Searching the Latent Space of a Generative Adversarial Network to Generate DOOM Levels

Edoardo Giacomello
Dipartimento di Elettronica, Informazione e Bioinformatica
Politecnico di Milano
edoardo.giacomello@polimi.it

Pier Luca Lanzi
Dipartimento di Elettronica, Informazione e Bioinformatica
Politecnico di Milano
pierluca.lanzi@polimi.it

Daniele Loiacono
Dipartimento di Elettronica, Informazione e Bioinformatica
Politecnico di Milano
daniele.loiacono@polimi.it

***Abstract*—In this work, following the same approach successfully applied to evolve *Super Mario* levels, we applied the CMA-ES to search the latent space of a GAN previously trained to generate DOOM levels. Combining a search algorithm with a model trained in a supervised setting, allows to take advantage from both these paradigms. From one hand, the GAN is able to generate contents exploiting the design patterns learned from all the examples it was trained from. On the other hand, the CMA-ES can effectively search this design space for *specific* contents that meet some given design objectives. In particular, we tested our approach evolving three very different type of levels: an *arena* level (i.e., few large areas), a *labyrinth* level (i.e., many corridors and small areas), and a *complex* level (i.e., a balanced mix of large and small areas). Our results show that the latent space of a GAN can be effectively searched by the CMA-ES to find DOOM levels that fit accurately the objectives but, at the same time, are also novel.**

I. Introduction

Nowadays, the increasing complexity and size of games require the creation of a huge amount of content, that is one of the most expensive and time consuming task of the whole development process. Among the game contents, such as weapons, enemies, texture, sprites, models, etc., the levels significantly affect the game experience. For this reason, game researchers are focusing on the study and development of AI systems that can either assist or partially replace human designers on this important process.

In a previous work [1] we proposed to apply Generative Adversarial Networks (GAN), a popular deep learning approach that proved successful for the generation of several kind of contents, to generate DOOM levels. DOOM is a milestone of the genre of the first person shooter and, despite being more than 25 years old, not only it is still played today but it is also possible to find online a large amount of newly created content for this game, including game levels.

A major limitation of using GAN to generate levels is that it does not allow to meet specific design objectives, i.e., the generated levels would basically represent novel data samples generated from a distribution modeled after the training dataset. Accordingly, in this work we extend our previous work on DOOM level generation by applying the CMA-ES search of the latent space of the GAN, following the same approach introduced in [2] and previously applied also to evolve *Super Mario Levels* [3]. Accordingly, we encode DOOM levels as vectors of 100 real values and map them into actual DOOM levels using the generator network of a previously trained GAN. We then extract a few features from the generated levels that describe effectively their design, e.g., the size and shape of the level layout, the number of rooms of the level and the distribution of their size, etc. Thus, we apply the CMA-ES to search the input space of the GAN that generates the DOOM levels and use the extracted features to set the design objectives, i.e., the fitness function, of this search. In particular, we tested our approach by applying the CMA-ES to generate novel DOOM levels with a design that is similar to three specific levels, that basically differ for the organization and complexity of their layout: an *arena* level, a *complex* level, and a *labyrinth* level. We compared the performance of CMA-ES with a stochastic hill climber and our results show that CMA-ES is able to effectively find levels with design features very similar to the one provided as a target, nevertheless the generated levels exhibit a good degree of novelty and variety with respect to the original content.

The rest of this paper is organized as follows. In Section II we provide an overview of the related work. In Section III we briefly describe how GAN is used to generate DOOM levels and in Section IV we detail our approach. The experimental design is described in Section V, while the results are reported and discussed in Section VI. Finally, we draw some conclusions in Section VII.

II. Related Work

Procedural Content Generation (PCG), used in the past to deal with memory and computational limitations [4], is today mainly used by game developers to generate a huge amount of contents (e.g., the procedurally generated worlds of *No Man's Sky* [1]) or to provide an endless gameplay experience (e.g., like in the popular endless runner *Canabalt* [2]). In the past few years, several works in the game research literature focused on using machine learning to improve and extend the procedural content generation systems. In particular, two major

[1]https://www.nomanssky.com/
[2]http://canabalt.com/

978-1-7281-1884-0/19/$31.00

approaches have been proposed in this field: the *search-based procedural content generation* (SB-PCG) and the PCG based on machine learning (PCGML).

SB-PCG, introduced in [5], combines the procedural content generation methods with a search-based algorithms, such as evolutionary algorithms, to automatically generate high-quality game content. So far, SB-PCG has been applied to several game genres, including platform games [6]–[8], racing games [9]–[11], RPG games [12], strategy games [13], and first-person shooters [14]–[18].

