A Survey of Procedural Content Generation via Machine Learning (PCGML) in Game Development: Approaches, Challenges, and Applications

George Kotti IV
Department of Computer Science and Engineering
Mississippi State University
Starkville, United States
gkotti4@gmail.com

***Abstract*— Procedural Content Generation via Machine Learning (PCGML) has emerged as a transformative approach to dynamically generating game assets, levels, textures, and other content elements. Traditional procedural generation techniques rely on deterministic algorithms, but the integration of machine learning models—such as Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Reinforcement Learning (RL)—has enabled more adaptive, diverse, and player-responsive content creation. This survey explores the state-of-the-art in PCGML, providing a detailed review of approaches, methodologies, and applications in gaming. A taxonomy is presented to classify different machine learning models in PCG, alongside a chronological overview of major developments. The survey also discusses challenges, such as computational constraints, evaluation metrics, and content controllability. Finally, we highlight emerging applications beyond gaming, including architecture, simulation training, and extended reality (XR) environments.**

Keywords—PCGML, Machine Learning, Procedural Content Generation, Generative Adversarial Network, Variational Autoencoder, Latent Space

I. Introduction (*Heading 1*)

Procedural Content Generation (PCG) has been a staple in the expansion of game development, enabling the creation of large, complex environments with minimal human intervention. Traditional PCG methods, such as Perlin noise for terrain generation or L-systems for plant growth, operate based on predefined rules. However, these deterministic approaches struggle to balance diversity, adaptability, and control—a key challenge in modern game development.

Machine Learning (ML) has revolutionized PCG by enabling data-driven content generation that adapts to user input, gameplay patterns, and creative constraints. The rise of GANs, VAEs, RL, and Transformer-based models has introduced novel ways to generate and adapt procedural content. This survey explores how ML models are applied in level design, texture generation, character modeling, and dynamic storytelling within procedural content generation.

Research Questions:

1. How do different ML models contribute to PCG?
2. What challenges remain in applying ML for game content generation?
3. How do evaluation metrics for PCGML compare to traditional PCG?
4. What are the broader applications of PCGML beyond gaming?

II. Background

A. *Procedural Content Generation (PCG)*

PCG refers to the automatic generation of content with minimal human intervention. Traditional PCG approaches include random generation, grammar-based methods, and search-based techniques. However, these methods often struggle with adaptability and producing high-quality, diverse content

B. *PCG via Machine Learning (PCGML)*

PCGML applies ML algorithms to procedural generation, enabling models to learn from existing game data and generate new content. The most common ML models used for PCG include:

- **Generative Adversarial Networks (GANs)** – Used for generating textures, game assets, and levels.
- **Variational Autoencoders (VAEs)** – Applied in level blending, character generation, and gameplay variation.
- **Reinforcement Learning (RL)** – Used for evolving game rules, procedural mechanics, and adaptive difficulty systems.
- **Transformers & Diffusion Models** – Emerging in text-to-game-content generation.

C. *Generative Adversarial Networks (GANs)*

GANs are a class of generative models that leverage adversarial training to learn high-dimensional data distributions. They consist of two networks: a generator (G) and a discriminator (D), which compete in a zero-sum game. GANs have been widely applied in areas such as image generation, super-resolution, and domain adaptation. [15] A generalized formula for the training process can be seen below:

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

Despite their success, GANs face several well-documented challenges, including training instability, mode

collapse, and the difficulty of robust evaluation. One of the primary reasons for instability arises from the adversarial nature of GAN training, where the generator and discriminator are engaged in a dynamic minimax optimization problem. As Goodfellow et al. and Salimans et al. observed, standard gradient descent methods often struggle with this formulation, as the optimization landscape contains high-dimensional saddle points rather than clear local minima, making convergence difficult [15]. Arjovsky et al. (2017) further demonstrated that, due to the non-overlapping support of real and generated data distributions in high-dimensional spaces, the discriminator can become too powerful, leading to vanishing gradients that hinder generator updates. Various techniques, such as Wasserstein loss (WGANs), gradient penalty regularization, and spectral normalization, have been proposed to address these issues by stabilizing training and ensuring meaningful gradient propagation. However, mode collapse—where the generator fails to capture the full diversity of the data distribution—remains a persistent challenge, often requiring architectural modifications or alternative loss formulations. Additionally, the absence of a reliable likelihood-based evaluation metric makes comparing different GAN variants difficult, leading to ongoing research into better quantitative and perceptual assessment methods [15, 16, 18].

