, were used in [48] to generate sprites' skeletons and color, respectively. Another potential application is GAN-based character generation [55] for video games, such as The Sims (Maxis, 2000). Wang and Kurabayashi [143] proposed Sketch2Map to generate 3D terrains from sketches. Sketch2Map used a conditional GAN (cGAN) to convert a sketch into an elevation bitmap, which is interpreted to generate the practical terrain asset by a deterministic algorithm [143].

More recently, Bontrager and Togelius [4] proposed a new training method similar to GANs, where the network consists of two parts: generator and agent. The generator is trying to generate new playable levels adapted to the agent's strength, while the agent plays the game and reports how playable it is and how hard it is to play. Similar to GANs, the agent will try to improve itself by playing the new generated levels, while the generator will improve itself based on the agent performance on its generated levels. In this work, RL is used to play the generated content and not to generate the content; an RL agent interacted with the generative model to create levels adapted to the agent's playing strength.

4.5 Evolutionary computation

There is a long tradition of using evolutionary computation (EC) approaches for training (deep) neural networks. While these are sometimes not regarded as DL, the standard definition of DL does in fact not reference gradient descent. Most evolved networks are deep, and architectures created by evolutionary algorithms such as NEAT [124] often have multiple layers and recurrent components [102].

For example, Hoover et al. [51] represented game levels as functional scaffolding for musical composition voices [49]. Taking Mario as an example, each level is presented by a set of voices with the size of possible tile types in a level. Each voice is a one-dimensional array of the same length of the level, in which each element indicates the vertical position of the tile if it presents on the corresponding column, otherwise 0. Neural networks were trained and evolved through neuroevolution of augmenting topologies (NEAT) [124] to suggest placements of tiles in Mario levels [51].

Hoover et al. [50] evolved CPPNs through NEAT for generating both audio and visual content in the game AudioInSpace. Risi et al. [94] evolved and trained CPPNs with NEAT to generate flower images for a flower-breeding video game *Petalz*.[3] The CPPNs of different flowers can be mated to generate new flowers.

Evolutionary computation techniques have also been combined with unsupervised DL methods for generating new content. A prominent example is the Deep Learning Novelty Explorer (DeLeNoX) [69]. DeLeNoX alternates phases of content exploration and content transformation for the generation of spaceships, depicted as 2D black and white images (Fig. 3). In the exploration phase, constrained novelty search seeks maximally diverse artifacts and generates a training set. In the transformation phase, a deep autoencoder learns to compress the variation between the found artifacts into a lower-dimensional space. The newly trained encoder is then used as the basis for a new fitness function, transforming the search criteria for the next exploration phase [69]. The process continues repeating exploration and transformation phases thereby iteratively refining and complexifying the generated outcomes.

Arguably one of the most popular examples of EC for DLPCG is the aforementioned Latent Variable approach [5], which combines unsupervised learning in the form of a GAN/VAE with evolutionary computation to search for content in the learned space of a GAN/VAE. Originating from synthesizing new fingerprint [95], in the context of games this approach has been employed to generate Super Mario Bros and Zelda levels [104, 140].

In the work of Volz et al. [140], a DCGAN [91] is trained on a set of level segments of Super Mario Bros represented by 2D array of tiles, and then latent variable evolution (LVE) [5] is applied to search for levels that are more playable and encourage particular behaviors evaluated by the games simulated by an A* agent. The overview process is illustrated in Fig. 4. The resulted framework, MarioGAN [140], certainly identified a new and creative way of generating game content. However, two issues have been observed: (i) broken pipes occur in some of the level segments generated by GANs, and (ii) the segments were connected directly in an arbitrary order to build complete levels, while how to combine segments to make the resulted levels more structured and organized was not exploited (Fig. 5). To tackle the former issue, Shu et al. [113] trained a MLP model to learn the surrounding information of tiles and detect wrong tiles in the generated segments (e.g., Fig. 6). An evolutionary repairer is designed to search for optimal replacement tiles for fixing the broken pipe [113]. To tackle the latter issue, a graph grammar was used to combine rooms of Zelda generated by a GAN into dungeons [36], and Schrum et al. [104] proposed CPPN2GAN which used Compositional Pattern

[3] https://www.facebook.com/Petalz-238904402867390/.

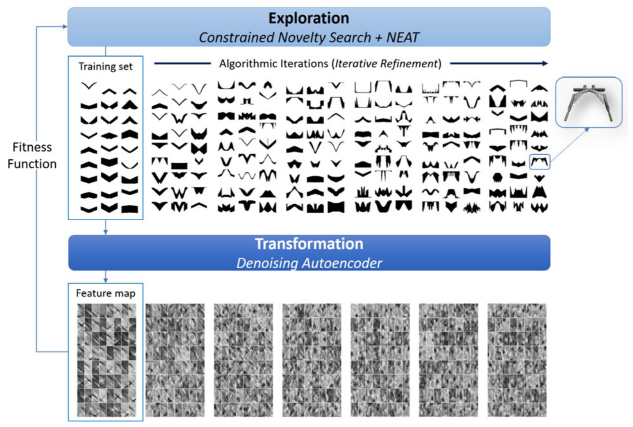

Fig. 3 The key phases of DeLeNoX for the autonomous generation of content [69]. DeLeNox adopts the principles of exploration (realized via constrained novelty search), transformation (realized via deep denoising autoencoders) and iterative refinement (realized through the increasing complexity of NEAT architectures). Image reproduced with authors' permission

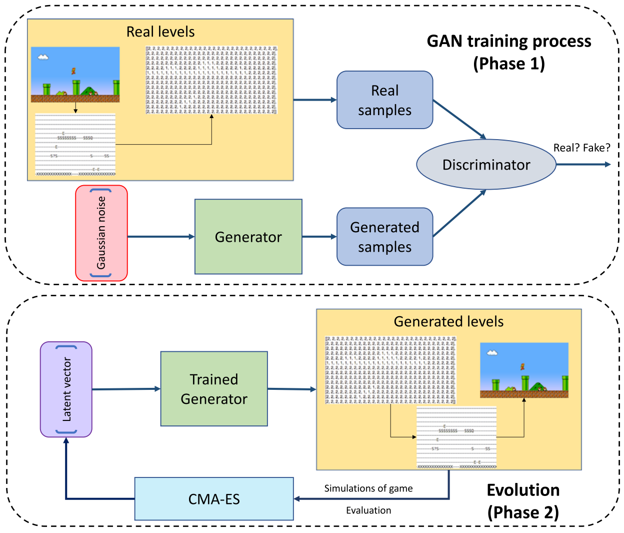

Fig. 4 Overview process of MarioGan [140], reproduced with authors' permission

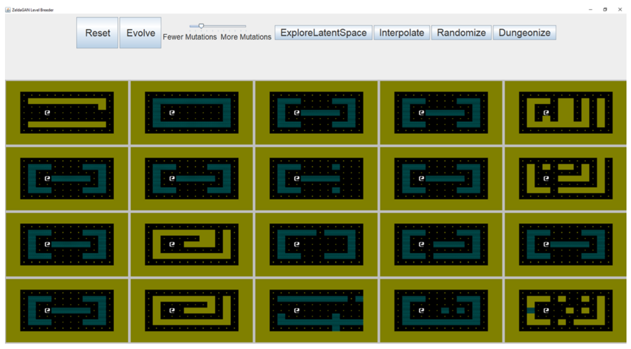

Fig. 5 Screenshot of interactive evolution interface in [103], reproduced with authors' permission

Producing Networks (CPPNs) to organize level segments generated by GANs into complete levels.

