Autoencoder and Evolutionary Algorithm for Level Generation in Lode Runner

Sarjak Thakkar, Changxing Cao, Lifan Wang, Tae Jong Choi, Julian Togelius
New York University
New York, USA
{tsarjak, cc5766, lw2435, tc3045, julian.togelius}@nyu.edu

***Abstract*— Procedural content generation can be used to create arbitrarily large amounts of game levels automatically, but traditionally the PCG algorithms needed to be developed or adapted for each game manually. Procedural Content Generation via Machine Learning (PCGML) harnesses the power of machine learning to semi-automate the development of PCG solutions, training on existing game content so as to create new content from the trained models. One of the machine learning techniques that have been suggested for this purpose is the autoencoder. However, very limited work has been done to explore the potential of autoencoders for PCGML. In this paper, we train autoencoders on levels for the platform game Lode Runner, and use them to generate levels. Compared to previous work, we use a multi-channel approach to represent content in full fidelity, and we compare standard and variational autoencoders. We also evolve the values of the hidden layer of trained autoencoders in order to find levels with desired properties.**

I. Introduction

Procedural content generation (PCG) techniques are used for game content for several reasons, including introducing variation to the player, enabling particular game aesthetics, and automating some parts of game development. Most PCG algorithms need to be developed or tailored to each game. In contrast, PCGML uses machine learning to generate content, using models trained on existing content [1].

One machine learning method that has only been cursorily investigated for PCGML is the autoencoder, a type of neural network used for feature compression. It was recently proposed to use autoencoders for tasks such as level generation, level repairing, and level recognition [2]. However, that study really only scratched the surface. Among other things, the autoencoder in that study using a single channel for all types of tiles, severely limiting the usefulness of the method.

In this paper, we build on the recent research on using autoencoders for PCG [2]. We propose a multi-channel encoding for autoencoders to learn the structure and properties of the levels of Lode Runner. Concretely, the proposed method uses a five channel encoding to separately learn the patterns of the bricks, ladders, ropes, enemies, golds of the levels of Lode Runner. In addition, we hse a multi-population evolutionary algorithm that contains patch-wise crossover and latent variable mutation operators, which can retain some interesting features of original levels while inserting less predictable variations. Our experimental results showed that the proposed method is able to create diverse levels. Besides, we found out that variational autoencoders prone to generate more complex levels than vanilla autoencoders.

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

II. Related Work

In the past, Summerville et al. used LSTMs on the string representation of the map for PCG [3]. Snodgrass et al. suggested learning statistical patterns from existing levels generated by humans, and use those patterns for training Markov Chains and for generating new levels from such Markov Chains [4]. In 2016, Font et al. suggested using evolutionary methods for generating levels by using two-phase level encoding [5]. In 2017, Samuel Alvernaz et al. trained an autoencoder to create a comparatively low-dimensional representation of the environment observation, and then use CMA-ES to train neural network controllers acting on this input data [6]. The bottom line of all these approaches is that different representations of the same game combined with different algorithms achieve varying results.

On the other hand, Lode Runner has a much larger variety of chunks, and Lode Runner maps rely heavily on puzzle-like structures and mazes in the level design [7]. However, our approach of using variational autoencoder to learn the complex underlying structure of Lode Runner levels, coupled with an evolutionary algorithm to increase the playability score of the levels, generated good results. Notably, we find most of the raw generated levels from autoencoder as not playable until the evolution step is carried out. We will show more details of the technique we used in the following sections.

III. Our Approach

We selected a game called Lode Runner in order to evaluate our approach. We used the multi-channel encoding for the 150 original levels and trained an autoencoder using that data. The autoencoder is used

to generate a pool of 10,000 levels by inputting random seeds to the decoder. We evolved these levels by passing them through the evolutionary process.

A. Multi-Channel Encoding

While selecting the format of level data to be used for an autoencoder, the primary goal was to make the computation for encoding and decoding more logically feasible and reduce as much complexity as possible. In order to achieve this, we divided each level into various blocks that represent entities such as bricks, ladders, ropes, etc. By doing so, we reduced each level to a small size of 22x32 blocks. This representation can be considered as a 2D string, where each character of the string represents a block of a level (Fig. 1). For example, we used 'b' for a brick and '-' for a rope. The string representation enables our autoencoder to learn the underlying structure of a level.

Each character of a string is then later converted to a binary encoding for separate channels. In other words, the proposed method uses separate channels for bricks, ladders, ropes, etc. Each channel has 1 at the position depending on whether the particular block exists in a level, and 0 otherwise. This representation can be compared to the RGB channels of a color image. We used five channels for the proposed method. Fig. 2 shows a brick and ladder encodings of a level.

