

# Towards Game Design via Creative Machine Learning (GDCML)

Anurag Sarkar  
Northeastern University  
Boston, USA  
sarkar.an@northeastern.edu

Seth Cooper  
Northeastern University  
Boston, USA  
se.cooper@northeastern.edu

**Abstract**—In recent years, machine learning (ML) systems have been increasingly applied for performing creative tasks. Such creative ML approaches have seen wide use in the domains of visual art and music for applications such as image and music generation and style transfer. However, similar creative ML techniques have not been as widely adopted in the domain of game design despite the emergence of ML-based methods for generating game content. In this paper, we argue for leveraging and repurposing such creative techniques for designing content for games, referring to these as approaches for Game Design via Creative ML (GDCML). We highlight existing systems that enable GDCML and illustrate how creative ML can inform new systems via example applications and a proposed system.

**Index Terms**—game design, creative AI, creative ML, computational creativity, procedural content generation, PCGML

## I. INTRODUCTION

Advances in machine learning, such as those in the field of computer vision and image processing [1]–[4], and the emergence of generative models such as GANs [5] and VAEs [6], [7], have enabled a wide variety of ML applications for performing creative and artistic tasks. In the domain of visual art, this has led to applications such as artistic style transfer [8]–[10], texture synthesis [11], [12] and image manipulation and translation [13], [14]. Similarly, the domain of music has also been a fertile ground for such creative applications including generation using a number of different approaches [15] and in different genres and styles [16]–[19]. Moreover, such *creative AI* [20] approaches and models have enabled the creation of generative systems and co-creative tools in both visual art [21]–[26] and music [27]–[29], helping in democratizing such AI and ML systems and better facilitating human creativity.

However, such ML-based creative approaches have not been as widely adopted for game design. Due to the relative dearth of good training data compared to visual art and music and the added complexity of ensuring that models can produce content that is functional and playable, most generative approaches for games, often termed as procedural content generation (PCG) [30], have primarily leveraged methods involving evolutionary search [31], generative grammars [32], constraint solving [33] and cellular automata [34]. More recently, a number of works have demonstrated the feasibility of using ML to build generative models for games and game content. PCG via

machine learning (PCGML) [35] has emerged as a subfield of games research concerned with generating new game content using models trained on existing game data. A number of ML techniques have been used for this purpose such as LSTMs [36], Bayes Nets [37], Markov models [38], GANs [39] and autoencoders [40], mostly for level generation for games such as *Super Mario Bros.* [41], *The Legend of Zelda* [42] and *Doom* [43]. While these works successfully used ML to generate levels, they mostly focused on a specific game and are more comparable to simpler generative applications in visual art and music rather than the more creative applications such as style transfer and ML-based co-creativity.

In this vein, a recent trend of more creative PCGML has emerged [44], focusing on applications such as domain transfer [45], [46], level and game blending [47]–[49] and generation of entire new games using ML models [50]. These new works combined with the emergence of new ML-powered game design tools [51] signal that creative ML approaches prevalent primarily in visual art and music thus far, can be repurposed for use with ML models for game design.

In this paper, we expand on our prior exploratory work [52] and introduce the term *Game Design via Creative Machine Learning* or GDCML to refer to such techniques and models. More specifically, we use GDCML to refer to a subset of PCGML techniques that use models trained on one or more games to enable creative ML applications and affordances for automated as well as mixed-initiative design tools for game design, similar to those seen in visual art and music as highlighted previously. In the remainder of the paper, we briefly survey creative ML approaches in visual art and music and connect them to related existing and potential applications in game design, highlight existing creative ML work in games and demonstrate applications for game design via a number of illustrative examples. We also discuss the blueprint for a proposed co-creative ML-based design tool that encompasses the demonstrated applications and conclude by highlighting future directions and challenges in the field.

## II. CREATIVE ML FOR VISUAL ART AND MUSIC

Seminal to the eventual emergence of creative ML in visual art was the work of Gatys et al. in using convolutional neural nets to perform texture synthesis [11] and image style transfer [8]. This has been followed by a large body of work both in

terms of research aimed at improving and building upon style transfer methods [53] as well as tools and implementations enabling users to interact and experiment with these methods [21], [22], [54]. Isola et al.'s pix2pix [13] model has been particularly helpful in popularizing creative ML applications. This model learns image transformation functions between sets of image pairs and has been used in a wide variety of interactive tools and demos such as Invisible Cities [25], Hesse's Image-to-Image Demo [24] as well as a number of live interfaces and installations [55]. Moreover, research into better understanding the inner workings of such ML models have also resulted in interesting artistic applications such as DeepDream [56], and interactive tools such as GAN Paint [23]. Along these lines, the software suite *RunwayML* [57] enables users to work with pretrained generative models for a number of artistic tasks, demonstrating how such applications can help democratize creative ML to non-practitioners. Instrumental to the rise of creative ML in visual art has been the increase in popularity and the rapid growth in scale, complexity and expressivity of Generative Adversarial Networks (GANs) [5] with models such as CycleGAN [14], SNGAN [58] and particularly BigGAN [59] and StyleGAN [60] being leveraged by artists such as Mario Klingemann, Helena Sarin, Robbie Barrat, Anna Ridler and Memo Akten, to name a few, to produce artworks and help usher in a new era of *AI Art* [61].