Inspired by [140], Irfan et al. [52] applied LVE and trained DCGANs on randomly generated levels of 3 single player games from the GVGAI framework [87], Freeway, Zelda and Colourescape. Based on the work of [140], Mott et al. [84] designed a new fitness function for CMA-ES as a weighted sum of the number of frames that an action is feasible, the fraction of agents that completed a level and the largest fraction to control the difficulty of generated levels. The weights are evaluated and tuned via the human playing tests performed on the levels generated using the corresponding fitness function [84].

Evolutionary methods for content generation can also be combined with user feedback, such as through interactive evolutionary computation (IEC), in which human evaluation is used instead of the fitness evaluation by a simulator. For example, Hastings et al. [44] used CPPNs to represent weapons in a multiplayer video game Galactic Arms Race.[4] The CPPNs are evolved during the game playing with the preferences abstracted from the past playing of players. IEC combined with LVE can allow users to breed their own game levels, such as Zelda and Mario [104]. Based on [36, 140], a mixed-initiative tile-based level design tool was implemented by Schrum et al. [103], which allows human to interact with the evolution and exploration within latent level-design space (interface illustrated in Fig. 5), and to play the generated levels in real-time.

EC methods can also collaborate with human to generate and evaluate or repair game content. Liapis et al. [71] presented *Sentient World* tool which allows interactions with human designers and generates game maps using Neuroevolution via novelty search. *Sentient World* can generate high resolution maps based on the rough terrain sketches drawn by designers, as well as the iterative refining via selection and editing options opened to designers.

Karavolos et al. [59] generated levels of a first-person shooter (FPS) game with targeting gameplay outcomes, in which a genetic algorithm is used to generate levels of specific fitness values based on the predicted outcomes by a CNN trained on simulated matches.

5 Using deep learning to evaluate content and content generators

Evaluating content generators is not a trivial task. Much of the ML and DL-based PCG work has focused their evaluations on the generated content and used those evaluations as proxies for evaluating the generator itself. However, the

[4] http://gar.eecs.ucf.edu/.

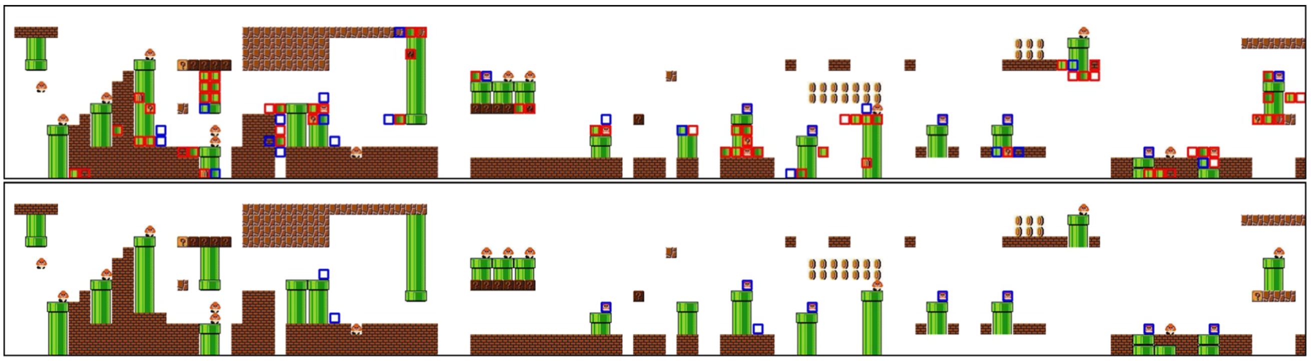

Fig. 6 Top: A MLP model trained on human-designed levels labels wrong tiles (in red rectangle) and unsure tiles (in blue rectangle) in a segment. Bottom: Segment fixed by an evolutionary repairer assisted by the trained MLP model [113]. Images reproduced with authors' permission

computational creativity community has identified that in order to get a full picture of the generator (or creative program) the process by which the output content is created should be evaluated as well. Colton [11] Jordanous [56] Pease and Colton [86] each propose frameworks and methodologies for evaluating the creativity of the process of a generator. Smith and Whitehead [116] (later expanded on by Summerville [125]) proposed methods for holistically evaluating a content generation approach, by evaluating large swaths of generated content to get a broader understanding of the generative space of a content generator and its biases within that generative space. Summerville [125] focused on ML-based generators and proposed approaches for highlighting the shortcomings and strengths of a generator through methodically highlighting generated artifacts (e.g., artifact most similar to an artifact in the training set).

In this section, we survey uses of deep learning for content generation in an *indirect* fashion. In particular, we list studies (cf. Fig. 7) that consider deep learning for testing or evaluating game content through the analysis of generated content (Sect. 5.1), construction of human-like playing bots (Sect. 5.2), or the construction of reliable models of player experience (Sect. 5.3). We additionally highlight which of these approaches focus on evaluating the generator itself instead of only the content.

5.1 Analyzing content

Statistical measures on the generated content and similarity measures based on the content used in training set (e.g., [77]) can give insight into the generative space of a content generator and its biases within that space. Statistical measures can be used to compare the distribution of generated content to the distribution of the training set [125]. Similarity measures can also be specifically designed for this task. For example, Lucas and Volz [77] compared occurrences of small structures in the generated set to their presence in the training set to measure similarity.

Many similarity and statistical measures suffer from the same drawback of only measuring what is quantifiable. Recent approaches in deep learning can help avoid this drawback by learning latent semantic features of the content. Recent work has developed approaches to style transfer [61, 74] by traversing the learned latent space of the model, and others have analyzed the learned latent space of their models to find semantic meaning in the features [1]. These advances have led to the use of latent space-based distance and similarity measures [144]. Leveraging the latent space learned by a model to create similarity measures between pieces of content might allow us to develop more semantically meaningful similarity measures in addition to the statistical measures currently in use. As an indicative example of such a research direction, Isaksen et al. [53] categorized tile-based 2D game levels with semantic hashing based on autoencoders. The proposed approach [53] can be used to categorize the generated level segments or rooms and group the ones sharing similar styles to build a complete game level or dungeon.

5.2 Playing content

In this section, we review methods based on ANNs and DL for reliable playtesting which can be used, in turn, to evaluate game content generators in an indirect fashion. Simulated playtesting [46, 47, 140] of generated content can give quick insights into the features of the content and the generative space of the content generator [116, 125]. Guzdial et al. [42] propose the use of deep reinforcement learning agents for simulated playtesting as a way of creating more human-like playtraces. Guzdial et al. [42] specifically focus on deep RL agents for Mario, where human-like control is simulated by giving the agent imprecise controls via stochastic effects on actions. Similarly, Min et al. [82] designed a goal recognition

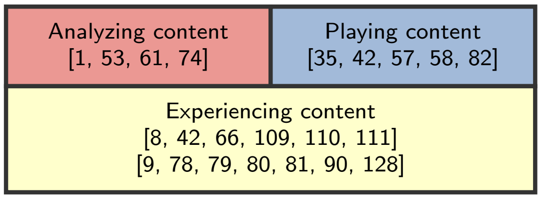

Fig. 7 Summary of the works that focused on analyzing, playing or experiencing generated content

framework based on stacked denoising autoencoders for open-ended games, which can be used to personalize games for different players according to their actions.