PCGML, introduced in [19], consists of training machine learning models (e.g., a deep neural network) on existing game contents and using trained models to generate novel contents. This approach already proved successful in several applications [19], such as the generation levels for 2d platformers [20]–[22] and levels of *Legend of Zelda* [3] [23]. Among the approaches used in PCGML, Generative Adversarial Networks (GANs) [24] are a promising approach to model a complex dataset and then generate synthetic data with a similar distribution to the real data. GANs already proved successful in several applications that involve image processing, such as handwritten digits [25], human faces [26], and bedrooms [27]. They have been also used for image colorization problems [28], frame prediction in videos [29] and sound generation [30]. Concerning the PCGML field, Beckham and Pal [31] proposed a method based on GANs for generating realistic height level maps for videogames, that is more focused on landscapes and might be difficult to apply to generate indoor environments. The same approach can be also applied to non-functional content generation, as done in [32], in which GANs are used to generate 2d sprites. Finally, in a previous work [1] we trained a GAN to generate novel DOOM levels from the *idgames archive*, a large dataset of levels created by the community of DOOM players.

Recently, a novel approach that combines both SB-PCG and PCGML has been introduced in [3] and applied to generate levels of *Super Mario* [4]. This approach relies on the *latent variable evolution* (LVE) introduced in [2]. The underlying idea of LVE is that, once a GAN is trained to generate a specific type of data (e.g., images, sounds, levels, etc.), the real-valued noise vector, provided as input to the GAN for generating a novel data sample, is actually a compact and convenient representation of the generated sample itself. Thus, the input space of the GAN, can be searched with an effective evolutionary algorithm such as CMA-ES to generate a data sample that maximizes a given objective function. This idea has been successfully applied by Botranger et al. to generate fingerprints [2] and different kind of images (e.g., faces, shoes, etc.) [33] that match a given target. Finally, the same approach was applied to procedural game content generation by Volz et al. [3] for the first time.

[3]https://www.zelda.com/

[4]https://en.wikipedia.org/wiki/Super_Mario_Bros.

Fig. 1. A screenshot of DOOM.

III. Doom Level Generation with GAN

DOOM[5] is a milestone in the videogame history, developed by *id Software* in 1993, that greatly influenced the design and the development of games for years. DOOM is a first person shooter, i.e., a game where players explore, from a first person perspective, levels populated by enemies to kill in order to reach the exit and complete the game (see Figure 1).

In DOOM, levels are stored as *WAD* files, that contain both the level data and the multimedia assets. In particular, level data are stored as an ordered sequence of records, called *lumps*, that contains information about sectors, doors, walls, tiles, textures, triggers, objects, enemies, etc. Such a level representation is unfortunately not suitable to be processed by a neural network, accordingly in [1] we proposed to extract an image-based representation from the *WAD*, consisting of an image of the level with multiple channels that encode the floor layout, the walls, the floor heights, the objects, etc.. Figure 2 shows the deep neural network architecture used in [1] to generate DOOM levels, that consists of two networks that are trained at the same time: (i) the generator (G) network receives a noise vector as input and provides as output an image-based representation of a novel level, that can be easily converted in a playable WAD; (ii) the discriminator (D) receives an image-based representation of a DOOM level as input, that can be either generated by the G network or extracted by the WAD of an existing level, and labels it either as computer-generated (*fake*) or human-designed (*real*). These two networks are trained from a rather large dataset of DOOM levels created by the players community with an adversarial scheme: D is trained to classify correctly levels as fake or real, G is trained to generate levels that are not correctly classified by D. As in [1], we used the Wasserstein GAN with Gradient Penalty (WGAN-GP) [34] architecture, the noise vector consists of 100 independent random variables uniformly sampled in $[-1, +1]$,

[5]https://en.wikipedia.org/wiki/Doom_(franchise)

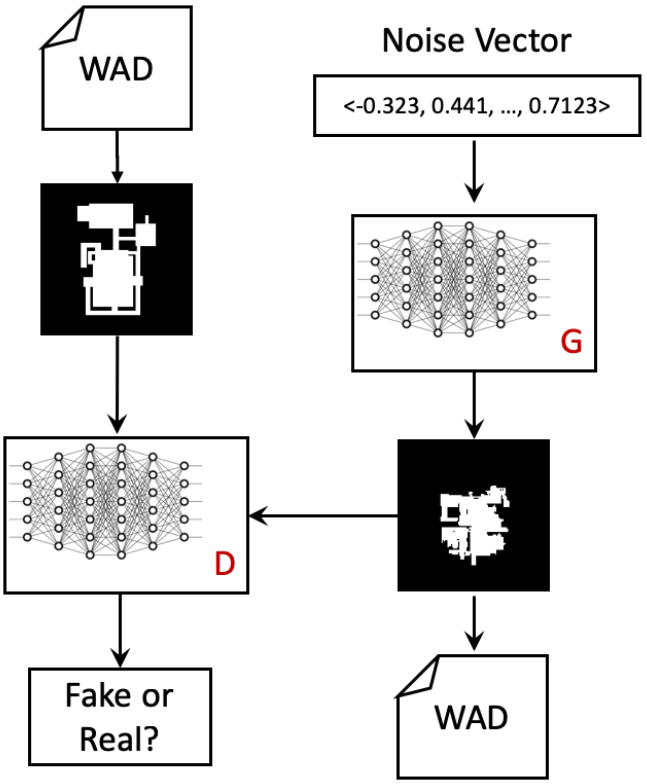


Fig. 2. Overview of the DOOM level generation approach using GAN proposed in [1]; G is the generator network in charge of generating new levels, D is the discriminator network in charge of classifying levels either as human-designed (real) or computer generated (fake).