The domain of music has also seen much ML-based research. A wide variety of different ML approaches have been used for building generative models of music [15] using both raw audio [62] as well as symbolic representations [63] and for diverse genres including Bach chorales [16], jazz [17], pop [18] and metal [19]. Like in visual art, ML research in music has also seen plenty of recent works use latent variable models such as GANs [29], [64] and VAEs [65]–[67]. These approaches all leverage the model's learned latent space to enable applications such as learning, blending and transfer of styles, instrument modeling and conditioning generation on desired attributes. Moreover, these models serve as the basis for co-creative, interactive design tools such as *Magenta Studio* [27] and *MidiMe* [68] which operationalize the affordances of the ML models underneath. OpenAI's *Jukebox* [69] is a high-profile recent example of a creative ML model for music.

Note that for both visual art and music, while the initial ML techniques enabling generative modeling were foundational, creative AI/ML did not flourish until the rise of more advanced latent variable models that enable applications such as blending, interpolation, style transfer and conditioning along with tools that operationalize and democratize them. Most current ML research in games is at the former, foundational stage. It is with a view to highlight and discuss existing and future methods to enable the latter that we write this paper.

### III. CREATIVE AI IN GAME DESIGN

In this section, we discuss existing co-creative approaches in games. First however, we would like to draw a distinction between *creative AI* and *creative ML*, two terms that are often used interchangeably but we differentiate for two reasons: 1)

in most uses of the term *creative AI*, the underlying method more specifically uses ML and 2) to focus our scope, we wish to concentrate on co-creative game design methods and tools that use ML, separate from the various co-creative game design tools that use more general AI methods.

While the previously mentioned PCGML works are analogous to ML models for music and art in general, the closest analogs to the related tools and applications are the numerous mixed-initiative, co-creative game design tools, most of which do not employ ML. Co-creative or mixed-initiative systems [70] refer to those that enable human designers to collaborate with the generative system. Notable earlier examples of such systems include *Tanagra* [32] for generating platformer levels, *Ropossum* [71] for generating levels for *Cut the Rope* [72] and *Sentient Sketchbook* [73] for generating strategy maps. More recent examples include *Cicero* [74] for designing GVG-AI games, the Evolutionary Dungeon Designer [75] and *Baba Is Y'All* [76] for generating levels for the puzzle game *Baba Is You* [77]. A related AI-based co-creative tool is *Danesh* [78] which allows users to adjust the parameters of generators and analyze their expressive range [79]. Finally, though not a co-creative tool, ANGELINA [80], [81] is an AI system capable of generating entirely new games using evolutionary computation. While all of these tools and systems enable the design and generation of new levels and games, they differ from the previously discussed counterparts for visual art and music in that they are not informed by ML models. That is, the systems do not leverage knowledge learned from an existing corpus of game data and subsequently are not able to harness the affordances that for example, a latent variable model would provide. In this work, we are interested in existing and potential approaches that could leverage much of the existing PCGML research to produce GDCML tools and it is this that we discuss in the next section.

### IV. THE CASE FOR CREATIVE ML FOR GAME DESIGN

This paper was motivated by a recent trend of a number of works that enable a more creative form of PCGML [44] via applications such as level and game blending, domain transfer and automated game generation. These in turn were motivated with wanting to incorporate computational creativity into ML models, specifically combinatorial creativity (also referred to as combinatorial creativity) [82], the branch of creativity focused on generating new concepts, domains and artifacts from combinations of existing ones. This view of creativity contends that innovation rarely happens in a vacuum and that new ideas usually build on existing ones. Maria Popova describes this as '*... the idea [is] that creativity is combinatorial, that nothing is entirely original, that everything builds on what came before, and that we create by taking existing pieces of inspiration, knowledge, skill and insight that we gather over the course of our lives and recombining them into incredible new creations*' [83]. Several other sources [84]–[87] also highlight the prevalence of combinatorial creativity throughout history in both the arts and sciences. Such creativity is evident throughout the history of games as well with new games and

game genres resulting from the recombination of existing ones. *Metroid* [88], for example, combines Mario’s platforming with the lock-and-key style progression of *Zelda*. *Spelunky* [89] similarly melds platformer mechanics with roguelike elements. We have seen the increasing use of terms such as *roguelite*, *procvania* and *soulslike* to describe new game genres that borrow and combine elements from multiple existing genres. Recent indie game *SuperMash* [90] lets players explicitly combine different genres to produce new games to play. Thus, imbuing ML models with combinational creativity techniques such as conceptual blending [91], amalgamation [92], compositional adaptation [93] and conceptual expansion [50] could enable tools to assist in such creative forms of game design and generation. Conceptual expansion has in fact been demonstrated to be able to generate entirely new games that combine the levels and mechanics of existing games [94].