1) GAN Variations

Variants of GANs include:

DCGAN (Deep Convolutional GAN)

- Introduced by Radford et al. (2016).
- Uses CNNs instead of fully connected layers.
- Generates higher-quality images.

WGAN (Wasserstein GAN)

- Introduced by Arjovsky et al. (2017).
- Uses Wasserstein distance instead of cross-entropy.
- Solves issues of mode collapse.

StyleGAN (Style-Based GAN)

- Introduced by Karras et al. (2018).
- Controls image style attributes at multiple levels.
- Used for high-resolution face synthesis.

VAE-GAN (Combining VAEs & GANs)

- Introduced by Larsen et al. (2016).
- Uses VAE's encoder-decoder structure with a GAN for better regularization.
- Balances variability (VAE) and realism (GAN).

D. Variable Autoencoders (VAEs)

Kingma & Welling (2013) introduced Variational Autoencoders (VAEs) as a probabilistic generative model that optimizes a latent variable framework using variational inference. Unlike GANs, VAEs explicitly model data distributions with a likelihood function and regularize the latent space using a KL divergence penalty. Their work demonstrated that VAEs can effectively generate diverse data samples while maintaining a structured and continuous latent space. They state that VAEs "introduce a reparameterization trick that allows backpropagation through stochastic variables, enabling deep generative modeling with efficient optimization" [38]. While VAEs offer stability in training and meaningful representations, they often produce blurrier samples than GANs due to their likelihood-based optimization. [36 39]

The internals of a VAE "can be viewed as two coupled, but independently parameterized models: the encoder or recognition model, and the decoder or generative model." [38] The process of compressing and reconstructing data in VAEs follows a basic scheme around the encoder and decoder. The model receives input, x. The encoder compresses it into the latent space, sometimes referred to as code. The decoder receives as input the information sampled from the latent space and produces x' as closely to x as possible—many times used to reconstruct clean versions of noisy inputs.

III. Broader Applications

PCGML extends beyond gaming into various domains:

- **Architecture & Urban Planning** – Procedural generation of cityscapes using GANs.
- **XR & Augmented Reality** – Adaptive AR environments with RL-enhanced PCG.
- **Education & Training Simulations** – ML-driven PCG for generating realistic training environments

IV. Taxonomy

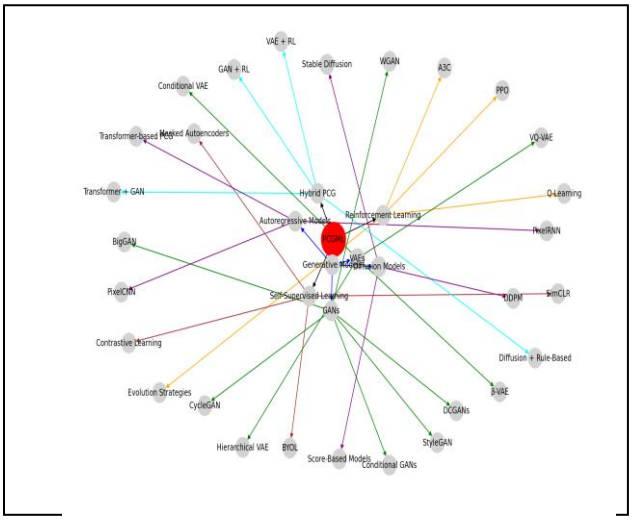

Fig. 1. Taxonomy of PCGMLs Technical Aspects

V. Chronological Overview

TABLE I.

Year	PCGML Overview	
	Contribution	***Contributor***
1980-1990	Rule-based, deterministic algorithms used for PCG (e.g., Perlin Noise for terrain generation in 1983)	n/a
1983	Perlin Noise – used in terrain generation	Ken Perlin
1987	L-Systems – Procedural generation of plants/trees	Aristid Lindenmayer

