Co-creating Platformer Levels with Constrained Adversarial Networks

PAOLO MORETTIN, KU Leuven, Belgium
ANDREA PASSERINI and STEFANO TESO, University of Trento, Italy

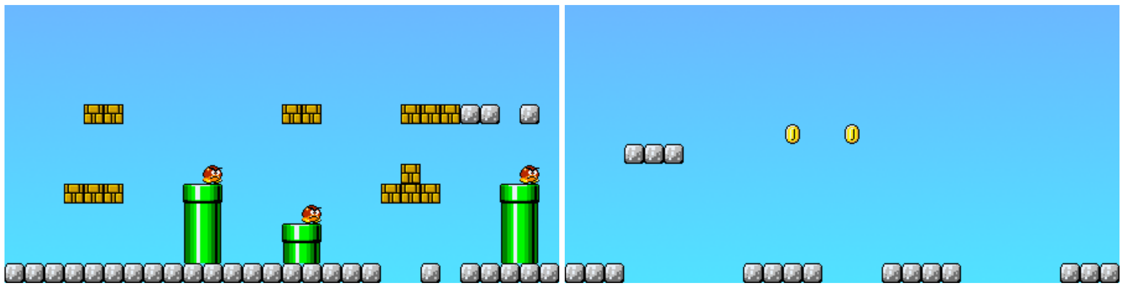

Fig. 1. Example Mario levels generated by a constrained adversarial network. Left: Pipe props are constrained to be complete and symmetric (coherency). Right: The rightmost column must be reachable from the starting position, regardless of gaps (playability). Images taken with permission from [6].

Given the success of deep generative models in many creative tasks, it is natural to ask how to best leverage them to support human designers. We study this problem in the context of mixed-initiative design of platformer levels, a paradigmatic co-creative task. This setting is especially challenging, because – like all functional content – platformer levels must satisfy complex validity constraints, like coherency and playability. We explore mixed-initiative interaction with constrained adversarial networks (CANs), a class of deep generative models that synthesize structures satisfying one or more validity constraints. As such, CANs can be used to complete user-supplied partial levels while retaining full control of the constraints to be applied. We go one step beyond, and consider the issue of *customizing* a pre-trained CAN to some target design task at hand and to the designer's preferences. We discuss how to achieve this by combining CANs with coactive learning, a very natural mixed-initiate interaction protocol that acquires the necessary supervision from the designer in a transparent manner. Finally, we illustrate how to extend coactive learning to acquire informative supervision in the form of interpretable constraints.

CCS Concepts: • **Human-centered computing → Collaborative interaction**; • **Computing methodologies → Unsupervised learning**; • **Theory of computation → Structured prediction**.

Additional Key Words and Phrases: mixed-initiative interaction, structured objects, interactive machine learning, generative adversarial networks, constraints

ACM Reference Format:
Paolo Morettin, Andrea Passerini, and Stefano Teso. 2018. Co-creating Platformer Levels with Constrained Adversarial Networks. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Deep generative models have achieved remarkable results in image synthesis [12, 26], music generation [20], style transfer [44], and many other creative tasks. It is only natural to ask how these models, which are designed to work in full autonomy, can be adapted for assisting and complementing human designers.

To ground the discussion, we focus on a concrete application, namely mixed-initiative design of 2-D platformer game levels [21, 22, 33]. This and other forms of semi-autonomated content generation are very relevant for the game industry, as they promise to lower production costs and facilitate scalability [36]. A key feature of this problem is that – in contrast to, e.g., natural images – game levels are *functional* [22], that is, they obey complex validity constraints like coherence (for instance props must be complete, non-flying enemies must touch the ground) and playability (the goal tile must be reachable), see Figure 1. Standard deep generative models, however, do not handle validity constraints and hence struggle to generate valid objects.

This motivates us to explore mixed-initiative level design with *constrained adversarial networks* (CANs) [6]. CANs are generative adversarial networks (GANs) [12] designed specifically for functional objects. Like other generative models, once trained on existing content, CANs can be invoked to create novel content from scratch [12] or to finish incomplete sketches supplied by a designer [43]. As a bonus, during interaction the designer can turn on and off different constraints, allowing her to synthesize different types of structures and boost diversity [6].