CNN to predict the outcomes [57] trained a CNN to predict the outcomes of a simplified 3 versus 3 multiplayer deathmatch shooter game to evaluate and determine if the levels, represented by maps and weapon parameters, are balanced or favoring a team. Based on the predictor for the same deathmatch shooter game, Karavolos et al. [58] further designed a DL surrogate model for pairing levels and character classes for desired game outcomes. Gudmundsson et al. [35] imitated the behavior of human through SL and performed experimental study on non-deterministic puzzle games *Candy Crush Saga* and *Candy Crush Soda Saga*. A CNN was trained on human player data and then used to predict the action that human players most likely to select when playing levels that were unseen during training [35]. This approach can be used to measure metrics such as the diversity of actions to evaluate generated new levels. Notice each of these methods focuses on evaluating the generated artifacts, but can be expanded to more broadly evaluating the generator itself if the results of artifact evaluations are used to stratify the generative space or further explore the biases and capabilities of the generator.

5.3 Experiencing content

Human user trials and surveys can provide the most useful insight into the less quantifiable (i.e. subjective) features of the content and the generation process, such as the human-perceived quality of the generated content over time. A large volume of studies focus on the use of deep learning for modeling aspects of player experience which can be used, in turn, to evaluate the content that is generated and experienced by the player. Player experience is usually provided as annotated labels (ratings or ranks) or even continuous traces via crowdsourcing. Running user evaluations and crowdsourcing labels of subjective aspects such as experience, however, can be a laborious task which may not be feasible if what is desired is the quick iteration on the generative system. One approach for further leveraging the output of a user evaluation is to treat the user evaluations as features to be learned. Larsson and Petri [66] trained neural networks using NEAT to predict the user rating of user-created StarCraft maps. This approach [66] can be extended to evaluate generated StarCraft maps.

Within the platformer genre, a series of studies by Shaker et al. [109–111] investigate the use of DL models of player experience for the generation of experience-tailored Super Mario Bros levels. Camilleri et al. [8] view a player's believability as a content generation problem and used various forms of deep networks to infer the mapping between game content, gameplay and believability in a Super Mario Bros variant. The networks of that study predict the degree to which a combination of gameplay behavior and a generated level can be considered believable. Guzdial et al. [42] trained a CNN to predict rate of the difficulty, enjoyment and aesthetics of game levels and performed case studies on Infinite Mario Bros, which was further enhanced by the features extracted from search history of an A* agent. Similarly, Summerville et al. [128] used a regression model on a large set of statistical measures to find measures that predict those same human evaluations of Mario levels. More recently, Pfau et al. [90] proposed deep player behavior modeling (DPBM) with a multilayer perceptron (MLP) trained on behavioral data and game observation to map game states to action probabilities. All aforementioned approaches can be used, for instance, to evaluate generated levels.

The first application of CNNs for modeling player experience is introduced by Martinez et al. [80]. CNNs in that study consider and fuse the content of a 3D maze prey–predator game and the in-game behavior of the player [79] and predict reported ranks of player experience via use deep preference learning. Looking at the challenge of player affect modeling by solely focusing on gameplay, Makantasis et al. [78] used various CNN models to predict the level of arousal of survival shooter games directly from the pixels of gameplay in a general player-agnostic fashion. Thus CNNs map between gameplay behavior and game content as represented by pixels—such as in-game play features and UI elements. In principle, such surrogate models of arousal can be used directly and evaluate video content of any game within the survival shooter genre. In a similar recent study, various types of neural networks have been trained to predict the continuous viewer engagement of PUBG streamed games on Twitch [81]; the engagement models obtained are highly accurate and general across different streamers. Camilleri et al. [9] took player experience modeling to the next level and built models that are general across many different games. The models are built on simple 1-hidden layer networks indicating the potential of the methodology with larger DL representations for the general evaluation of the experience of game content across games. Similar to the previous section, each of these methods is predominantly used to evaluate content. However, using these methods to evaluate large samples of content from a generator can enable a meta-analysis of the types of content a particular generator tends toward creating.

6 Discussion and outlook

The combination of deep learning and PCG in games is beneficial for both game research—as deep learning enhances our capacity to generate content—and deep learning research since games pose challenging problems for deep learning to solve. Deep learning opens new opportunities for the autonomous generation of content of any type and has a plethora of use cases within games. As we saw throughout this article, deep learning may serve as a content generator, as a content evaluator, as a gameplay outcome predictor, as a driver of search, and as a pattern recognizer for repair and style transfer. This section surveys the areas with a particular importance for the current and future use of DLPCG in games with an emphasis on mixed-initiative generation, style transfer and breeding, underexplored content types, learning from small datasets, orchestrating different content types within a game, and generalizing generation across games.

6.1 Mixed-initiative DLPCG

Autonomous PCG systems, including the cases where the initiative of the human designer is limited to algorithmic parameterizations [148], can hardly generate content with target quality or features. Recently, more and more work takes into account the preferences or input of designers or players in different ways while generating content. Mixed-initiative PCG [149], formally defined as "the process that considers both the human and the computer proactively making content contributions to the game design task" [148], offers a more controllable and practical design process that may involve the use of DLPCG algorithms but their use is limited so far.

Level generation in games, as a popular application of mixed-initiative DLPCG, requires some initial specifications (i.e. the initiative) from the designer—e.g. in the form of sketches [43]—to assist the design process. A popular example of the mixed-initiative paradigm is the shallow neural network model presented in [70] which generates game strategy maps based on the terrain sketches drawn by designers. The map generation feature of Sentient Sketchbook features neuroevolutionary search which is driven by design objectives and the novelty of the map. Moving from level to image generation, Serpa and Rodrigues [107] adapted the GAN-based Pix2Pix architecture to generate both gray and color pixel art sprites from sketches using a single network.

Taking platform games as the domain under investigation, Guzdial et al. [39] developed a mixed-initiative Super Mario Bros level design tool that leveraged several existing PCGML techniques, including Markov chains [117], LSTM [126] and Bayes Net [37], to assist the user in creating levels. user in creating levels. [39] gathered data on how the users interacted with the models in the tool, and trained a CNN on that collected data. This CNN was then used to better predict and generate level sections along with the user. Later, Guzdial et al. [41] used the trained CNN with active learning based on the user current interaction to generate levels for Super Mario in a mixed-initiative fashion [68, 149]. Recently, Schrum et al. [103] have allowed the designers to change manually the latent vectors of the trained generative model or define the mutation strength of their evolutionary generator for tile-based 2D levels. Delarosa et al. [17] presented *RL Brush*, a human-driven, AI-augmented design tool also for tile-based 2D levels, in which RL-based models have been used to enhance human design with suggestions generated by PCG methods.