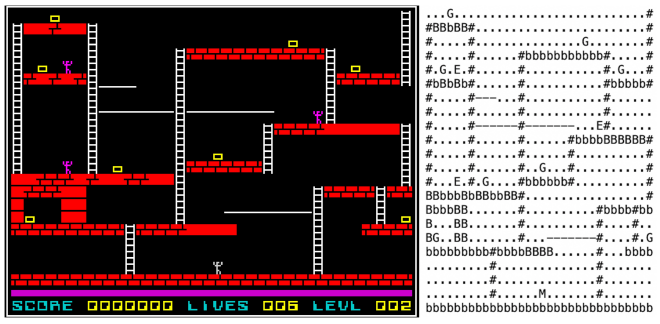


Fig. 1: Example of String Representation of Level

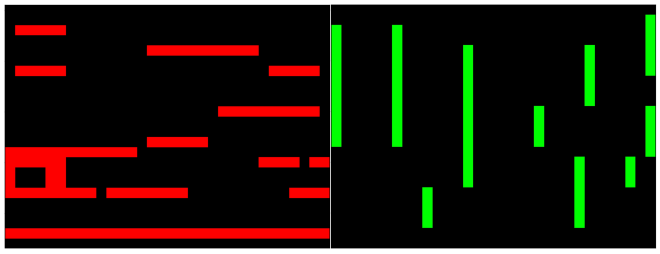


Fig. 2: Bricks (Red) and Ladders (Green) Encoding

B. Variational Autoencoder

Autoencoders have been used a variety of fields such as images, sounds, music. The representation of any data in the latent variable of an autoencoder can be used to reconstruct the original data from it.

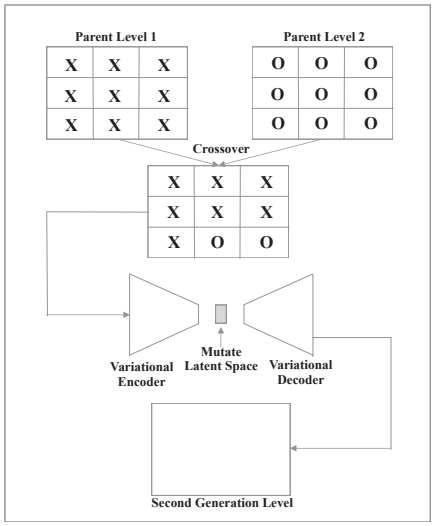


Fig. 3: Evolutionary Process of Proposed Method

Diederik et al. showed that the posterior inference can be made especially efficient by fitting an approximate inference mode [8]. The variational autoencoder is more advanced as it creates a distribution for each latent variable [9]. Hence, the variational autoencoder is able to reproduce images with more variations, compared to the vanilla autoencoder. We employed the variational autoencoder in the proposed method. One is to produce a large dataset that is up to 10,000 images as a global gene pool before the evolutionary process; the other is used in the evolutionary process to perform mutation on the latent representation of the level.

C. Evolutionary Process

After creating a global gene pool, the proposed method uses the evolutionary operators such as crossover and mutation to generate playable levels. For the level generation in Lode Runner, the proposed method uses the specially designed crossover and mutation operators, fitness function, and multi-population structure, which will be explained in the following subsections.

1) Crossover and Mutation Operators: In order to retain some interesting features of original levels, we applied a patch-wise crossover. As illustrated in Fig. 3, the parent images are divided into nine patches, and a child image inherits the first seven patches from parent 1 and the last two from parent 2. For the mutation, the proposed method takes the child image, passes it through the autoencoder, and randomly mutate its latent variable. The parameters for the crossover and mutation operators can be found in Table I. Fig. 3 illustrates the process of the mutation operator.

2) Fitness Function: This section describes the fitness function of the proposed method, which is consists of playability and connectivity.

- Playability: We employed a path-finding algorithm (PFA) to check if all the golds of a generated level

TABLE I: Used Control Parameters

Crossover Rate	Mutation Rate	Population Size	Number of Generations
0.78	0.1	32	150

can be collected. If they can be collected, the level is playable; otherwise, not playable.

- Connectivity: The connectivity measure evaluates the percentage of coordinate points in a generated level that can be reached.

The proposed method checks the playability measure first, and only playable levels are chosen to evolve. After that, the proposed method applies the connectivity measure in order to evaluate the fitness of each playable level. In other words, the more blocks are connected, the higher fitness scores are returned. We employed a PFA to check what proportion of the coordinate in a level is reachable.