Along these lines, Gow and Corneli [95] were among the first to propose a framework for blending existing games together to produce new ones. With a view towards building ML models capable of doing this, we used LSTMs [48] to blend models of Mario and *Kid Icarus* [96] and improved on that by using VAEs to perform controllable blending using the VAE latent space [49]. The latter was also inspired by Volz et al.’s work [39] in training GANs for generating Mario levels and using the GAN latent space to evolve variations. Since then, other works have used GANs and VAEs for PCG [97]–[99]. These approaches share similarities with analogous approaches discussed previously for visual art and music in their explicit use of the latent space and its encodings or at least their potential to do so. This in turn enables similar affordances within games (i.e. blending, interpolation, conditioned generation) as they do in other domains and thus prime these methods for serving as the foundation for co-creative GDCML tools.

To this end, Schrum et al. [100] recently presented a latent model-based co-creative game design tool, developing a system based on their GAN models for Mario and *Zelda* that allows users to design and generate levels via interactive evolution and exploration of the GAN’s latent space. Thus, this tool has much in common with similar tools for visual art and music and represents the type of application we wish to highlight and build on under the GDCML definition. Another such co-creative ML-based game design tool, though not based on latent variable models but still influential to our recommendations in the following sections, is Guzdial et al.’s *Morai Maker* [51], a Unity tool for designing Mario levels using generative models from past PCGML works [36], [38], [101] as co-creative partners. In addition to these works, recent PCGML work has also looked at domain transfer methods such as Snodgrass and Ontañón’s work [45] in learning mappings between platformer games and Snodgrass’ [102] newer work that uses binary space partitioning to generate levels from a low-resolution sketch representation, an approach that seems particularly suited to inform co-creative design tools.

Overall, such tools represent promising first steps towards realizing creative ML for game design in the future, as it

currently exists for visual art and music. For this to happen, existing tools need to be built upon and enhanced in terms of scope and affordances. Both *Morai Maker* and Schrum et al.’s tool are restricted to a single domain. While effective, this necessarily limits the possibility space of design. Building tools that leverage existing PCGML works in blending and domain transfer described above is necessary for enabling more creative applications such as style transfer and the design and discovery of new domains and genres of games. Moreover, not much attempt has been made to borrow creative ML ideas from other domains into games. This is a missed opportunity as creative ML for visual art and music, due to its increased maturity as a field, offers many approaches and applications that can be repurposed for games. In the next section, we discuss example applications, inspired in part by those in visual art and music and in part by the existing GDCML tools above, that leverage the affordances of latent models and that we hope to implement in GDCML tools in the future.

## V. APPLICATIONS

In this section, we demonstrate example applications that we hope to implement and operationalize in future creative ML tools for game design. For some applications, we provide example figures generated from techniques and systems developed in our prior work. For these examples, we trained variational autoencoders on level data from the Video Game Level Corpus (VGLC) [103] for the games *Super Mario Bros.* and *Kid Icarus*—both classic NES-era platformers. Models were trained on 16x16 level segments using PyTorch [104].

### A. Game Blending

Game blending refers to combining the levels and/or mechanics of two or more existing games to produce an entirely new game. Thus, it is comparable to style transfer techniques in visual art and music. Our prior work [49] has demonstrated the feasibility of using VAEs for blending levels from separate games, motivated by implementing Gow and Corneli’s VGDL game blending framework [95] as an ML model able to



Fig. 1. Example level that blends *Super Mario Bros.* and *Kid Icarus*.



Fig. 2. Interpolation in Mario. Segments at each end are from the actual game.

perform blending in an automated manner. The basic idea is that training a latent variable model on levels from multiple games enables it to learn a latent representation that spans and encodes levels from all games. Thus levels generated using this representation necessarily blend the properties of the original games. An example blended level is shown in Figure 1. In addition to enabling users to generate such blended levels and games, tools should also allow blends to be controllable in terms of the amount of each game desired in the final blend as well as properties such as difficulty and level topology. Previous works [39], [49] have demonstrated that evolving vectors in the latent space using various objective functions can make generation controllable and find latent vectors corresponding to playable levels. Thus, GDCML tools can allow users to select different objective functions and parameters to optimize to generate desired types of blends.

### B. Interpolation

Latent variable models learn encodings of data within a continuous, latent space. When trained on game levels, such models thus enable generation of new levels that inhabit the space between existing levels via interpolation. When these levels are from different games, it enables blending as shown above but when the levels are from one specific game, we can obtain new levels not in that game. Examples of this are given in Figure 2. Schrum et al.’s [100] GAN-based tool already implements such functionality by allowing users to interpolate between two existing levels via a slider and then interact with the resulting interpolated level. We envision such features to be incorporated in GDCML tools moving forward.

### C. Level Search

This could allow designers to search for new levels given an input level and an objective. While similar to the aforementioned latent vector evolution methods, here we specifically refer to queries of the form: *generate new level given input level X, metric Y and comparison condition Z*. In other words, this would enable users to generate levels that are similar/dissimilar to an input level using a given metric. Examples of this can be found in our prior work [52].