6.2 Style transfer, breeding and blending

Most style transfer methods and generative models for image, music and sound [6] can be applied to generate game content. So far, only a few work focused on the style transfer for game content (e.g., [71, 107, 150]). Liapis et al. [71] generated game maps based on the terrain sketches, and Serpa and Rodrigues [107] generated art sprites from sketches drawn by human. However, a number of diverse input sketches to these two work can also be generated using deep learning approaches based on a single human sketch [43]. Moreover, algorithms and techniques designed for image generation can often be adapted to the automatic generation of faces and sprites in games. For instance, Yoo and Kim [150] applied a neural styling algorithm [30] to change artistic style of graphics in a strategy game *Hedgewars*.[5] Another example is *ArtBreeder*,[6] which contains several generative models for creating new images by image breeding, among which, the models for portraits and anime-style faces, can be used to generate comic or video game characters and the one for landscapes can be used to generate background images for games. Blending levels from different games has recently gained more attention from the research community, with much recent work focusing on blending platformer levels. Sarkar and Cooper [96, 99] trained separate models on two different

[5] http://www.hedgewars.org/.

[6] https://artbreeder.com/.

games, and then blended new levels using these trained models via interpolation or alternation. Snodgrass and Sarkar [121] used VAEs to generate level structures, and a search-based approach to blend details from various platformers, while Sarkar et al. [100] directly trained VAEs on levels from several platforming games and interpolated the latent vectors between domains for blending.

6.3 Underexplored content types

Most of the reviewed works focus on the design of content that can be represented by 2D images of tiles or pixels, such as 2D levels, landscapes and sprites (cf. Sect. 4). Only a few of them considered text and narrative generation, music and rhythm generation, weapons generation for FPS, etc.

In the research we have surveyed, platformer and dungeon-like games (e.g., arcade games, FPS games and adventure games) are clearly over-represented. In particular, Super Mario Bros and Zelda are usually used for testing the GAN-based level generation approaches.

However, the types of games are not limited to arcade games and the generation of some commonly seen types of game content are rarely investigated. For instance, the generation of characters (skills, actions, and images) for fighting games and multi-player online battle games; the generation of cards and rules for strategy card games (e.g., Hearthstone); event generation (stories and effects) (e.g., for The Sims); goal generation in all kinds of games. Several approaches from other fields can be adapted to DLPCG, such as transfer learning for image generation in games, story generation for text-based adventure games and conversational NPCs.

6.4 Content generation in real time—personalized game content

Another less explored area is content generation in real time, such as generating level segments during gameplay, according to the actual player's playing skill-depth, style and preferences. Taking Super Mario Bros as an example, several MarioGAN models can be trained offline using a variety of fitness functions with different aims (e.g., encourage more jumps by putting more pipes, put more coins for players to collect, adjust the difficulty by controlling the number of enemies), and then be selected to generate new level segments during the game after determining the player's preferences and performance according to the gameplay data during first segments.

6.5 Learning from small data

One of the main limitations for most forms of PCG based on deep learning, or PCGML in general, is the access to training data. Some games have a large amount of existing content, either made by developers or by users. However, for a game in development there may not be content to learn from, because the content may not be made yet. In fact, not having to produce all of that content may be a prime reason for wanting to train a content generator in the first place. What would be desirable here would be a way of training a generator based on only a few pieces of hand-designed content, such as items, levels, or characters.

One approach to doing this is *bootstrapping*, where a generator is first trained on just a few examples, and whenever it produces new content that satisfies the functionality constraints, this content gets added to the training set for continued training of the generator [137]. This approach requires a reliable test of the functionality constraints, for example the playability of a level can be tested with game-playing agents.

Note that the amount of data required to train a reliable model varies greatly depending on the complexity of the model, the complexity of the data, and the training procedures of the model. For example, the training data limitation does not apply to PCG methods based on reinforcement learning. Further, MarioGAN [140] was trained on a single Mario level broken into many sections. Snodgrass et al. [122] explored the effects of the amount and diversity of training data on a simple Markov chain model and an LSTM and found that the benefits of additional data dropped off after several levels. Further studies exploring the data requirements of DLPCG models can help illuminate the usability and scalability of these approaches.

6.6 Generalization across games

Another, and arguably better, approach to learning generators for games for which you do not (yet) have much content would be trained on content from other games. After all, games from a particular genre have much in common, and it should arguably be possible to train on FPS levels from *Quake*, *Halo* and *Call of Duty* to learn to generate new levels for *Half-Life*. It should be even easier to train character models on existing human-designed characters from several open-world games, as they share the same functionality constraints. The trained generator would likely be a conditional model that takes some encoding of the characteristics of a game as input. In all of these cases, the deep learning model would have to learn to represent the underlying similarities between content for the games it was trained on, as well as the differences.

6.7 Orchestration for game generation

A key future research direction for any PCG framework is the generation of more than one domain of computational creativity within games. The six key computational game creativity domains as defined by Liapis et al. [72] include visuals, audio, narrative, levels, rules and gameplay. A process that considers the output of two or more of these domain generators up to the generation of a complete game is referred to as *orchestration* [73]. In other words, orchestration can be defined as the "harmonization of the game generation process" [73].

While orchestration is a core aim for the autonomous generation of complete games, Liapis et al. [73] reported only a few game generation systems that considered more than one generation domain. These include Angelina [12, 13], Game-O-matic [138], Sonancia [76], AudioInSpace [50] and the FPS generator by Karavolos et al. [58, 60]. Among these case studies of orchestrated game generation, only a few can be considered early embryos of DLPCG-based game orchestration. In particular, the work by Karavolos et al. [58, 60], Sonancia [76], and AudioInSpace [50] use various forms of shallow and deep neural networks—both as surrogate models (indirectly) and as generative functions (directly)—to generate content for multiple domains within games. As deep learning is of particular importance for fusing the generation process across content representations of dissimilar resolutions and characteristics [148], we expect to witness an increase in DL research work toward achieving game orchestration.

7 Conclusions

The work surveyed in this paper is the result of two convergent trends from the last few years. One is the increasing use of deep learning for generative tasks in non-game contexts, such as GANs and VAEs used for generating pictures of faces and RNNs used for generating voices and music. The other is the increasing use of machine learning in PCG, something that was unheard of until five years or so ago. Both of these trends build on the deep learning revolution itself, which has made machine learning effective on completely new classes of problems.

As a result, interest in deep learning for PCG has exploded. Examples abound, as our survey shows. It is very likely that we will see rapid progress in this research direction in the near future. This survey paper attempts to contribute to this progress by surveying and systematizing this work and implicitly and explicitly pointing out relevant and fertile research problems. We believe that this is a very timely effort given the exciting pace of this field.

Deep learning methods have been applied alone or in collaboration with other PCG methods to generate game content and to analyze, play and experience content. Due to the characteristics of different types of content, different types of deep neural architectures have been used. Among the reviewed work, the widely used neural architectures include convolutional neural networks for supervised learning tasks, varying from generating texture or music for target emotion to predicting game outcomes or difficulty rate; long short-term memory for generating sequential data like charts for rhythm and narrative or for predicting action sequences; deep variational autoencoders, mostly used for generating level maps and sometimes for classifying NPCs' or players' behaviors; and generative adversarial networks for creating image-like content (e.g., level maps, landscapes, faces and sprites). A part from the direct use of deep learning methods or their alliance with evolutionary computation to generate game content, they have also been used for evaluating content and content generators in an indirect manner.