In this setup the model is trained to generate content that mimics the training data, which might be quite different from the content needed by the designer for the task at hand. Compatibly with recent work [13, 19], we argue that effective content synthesis requires *customizing* the generative model for the target domain. To this end, we propose to leverage *coactive learning* (CL) [32], an interaction protocol whereby the machine acquires pairwise preferences (that is, "object x is better than object x'") by tracking the modifications made by the designer to its own suggestions. The generative model is then periodically adjusted to comply with the collected preferences. Coactive learning integrates seamlessly into mixed-initiative workflows and is completely transparent to the designer [7, 39].

Given the tension between the amount of feedback that can reasonably be acquired during a design session and the number of preferences needed to adjust deep generative models like CANs [19], we also discuss how to extend CL to collect both preferences and constraints. In our view, constraints are a powerful form of feedback as they provide information about entire *sets* of candidate levels: levels that *do* satisfy a constraint are preferred to levels that *do not*, all else being equal. Importantly, constraints can be imposed on top of both outputs (e.g., tile and enemy patterns) and disentangled latent representations of the generative models.

In the following, we overview GANs and CANs. Next, we discuss mixed-initiative level design with CANs and distinguish between strategies based on conditional generation and adaptive strategies based on coactive learning and constraint acquisition. In Section 4 we discuss related work and then conclude with some final remarks.

2 BACKGROUND

Generative Adversarial Networks (GANs) [12] are a popular class of generative models where two neural networks, the *generator* g and the *discriminator* d, are trained jointly in an adversarial fashion. Specifically, the discriminator d is trained to distinguish "real" objects in the training set $\mathcal{X}$ from "fake" objects synthesized by the generator g. At the same time, g is trained to output objects that fool d. The generator is typically implemented as a feed-forward or a deconvolutional neural network that maps random vectors $\mathbf{z}$ (sampled from some simple distribution, like a multivariate normal) into objects $\mathbf{x}$. The discriminator is a feed-forward or convolutional neural network that takes objects $\mathbf{x}$ and output their probability of being real or fake.

Training is typically performed using (variants of) stochastic gradient descent by interleaving updates to g and d, and it proceeds until the synthesized objects look indistinguishable from those in the training set $\mathcal{X}$. Under idealized assumptions [12], the learned generator eventually recovers the data distribution. Once trained, the generator g can be used to synthesize *new* objects $\mathbf{x}$ by simply sampling random vectors $\mathbf{z}$ and computing their image under the generator, that is, $\mathbf{x} = g(\mathbf{z})$. This operation is extremely efficient. This form of training has the advantage of not requiring to compute an objective based on likelihood or other probabilistic metrics, which are in general intractable and thus approximated. For this reason, GANs have found successful applications in domains where formalizing the target distribution is hard.

GANs and other standard deep generative models are not designed to generate valid structures [6]. Indeed, generating objects consistent with a constraint involves first discovering that such a constraint exists (at least implicitly) by looking at the data, and generative models are not designed to carry out this non-trivial task. Furthermore, if the training data contains infeasible examples (because of, e.g., mistakes during the data collection), then the validity constraint cannot be acquired perfectly even in principle.[1]

2.1 Constrained Adversarial Networks

Constrained Adversarial Networks (CANs) address this issue by taking both training examples and validity constraints into consideration. CANs train the generator g so that it jointly maximizes the probability of fooling the discriminator d *and* the probability of synthesizing valid objects. This is achieved by augmenting the adversarial loss used in standard GANs (denoted by ℓ_{adv}) with the *semantic loss* (SL), a technique proposed in [41] to encourage neural networks to output predictions consistent with constraints. Letting ψ be a user-supplied validity constraint (encoded as a propositional logic formula over the elements of $\mathbf{x}$), CANs train g and d to optimize (resp. minimize and maximize) the following expression:

$$\ell_{adv}(g, d) + \lambda \cdot SL_{\psi}(g) \tag{1}$$

where $\lambda > 0$ is a hyper-parameter controlling the importance of the constraint. The SL is designed to be large whenever the probability that the generator outputs a valid object is small. To this end, the SL is defined as the negative logarithm of this probability:

$$SL_{\psi}(g) = -\log \left[\sum_{\mathbf{x} \text{ satisfies } \psi} P_g(\mathbf{x}) \right] \tag{2}$$

The sum measures the total probability allocated by the generator to valid objects. Evaluating the sum involves enumerating *all* possible objects (e.g., tile arrangements) $\mathbf{x}$, checking which ones are feasible, and computing their probability with respect to the generator (written $P_g(\mathbf{x})$). Since the number of possible objects is typically exponential, naïve enumeration is infeasible. Knowledge compilation (KC) [4] is thus used to compile the sum into a *compact* polynomial (or more precisely, an arithmetic circuit) that can be evaluated efficiently during training. KC works by leveraging distributivity to rewrite $SL_{\psi}(g)$ as compactly as possible, enormously speeding up evaluation. This is achieved by identifying shared sub-components and compactly representing the factorized expression using a DAG.