### D. Conditioned Generation

While above methods describe generating desired levels by searching the latent space via vector evolution, this can also be done by directly conditioning generation on either an input segment or a label, i.e. the model generates desired levels without having to use latent search. For example, a model can be trained to predict the next segment of a level given the current segment. Examples of such generations are given in

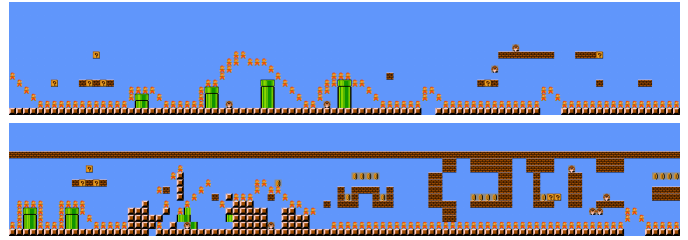


Fig. 3. Mario levels generated from an initial given segment. The top level is generated using an initial segment taken from the original Level 1-1. The bottom level is generated using a custom initial segment.

Figure 3 using an approach from our prior work [105]. Such a model could be used co-creatively to generate additional content based on designer input. Alternatively, conditional VAEs allow generation to be conditioned on labels provided during training such that the model can generate levels using these labels. Such techniques could allow users to choose to generate new levels by selecting from a list of labels.

### E. Latent Space Visualization

t-distributed Stochastic Neighbor Embedding or t-SNE [106] is a dimensionality reduction technique that allows visualizing datapoints in a high-dimensional space by assigning them to coordinates in a lower-dimensional space, usually 2D or 3D. Within creative ML, it has been used to cluster images, paintings and audio clips based on features learned by trained models [107], [108]. In game design however, the use of t-SNE has not been as widespread with Zhang et al.’s [109] use of t-SNE to visualize clusters of videogame moments being an example of its use in games. Similar to clustering related game moments, and analogous to clustering paintings and audio based on features, t-SNE could be used to visualize clusters of similar levels. Such visualizations, particularly when made interactive, could allow designers to explore the learned latent space and interactively search for desired levels and other content. The t-SNE visualizations of encodings of level segments from the original games are shown in Figure 4. These depict clusters of level segments that share structural properties as seen by the Mario segments with pipes in the lower middle part and with pyramid like structures in the lower left part of the Mario visualization. Similar structure-based groupings can be observed for *Kid Icarus* as well. Additionally, the plot for both games depicts the two games in separate clusters but more usefully, one may expect that the segments along the edge between the two clusters may be amenable for blending or combination. Thus, these visualizations can help designers interactively search the latent space for blended segments that are learned by models trained on data from multiple games. Overall, such plots help visualize the landscape of the latent space learned by the VAEs. Interactive versions of these visualizations, such as Taylor Denouden’s interactive visualization of a VAE trained on MNIST digits [110], could let users explore and search the latent space to find desired levels. Thus, we wish to develop such interactive visualizations as components of



Fig. 4. From left to right, t-SNE visualizations for training segments using VAEs trained on only *Super Mario Bros.*, only *Kid Icarus*, and both.

GDCML tools. Future work could also look into using other visualization techniques such as UMAP [111] which has been shown to exhibit certain benefits over t-SNE [112].

## VI. PROPOSED SYSTEM

We intend to implement a system that enables the above applications, focusing on 2D-side scrolling platformers from the NES era. In this section, we discuss its proposed features and functionalities in more detail.

### A. Platform and Interface

The existing tools we draw the most inspiration from are *Magenta Studio* [27], Google’s co-creative music tool based on their MusicVAE model [65]; *RunwayML* [57], a recently released application that lets users run pretrained creative ML models for performing a variety of artistic tasks; and the previously described *Morai Maker* [51]. Similar to how *Magenta Studio* is a suite of plugins for Ableton Live, a popular digital audio workstation, we envision our tool to consist of a suite of modules for popular game development engines, with each module corresponding to one of the applications mentioned previously. Additionally, like how *RunwayML* streamlines the use of existing models for artistic applications, we intend to do the same for game design applications.

### B. Modules

Our proposed system would consist of two sets of ML models—single domain, i.e. trained on a single game, and multi-domain, i.e. trained on multiple games taken together. Thus, modules pertaining to game blending would apply only for the multi-domain models while all other modules would be applicable for both. For single-domain models, users would select which game they want to work with while for multi-domain, they would select which subset of all possible games to work with. We currently envision the following modules corresponding to the previously discussed applications:

- *Generate* - Generate a level of user-defined length starting from a random segment. The system would pick a random

vector in the latent space and forward it through the VAE’s decoder until the desired size is reached.

- *Continue* - Generate a level of user-defined length starting from a segment selected or defined by the user. Here the user supplies the segment (or selects it from a group of existing ones). This is encoded into a latent space vector and then generation proceeds as in *Generate*.
- *Interpolate* - The user defines or selects two segments, both are encoded into latent space and interpolation is done between the resulting vectors to generate new segments.
- *Search* - This would be associated with a number of metrics and search criteria. The user defines/selects an input segment, metric and criterion which the module then uses to return a new segment based on the search results.
- *Condition* - This would use a set of conditional VAEs each trained using separate sets of labels such as those based on level elements and level structures. In the multi-domain case, labels could correspond to separate games.
- *Visualize* - A latent space visualizer which would allow the user to interactively explore the latent space to find segments and content to their liking.