Year	PCGML Overview	
	Contribution	***Contributor***
1996	Genetic Algorithms (GAs) in PCG	John H. Holland
1998	Cellular Automata for Dungeon Generation	Bay 12 Games
2005	Markov Chains for 2D level generation	Jorg Schroder, Julian Togelius
2007	Monte Carlo Tree Search (MCTS) – applied to procedural content placement	Remi Coulom
2009	Neuroevolution in PCG – Neural networks (NEAT) were proposed for level generation in Super Mario Bros	Kenneth O. Stanley
2010	Bayesian Networks for predictive modeling in structured game maps	Sebastian Risi, Julian Togelius
2013	Reinforcement learning for level design	DeepMind, Google
2013	Introduced VAEs	Kingma & Welling
2014	Introduced GANs	Goodfellow et al.
2015	VAEs used for level generation – encoding level layouts	Kingma & Welling, Google DeepMind
2016	Deep Convolutional GANs (DCGANs) for Texture Generation	Radford et al., OpenAI
2017	Unified VAEs and GANs	Mescheder et al.
2018	LSTM Networks for Tile-based PCG	Nintendo AI Lab, Julian Togelius
2019	Self-Organizing Maps (SOMs) for Game Content Analysis – Clustering PCG results	Unity AI Research
2020	GANs for Super Mario level Generation	Green et al.
2021	VAEs for adaptive level generation	Truong et al.
2021	Diffusion Models for Texture & Asset Generation	Google DeepMind, NVIDIA
2022	Self-Supervised Learning (SSL) introduced to PCG	Facebook AI, MIT-IBM Watson Lab
2024	Multi-Agent PCGML & Autonomous Game Design – fully automated procedural pipelines	DeepMind, NVIDIA Omniverse

VI. Conclusion

Procedural Content Generation via Machine Learning (PCGML) has evolved significantly, leveraging advancements in Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), Reinforcement Learning (RL), and Self-Supervised Learning (SSL). These approaches have enabled the creation of high-quality, diverse, and adaptive content in games, from textures and levels to dynamic narratives and character behaviors. Despite these advancements, challenges such as mode collapse in GANs, interpretability in VAEs, and sample efficiency in RL remain key hurdles. Future research should focus on hybrid models, improved training stability, and better evaluation metrics to enhance controllability and creativity in PCGML. As the field progresses, these innovations will contribute to more immersive, scalable, and player-adaptive game worlds.

References

[1] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, "Procedural content generation for games," ACM Transactions on Multimedia Computing, Communications, and Applications, vol. 9, no. 1, pp. 1–22, Feb. 2013, doi: https://doi.org/10.1145/2422956.2422957.

[2] M. Freiknecht, "Procedural Content Generation for Games." Available: https://madoc.bib.uni-mannheim.de/59000/1/Procedural%20Content%20Generation%20for%20Games.pdf.

[3] A. Summerville et al., "Procedural Content Generation via Machine Learning (PCGML)," IEEE Transactions on Games, vol. 10, no. 3, pp. 257–270, Sep. 2018, doi: https://doi.org/10.1109/tg.2018.2846639.

[4] J. Liu, S. Snodgrass, A. Khalifa, S. Risi, G. N. Yannakakis, and J. Togelius, "Deep learning for procedural content generation," Neural Computing and Applications, vol. 33, no. 1, pp. 19–37, Oct. 2020, doi: https://doi.org/10.1007/s00521-020-05383-8.

[5] "Procedural Content Generation in Games: A Survey with Insights on Emerging LLM Integration," Arxiv.org, 2022. https://arxiv.org/html/2410.15644v1.

[6] "Procedural Content Generation via Generative Artificial Intelligence," Arxiv.org, 2019. https://arxiv.org/html/2407.09013v1?utm_source=chatgpt.com.

[7] "Reinforcement Learning-Enhanced Procedural Generation for Dynamic Narrative-Driven AR Experiences Accepted at GRAPP 2025 - 20th International Conference on Computer Graphics Theory and Applications," Arxiv.org, 2025. https://arxiv.org/html/2501.08552v1.

[8] S. Saffari, M. Dorrigiv, and F. Yaghmaee, "Harnessing Machine Learning for Procedural Content Generation in Gaming: A Comprehensive Review," Technology Journal of Artificial Intelligence and Data Mining, vol. 12, no. 4, pp. 583–597, 2024, doi: https://doi.org/10.22044/jadm.2025.15016.2603.

[9] M. C. Green, L. Mugrai, A. Khalifa, and J. Togelius, "Mario Level Generation From Mechanics Using Scene Stitching," arXiv.org, 2020. https://arxiv.org/abs/2002.02992.

[10] A. Sarkar, Z. Yang, and S. Cooper, "Conditional Level Generation and Game Blending," arXiv.org, 2020. https://arxiv.org/abs/2010.07735.