If ψ is very complex, the polynomial output by KC may be large. This is not a huge issue during training, which is performed once (or infrequently) on powerful machines, but it could be problematic for inference. A major advantage of

[1]It can be shown that in this case GANs will acquire a wrong validity constraint [6].

CANs is that – as in regular GANs – synthesizing new objects boils down to a simple forward pass over the generator, independently of the KC polynomial, which can thus be thrown away after training.

3 MIXED-INITIATIVE LEVEL DESIGN WITH (CONDITIONAL) CANS

We are concerned with mixed-initiative design of 2-D platformer levels. In this setting, a human designer and a generative model take turns in proposing modifications to a shared level map [21, 22, 33]. The designer can reject or modify any suggestions made by the model. The process can be considered a success if the machine proposes useful suggestions (that is, modifications that improve the overall quality of the level) thus saving time and resources, or if it inspires the human designer to create an overall better level [18, 42].

Co-designing with a pre-trained model. Perhaps the most straightforward co-design strategy is to use a pre-trained generative model to synthesize levels *conditioned* on the designer's input. The simplest setup is *inpainting*: in this case, the user supplies an incomplete level and the network fills in the missing parts based on the context [43]. CANs, in particular, can be easily adapted to inpainting under constraints [6]. In a second, orthogonal approach, the designer specifies some desired features of the output and lets the network generate a level compatible with those features. For instance, one could condition the generator to output objects from a specific class (e.g., underground rather than an open air levels) [24], to include a given quantity of certain elements (enemies, amount of water) [15], or to satisfy expected length/playtime or leniency (a proxy of expected difficulty) requirements [31].

CANs support an additional form of conditional generation that makes it possible during synthesis to enable or disable different groups of constraints. A technical description of this mechanism can be found in [6]. Hence CANs not only inherit the conditional capabilities of traditional GANs, but also support the generation of objects conditioned on properties expressed in logical terms. Using this technique, the designer can generate diverse levels by turning on and off constraints like "*a room contains a boss fight if and only if it contains a treasure chest*" or "*if the exit is locked then the room must contain a key somewhere*". The requirement is that these optional constraints are all baked into the CAN generator during training.

In complex co-creative scenarios like level generation, the output of the model is unlikely to satisfy all of the designer's desiderata from the get go. To solve this issue, recent work has investigated strategies that allow the designer to iteratively refine an initial suggestion by interacting with the generative model [3, 9, 10]. In this sequential setting, a crucial aspect is how to ensure consistency of the generated object with respect to previous iterations. This is particularly challenging when the designer's feedback is expressed in natural language and the output is relatively unstructured [23]. Since game levels are inherently structured it is easier to unambiguously describe the desired changes, to maintain consistency with respect to previous iterations, and to identify possibly contradictory feedback from the user. For instance, it is much easier to effectively account for feedback like "*turn all the water tiles into lava*" or "*the first half of the level should not contain enemies*" than "*the subject in the photo should smile*".

Co-designing and customization. So far, we considered interaction based on conditioning a pre-trained generative model during synthesis. Since the model generates content that mimics the training data, unless the training data is designed appropriately, the synthesized content will not fit the target application. This introduces a fundamental tension between the effort required to create the training data and the effort saved by using the generative model [19].

A sensible option is then to collect feedback on the model's suggestions while interacting with the designer, and then to incorporate the latter into the generative model itself [7, 13, 39]. We propose to do so by leveraging *coactive learning* [7, 32], an interactive learning protocol in which the machine iteratively suggests content to the user, the user improves – even by changing just a few tiles – the suggested content, and the process repeats. The meaning of "improvement" is entirely defined by the designer's preferences and requirements. At the end of each iteration, the machine has access to a suggested object $\mathbf{x}$ and an improved configuration $\mathbf{x}'$, and extracts a pairwise preference of the form "$\mathbf{x}'$ is better than $\mathbf{x}$". Such preferences are collected in a data set and then used to adjust the generative model. A simple procedure to do this is to re-train the current generative model to better comply to the collected preferences using a ranking loss, as proposed in [16]. As interaction proceeds, more preferences are collected and the model progressively aligns to the designer's needs. Notice that coactive learning is completely transparent to the user and integrates naturally in mixed-initiative interaction via manipulative interfaces [7], which are commonplace in level co-design [21, 22, 33].