both G and D networks have 4 layers with respectively 1024, 512, 26 and 128 filters, and the image-based representation of the levels is of size 128x128 pixels. The data used to train the GAN consists of a selection of the levels available in the *idgames archive*, the largest online archive of levels, modifications, tools, and resources for the games based on the DOOM engine. We refer the interested reader to [1] for additional details about the GAN architecture and the training process, not reported in this paper for brevity.

IV. Searching the Latent Space of GAN

The approach presented in this paper (see Figure 3) is based on the one introduced in [3] to evolve levels for *Super Mario* levels. In this section we will describe how we apply the CMA-ES to evolve DOOM levels, in particular we describe (i) how levels are encoded, (ii) how they are evaluated, and (iii) how the overall process works.

A. Level Encoding

As described in the previous section, DOOM levels are encoded as records stored in a WAD file. From the WAD file is possible to extract a multi-channels image that encodes the most relevant information about the level, i.e., the floor map, the floor heights, the walls, and the game objects. Unfortunately, this image-based encoding might not be an

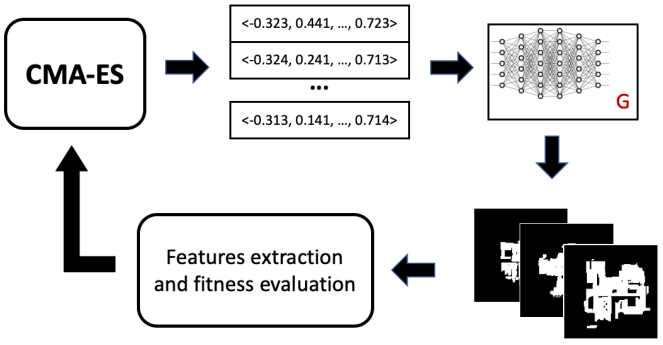


Fig. 3. An overview of our approach for searching the latent space of GAN to generate DOOM levels.

effective choice to apply an evolutionary algorithm to search the level design space. In fact, using an image-based encoding, the search space would not only be huge, i.e., a multi-channels image of size 128x128, but it would also include a large amount of meaningless and ill-formed levels. Accordingly, our approach consists of using an indirect encoding of the level that relies on a GAN previously trained to generate DOOM levels, as described in the previous section. The underlying idea is that the input vector of the generator network of the GAN is actually a well suited encoding of DOOM level; thus, the generator network can be basically used as a genotype-phenotype mapping, i.e., the generator network can be used to generate an image-based representation of a DOOM level starting from an encoding consisting of an input vector of real numbers. In particular, in this work we trained a GAN following the same approach introduced in [1] (see Section III), and thus we use a vector of 100 real variables to encode a DOOM level. A level encoded as a vector can be then mapped into an actual DOOM level using the generator network of a trained GAN and, eventually, generating a WAD file from the level image provided by the network.

B. Level Evaluation

To evaluate the generated levels and, thus, compute their fitness values we extract from them some high-level features that are somehow relevant from a design perspective. These features are extracted from the image-based representation of the levels that is provided as output by the generator network when the vector encoding of the level is provided as input. Table I shows the 7 features extracted from the levels. The underlying idea of our approach is that these features are able to capture the key elements of the level design layout and, therefore, they can be used to guide the evolutionary search. Therefore the fitness of a level is computed as the L^2 norm of the difference between the vector of features extracted from it and a vector of target features, i.e., a vector that contains a target value for each one of the 7 features.

C. Search Process

Figure 3 depicts an overview of our approach to search the level design space. A population of level, encoded as real-

TABLE I
THE LIST OF FEATURES EXTRACTED FROM THE IMAGE-BASED REPRESENTATION OF THE GENERATED DOOM LEVELS.

Name	Definition
diameter (D)	This feature is defined as the diameter of the smallest circle that cover the whole layout of the levels in the 128x128 image representation; accordingly, it measures the distance between the furthest points of the level.
major (M) and minor (m) axis	These two features measure the distance of the furthest points of the level along the horizontal and vertical axes of the level image; *major axis* (M) is then defined as the largest of the two distances computed, while *minor axis* (m) as the smallest one; these two features together give a rough idea of the form factor (i.e., the aspect ratio) of the level (i.e., how close to a square it is) and of its size.
solidity (S)	This features measures the size of the walkable area of the level as a fraction of the area of the convex hull that encloses the whole level; thus, it measures how *dense* is the layout of the level.
rooms (R)	This feature is measured as the number of *rooms* in the level; to compute this feature, it is not possible relying on the doors placed into the levels, because they are often not used in DOOM levels; therefore, we applied an algorithm introduced in [35] to generate a segmentation of an indoor map into separated rooms.
skewness (γ) and kurtosis (β)	These two features are respectively the skewness and the kurtosis of the distribution of the distance between each pixel of the level image and the closest wall; these features account for the variety and balance of large and small areas of the level.