Note that the first 3 modules above are analogous to the similarly named plugins in *Magenta Studio* that respectively sample random bars from MusicVAE, extend a given music sequence using an LSTM model and interpolate between pairs of sequences using MusicVAE.

In addition to the above, for the multi-domain case, we presently plan on two blending modules:

- *Blend Canvas* - This is partly inspired by *Beat Blender* [113], which allows users to interactively define and blend drum beats. We envision this module to be a similar interactive palette that lets users specify which subset of games to blend together. This would then form the single domain which can then use all of the above modules. Multi-domain models can be built either by combining single-domain models or by training a model on all available games and then extracting attribute vectors corresponding to specific games by averaging that game’s latent vectors. Extracted vectors could then be combined to form the desired subset,

for e.g.  $\text{Vector}_{\text{Mario}} + \text{Vector}_{\text{KidIcarus}} - \text{Vector}_{\text{MegaMan}}$ .

- *Blend Progression* - A second blending module would allow users to select desired games to blend and then specify the blend proportions as the level progresses. For example, users could choose to blend Mario and Icarus and then specify that the first  $x\%$  of the level should be  $a\%$  Mario and  $b\%$  Icarus, the next  $y\%$  should be  $c\%$  Mario and  $d\%$  Icarus and so on. Such a module would necessitate solving interesting challenges related to generating specific blend proportions, ensuring playability, and level layout.

Tools similar to *Beat Blender* such as *Melody Mixer* [114] and *Latent Loops* [115] could also inspire modules, particularly focusing on mechanics and gameplay. This would be similar to Hoyt et al.’s [116] path-based enhancements to the *Morai Maker*. Compared to levels, there has been less PCGML work on mechanics with Guzdial and Riedl’s [101] learning of game rules from video being among the few.

### C. ML Models and Data

We intend to use different variants of VAEs trained on level data from the VGLC for all our models. These would include VAEs trained to reconstruct input segments like our prior work in [49] as well as VAEs trained to generate the next segment given the current segment like our prior work in [105]. Further, we wish to explore conditional VAEs [117] to enable some of the desired functionality described previously.

## VII. CONCLUSION AND FUTURE DIRECTIONS

In this paper, we focused on existing and potential applications for game design via creative ML (GDCML). We discussed how creative ML approaches in visual art and music can inform similar approaches in games, highlighted applications that can be enabled by the affordances of creative PCGML models and gave a brief description of a proposed system. Apart from implementing this system, there are a number of interesting avenues to consider for future work.

### A. Latent Space Disentanglement

We discussed methods of enabling controllable generation via latent vector evolution and training conditional models. Increased controllability can also be achieved by learning disentangled representations, i.e. different dimensions of the latent space encoding different attributes. Future work could thus look at leveraging such disentanglement approaches [118].

### B. Datasets

A major challenge in PCGML is the lack of large datasets especially compared to other creative domains. Much of the advancements in these fields is due to the ability to train increasingly complex ML models on massive, publicly available datasets. While the VGLC has been a useful resource for PCGML work, it is tiny compared to traditional ML datasets. Moreover, most games only contain a handful of levels, thus necessitating models to be trained on segments of levels rather than the levels themselves to account for data scarcity. Hence,

future work could look at assembling larger datasets for games and for game content not restricted to levels. A step in this direction is the Video Game Affordances Corpus [119], which attempts to gather data about the affordances and interactions enabled by game levels. Sites such as the Video Game Atlas [120] could also serve as a source for building game datasets.

### C. Blending Genres

Most blending and domain transfer work in PCGML has focused on games of one genre, particularly platformers and even for our proposed system, we intend to focus on platformers. However, it would be interesting to blend games from different genres. What would a combination of Mario and Zelda look and play like, for example? The previously mentioned *SuperMash* attempts to do this by treating one genre as primary and the other as secondary, and draws elements from the latter to incorporate into the former. Can we build systems that can help design such games? This likely requires new forms of game representation that can simultaneously encode information about different genres. For example, platformers are amenable to text-based representations while dungeons are better represented using graphs. Are there representations that offer the best of both these worlds? Investigating such questions could help bring GDCML approaches closer to their counterparts in music and visual art.

## REFERENCES

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, 2009, pp. 248–255.
- [3] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [4] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative Adversarial Nets,” in *Advances in Neural Information Processing Systems*, 2014.
- [6] D. P. Kingma and M. Welling, “Auto-encoding Variational Bayes,” in *The 2nd International Conference on Learning Representations (ICLR)*, 2013.
- [7] D. J. Rezende, S. Mohamed, and D. Wierstra, “Stochastic backpropagation and approximate inference in deep generative models,” *arXiv preprint arXiv:1401.4082*, 2014.
- [8] L. A. Gatys, A. S. Ecker, and M. Bethge, “A neural algorithm of style,” *arXiv preprint arXiv:1508.06576*, 2015.
- [9] X. Huang and S. Belongie, “Arbitrary style transfer in real-time with adaptive instance normalization,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1501–1510.
- [10] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens, “Exploring the structure of a real-time, arbitrary neural artistic stylization network,” *arXiv preprint arXiv:1705.06830*, 2017.
- [11] L. Gatys, A. S. Ecker, and M. Bethge, “Texture synthesis using convolutional neural networks,” in *Advances in neural information processing systems*, 2015, pp. 262–270.
- [12] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky, “Texture networks: Feed-forward synthesis of textures and stylized images,” in *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, 2016.
- [13] P. Isola, J.-Y. Zhu, T. Zhou, and A. Efros, “Image-to-image translation with conditional adversarial networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