Although a variety of game content (e.g., levels, text, character models, textures, music and sound) have been investigated, the generation of content like event, goals or character features with skill-depth can be exploited more. As a future research, evolving or training game-playing agents and content generators in parallel, such as in the recent work of Dharna et al. [19], is of great interest, as well as the generalization across games. Besides those, online generation of game content to adapt players' skill and preferences in real time will accelerate the realization of personalized games.

Acknowledgements J. Liu was supported by the National Key R&D Program of China (Grant No. 2017YFC0804003), the National Natural Science Foundation of China (Grant No. 61906083), the Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (Grant No. 2017ZT07X386), the Science and Technology Innovation Committee Foundation of Shenzhen (Grant No. JCYJ20190809121403553), the Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531) and the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008). S. Risi was supported by a Google Faculty Research award and a Sapere Aude: DFF-Starting Grant. A. Khalifa and J. Togelius acknowledge the financial support from National Science Foundation (NSF) award number 1717324 - "RI: Small: General Intelligence through Algorithm Invention and Selection". G. N. Yannakakis was supported by European Union's Horizon 2020 AI4Media (951911) and TAMED (101003397) projects.

Compliance with ethical standards

Conflict of interest The authors declare that they have a conflict of interest with *modl.ai*.

References

1. Abdal R, Qin Y, Wonka P (2019) Image2StyleGAN: how to embed images into the StyleGAN latent space? In: Proceedings of the IEEE International Conference on Computer Vision, pp 4432–4441
2. Ammanabrolu P, Cheung W, Tu D, Broniec W, Riedl MO (2020) Bringing stories alive: generating interactive fiction worlds. In: Proceedings of the sixteenth annual AAAI conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2020)
3. Awiszus M, Schubert F, Rosenhahn B (2020) TOAD-GAN: coherent style level generation from a single example. In: Proceedings of the sixteenth annual AAAI conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2020)
4. Bontrager P, Togelius J (2020) Fully differentiable procedural content generation through generative playing networks. arXiv preprint arXiv:200205259
5. Bontrager P, Roy A, Togelius J, Memon N, Ross A (2018) DeepMasterPrints: generating masterprints for dictionary attacks via latent variable evolution. In: 2018 IEEE 9th International Conference on Biometrics Theory. Applications and Systems (BTAS). IEEE, pp 1–9
6. Briot JP, Hadjeres G, Pachet F (2019) Deep learning techniques for music generation, vol 10. Springer, Berlin
7. Browne C, Maire F (2010) Evolutionary game design. IEEE Trans Comput Intell AI Games 2(1):1–16
8. Camilleri E, Yannakakis GN, Dingli A (2016) Platformer level design for player believability. In: 2016 IEEE Conference on Computational Intelligence and Games (CIG). IEEE, pp 1–8
9. Camilleri E, Yannakakis GN, Liapis A (2017) Towards general models of player affect. In: 2017 seventh international conference on Affective Computing and Intelligent Interaction (ACII). IEEE, pp 333–339
10. Chen Z, Amato C, Nguyen THD, Cooper S, Sun Y, El-Nasr MS (2018) Q-deckrec: a fast deck recommendation system for collectible card games. In: 2018 IEEE conference on Computational Intelligence and Games (CIG), pp 1–8. https://doi.org/10.1109/CIG.2018.8490446
11. Colton S (2008) Creativity versus the perception of creativity in computational systems. In: AAAI spring symposium: creative intelligent systems, vol 8
12. Cook M, Colton S, Raad A, Gow J (2013) Mechanic miner: reflection-driven game mechanic discovery and level design. In: European conference on the applications of evolutionary computation. Springer, pp 284–293
13. Cook M, Colton S, Gow J (2016) The angelina videogame design system—part I. IEEE Trans Comput Intell AI Games 9(2):192–203
14. Dahlskog S, Togelius J, Nelson MJ (2014) Linear levels through n-grams. In: Proceedings of the 18th International Academic MindTrek Conference: Media Business, Management, Content & Services, pp 200–206
15. Davoodi O, Ashtiani M, Rajabi M (2020) An approach for the evaluation and correction of manually designed video game levels using deep neural networks. Comput J. https://doi.org/10.1093/comjnl/bxaa071
16. De Kegel B, Haahr M (2020) Procedural puzzle generation: a survey. IEEE Trans Games 12(1):21–40
17. Delarosa O, Dong H, Ruan M, Khalifa A, Togelius J (2020) Mixed-initiative level design with RL brush. arXiv preprint arXiv:200802778
18. Dhariwal P, Jun H, Payne C, Kim JW, Radford A, Sutskever I (2020) Jukebox: a generative model for music. arXiv preprint arXiv:200500341
19. Dharna A, Togelius J, Soros L (2020) Coevolution of game levels and game-playing agents. In: Proceedings of the sixteenth annual AAAI conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2020)
20. Di Liello L, Ardino P, Gobbi J, Morettin P, Teso S, Passerini A (2020) Efficient generation of structured objects with constrained adversarial networks. arXiv preprint arXiv:200713197
21. Donahue C, Lipton ZC, McAuley J (2017) Dance dance convolution. In: International conference on machine learning, pp 1039–1048
22. Dormans J (2010) Adventures in level design: generating missions and spaces for action adventure games. In: Proceedings of the 2010 workshop on procedural content generation in games, pp 1–8
23. Earle S (2019) Using fractal neural networks to play SimCity 1 and Conway's Game of Life at variable scales. In: Proceedings of the Experimental AI in Games (EXAG) Workshop at AIIDE
24. Ebert DS, Musgrave FK, Peachey D, Perlin K, Worley S (2003) Texturing & modeling: a procedural approach. Morgan Kaufmann, Burlington
25. Fadaeddini A, Majidi B, Eshghi M (2018) A case study of generative adversarial networks for procedural synthesis of original textures in video games. In: 2018 2nd National and 1st International Digital Games Research Conference: trends, technologies, and applications (DGRC). IEEE, pp 118–122
26. Fang K, Zhu Y, Savarese S, Fei-Fei L (2020) Adaptive procedural task generation for hard-exploration problems. arXiv preprint arXiv:200700350
27. Ferreira LN, Lelis LH, Whitehead J (2020) Computer-generated music for tabletop role-playing games. In: Proceedings of the sixteenth annual AAAI conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2020)
28. Fontaine M, Togelius J, Nikolaidis S, Hoover AK (2020) Covariance matrix adaptation for the rapid illumination of behavior space. In: Proceedings of the 2020 genetic and evolutionary computation conference
29. Fontaine MC, Liu R, Togelius J, Hoover AK, Nikolaidis S (2020) Illuminating Mario scenes in the latent space of a generative adversarial network. arXiv preprint arXiv:200705674
30. Gatys LA, Ecker AS, Bethge M (2015) A neural algorithm of artistic style. arXiv preprint arXiv:150806576
31. Giacomello E, Lanzi PL, Loiacono D (2018) Doom level generation using generative adversarial networks. In: 2018 IEEE Games, Entertainment, Media Conference (GEM). IEEE, pp 316–323
32. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ (eds) Advances in neural information processing systems, vol 27. Curran Associates, Inc., Red Hook, pp 2672–2680
33. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press. http://www.deeplearningbook.org
34. Greff K, Srivastava RK, Koutník J, Steunebrink BR, Schmidhuber J (2017) LSTM: a search space odyssey. IEEE Trans Neural Netw Learn Syst 28(10):2222–2232
35. Gudmundsson SF, Eisen P, Poromaa E, Nodet A, Purmonen S, Kozakowski B, Meurling R, Cao L (2018) Human-like playtesting with deep learning. In: 2018 IEEE conference on Computational Intelligence and Games (CIG). IEEE, pp 1–8
36. Gutierrez J, Schrum J (2020) Generative adversarial network rooms in generative graph grammar dungeons for the Legend of Zelda. In: 2020 IEEE Congress on Evolutionary Computation (CEC). IEEE