Fig. 4. Distribution of the features extracted from of a selection of the levels available in the *idgames archive*; each level has been labeled either as *arena* (green dots), *complex* (blue dots), or *labyrinth* (red dots); only the values of four features are reported in this analysis.

valued vectors, is generated by the CMA-ES according to the current values of the centroids and of the covariance matrix. Each vector in the population is then provided as input to the generator network (G in Figure 3) and an image-based representation of the level is generated. Then, from this image-based representation the vector of features described in Table I is extracted and used to compute the fitness of the level. Finally, the population ranked by the fitness values is used to update the parameters of the CMA-ES and the following iteration can start over.

V. EXPERIMENTAL DESIGN

To test our approach, we run three experiments that differ only for the target values of the level features used to compute the fitness and, hence, to guide the search in the level design space. To choose three meaningful targets, we performed an analysis of the features distribution of the large dataset of human-designed DOOM levels used to train our GAN level generator (see Section III). As a result of this analysis, we identified three kinds of levels corresponding to three clusters in the feature space as shown in Figure 4: the first kind of level, *arena*, has a simple layout and very large open areas; the second kind of level, *complex*, has a more complex layout that include both rather large rooms and smaller ones; the third kind of level, *labyrinth*, has a very complex layout with a large number of small closed areas.

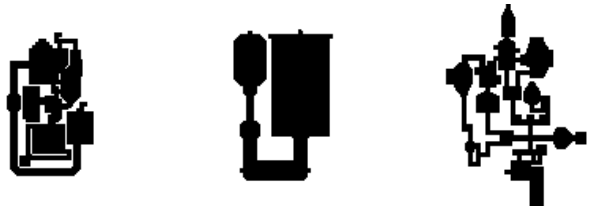


Fig. 5. The three maps selected to extract the features used as target in the experiments reported in this paper: (a) the target *complex* level, (b) the target *arena* level, and (c) the target *labyrinth* level.

Accordingly, we designed three experiments choosing, as target, the features extracted from a human-designed level for each kind of level described before, i.e., *arena*, *complex*, and *labyrinth*. Figure 5 shows the three maps selected to extract the target features for our experiments; these maps are taken from the *idgames archive* but they were not included in the dataset used to train the GAN. We decided to use target values extracted from actual DOOM levels, instead of simply using arbitrarily chosen values, basically because this guarantees that the target values correspond to a meaningful level design and it also allows to do a visual comparison between the evolved levels and the one used as a target.

VI. Experimental Results

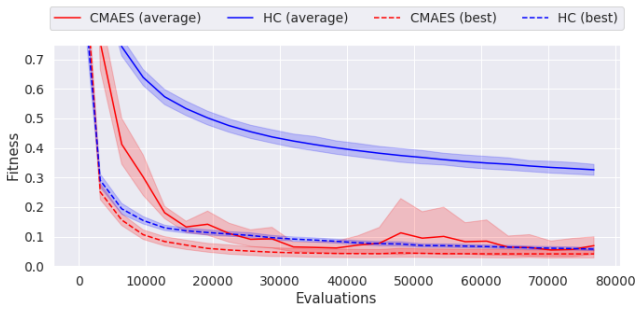
We performed three experiments, one for each of the levels selected as a target, as described in the previous section. The experiments have been carried out using the CMA-ES implementation available in the DEAP software library [36]. On each experiment, we compared the CMA-ES with stochastic hill climbing (HC) [37] and we performed 30 runs with the following parameters setting. The parameter λ is set to 32 [6], μ is set to 16 (i.e., $\lambda/2$), each element of the centroid vector is initially set to 0, and σ is set to 1. All the other parameters are set to the default values, based on the guideline provided in [38]. Each run consisted of 2500 generations, i.e., 8000 evaluations. Concerning the stochastic hill climbing, in each run we performed 2500 iterations with 32 parallel instances, leading to 8000 evaluations as the ones performed with CMAE-ES.