[11] I. Srivastava, "A comparative analysis of generative models for terrain generation in open-world video games," Journal of High School Science, vol. 8, no. 1, pp. 120–143, Feb. 2024, Accessed: Feb. 21, 2025. [Online]. Available: https://jhss.scholasticahq.com/article/92856-a-comparative-analysis-of-generative-models-for-terrain-generation-in-open-world-video-games.

[12] "A Case Study of Generative Adversarial Networks for Procedural Synthesis of Original Textures in Video Games | IEEE Conference Publication | IEEE Xplore," ieeexplore.ieee.org. https://ieeexplore.ieee.org/document/8712070.

[13] L. Z. Kelvin and Anand Bhojan, "Procedural Generation of Roads with Conditional Generative Adversarial Networks," pp. 1–2, Aug. 2020, doi: https://doi.org/10.1145/3388770.3407422.

[14] M .-Y. Liu, X. Huang, J. Yu, T.-C. Wang, and A. Mallya, "Generative Adversarial Networks for Image and Video Synthesis: Algorithms and Applications," Proceedings of the IEEE, pp. 1–24, 2021, doi: https://doi.org/10.1109/jproc.2021.3049196.

[15] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative Adversarial Networks: An Overview," IEEE Signal Processing Magazine, vol. 35, no. 1, pp. 53–65, Jan. 2018, doi: https://doi.org/10.1109/msp.2017.2765202.

[16] A. Aggarwal, M. Mittal, and G. Battineni, "Generative adversarial network: An overview of theory and applications," International Journal of Information Management Data Insights, vol. 1, no. 1, p. 100004, Jan. 2021, doi: https://doi.org/10.1016/j.jjimei.2020.100004.

[17] M. Ben-Yosef and D. Weinshall, "Gaussian Mixture Generative Adversarial Networks for Diverse Datasets, and the Unsupervised Clustering of Images," arXiv.org, 2018. https://arxiv.org/abs/1808.10356.

[18] A. Wulff-Jensen, N. N. Rant, T. N. Møller, and J. A. Billeskov, "Deep Convolutional Generative Adversarial Network for Procedural 3D Landscape Generation Based on DEM," Springer eBooks, pp. 85–94, Jan. 2018, doi: https://doi.org/10.1007/978-3-319-76908-0_9.

[19] A. Hald, J. S. Hansen, J. Kristensen, and Paolo Burelli, "Procedural Content Generation of Puzzle Games using Conditional Generative Adversarial Networks," arXiv (Cornell University), pp. 1–9, Sep. 2020, doi: https://doi.org/10.1145/3402942.3409601.

[20] K. Ping and Luo Dingli, “Conditional Convolutional Generative Adversarial Networks Based Interactive Procedural Game Map Generation,” Advances in intelligent systems and computing, pp. 400–419, Jan. 2020, doi: https://doi.org/10.1007/978-3-030-39445-5_30.

[21] inus Gisslén, A. Eakins, C. Gordillo, J. Bergdahl, and Konrad Tollmar, “Adversarial Reinforcement Learning for Procedural Content Generation,” arXiv (Cornell University), Aug. 2021, doi: https://doi.org/10.1109/cog52621.2021.9619053.

[22] M. Saito, E. Matsumoto, and S. Saito, “Temporal Generative Adversarial Nets with Singular Value Clipping,” International Conference on Computer Vision, Oct. 2017, doi: https://doi.org/10.1109/iccv.2017.308.

[23] Z. Lin, A. Khetan, G. Fanti, and S. Oh, “PacGAN: The power of two samples in generative adversarial networks,” arXiv.org, 2017. https://arxiv.org/abs/1712.04086.

[24] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral Normalization for Generative Adversarial Networks,” arXiv:1802.05957 [cs, stat], Feb. 2018, Available: https://arxiv.org/abs/1802.05957.

[25] I. J. Goodfellow et al., “Generative Adversarial Networks,” arXiv.org, Jun. 10, 2014. https://arxiv.org/abs/1406.2661.

[26] B. Gan, Y. Saatchi, and A. Wilson, Accessed: Feb. 21, 2025. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/312351bff07989769097660a56395065-Paper.pdf.

[27] Z. Guo, H. Liu, Y.-S. Ong, X. Qu, Y. Zhang, and J. Zheng, “Generative Multiform Bayesian Optimization,” IEEE Transactions on Cybernetics, vol. 53, no. 7, pp. 4347–4360, Jul. 2023, doi: https://doi.org/10.1109/tcyb.2022.3165044.