The nature of the feedback highly impacts the performance of the model. Preferences are very effective for capturing the designer's needs [28], however they do not explanation *why* one object is preferred to the other. As a concrete example, if the user is presented with a Super Mario level $\mathbf{x}$ and improves it by removing some coins from it, leading to $\mathbf{x}'$, the system does not know whether the coins were removed because they were unreachable, because there were too abundant already, because they did not look good, etc. This kind of explanatory supervision conveys a lot of information [30] and can dramatically improve the speed of adaptation, especially for data hungry models like GANs. We propose to do so by combining constraint acquisition [5] with CANs. The human designer not only can adjust the generative model by modifying the presented output, but is also empowered with tools for specifying constraints on the desired output. Of course, encoding preferences in formal/logical terms can be hard for the end user. Nonetheless, there exists a body of work on translating natural language into various formal representations [2, 8, 38]. Most crucially, the system must be able to learn the building blocks of this interaction language, that is, the logical predicates that the designer uses to specify her needs. While many predicates of practical interest can be specified a priori, like the presence or number of specific elements in a portion of the level, we can go further and provide an interface for learning new predicates from the user. As shown in [6], it is indeed possible to use learned (neural) predicates in the CAN framework, opening up the possibility of extending the base language with new high-level concepts. For instance, the designer may want to specify that a portion of the level referenced by its coordinates represents a castle, $isCastle(x1, y1, x2, y2)$. Even if the concept is unknown to the system, it can be learned from user examples and possibly refined with interactions where the user is presented with some positive examples and labels whether they represent a castle or not.

CANs readily provide an interface for learning from such constraints, hence once collected these constraints can simply used to re-train the CAN until it complies with them. It is to be expected that very complex constraints require an expensive compilation step and many re-training epochs. We admit for adaptation to occur in the background or during periodic sleep cycles, rather than *during* the design session as in CL. Delayed learning of this kind is completely sound from a machine learning perspective [45] and widely adopted in settings characterized by computationally demanding *concept drifts* like anomaly detection [27] and some reinforcement learning applications [1, 25].

4 RELATED WORK

Procedural (game) content generation (PCG) has been traditionally addressed with search-based or rule-based methods. In recent years, the focus has shifted toward techniques based on machine learning [17, 36] and deep learning [22]. Game content can be divided into *functional* content, like game levels, game mechanics or behavioural rules for non-player characters, and *cosmetic* content, such as textures, music and sound effects. Functional content is arguably more challenging for automatic generators. While tasks in computer vision or natural language processing are supported by very large and accessible datasets, the training data for the generation of functional content for games is typically very scarce. Game level generation is by far the most studied problem in this area, with most works focusing on 2D tile-based video games. Although annotated datasets of levels exist [37], they are relatively small in size and cover a handful of popular games. The available data is often insufficient for effectively training a fully autonomous deep generative model and, most crucially, different approaches must be adopted for training a generator for a novel game with no data available. While some works mitigate this problem with bootstrapping techniques [40] or by allowing for diverse sources of supervision (like gameplay videos [14] or transfer learning across different games [29, 34]), involving human interaction in the training of these systems is likely a necessity for all but the most trivial settings.

Automated creation tools are unlikely to output content that fits the target application perfectly. Successful applications of computer-aided design (CAD) tools have spurred research in developing mixed-initiative co-creation techniques for level generation. This interaction paradigm is not only deemed effective for reaching satisfiable end results, but it also useful in fostering the creativity of the human designer [42].

Coactive learning was first proposed in the context of interface optimization [11] and information retrieval [32]. A useful feature of CL is that it integrates seamlessly with mixed-initiative interaction: preferences are extracted whenever the human supervisor modifies, explicitly or implicitly, any suggestion made by the machine. For this reason, CL was adopted in constructive preference elicitation with manipulative interaction [7]. Our proposed approach is directly inspired by this line of work.

Recently, Guzdial et al. introduced an interaction protocol reminiscent of CL. The difference is that, whereas in CL the model used to synthesize structures and the model used to learn from the designer's feedback are the very same, Guzdial et al. allow the two models to be different [13]. From a learning perspective this is sub-optimal, as the two models might make different mistakes and have different biases, and therefore feedback useful for one model might be less (than) useful for the other. Finally, our idea of integrating interpretable constraint acquisition into coactive learning extends prior work in constructive preference elicitation [39] by combining it with constraint learning [5] and explanatory interactive learning [30, 35].