3) Multi-Population Structure: Since we faced the lack of variations in generated levels from one evolution iteration, we applied a multi-population structure to avoid the premature convergence. The proposed method maintains ten populations separately and evolves ten unique and better image sets, which are then passed onto the next iteration. Since each of the ten populations contains its unique local characteristics, combining them creates more diverse image sets with several different characteristics revealed. Fig 4 shows the multi-population structure.

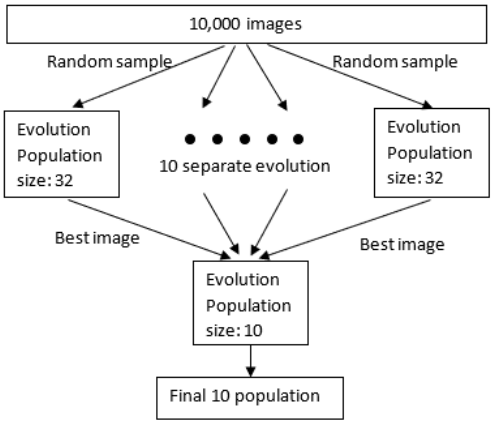


Fig. 4: Multi-Population Structure of Proposed Method

D. Path Finding Algorithm

A PFA in the proposed method is to check the playability and connectivity of generated levels. Based on the rules regarding the player movement in Lode Runner, we developed an algorithm, which acts as a playing agent. Some of those rules include - the player can move only up or down on the ladder; the player can dig the bricks; if the player falls off the edge of a brick, it keeps falling until it reaches a rope, ladder or brick. We employed A* algorithm as a PFA of the proposed method to find the shortest path for collecting all golds. Fig 5 is the solution path of an original level from A* algorithm. An extension of this approach, where the PFA tries to reach every block of the level helps to determine the connectivity of the level.

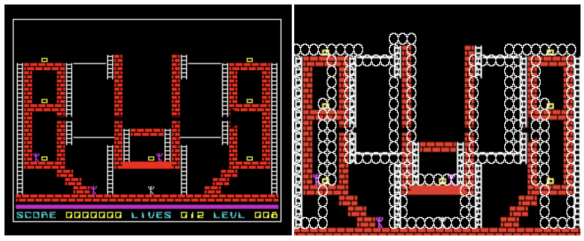


Fig. 5: Shortest Path from A* Algorithm

IV. Results

A. Results of Proposed Method with Variational Autoencoder

Fig. 6(a) consists of four randomly chosen levels out of 10,000 generated levels from the variational autoencoder (before evolution) with the parameters as described in Table II.

TABLE II: Parameters of Variational Autoencoder

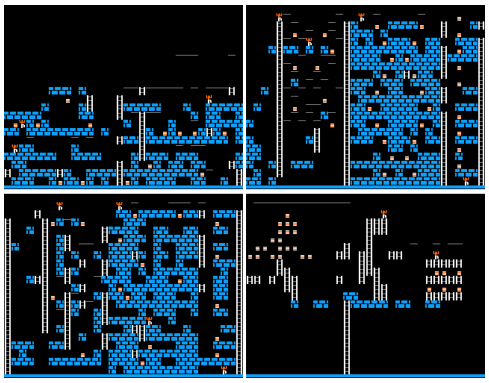
Training Data	Noise Dimension	Learning Rate	Number of Epochs
300	16	1e-3	1000

From Fig. 6(a) we can observe the variational autoencoder results. In these results, the left-upper one is not valid because all of the golds cannot be collected. As we mentioned above, only playable levels are used to generate child levels. Fig. 7(b) consists of two randomly chosen results after the evolutionary process combined with the variational autoencoder. Since the vanilla and variational autoencoders generate levels with varying densities, we decided to use them separately to generate dense or sparse levels as hard and easy levels, shown in the next.

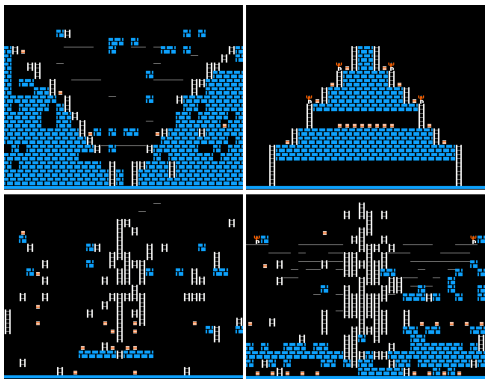
B. Results of Proposed Method with Vanilla Autoencoder

Fig. 7(a) consists of two randomly chosen results after the evolutionary process combined with the vanilla autoencoder, which are sparse and easy for rookies. From Fig. 7(b), we can see that the results generated with variational autoencoder are dense and hard for rookies.