Figure 6 compares the performances of CMA-ES with the ones of stochastic hill climbing on the three experiments. The results show that in all the experiments the CMA-ES outperforms (blue lines in Figure 6) the hill climbing (red lines in Figure 6). Although hill climbing is able to find at least one level with a quite good fitness in each experiment (see dashed lines in Figure 6), the best fitness achieved by the CMA-ES is always lower. Moreover, the average fitness of the solution found with hill climbing, despite decreasing with the evaluation, is much higher than the one achieved by the CMA-ES (see solid lines in Figure 6). Finally it is worthwhile to notice that, except when the level of type *labyrinth* (see Figure IV-Bc) is used as a target, the population of levels evolved by the CMA-ES does not converge. Our analysis of the population suggest that this is mainly due to the fact that the search space of the CMA-ES is not bounded, while the input vector provided to the generator network (see Figure 3) is supposed to be bound between -1 and 1. As a result, we found some numerical instability that might prevent the convergence of the CMA-ES.

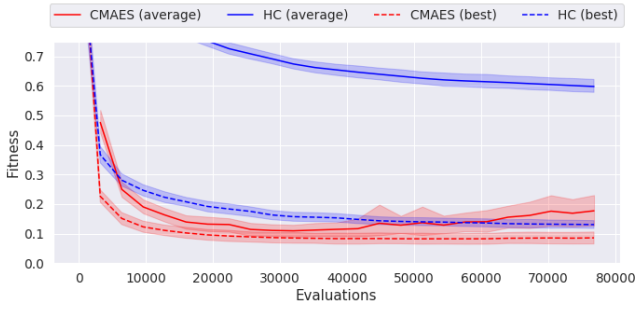
To assess the quality of the results achieved by the CMA-ES, in Figure 7 we plotted the distribution of the features extracted [7] from the levels evolved at different stages of the evolutionary process, i.e., at the beginning, roughly in the middle, and at the end. The distribution of these values is showed together with the target values (dashed red line in Figure 7). This analysis shows that at the beginning the features are spread across a large range of values, often not even centered around the target value. However, at the end of the evolutionary process, the features of evolved levels converge effectively toward the target values, in all the three experiments. It is also interesting to note that some features converge more quickly than others. In particular, the diameter of almost all the levels evolved at generation 1000 is already

[6]Please notice that this value differs from the one suggested in [38] and it was set to 32 for practical reasons: in fact, 32 is the batch size used in our GAN architecture, making it very convenient to evaluate the whole population of 33 individuals at once.

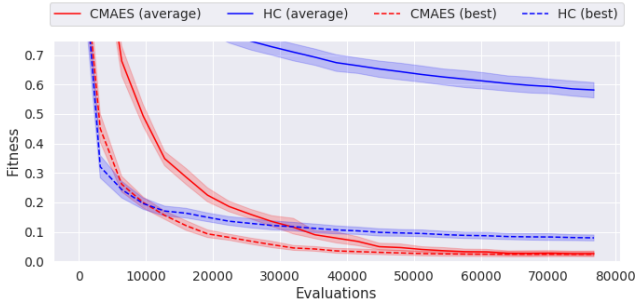
[7]For brevity, we reported the data only for diameter, rooms, solidity, and skewness features.



(a)



(b)



(c)

Fig. 6. Performance of CMA-ES (in red) and HC (in blue) when the target features are extract from a level of type (a) *complex*, (b) *arena*, and (c)*labyrinth*; average fitness is reported as a solid line, best fitness as a dashed line (the lower the fitness value, the better); each curve is averaged over 30 runs.

equal to the target value; in contrast, it takes more time to the solidity values to converge toward to the target values. This suggests that setting the level diameter is a much easier design objective with respect to the solidity, probably because the GAN is able to generate levels with a better variety of diameter values than it is for solidity.

Finally, Figure 8 shows some examples of the evolved levels. From these results it is possible to see that, despite having different layouts, the evolved levels are actually able to capture the design features of the target levels.

VII. Conclusions

In this paper, we extended our previous work on using GAN for the level design of DOOM [1]. Following the approach introduced in [3], we applied CMA-ES to search the latent