5 CONCLUSION

We discussed mixed-initiative level design with constrained adversarial networks. Our contribution is conceptual: on the one hand, we show that these models – which are designed for autonomously generating functional content – can be used for helping human designers, and on the other that they can be adapted to the task at hand by combining them with coactive learning and constraint acquisition. Of course, these insights must be validated through extensive experiments and user studies. This is left to future work.

ACKNOWLEDGMENTS

This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. [694980] SYNTH: Synthesising Inductive Data Models). The research of ST and AP was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215.

REFERENCES

[1] Noam Brown and Tuomas Sandholm. 2018. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* 359, 6374 (2018), 418–424.

[2] Andrea Brunello, Angelo Montanari, and Mark Reynolds. 2019. Synthesis of LTL formulas from natural language texts: State of the art and research directions. In *26th International Symposium on Temporal Representation and Reasoning (TIME 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer

Informatik.

[3] Yu Cheng, Zhe Gan, Yitong Li, Jingjing Liu, and Jianfeng Gao. 2020. Sequential attention GAN for interactive image editing. In *Proceedings of the 28th ACM International Conference on Multimedia*. 4383–4391.

[4] Adnan Darwiche and Pierre Marquis. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research* 17 (2002), 229–264.

[5] Luc De Raedt, Andrea Passerini, and Stefano Teso. 2018. Learning constraints from examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[6] Luca Di Liello, Pierfrancesco Ardino, Jacopo Gobbi, Paolo Morettin, Stefano Teso, and Andrea Passerini. 2020. Efficient Generation of Structured Objects with Constrained Adversarial Networks. *Advances in Neural Information Processing Systems* 33 (2020).

[7] Paolo Dragone, Stefano Teso, and Andrea Passerini. 2018. Constructive preference elicitation. *Frontiers in Robotics and AI* 4 (2018), 71.

[8] Juraj Dzifcak, Matthias Scheutz, Chitta Baral, and Paul Schermerhorn. 2009. What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *2009 IEEE International Conference on Robotics and Automation*. IEEE, 4163–4168.

[9] Alaaeldin El-Nouby, Shikhar Sharma, Hannes Schulz, Devon Hjelm, Layla El Asri, Samira Ebrahimi Kahou, Yoshua Bengio, and Graham W Taylor. 2019. Tell, draw, and repeat: Generating and modifying images based on continual linguistic instruction. In *Proceedings of the IEEE International Conference on Computer Vision*. 10304–10312.

[10] Alaaeldin El-Nouby, Shikhar Sharma, Hannes Schulz, Devon Hjelm, Layla El Asri, Samira Ebrahimi Kahou, Yoshua Bengio, and Graham W Taylor. 2018. Keep Drawing It: Iterative language-based image generation and editing. In *Neural Information Processing Systems: Visually Grounded Interaction and Language Workshop*.

[11] Krzysztof Gajos and Daniel S Weld. 2005. Preference elicitation for interface optimization. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*. 173–182.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[13] Matthew Guzdial, Nicholas Liao, and Mark Riedl. 2018. Co-creative level design via machine learning. *arXiv preprint arXiv:1809.09420* (2018).

[14] Matthew Guzdial and Mark O Riedl. 2016. Game Level Generation from Gameplay Videos.. In *AIIDE*. 44–50.

[15] Andreas Hald, Jens Struckmann Hansen, Jeppe Kristensen, and Paolo Burelli. 2020. Procedural Content Generation of Puzzle Games using Conditional Generative Adversarial Networks. In *International Conference on the Foundations of Digital Games*. 1–9.

[16] Eric Heim. 2019. Constrained Generative Adversarial Networks for Interactive Image Generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 10753–10761.

[17] Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. 2013. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 9, 1 (2013), 1–22.

[18] Pegah Karimi, Jeba Rezwana, Safat Siddiqui, Mary Lou Maher, and Nasrin Dehbozorgi. 2020. Creative sketching partner: an analysis of human-AI co-creativity. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*. 221–230.

[19] Isaac Karth and Adam M Smith. 2019. Addressing the fundamental tension of PCGML with discriminative learning. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*. 1–9.

[20] Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C Courville. 2019. Melgan: Generative adversarial networks for conditional waveform synthesis. *Advances in Neural Information Processing Systems* 32 (2019), 14910–14921.