1) Level Differences Between Variational and Vanilla Autoencoders: Compared to a vanilla autoencoder, a variational autoencoder adds regularization constraints to the encoding process by forcing the generated implicit vector to follow a standard normal distribution. On



(a) Variational Autoencoder with Gaussian Noise



(b) Vanilla Autoencoder with Uniform Noise

Fig. 6: Results of Variational and Vanilla Autoencoders

applying evolutionary algorithm on variational autoencoder, generated levels grow more branches of details with small mutations in the latent space, eventually making generations different. On the other hand, the vanilla autoencoder does not regularize the implicit vector, as a result of which, the mutation results in lesser details.

C. Similarity of Generated Levels to Original Levels

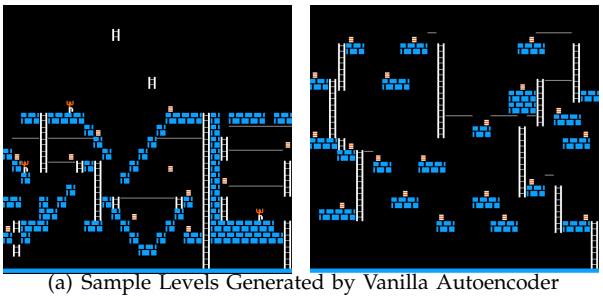
We compared all of the 330 generated levels with the original levels of the game, block by block (ignoring the blank space) to calculate the similarity proportion. Table III shows the mean similarity and the number of generated levels that are over 30% and 50% similar to any of the existing levels of the game, respectively.

TABLE III: Similarity of Generated Levels

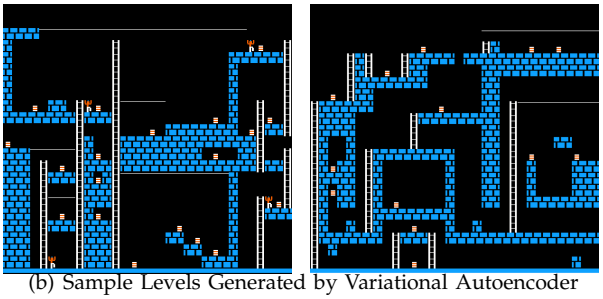
	Mean Similarity	Num Levels >30% Similarity	Num Levels >50% Similarity
Variational	25.36%	73 / 330	3 / 330
Vanilla	27.05%	130 / 330	38 / 330

V. Conclusions

In this paper, we proposed a multi-channel encoding for an autoencoder with a multi-population evolutionary algorithm to generate levels for Lode Runner. We used patch-wise crossover and latent variable



(a) Sample Levels Generated by Vanilla Autoencoder



(b) Sample Levels Generated by Variational Autoencoder

Fig. 7: Results After Evolutionary Process

mutation, which helps to retain some interesting features of original levels while inserting less predictable variations. Our experimental results showed that the proposed method can create large amounts of significantly different playable levels. We also discussed the difference in generated levels of the variational and vanilla autoencoders. For future work, we will apply the proposed method to other games such as Super Mario Bros and Mario Kart.

References

[1] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgård, A. K. Hoover, A. Isaksen, A. Nealen, and J. Togelius, "Procedural content generation via machine learning (pcgml)," *IEEE Transactions on Games*, vol. 10, no. 3, pp. 257–270, 2018.

[2] R. Jain, A. Isaksen, C. Holmgård, and J. Togelius, "Autoencoders for level generation, repair, and recognition."

[3] A. J. Summerville and M. Mateas, "Super mario as a string: Platformer level generation via lstms," 2016.

[4] S. Snodgrass and S. Ontañón, "Experiments in map generation using markov chains."

[5] J. M. Font, R. Izquierdo, D. Manrique, and J. Togelius, "Constrained level generation through grammar-based evolutionary algorithms," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2016, pp. 558–573.

[6] S. Alvernaz and J. Togelius, "Autoencoder-augmented neuroevolution for visual doom playing," in *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, Aug 2017, pp. 1–8.

[7] S. Snodgrass and S. Ontanon, "A hierarchical mdmc approach to 2d video game map generation," in *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2015.

[8] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," Dec 2013, p. arXiv:1312.6114.

[9] I. Gulrajani, K. Kumar, F. Ahmed, A. A. Taiga, F. Visin, D. Vazquez, and A. Courville, "PixelVAE: A Latent Variable Model for Natural Images," *arXiv e-prints*, p. arXiv:1611.05013, Nov 2016.