- [14] J.-Y. Zhu, T. Park, P. Isola, and A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [15] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, "Deep learning techniques for music generation—a survey," *arXiv preprint arXiv:1709.01620*, 2017.
- [16] G. Hadjeres, F. Pachet, and F. Nielsen, "DeepBach: A steerable model for Bach chorales generation," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017, pp. 1362–1371.
- [17] N. Trieu and R. M. Keller, "JazzGAN: Improvising with generative adversarial networks," in *Proc. of the 6th International Workshop on Musical Metacreation (MUME)*, 2018.
- [18] H. Chu, R. Urtasun, and S. Fidler, "Song from PI: A musically plausible network for pop music generation," *arXiv preprint arXiv:1611.03477*, 2016.
- [19] Z. Zukowski and C. Carr, "Generating black metal and math rock: Beyond Bach, Beethoven, and Beatles," *arXiv preprint arXiv:1811.06639*, 2018.
- [20] R. Pieters and S. Winiger, "Creative AI: On the democratisation and escalation of creativity," <https://medium.com/@creativeai/creativeai-9d4b2346faf3>, 2016.
- [21] D. Ulyanov, "Fast neural doodle," <https://github.com/DmitryUlyanov/fast-neural-doodle>, 2016.
- [22] A. J. Champandard, "Semantic style transfer and turning two-bit doodles into fine artworks," *arXiv preprint arXiv:1603.01768*, 2016.
- [23] D. Bau, J.-Y. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, and A. Torralba, "GAN dissection: Visualizing and understanding generative adversarial networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.
- [24] C. Hesse, "Image-to-image demo: Interactive image translation with pix2pix-tensorflow," <https://affinelayer.com/pixsrv/index.html>, 2017.
- [25] OpenDotLab, "Invisible cities," <https://opendot.github.io/ml4a-invisible-cities/>, 2016.
- [26] L. Karacan, Z. Akata, A. Erdem, and E. Erdem, "Manipulating attributes of natural scenes via hallucination," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 1, 2019.
- [27] A. Roberts, J. Engel, Y. Mann, J. Gillick, C. Kayacik, S. Nørly, M. Dinculescu, C. Radebaugh, C. Hawthorne, and D. Eck, "Magenta studio: Augmenting creativity with deep learning in ableton live," in *Proceedings of the International Workshop on Musical Metacreation (MUME)*, 2019.
- [28] C. Donahue, I. Simon, and S. Dieleman, "Piano genie," in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, 2019, pp. 160–164.
- [29] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "Gansynth: Adversarial neural audio synthesis," *arXiv preprint arXiv:1902.08710*, 2019.
- [30] N. Shaker, J. Togelius, and M. Nelson, *Procedural Content Generation in Games*. Springer International Publishing, 2016.
- [31] J. Togelius, G. Yannakakis, K. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172–186, 2011.
- [32] G. Smith, J. Whitehead, and M. Mateas, "Tanagra: Reactive planning and constraint solving for mixed-initiative level design," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 201–215, 2011.
- [33] A. Smith and M. Mateas, "Answer set programming for procedural content generation," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 187–200, 2011.
- [34] L. Johnson, G. N. Yannakakis, and J. Togelius, "Cellular automata for real-time generation of infinite cave levels," in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, 2010, pp. 1–4.
- [35] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgård, A. K. Hoover, A. Isaksen, A. Nealen, and J. Togelius, "Procedural content generation via machine learning (PCGML)," *IEEE Transactions on Games*, vol. 10, no. 3, pp. 257–270, 2018.
- [36] A. Summerville and M. Mateas, "Super Mario as a string: Platformer level generation via LSTMs," in *Proceedings of the 1st International Joint Conference on DiGRA and FDG*, 2016.
- [37] M. Guzdial and M. Riedl, "Game level generation from gameplay videos," in *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2016.
- [38] S. Snodgrass and S. Ontañón, "Experiments in map generation using Markov Chains," in *Proceedings of the 9th International Conference on the Foundations of Digital Games*, 2014.
- [39] V. Volz, J. Schrum, J. Liu, S. Lucas, A. Smith, and S. Risi, "Evolving Mario levels in the latent space of a deep convolutional generative adversarial network," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018.
- [40] R. Jain, A. Isaksen, C. Holmgård, and J. Togelius, "Autoencoders for level generation, repair and recognition," in *Proceedings of the ICCG Workshop on Computational Creativity and Games*, 2016.
- [41] Nintendo, "Super Mario Bros." Game [NES], 1985.
- [42] —, "The Legend of Zelda," Game [NES], 1986.
- [43] id Software, "Doom," Game [PC], 1993.
- [44] M. J. Guzdial and M. O. Riedl, "Combinatorial creativity for procedural content generation via machine learning," in *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [45] S. Snodgrass and S. Ontañón, "An approach to domain transfer in procedural content generation of two-dimensional videogame levels," in *Proceedings of the Twelfth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-17)*, 2017.
- [46] S. Snodgrass and A. Sarkar, "Multi-domain level generation and blending with sketches via example-driven BSP and variational autoencoders," in *Proceedings of the 15th Conference on the Foundations of Digital Games*, 2020.
- [47] M. Guzdial and M. Riedl, "Learning to blend computer game levels," in *Proceedings of the Seventh International Conference on Computational Creativity*, 2016.
- [48] A. Sarkar and S. Cooper, "Blending levels from different games using LSTMs," in *Proceedings of the AIIDE Workshop on Experimental AI in Games*, 2018.
- [49] A. Sarkar, Z. Yang, and S. Cooper, "Controllable level blending between games using variational autoencoders," in *Proceedings of the AIIDE Workshop on Experimental AI in Games*, 2019.
- [50] M. Guzdial and M. Riedl, "Automated game design via conceptual expansion," in *Proceedings of the Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2018.
- [51] M. Guzdial, N. Liao, and M. Riedl, "Co-creative level design via machine learning," *arXiv preprint arXiv:1809.09420*, 2018.
- [52] A. Sarkar, "Game design using creative AI," in *NeurIPS Workshop on Machine Learning for Creativity and Design*, 2019.
- [53] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song, "Neural style transfer: A review," *IEEE transactions on visualization and computer graphics*, 2019.
- [54] J. Johnson, "neural-style," <https://github.com/jcjohnson/neural-style>, 2015.
- [55] ml4a, "Pix2pix," <https://ml4a.github.io/guides/Pix2Pix/>, 2016.
- [56] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," <https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html>, 2015.
- [57] "Runwayml," <https://runwayml.com/>, 2018.
- [58] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," *arXiv preprint arXiv:1805.08318*, 2018.
- [59] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.
- [60] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401–4410.
- [61] A. Hertzmann, "Aesthetics of neural network art," *arXiv preprint arXiv:1903.05696*, 2019.
- [62] A. V. D. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.
- [63] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation," *arXiv preprint arXiv:1703.10847*, 2017.
- [64] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