[21] Antonios Liapis, Georgios N Yannakakis, and Julian Togelius. 2013. Sentient sketchbook: computer-assisted game level authoring. (2013).

[22] Jialin Liu, Sam Snodgrass, Ahmed Khalifa, Sebastian Risi, Georgios N Yannakakis, and Julian Togelius. 2020. Deep learning for procedural content generation. *Neural Computing and Applications* (2020), 1–19.

[23] Yahui Liu, Marco De Nadai, Deng Cai, Huayang Li, Xavier Alameda-Pineda, Nicu Sebe, and Bruno Lepri. 2020. Describe What to Change: A Text-guided Unsupervised Image-to-Image Translation Approach. In *Proceedings of the 28th ACM International Conference on Multimedia*. 1357–1365.

[24] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[25] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. 2017. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science* 356, 6337 (2017), 508–513.

[26] Augustus Odena, Christopher Olah, and Jonathon Shlens. 2017. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*. PMLR, 2642–2651.

[27] Murugaraj Odiathevar, Winston KG Seah, and Marcus Frean. 2019. A hybrid online offline system for network anomaly detection. In *2019 28th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 1–9.

[28] Gabriella Pigozzi, Alexis Tsoukias, and Paolo Viappiani. 2016. Preferences in artificial intelligence. *Annals of Mathematics and Artificial Intelligence* 77, 3-4 (2016), 361–401.

[29] Anurag Sarkar and Seth Cooper. 2018. Blending Levels from Different Games using LSTMs.. In *AIIDE Workshops*.

[30] Patrick Schramowski, Wolfgang Stammer, Stefano Teso, Anna Brugger, Franziska Herbert, Xiaoting Shao, Hans-Georg Luigs, Anne-Katrin Mahlein, and Kristian Kersting. 2020. Making deep neural networks right for the right scientific reasons by interacting with their explanations. *Nature Machine Intelligence* 2, 8 (2020), 476–486.

[31] Jacob Schrum, Vanessa Volz, and Sebastian Risi. 2020. CPPN2GAN: Combining Compositional Pattern Producing Networks and GANs for Large-scale Pattern Generation. *arXiv preprint arXiv:2004.01703* (2020).

[32] Pannaga Shivaswamy and Thorsten Joachims. 2015. Coactive learning. *Journal of Artificial Intelligence Research* 53 (2015), 1–40.

[33] Gillian Smith, Jim Whitehead, and Michael Mateas. 2010. Tanagra: A mixed-initiative level design tool. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*. 209–216.

[34] Sam Snodgrass and Santiago Ontanon. 2016. An approach to domain transfer in procedural content generation of two-dimensional videogame levels. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

[35] Wolfgang Stammer, Patrick Schramowski, and Kristian Kersting. 2020. Right for the Right Concept: Revising Neuro-Symbolic Concepts by Interacting with their Explanations. *arXiv preprint arXiv:2011.12854* (2020).

[36] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. 2018. Procedural Content Generation via Machine Learning (PCGML). *IEEE Transactions on Games* 10, 3 (2018), 257–270.

[37] Adam James Summerville, Sam Snodgrass, Michael Mateas, and Santiago Onta n'on Villar. 2016. The VGLC: The Video Game Level Corpus. *Proceedings of the 7th Workshop on Procedural Content Generation* (2016).

[38] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 25.

[39] Stefano Teso, Paolo Dragone, and Andrea Passerini. 2017. Coactive critiquing: Elicitation of preferences and features. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[40] Ruben Rodriguez Torrado, Ahmed Khalifa, Michael Cerny Green, Niels Justesen, Sebastian Risi, and Julian Togelius. 2020. Bootstrapping conditional gans for video game level generation. In *2020 IEEE Conference on Games (CoG)*. IEEE, 41–48.

[41] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Broeck. 2018. A Semantic Loss Function for Deep Learning with Symbolic Knowledge. In *International Conference on Machine Learning*. 5498–5507.

[42] Georgios N Yannakakis, Antonios Liapis, and Constantine Alexopoulos. 2014. Mixed-initiative co-creativity. (2014).

[43] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. 2018. Generative image inpainting with contextual attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5505–5514.

[44] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*. 2223–2232.

[45] Martin Zinkevich, John Langford, and Alex Smola. 2009. Slow learners are fast. *Advances in neural information processing systems* 22 (2009), 2331–2339.