[28] T. S. Rodrigues and P. R. Pinheiro, “Hyperparameter Optimization in Generative Adversarial Networks (GANs) Using Gaussian AHP,” IEEE Access, pp. 1–1, Jan. 2024, doi: https://doi.org/10.1109/access.2024.3518979.

[29] Lars Mescheder, S. Nowozin, and A. Geiger, “Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks,” PMLR, pp. 2391–2400, Jul. 2017, Available: https://proceedings.mlr.press/v70/mescheder17a.html?ref=https://githubhelp.com.

[30] S. Vivekananthan, “Comparative Analysis of Generative Models: Enhancing Image Synthesis with VAEs, GANs, and Stable Diffusion,” arXiv.org, 2024. https://arxiv.org/abs/2408.08751.

[31] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, “Deep Generative Modelling: a Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 11, pp. 1–1, 2021, doi: https://doi.org/10.1109/TPAMI.2021.3116668.

[32] D. Bank, N. Koenigstein, and R. Giryes, “Autoencoders,” arXiv:2003.05991 [cs, stat], Apr. 2021, Available: https://arxiv.org/abs/2003.05991.

[33] Umberto Michelucci, “An Introduction to Autoencoders,” arXiv (Cornell University), Jan. 2022, doi: https://doi.org/10.48550/arxiv.2201.03898.

[34] P. Baldi, “Autoencoders, Unsupervised Learning, and Deep Architectures,” vol. 27, pp. 37–50, 2012, Available: https://proceedings.mlr.press/v27/baldi12a/baldi12a.pdf.

[35] Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, “Variational Autoencoders for Collaborative Filtering,” Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW ’18, 2018, doi: https://doi.org/10.1145/3178876.3186150.

[36] A. Cemgil et al., “The Autoencoding Variational Autoencoder.” Available: https://papers.nips.cc/paper/2020/file/ac10ff1941c540cd87c107330996f4f6-Paper.pdf.

[37] “Papers with Code - VAE Explained,” Paperswithcode.com, 2020. https://paperswithcode.com/method/vae.

[38] D. P. Kingma and M. Welling, “An Introduction to Variational Autoencoders,” Foundations and Trends® in Machine Learning, vol. 12, no. 4, pp. 307–392, 2019, doi: https://doi.org/10.1561/2200000056.

[39] D. P. Kingma and M. Welling, “Auto-Encoding Variational Bayes,” arXiv.org, Dec. 20, 2013. https://arxiv.org/abs/1312.6114.

[40] .-T. Truong, A. Salah, and H. W. Lauw, “Bilateral Variational Autoencoder for Collaborative Filtering,” Web Search and Data Mining, Mar. 2021, doi: https://doi.org/10.1145/3437963.3441759.

[41] A. Sarkar and S. Cooper, “Sequential Segment-based Level Generation and Blending using Variational Autoencoders,” arXiv.org, 2020. https://arxiv.org/abs/2007.08746?utm_source=chatgpt.com.

[42] S. Thakkar, C. Cao, L. Wang, Tae Jong Choi, and J. Togelius, “Autoencoder and Evolutionary Algorithm for Level Generation in Lode Runner,” 2019 IEEE Conference on Games (CoG), Aug. 2019, doi: https://doi.org/10.1109/cig.2019.8848076.

[43] A. Sarkar, Z. Yang, and S. Cooper, “Controllable Level Blending between Games using Variational Autoencoders,” arXiv (Cornell University), Feb. 2020, doi: https://doi.org/10.48550/arxiv.2002.11869.

[44] S. Saito, L. Wei, L. Hu, K. Nagano, and H. Li, “Photorealistic Facial Texture Inference Using Deep Neural Networks,” arXiv.org, 2016. https://arxiv.org/abs/1612.00523.

[45] Y. Song, L. Bao, S. He, Q. Yang, and M.-H. Yang, “Stylizing face images via multiple exemplars,” Computer Vision and Image Understanding, vol. 162, pp. 135–145, Sep. 2017, doi: https://doi.org/10.1016/j.cviu.2017.08.009.

[46] Y. Zhou et al., “HairNet: Single-View Hair Reconstruction using Convolutional Neural Networks,” arXiv.org, 2018. https://arxiv.org/abs/1806.07467.

[47] K. Wang, M. Savva, A. X. Chang, and D. Ritchie, “Deep convolutional priors for indoor scene synthesis,” ACM Transactions on Graphics, vol. 37, no. 4, pp. 1–14, Aug. 2018, doi: https://doi.org/10.1145/3197517.3201362.