- [65] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long term structure in music," in *International Conference on Machine Learning*, 2018.
- [66] J. Engel, M. Hoffman, and A. Roberts, "Latent constraints: Learning to generate conditionally from unconditional generative models," *arXiv preprint arXiv:1711.05772*, 2017.
- [67] A. Roberts, J. Engel, S. Oore, and D. Eck, "Learning latent representations of music to generate interactive musical palettes." in *IUI Workshops*, 2018.
- [68] M. Dinculescu, J. Engel, and A. Roberts, "Midime: Personalizing a musicvae model with user data," in *NeurIPS Workshop on Machine Learning for Creativity and Design*, 2019.
- [69] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.
- [70] G. Yannakakis, A. Liapis, and C. Alexopoulos, "Mixed-initiative co-creativity," in *Foundations of Digital Games Conference*, 2014.
- [71] N. Shaker, M. Shaker, and J. Togelius, "Ropossum: An authoring tool for designing, optimizing and solving Cut the Rope levels," in *Proceedings of the Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2013.
- [72] ZeptoLab, "Cut the Rope," Game, 2010.
- [73] A. Liapis, G. Yannakakis, and J. Togelius, "Sentient Sketchbook: Computer-aided game level authoring," in *Proceedings of the 8th International Conference on the Foundations of Digital Games*, 2013.
- [74] T. Machado, D. Gopstein, A. Nealen, O. Nov, and J. Togelius, "AI-assisted game debugging with Cicero," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–8.
- [75] A. Alvarez, S. Dahlskog, J. Font, and J. Togelius, "Interactive constrained MAP-Elites analysis and evaluation of the expressiveness of the feature dimensions," *arXiv preprint arXiv:2003.03377*, 2020.
- [76] M. Charity, A. Khalifa, and J. Togelius, "Baba is y'all: Collaborative mixed-initiative level design," *arXiv preprint arXiv:2003.14294*, 2020.
- [77] Hempuli, "Baba Is You," Game, 2019.
- [78] M. Cook, J. Gow, and S. Colton, "Danesh: Helping bridge the gap between procedural generators and their output," in *7th Workshop on Procedural Content Generation*. Goldsmiths, University of London, 2016.
- [79] G. Smith, "Understanding Procedural Content Generation: A design-centric analysis of the role of PCG in games," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2014.
- [80] M. Cook, S. Colton, and J. Gow, "The ANGELINA videogame design system—part I," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 9, no. 2, pp. 192–203, 2016.
- [81] —, "The ANGELINA videogame design system—part II," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 9, no. 3, pp. 254–266, 2016.
- [82] M. A. Boden, *The Creative Mind: Myths and Mechanisms*. Psychology Press, 2004.
- [83] M. Popova, "Networked knowledge and combinatorial creativity," <https://www.brainpickings.org/2011/08/01/networked-knowledge-combinatorial-creativity/>, 2011.
- [84] —, "How Einstein thought: Why combinatory play is the secret genius," <https://www.brainpickings.org/2013/08/14/how-einstein-thought-combinatorial-creativity/>, 2013.
- [85] K. Ferguson, "Creativity is a remix," <https://www.youtube.com/watch?v=zd-dqUuvLk4>, 2012.
- [86] A. Kleon, "Steal like an artist," <https://www.youtube.com/watch?v=oww7oB9rjgw>, 2012.
- [87] K. Kowalski, "The art of new ideas: Combinatorial creativity and why everything is a remix," <https://www.sloww.co/combinatorial-creativity/>, 2019.
- [88] Nintendo, "Metroid," Game [NES], 1986.
- [89] Mossmouth, "Spelunky," Game [PC], 2008.
- [90] DigitalContinue, "SuperMash," Game, 2019.
- [91] G. Fauconnier and M. Turner, "Conceptual integration networks," *Cognitive Science*, vol. 22, no. 2, pp. 133–187, 1998.
- [92] S. Ontañón and E. Plaza, "Amalgams: A formal approach for combining multiple case solutions," in *International Conference on Case-Based Reasoning*, 2010.