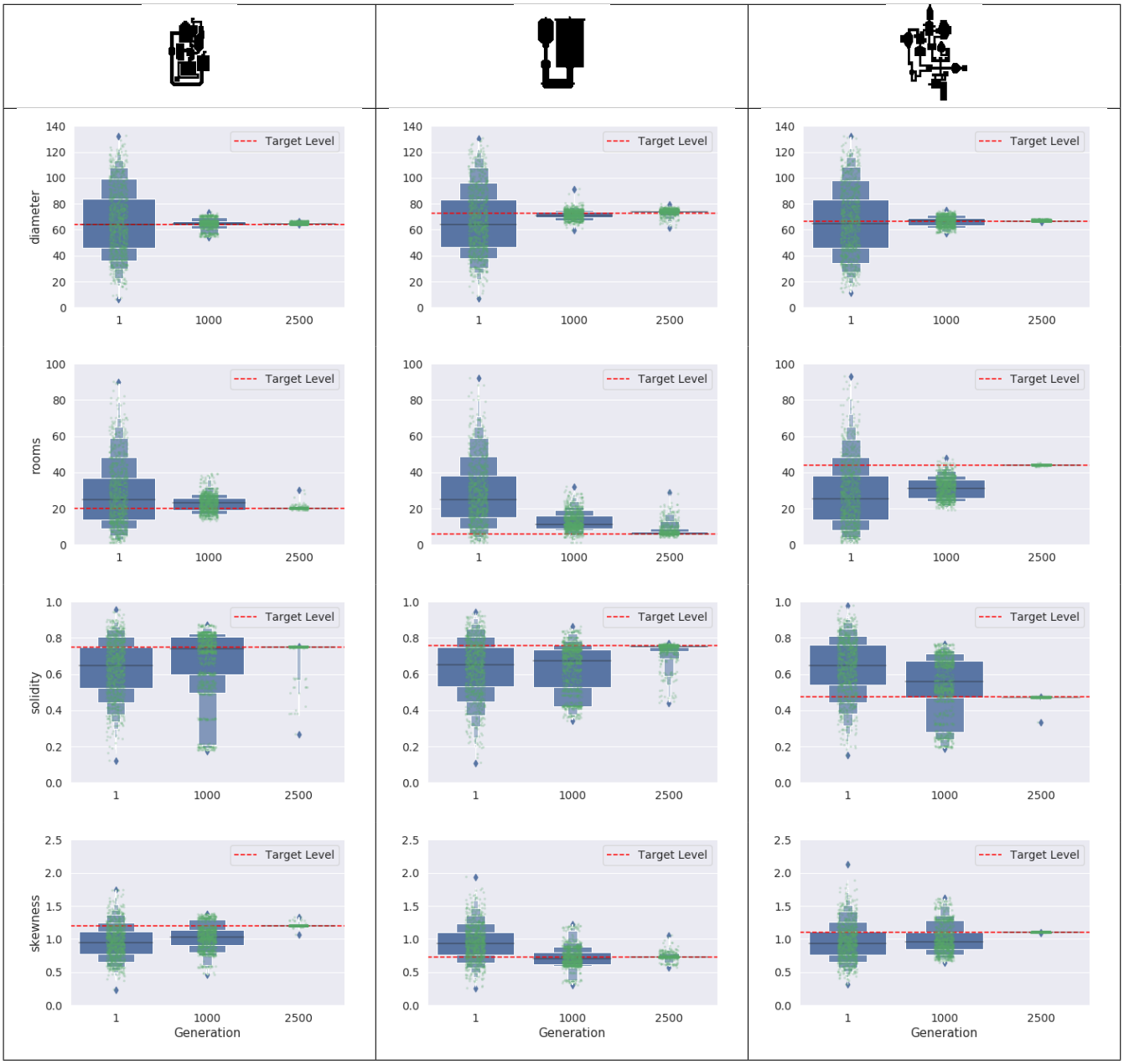


Fig. 7. Distribution of the features extracted from the levels evolved by the CMA-ES (at generation 1, 1000 and 2500), when the target is either a *complex* level (left column), an *arena* level (middle column), or a *labyrinth* level (right column); the distributions of the features are reported in blue using letter-valued plots [39], all the single values of the features are reported as green dots, and the target value is reported with a dashed red line; the reported data refer to 30 runs.

space of a GAN previously trained to generate DOOM levels. In the approach proposed in this work, a level is encoded as a vector of real values and mapped into an actual level by providing this vector as input to the generator network of the trained GAN. We computed the fitness of each level as the L^2 norm of the difference between the vector of features extracted from the levels and a target vector of features. Accordingly, our approach allows to evolve novel DOOM levels that are based on a set of human-designed levels, i.e., the ones used to train the GAN, but at the same time they meet the design objectives identified by the target vector of features, such as the number of rooms, the size, the density of the level layout, etc. We designed three experiments to test our approach, using as target the features extracted from three human-designed levels very different among them: (i) an *arena* level, that contains only few large areas, (ii) a *labyrinth* level, that consists of many corridors and small areas, and (iii) a *complex* level, that is well balanced mix of the previous ones. Our results are very promising and show that the CMA-ES is able to effectively search the latent space of the GAN to evolve DOOM levels with features that are very close to the ones provided as target, despite showing a good degree of novelty with respect to the