37. Guzdial M, Riedl M (2016) Game level generation from gameplay videos. In: Twelfth artificial intelligence and interactive digital entertainment conference
38. Guzdial M, Long D, Cassion C, Das A (2017) Visual procedural content generation with an artificial abstract artist. In: Proceedings of ICCC computational creativity and games workshop
39. Guzdial M, Liao N, Riedl M (2018) Co-creative level design via machine learning. In: Proceedings of the Experimental AI in Games (EXAG) workshop at AIIDE
40. Guzdial M, Reno J, Chen J, Smith G, Riedl M (2018) Explainable PCGML via game design patterns. In: Proceedings of the Experimental AI in Games (EXAG) workshop at AIIDE
41. Guzdial M, Liao N, Chen J, Chen SY, Shah S, Shah V, Reno J, Smith G, Riedl MO (2019) Friend, collaborator, student, manager: how design of an AI-driven game level editor affects creators. In: Proceedings of the 2019 CHI conference on human factors in computing systems, pp 1–13
42. Guzdial MJ, Sturtevant N, Li B (2016) Deep static and dynamic level analysis: a study on infinite mario. In: Twelfth artificial intelligence and interactive digital entertainment conference
43. Ha D, Eck D (2017) A neural representation of sketch drawings. arXiv preprint arXiv:170403477
44. Hastings EJ, Guha RK, Stanley KO (2009) Automatic content generation in the galactic arms race video game. IEEE Trans Comput Intell AI Games 1(4):245–263
45. Hochreiter S, Schmidhuber J (1997) LSTM can solve hard long time lag problems. In: Advances in neural information processing systems, pp 473–479
46. Holmgård C, Liapis A, Togelius J, Yannakakis GN (2014) Evolving personas for player decision modeling. In: 2014 IEEE conference on computational intelligence and games. IEEE, pp 1–8
47. Holmgard C, Green MC, Liapis A, Togelius J (2018) Automated playtesting with procedural personas with evolved heuristics. IEEE Trans Games 11(4):352–362
48. Hong S, Kim S, Kang S (2019) Game sprite generator using a multi discriminator GAN. KSII Trans Internet Inf Syst 13(8):4255–4269
49. Hoover AK, Szerlip PA, Stanley KO (2014) Functional scaffolding for composing additional musical voices. Comput Music J 38(4):80–99
50. Hoover AK, Cachia W, Liapis A, Yannakakis GN (2015) Audioinspace: exploring the creative fusion of generative audio, visuals and gameplay. In: International conference on evolutionary and biologically inspired music and art. Springer, pp 101–112
51. Hoover AK, Togelius J, Yannakis GN (2015) Composing video game levels with music metaphors through functional scaffolding. In: First computational creativity and games workshop. ACC
52. Irfan A, Zafar A, Hassan S (2019) Evolving levels for general games using deep convolutional generative adversarial networks. In: 2019 11th Computer Science and Electronic Engineering (CEEC). IEEE, pp 96–101
53. Isaksen A, Holmgård C, Togelius J (2017) Semantic hashing for video game levels. Game Puzzle Des 3(1):10–16
54. Jain R, Isaksen A, Holmgård C, Togelius J (2016) Autoencoders for level generation, repair, and recognition. In: Proceedings of the ICCC workshop on computational creativity and games
55. Jin Y, Zhang J, Li M, Tian Y, Zhu H, Fang Z (2017) Towards the automatic anime characters creation with generative adversarial networks. CoRR arXiv:1708.05509
56. Jordanous A (2012) A standardised procedure for evaluating creative systems: computational creativity evaluation based on what it is to be creative. Cogn Comput 4(3):246–279
57. Karavolos D, Liapis A, Yannakakis G (2017) Learning the patterns of balance in a multi-player shooter game. In: Proceedings of the 12th international conference on the foundations of digital games, pp 1–10
58. Karavolos D, Liapis A, Yannakakis GN (2018) Pairing character classes in a deathmatch shooter game via a deep-learning surrogate model. In: Proceedings of the 13th international conference on the Foundations of digital games, pp 1–10
59. Karavolos D, Liapis A, Yannakakis GN (2018) Using a surrogate model of gameplay for automated level design. In: 2018 IEEE conference on Computational Intelligence and Games (CIG). IEEE, pp 1–8
60. Karavolos D, Liapis A, Yannakakis GN (2019) A multi-faceted surrogate model for search-based procedural content generation. IEEE Trans Games. https://doi.org/10.1109/TG.2019.2931044
61. Karras T, Laine S, Aittala M, Hellsten J, Lehtinen J, Aila T (2019) Analyzing and improving the image quality of stylegan. arXiv preprint arXiv:191204958
62. Khalifa A, Bontrager P, Earle S, Togelius J (2020) PCGRL: procedural content generation via reinforcement learning. arXiv preprint arxiv:2001.09212
63. Kingma DP, Welling M (2013) Auto-encoding variational bayes. arXiv preprint arXiv:13126114
64. Kuang P, Luo D (2020) Conditional convolutional generative adversarial networks based interactive procedural game map generation. In: Future of information and communication conference. Springer, pp 400–419
65. Kumaran V, Mott BW, Lester JC (2020) Generating game levels for multiple distinct games with a common latent space. In: Proceedings of the sixteenth annual AAAI conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2020)
66. Larsson S, Petri O (2016) Content evaluation of starcraft maps using neuroevolution. Dissertation. Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:bth-11684
67. Liang Y, Li W, Ikeda K (2019) Procedural content generation of rhythm games using deep learning methods. In: Joint international conference on entertainment computing and serious games. Springer, pp 134–145
68. Liapis A, Yannakis GN (2016) Boosting computational creativity with human interaction in mixed-initiative co-creation tasks. In: Proceedings of the ICCC workshop on computational creativity and games
69. Liapis A, Martínez HP, Togelius J, Yannakakis GN (2013) Transforming exploratory creativity with delenox. In: International conference on computational creativity
70. Liapis A, Yannakakis GN, Togelius J (2013) Sentient sketchbook: computer-aided game level authoring. In: Proceedings of the 2013 ACM conference on foundations of digital games
71. Liapis A, Yannakakis GN, Togelius J (2013) Sentient world: human-based procedural cartography. In: International conference on evolutionary and biologically inspired music and art. Springer, pp 180–191
72. Liapis A, Yannakakis GN, Togelius J (2014) Computational game creativity. In: ICCC
73. Liapis A, Yannakakis GN, Nelson MJ, Preuss M, Bidarra R (2018) Orchestrating game generation. IEEE Trans Games 11(1):48–68
74. Liu MY, Breuel T, Kautz J (2017) Unsupervised image-to-image translation networks. In: Advances in neural information processing systems, pp 700–708
75. Liu Z, Luo P, Wang X, Tang X (2015) Deep learning face attributes in the wild. In: Proceedings of International Conference on Computer Vision (ICCV)
76. Lopes P, Liapis A, Yannakakis GN (2015) Sonancia: sonification of procedurally generated game levels. In: ICCC