[48] Anđelo Martinović and Luc Van Gool, “Bayesian Grammar Learning for Inverse Procedural Modeling,” CiteSeer X (The Pennsylvania State University), Jun. 2013, doi: https://doi.org/10.1109/cvpr.2013.33.

[49] İ. Demir and D. G. Aliaga, “Guided proceduralization: Optimizing geometry processing and grammar extraction for architectural models,” Computers & Graphics, vol. 74, pp. 257–267, Aug. 2018, doi: https://doi.org/10.1016/j.cag.2018.05.013.

[50] J. O. Talton, L. Yang, R. Kumar, M. Lim, N. D. Goodman, and R. Mech, “Learning design patterns with bayesian grammar induction,” CiteSeer X (The Pennsylvania State University), Oct. 2012, doi: https://doi.org/10.1145/2380116.2380127.

[51] T. Kelly, P. Guerrero, A. Steed, P. Wonka, and N. J. Mitra, “FrankenGAN,” ACM Transactions on Graphics, vol. 37, no. 6, pp. 1–14, Dec. 2018, doi: https://doi.org/10.1145/3272127.3275065.

[52] C. Beckham and C. Pal, “A step towards procedural terrain generation with GANs,” arXiv.org, 2017. https://arxiv.org/abs/1707.03383.

[53] C. Tang, K. Zhang, C. Xing, Y. Ding, and Z. Xu, “Perlin Noise Improve Adversarial Robustness,” arXiv.org, 2021. https://arxiv.org/abs/2112.13408.

[54] H.-J. Bae et al., “A Perlin Noise-Based Augmentation Strategy for Deep Learning with Small Data Samples of HRCT Images,” Scientific Reports, vol. 8, no. 1, p. 17687, Dec. 2018, doi: https://doi.org/10.1038/s41598-018-36047-2.

[55] H. Irobe et al., “Robust VAEs via Generating Process of Noise Augmented Data,” arXiv.org, 2024. https://arxiv.org/abs/2407.18632.

[56] T. Kaneko and T. Harada, “Noise Robust Generative Adversarial Networks,” arXiv.org, 2019. https://arxiv.org/abs/1911.11776.

[57] Y. Peng, “A Comparative Analysis Between GAN and Diffusion Models in Image Generation,” vol. 5, pp. 189–195, Aug. 2024, doi: https://doi.org/10.62051/0f1va465.

[58] S. J. Barigye, J. M. García de la Vega, and Y. Perez-Castillo, “Generative Adversarial Networks (GANs) Based Synthetic Sampling for Predictive Modeling,” Molecular Informatics, Jul. 2020, doi: https://doi.org/10.1002/minf.202000086.

[59] Pedro, R. Lorenz, R. P. Monti, E. Jones, and R. Leech, “Bayesian optimization for automatic design of face stimuli,” arXiv.org, 2020. https://arxiv.org/abs/2007.09989.

[60] C. Badrinath, U. Bhalla, A. Oesterling, S. Srinivas, and H. Lakkaraju, “All Roads Lead to Rome? Exploring Representational Similarities Between Latent Spaces of Generative Image Models,” arXiv.org, 2024. https://arxiv.org/abs/2407.13449?utm_source=chatgpt.com (accessed Feb. 21, 2025).

[61] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual Losses for Real-Time Style Transfer and Super-Resolution,” Computer Vision –

ECCV 2016, pp. 694–711, 2016, doi: https://doi.org/10.1007/978-3-319-46475-6_43.

[62] J. Togelius, G. N. Yannakakis, K. O. 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, Sep. 2011, doi: https://doi.org/10.1109/tciaig.2011.2148116.

[63] . Sarkar and S. Cooper, “Towards Game Design via Creative Machine Learning (GDCML),” arXiv.org, 2020. https://arxiv.org/abs/2008.13548.

[64] S. Snodgrass and S. Ontanon, “Procedural level generation using multi-layer level representations with MdMCs,” Aug. 2017, doi: https://doi.org/10.1109/cig.2017.8080447.

[65] X. Wei, B. Gong, Z. Liu, W. Lu, and L. Wang, “Improving the Improved Training of Wasserstein GANs: A Consistency Term and Its Dual Effect,” arXiv.org, 2018. https://arxiv.org/abs/1803.01541.

[66] T. Hu et al., “Complexity Matters: Rethinking the Latent Space for Generative Modeling,” arXiv.org, 2023. https://arxiv.org/abs/2307.08283.