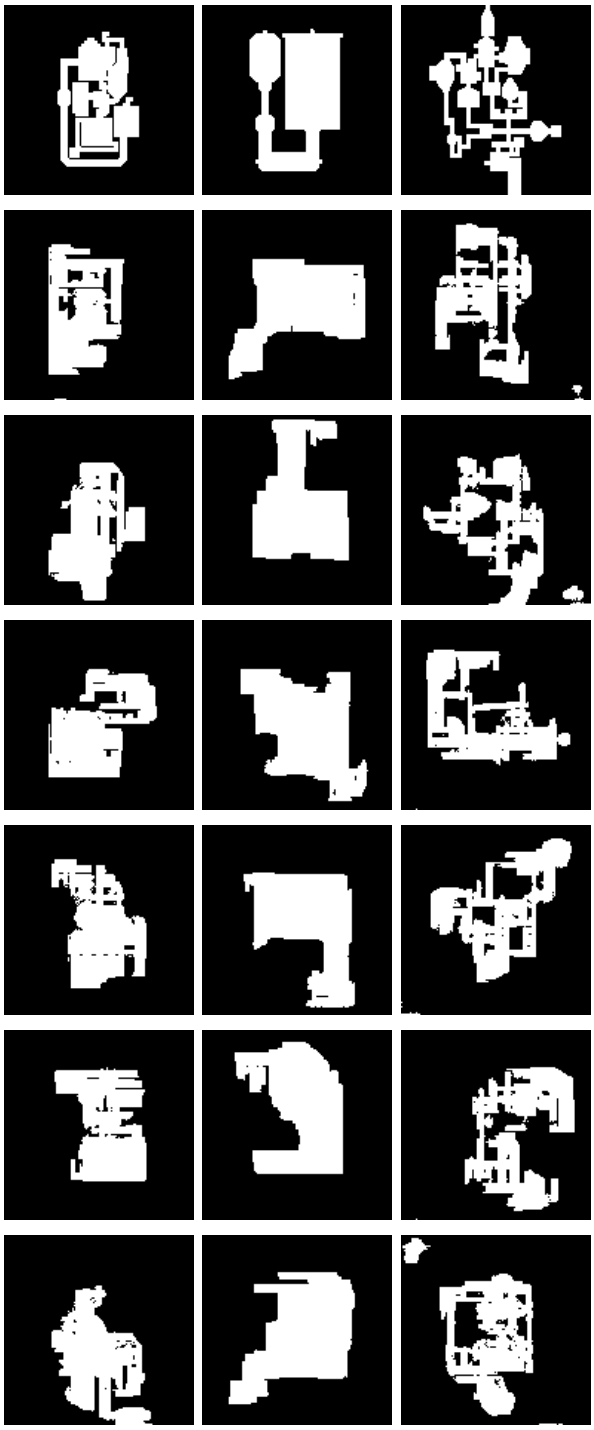


Fig. 8. Examples of levels evolved by the CMA-ES at the end of the evolutionary search. The levels in the same column are evolved in the same experiment and the target levels are depicted in the first row.

target levels.

REFERENCES

[1] E. Giacomello, P. L. Lanzi, and D. Loiacono, "DOOM level generation using generative adversarial networks," in *IEEE Games, Entertainment, Media Conference, GEM 2018, Galway, Ireland, August 15-17, 2018*. IEEE, 2018, pp. 316–323. [Online]. Available: https://doi.org/10.1109/GEM.2018.8516539

[2] P. Bontrager, J. Togelius, and N. D. Memon, "Deepmasterprint: Generating fingerprints for presentation attacks," *CoRR*, vol. abs/1705.07386, 2017. [Online]. Available: http://arxiv.org/abs/1705.07386

[3] V. Volz, J. Schrum, J. Liu, S. M. 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*, ser. GECCO '18. New York, NY, USA: ACM, 2018, pp. 221–228. [Online]. Available: http://doi.acm.org/10.1145/3205455.3205517

[4] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural content generation in games*. Springer, 2016.

[5] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation," in *Proceedings of EvoApplications*, C. Di Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekárt, A. I. Esparcia-Alcazar, C.-K. Goh, J. J. Merelo, F. Neri, M. Preuß, J. Togelius, and G. N. Yannakakis, Eds., vol. 6024. Berlin, Heidelberg: Springer LNCS, 2010, pp. 141–150.

[6] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Optimization of platform game levels for player experience," in *AIIDE*, C. Darken and G. M. Youngblood, Eds. The AAAI Press, 2009.

[7] K. Compton and M. Mateas, "Procedural level design for platform games," in *AIIDE*, J. E. Laird and J. Schaeffer, Eds. The AAAI Press, 2006, pp. 109–111.

[8] N. Shaker, G. N. Yannakakis, and J. Togelius, "Towards automatic personalized content generation for platform games," in *AIIDE*, G. M. Youngblood and V. Bulitko, Eds. The AAAI Press, 2010.

[9] J. Togelius, R. De Nardi, and S. Lucas, "Towards automatic personalised content creation for racing games," in *Proc. IEEE Symposium on Computational Intelligence and Games CIG 2007*, 2007, pp. 252–259.

[10] D. Loiacono, L. Cardamone, and P. L. Lanzi, "Automatic track generation for high-end racing games using evolutionary computation," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 3, no. 3, pp. 245 –259, sept. 2011.

[11] L. Cardamone, D. Loiacono, and P. L. Lanzi, "Interactive evolution for the procedural generation of tracks in a high-end racing game," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, ser. GECCO '11. New York, NY, USA: ACM, 2011, pp. 395–402. [Online]. Available: http://doi.acm.org/10.1145/2001576.2001631

[12] J. Dormans and S. Bakkes, "Generating missions and spaces for adaptable play experiences," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 3, no. 3, pp. 216–228, 2011.