- [93] W. Wilke and R. Bergmann, "Techniques and knowledge used for adaptation during case-based problem solving," in *Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, 1998.
- [94] M. Guzdial and M. Riedl, "Conceptual game expansion," *arXiv preprint arXiv:2002.09636*, 2020.
- [95] J. Gow and J. Corneli, "Towards generating novel games using conceptual blending," in *Proceedings of the Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2015.
- [96] Nintendo, "Kid Icarus," Game [NES], 1986.
- [97] J. Gutierrez and J. Schrum, "Generative adversarial network rooms in generative graph grammar dungeons for the legend of zelda," *arXiv preprint arXiv:2001.05065v1*, 2020.
- [98] S. Thakkar, C. Cao, L. Wang, T. J. Choi, and J. Togelius, "Autoencoder and evolutionary algorithm for level generation in lode runner," in *IEEE Conference on Games*, 2019.
- [99] E. Giacomello, P. L. Lanzi, and D. Lioacono, "Doom level generation using generative adversarial networks," in *IEEE Games*, 2018.
- [100] J. Schrum, J. Gutierrez, V. Volz, J. Liu, S. Lucas, and S. Risi, "Interactive evolution and exploration within latent level-design space of generative adversarial networks," *arXiv preprint arXiv:2004.00151*, 2020.
- [101] M. Guzdial, B. Li, and M. Riedl, "Game engine learning from video," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017.
- [102] S. Snodgrass, "Levels from sketches with example-driven binary space partition," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 15, no. 1, 2019, pp. 73–79.
- [103] A. Summerville, S. Snodgrass, M. Mateas, and S. Ontañón, "The VGLC: The Video Game Level Corpus," *Proceedings of the 7th Workshop on Procedural Content Generation*, 2016.
- [104] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.
- [105] A. Sarkar and S. Cooper, "Sequential segment-based level generation and blending using variational autoencoders," in *Proceedings of the 11th Workshop on Procedural Content Generation in Games*, 2020.
- [106] L. V. D. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.
- [107] P. Hamel and D. Eck, "Learning features from music audio with deep belief networks," in *11th International Society for Music Information Retrieval Conference*, 2010.
- [108] C. Carr and Z. Zukowski, "Curating generative raw audio music with D.O.M.E." in *Joint Proceedings of the ACM IUI 2019 Workshops*, 2019.
- [109] X. Zhang, Z. Zhan, M. Holtz, and A. Smith, "Crawling, indexing and retrieving moments in videogames," in *Proceedings of the Foundations of Digital Games*, 2018.
- [110] T. Denouden, "VAE latent space explorer," <https://denouden.dev/VAE-Latent-Space-Explorer/>, 2018.
- [111] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.
- [112] E. Becht, C. Dutertre, I. W. H. Kwok, L. Ng, F. Ginhoux, and E. Newell, "Evaluation of UMAP as an alternative to t-SNE for single cell data," in *bioRxiv*, 298430, 2018.
- [113] K. Phillips, T. Blankensmith, and A. Roberts, "Beat blender," <https://experiments.withgoogle.com/ai/beat-blender/view/>, 2018.
- [114] T. Blankensmith, "Melody mixer," <https://experiments.withgoogle.com/ai/melody-mixer/view/>, 2018.
- [115] "Latent loops," <https://teampieshop.github.io/latent-loops/>, 2018.
- [116] A. Hoyt, M. Guzdial, Y. Kumar, G. Smith, and M. O. Riedl, "Integrating automated play in level co-creation," *arXiv preprint arXiv:1911.09219*, 2019.
- [117] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in neural information processing systems*, 2015, pp. 3483–3491.
- [118] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-VAE: Learning basic visual concepts with a constrained variational framework," *Proceedings of the International Conference on Learning Representations*, 2017.
- [119] G. R. Bentley and J. C. Osborn, "The videogame affordances corpus," in *2019 Experimental AI in Games Workshop*, 2019.
- [120] "The video game atlas," <https://vgmaps.com>.