77. Lucas SM, Volz V (2019) Tile pattern KL-divergence for analysing and evolving game levels. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-19. Association for Computing Machinery, New York, NY, USA, pp 170–178. https://doi.org/10.1145/3321707.3321781
78. Makantasis K, Liapis A, Yannakakis GN (2019) From pixels to affect: a study on games and player experience. In: 2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII). IEEE, pp 1–7
79. Martínez HP, Yannakakis GN (2014) Deep multimodal fusion: combining discrete events and continuous signals. In: Proceedings of the 16th international conference on multimodal interaction, pp 34–41
80. Martinez HP, Bengio Y, Yannakakis GN (2013) Learning deep physiological models of affect. IEEE Comput Intell Mag 8(2):20–33
81. Melhart D, Gravina D, Yannakakis GN (2020) Moment-to-moment engagement prediction through the eyes of the observer: PUBG streaming on twitch. In: Foundations of digital games
82. Min W, Ha EY, Rowe J, Mott B, Lester J (2014) Deep learning-based goal recognition in open-ended digital games. In: Tenth artificial intelligence and interactive digital entertainment conference
83. Mordvintsev A, Randazzo E, Niklasson E, Levin M (2020) Growing neural cellular automata. Distill 5:e23. https://doi.org/10.23915/distill.00023
84. Mott J, Nandi S, Zeller L (2019) Controllable and coherent level generation: a two-pronged approach. In: Experimental AI in games workshop
85. Park K, Mott BW, Min W, Boyer KE, Wiebe EN, Lester JC (2019) Generating educational game levels with multistep deep convolutional generative adversarial networks. In: 2019 IEEE Conference on Games (CoG). IEEE, pp 1–8
86. Pease A, Colton S (2011) On impact and evaluation in computational creativity: a discussion of the turing test and an alternative proposal. In: Proceedings of the AISB symposium on AI and philosophy, vol 39
87. Perez-Liebana D, Liu J, Khalifa A, Gaina RD, Togelius J, Lucas SM (2019a) General video game AI: a multitrack framework for evaluating agents, games, and content generation algorithms. IEEE Trans Games 11(3):195–214
88. Perez-Liebana D, Lucas SM, Gaina RD, Togelius J, Khalifa A, Liu J (2019) General video game artificial intelligence. Morgan & Claypool Publishers. https://gaigresearch.github.io/gvgaibook/
89. Perlin K (1985) An image synthesizer. ACM Siggraph Comput Graph 19(3):287–296
90. Pfau J, Liapis A, Volkmar G, Yannakakis GN, Malaka R (2020) Dungeons & replicants: automated game balancing via deep player behavior modeling. In: Proceedings of the 2020 IEEE Conference on Games (CoG)
91. Radford A, Metz L, Chintala S (2015) Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:151106434
92. Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I (2019) Language models are unsupervised multitask learners. OpenAI Blog
93. Risi S, Togelius J (2019) Increasing generality in machine learning through procedural content generation. arXiv preprint arXiv:1911.13071
94. Risi S, Lehman J, D'Ambrosio DB, Hall R, Stanley KO (2015) Petalz: search-based procedural content generation for the casual gamer. IEEE Trans Comput Intell AI Games 8(3):244–255
95. Roy A, Memon N, Ross A (2017) Masterprint: exploring the vulnerability of partial fingerprint-based authentication systems. IEEE Trans Inf Forensics Secur 12(9):2013–2025
96. Sarkar A, Cooper S (2018) Blending levels from different games using LSTMs. In: Proceedings of the Experimental AI in Games (EXAG) workshop at AIIDE
97. Sarkar A, Cooper S (2020) Sequential segment-based level generation and blending using variational autoencoders. arXiv preprint arXiv:200708746
98. Sarkar A, Cooper S (2020) Towards game design via creative machine learning (GDCML). In: Proceedings of the 2020 IEEE Conference on Games (CoG)
99. Sarkar A, Yang Z, Cooper S (2019) Controllable level blending between games using variational autoencoders. In: Proceedings of the Experimental AI in Games (EXAG) workshop at AIIDE
100. Sarkar A, Summerville A, Snodgrass S, Bentley G, Osborn J (2020) Exploring level blending across platformers via paths and affordances. In: Sixteenth artificial intelligence and interactive digital entertainment conference
101. Schaul T (2013) A video game description language for model-based or interactive learning. In: Proceedings of the IEEE conference on computational intelligence in games. IEEE Press, Niagara Falls
102. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85–117. https://doi.org/10.1016/j.neunet.2014.09.003
103. Schrum J, Gutierrez J, Volz V, Liu J, Lucas SM, Risi S (2020) Interactive evolution and exploration within latent level-design space of generative adversarial networks. In: Proceedings of the genetic and evolutionary computation conference. ACM
104. Schrum J, Volz V, Risi S (2020) CPPN2GAN: combining compositional pattern producing networks and GANs for large-scale pattern generation. In: Proceedings of the genetic and evolutionary computation conference. ACM
105. Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Trans Signal Process 45(11):2673–2681
106. Scirea M, Eklund P, Togelius J, Risi S (2018) Evolving in-game mood-expressive music with metacompose. In: The audio mostly 2018 on sound in immersion and emotion, pp 1–8
107. Serpa YR, Rodrigues MAF (2019) Towards machine-learning assisted asset generation for games: a study on pixel art sprite sheets. In: 2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames). IEEE, pp 182–191
108. Shaham TR, Dekel T, Michaeli T (2019) Singan: learning a generative model from a single natural image. In: Proceedings of the IEEE international conference on computer vision, pp 4570–4580
109. Shaker N, Yannakakis G, Togelius J (2010) Towards automatic personalized content generation for platform games. In: Sixth artificial intelligence and interactive digital entertainment conference
110. Shaker N, Togelius J, Yannakakis GN, Weber B, Shimizu T, Hashiyama T, Sorenson N, Pasquier P, Mawhorter P, Takahashi G et al (2011) The 2010 Mario AI championship: level generation track. IEEE Trans Comput Intell AI Games 3(4):332–347
111. Shaker N, Nicolau M, Yannakakis GN, Togelius J, O'neill M (2012) Evolving levels for Super Mario Bros using grammatical evolution. In: Computational intelligence and games. IEEE, pp 304–311
112. Shaker N, Togelius J, Nelson MJ (2016) Procedural content generation in games. Springer, Berlin
113. Shu T, Wang Z, Liu J, Yao X (2020) A novel CNET-assisted evolutionary level repairer and its applications to Super Mario Bros. In: 2020 IEEE Congress on Evolutionary Computation (CEC). IEEE