[13] J. Togelius, M. Preuss, N. Beume, S. Wessing, J. Hagelbäck, and G. N. Yannakakis, "Multiobjective exploration of the starcraft map space," in *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, 2010, pp. 265–272.

[14] L. Cardamone, G. N. Yannakakis, J. Togelius, and P. L. Lanzi, "Evolving interesting maps for a first person shooter," in *Proceedings of the 2011 international conference on Applications of evolutionary computation - Volume Part I*, ser. EvoApplications'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 63–72. [Online]. Available: http://dl.acm.org/citation.cfm?id=2008402.2008411

[15] P. L. Lanzi, D. Loiacono, and R. Stucchi, "Evolving maps for match balancing in first person shooters," in *2014 IEEE Conference on Computational Intelligence and Games, CIG 2014, Dortmund, Germany, August 26-29, 2014*. IEEE, 2014, pp. 1–8.

[16] D. Loiacono and L. Arnaboldi, "Fight or flight: Evolving maps for cube 2 to foster a fleeing behavior," in *IEEE Conference on Computational Intelligence and Games, CIG 2017, New York, NY, USA, August 22-25, 2017*. IEEE, 2017, pp. 199–206. [Online]. Available: https://doi.org/10.1109/CIG.2017.8080436

[17] D. Loiacono and L. Arnaboldi, "Multiobjective evolutionary map design for cube 2: Sauerbraten," *IEEE Transactions on Games*, vol. 11, no. 1, pp. 36–47, March 2019.

[18] W. Cachia, A. Liapis, and G. N. Yannakakis, "Multi-level evolution of shooter levels," in *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2015.
[19] 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)," *CoRR*, vol. abs/1702.00539, 2017. [Online]. Available: http://arxiv.org/abs/1702.00539
[20] S. Dahlskog, J. Togelius, and M. J. Nelson, "Linear levels through n-grams," in *Proceedings of the 18th International Academic MindTrek Conference: Media Business, Management, Content & Services*. ACM, 2014, pp. 200–206.
[21] R. Jain, A. Isaksen, C. Holmgård, and J. Togelius, "Autoencoders for level generation, repair, and recognition," in *Proceedings of the ICCC Workshop on Computational Creativity and Games*, 2016.
[22] S. Snodgrass and S. Ontañón, "Experiments in map generation using markov chains." in *FDG*, 2014.
[23] A. J. Summerville and M. Mateas, "Sampling hyrule: Multi-technique probabilistic level generation for action role playing games," in *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2015.
[24] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial networks," *CoRR*, vol. abs/1406.2661, 2014. [Online]. Available: http://arxiv.org/abs/1406.2661
[25] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.
[26] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.
[27] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.
[28] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arxiv*, 2016.
[29] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," *CoRR*, vol. abs/1511.05440, 2015. [Online]. Available: http://arxiv.org/abs/1511.05440
[30] L. Yang, S. Chou, and Y. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation using 1d and 2d conditions," *CoRR*, vol. abs/1703.10847, 2017. [Online]. Available: http://arxiv.org/abs/1703.10847
[31] C. Beckham and C. J. Pal, "A step towards procedural terrain generation with gans," *CoRR*, vol. abs/1707.03383, 2017. [Online]. Available: http://arxiv.org/abs/1707.03383
[32] L. Horsley and D. Perez-Liebana, "Building an automatic sprite generator with deep convolutional generative adversarial networks," in *Computational Intelligence and Games (CIG), 2017 IEEE Conference on*. IEEE, 2017, pp. 134–141.
[33] P. Bontrager, W. Lin, J. Togelius, and S. Risi, "Deep interactive evolution," in *Computational Intelligence in Music, Sound, Art and Design*, A. Liapis, J. J. Romero Cardalda, and A. Ekárt, Eds. Cham: Springer International Publishing, 2018, pp. 267–282.
[34] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *CoRR*, vol. abs/1704.00028, 2017. [Online]. Available: http://arxiv.org/abs/1704.00028
[35] M. Luperto and F. Amigoni, "Predicting the global structure of indoor environments: A constructive machine learning approach," *Autonomous Robots*, Apr 2018. [Online]. Available: https://doi.org/10.1007/s10514-018-9732-7
[36] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, jul 2012.
[37] S. Luke, *Essentials of Metaheuristics*, 2nd ed. Lulu, 2013, available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.
[38] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001. [Online]. Available: https://doi.org/10.1162/106365601750190398
[39] H. Hofmann, H. Wickham, and K. Kafadar, "Letter-value plots: Boxplots for large data," *Journal of Computational and Graphical Statistics*, vol. 26, no. 3, pp. 469–477, 2017. [Online]. Available: https://doi.org/10.1080/10618600.2017.1305277