114. Sirota J, Bulitko V, Brown MR, Hernandez SP (2019) Towards procedurally generated languages for non-playable characters in video games. In: 2019 IEEE Conference on Games (CoG). IEEE, pp 1–4
115. Smith AM, Mateas M (2011) Answer set programming for procedural content generation: a design space approach. IEEE Trans Comput Intell AI Games 3(3):187–200
116. Smith G, Whitehead J (2010) Analyzing the expressive range of a level generator. In: Proceedings of the 2010 workshop on procedural content generation in games, pp 1–7
117. Snodgrass S, Ontañón S (2014) Experiments in map generation using Markov chains. In: Proceedings of the 9th conference on the foundations of digital games
118. Snodgrass S, Ontanon S (2015) A hierarchical MDMC approach to 2D video game map generation. In: Eleventh artificial intelligence and interactive digital entertainment conference
119. Snodgrass S, Ontanón S (2016) Controllable procedural content generation via constrained multi-dimensional Markov chain sampling. In: IJCAI, pp 780–786
120. Snodgrass S, Ontanón S (2016b) Learning to generate video game maps using Markov models. IEEE Trans Comput Intell AI Games 9(4):410–422
121. Snodgrass S, Sarkar A (2020) Multi-domain level generation and blending with sketches via example-driven BSP and variational autoencoders. In: Proceedings of the 15th international conference on the foundations of digital games
122. Snodgrass S, Summerville A, Ontañón S (2017) Studying the effects of training data on machine learning-based procedural content generation. In: Thirteenth artificial intelligence and interactive digital entertainment conference
123. Soares ES, Bulitko V (2019) Deep variational autoencoders for NPC behaviour classification. In: 2019 IEEE Conference on Games (CoG). IEEE, pp 1–4
124. Stanley KO, Miikkulainen R (2002) Evolving neural networks through augmenting topologies. Evol Comput 10(2):99–127
125. Summerville A (2018) Expanding expressive range: evaluation methodologies for procedural content generation. In: Fourteenth artificial intelligence and interactive digital entertainment conference
126. Summerville A, Mateas M (2016) Super Mario as a string: platformer level generation via LSTMs. In: International Joint Conference of DiGRA and FDG
127. Summerville A, Guzdial M, Mateas M, Riedl MO (2016) Learning player tailored content from observation: platformer level generation from video traces using LSTMs. In: Twelfth artificial intelligence and interactive digital entertainment conference
128. Summerville A, Mariño JR, Snodgrass S, Ontañón S, Lelis LH (2017) Understanding Mario: an evaluation of design metrics for platformers. In: Proceedings of the 12th international conference on the foundations of digital games, pp 1–10
129. Summerville A, Snodgrass S, Guzdial M, Holmgård C, Hoover AK, Isaksen A, Nealen A, Togelius J (2018) Procedural content generation via machine learning (PCGML). IEEE Trans Games 10(3):257–270
130. Summerville AJ, Mateas M (2016) Mystical tutor: a magic: the gathering design assistant via denoising sequence-to-sequence learning. In: Twelfth artificial intelligence and interactive digital entertainment conference
131. Summerville AJ, Philip S, Mateas M (2015) MCMCTS PCG 4 SMB: Monte Carlo tree search to guide platformer level generation. In: Artificial intelligence and interactive digital entertainment
132. Togelius J, Kastbjerg E, Schedl D, Yannakakis GN (2011) What is procedural content generation? Mario on the borderline. In: Proceedings of the 2nd international workshop on procedural content generation in games. ACM, p 3
133. Togelius J, Yannakakis GN, Stanley KO, Browne C (2011b) Search-based procedural content generation: a taxonomy and survey. IEEE Trans Comput Intell AI Games 3(3):172–186
134. Togelius J, Champandard AJ, Lanzi PL, Mateas M, Paiva A, Preuss M, Stanley KO (2013) Procedural content generation: goals, challenges and actionable steps. In: Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik
135. Togelius J, Shaker N, Karakovskiy S, Yannakakis GN (2013b) The Mario AI championship 2009–2012. AI Mag 34(3):89–92
136. Torrado RR, Bontrager P, Togelius J, Liu J, Perez-Liebana D (2018) Deep reinforcement learning for general video game AI. In: Proceedings of the 2018 IEEE Conference on Computational Intelligence and Games (CIG). IEEE, pp 1–8
137. Torrado RR, Khalifa A, Green MC, Justesen N, Risi S, Togelius J (2019) Bootstrapping conditional gans for video game level generation. arXiv preprint arXiv:1910.01603
138. Treanor M, Blackford B, Mateas M, Bogost I (2012) Game-o-matic: generating videogames that represent ideas. In: Proceedings of the the third workshop on procedural content generation in games, pp 1–8
139. Tsujino Y, Yamanishi R (2018) Dance dance gradation: a generation of fine-tuned dance charts. In: International conference on entertainment computing. Springer, pp 175–187
140. Volz V, Schrum J, Liu J, Lucas SM, Smith A, 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. ACM, pp 221–228
141. Volz V, Justesen N, Snodgrass S, Asadi S, Purmonen S, Holmgård C, Togelius J, Risi S (2020) Capturing local and global patterns in procedural content generation via machine learning. In: Proceedings of the 2020 IEEE Conference on Games (CoG)
142. Walton N (2019) AI Dungeon 2: creating infinitely generated text adventures with deep learning language models. https://pcc.cs.byu.edu/2019/11/21/ai-dungeon-2-creating-infinitely-generated-text-adventures-with-deep-learning-language-models/. Accessed 2 May 2020
143. Wang T, Kurabayashi S (2020) Sketch2map: a game map design support system allowing quick hand sketch prototyping. In: Proceedings of the 2020 IEEE Conference on Games (CoG)
144. Wong A, Wang GH (2017) Image_retrieval_demo: a demo for image retrieval. https://github.com/DoctorKey/image_retrieval_demo
145. Wulff-Jensen A, Rant NN, Møller TN, Billeskov JA (2017) Deep convolutional generative adversarial network for procedural 3D landscape generation based on DEM. In: Interactivity, game creation, design, learning, and innovation. Springer, pp 85–94
146. Yang Z, Sarkar A, Cooper S (2020) Game level clustering and generation using Gaussian mixture VAEs. In: Proceedings of the sixteenth annual AAAI conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2020). AAAI
147. Yannakakis GN, Togelius J (2011) Experience-driven procedural content generation. IEEE Trans Affect Comput 2(3):147–161
148. Yannakakis GN, Togelius J (2018) Artificial intelligence and games. Springer. http://gameaibook.org
149. Yannakakis GN, Liapis A, Alexopoulos C (2014) Mixed-initiative co-creativity. In: Proceedings of the 9th conference on the foundations of digital games
150. Yoo B, Kim KJ (2016) Changing video game graphic styles using neural algorithms. In: 2016 IEEE conference on Computational Intelligence and Games (CIG). IEEE, pp 1–2

151. Yumer ME, Asente P, Mech R, Kara LB (2015) Procedural modeling using autoencoder networks. In: Proceedings of the 28th annual ACM Symposium on User Interface Software & Technology, UIST '15. Association for Computing Machinery, New York, NY, USA, pp 109–118. https://doi.org/10.1145/2807442.2807448

152. Zafar A, Irfan A, Sabir MZ (2019) Generating general levels using Markov chains. In: 2019 11th Computer Science and Electronic Engineering (CEEC). IEEE, pp 134–138