Emerging Complexity in Lenia

Sanyam Jain
Dept of Computer Science
Østfold University College
Halden, Norway
sanyamj@hiof.no

Aarati Shrestha
Dept of Computer Science
Østfold University College
Halden, Norway
Aarati.Shrestha@hiof.no

***Abstract*—This research project investigates Lenia, an artificial life platform that simulates ecosystems of digital creatures. Lenia's ecosystem consists of simple, artificial organisms that can move, consume, grow, and reproduce. The platform is important as a tool for studying artificial life and evolution, as it provides a scalable and flexible environment for creating a diverse range of organisms with varying abilities and behaviors. Measuring complexity in Lenia is key aspect of the study, which identifies the metrics for measuring long-term complex emerging behavior of rules, with the aim of evolving better Lenia behaviors which are yet not discovered. The Genetic Algorithm uses neighborhoods or kernels as genotype while keeping rest of the parameters of lenia as fixed to produce different behaviors respective to the population and then measures fitness value to decide the complexity of the resulting behaviour. The world with best fitness are selected and given chance to carry the genetic material forward in further generations. Overall, this project aims to deepen our understanding of artificial life and evolution by investigating the potential of Lenia as tools for modeling complex systems and emergent phenomena in open-ended ecosystem.**

***Index Terms*—Continuous CA, Lenia, Evolutionary Computation, Artificial Life**

I. Introduction

Open-endedness is important in AI [1], [2] because it enables the development of more flexible, creative, and autonomous systems that can solve a wider range of problems and tasks. It also facilitates the emergence of unexpected and potentially useful behaviors or solutions that may not have been anticipated by human designers. In addition, open-endedness allows AI systems to continue to learn and improve over time, adapting to changing environments and evolving to meet new challenges.Lenia is a digital life form created by mathematician and computer scientist Bert Wang-Chak Chan in 2019 as Independent researcher [3], [4]. It is a fascinating and beautiful example of how simple rules can give rise to complex behavior in a digital system. It is a type of artificial life simulation that uses a continuous cellular automaton to generate complex, evolving patterns. Lenia differs from traditional cellular automata in that its cells are not limited to discrete states like "on" or "off". Instead, each cell can take on a continuous range of values, which allows for more fluid and organic patterns to emerge. Lenia is also unique in that it allows for a wide range of parameters to be adjusted, such as the size and shape of the grid, the rules for how cells interact (kernel), and the speed at which the simulation runs (growth function). This flexibility has led to a diverse array of visual and auditory creations, ranging from mesmerizing abstract patterns to complex rhythmic compositions.

One of the reasons why most of the research in Lenia is focused on finding new species is because Lenia is an open-ended, evolving system that has the potential to generate an infinite number of unique and complex patterns. This makes it an ideal system for exploring the principles of emergent behavior and evolutionary dynamics. By searching for new species in Lenia, researchers can uncover new and unexpected patterns that may have applications in fields such as digital art, music, and data visualization. These patterns can also provide insights into the underlying principles that govern complex systems and help us better understand the behavior of biological systems. In addition, the search for new species in Lenia can also lead to the discovery of novel algorithms and computational techniques that can be applied to other fields. For example, the methods used to analyze and classify Lenia patterns can be applied to biotechnology, genetic evolution of real organisms and synthetic biology. However in our research, we keep the system open-ended, self organising and let evolution produce behaviours in the Lenia subsystem.

Cellular automata (CA) have proven applications in the field of self-organized control. In a paper [5] authors describe how they used a variant of CA called Neural Cellular Automata (NCA) to develop a control system for a cart-pole agent. The goal of the control system is to keep the pole balanced while moving the cart in response to external disturbances. This work used neural cellular automata to control a cart-pole agent through deep Q-learning, demonstrating stable behavior over many iterations and exhibiting life-like phenomena such as regeneration and robustness to disruptions. Further, highly complex and self organising versions of classical Lenia such as Sensorimotor Lenia [6], Flow Lenia [7], and Energy based Particle Lenia [8] shows advanced physics, chemistry and biology of the matter.

II. Related work

A. Lenia

In the Lenia system [3], [4], altering the values of mu and sigma for the growth and kernel functions can generate different species (we describe growth function and kernel in next paragraph). Over 500 species have been discovered so far, with each residing in a unique combination of these parameters. The values can be varied through evolutionary

computation, but the outcome is often a dead or chaotic grid. Researchers typically use smooth ring kernel shells with different values of mu and sigma. To conform to the taxonomy, the discovery of new species must pass specific benchmarks. Lenia uses a kernel, a circular matrix that acts as a filter to smooth the state of the simulation board. The kernel is circular, with a diameter that is equal to the size of the kernel and centered around the current cell being processed. During convolution, the kernel is slid over the entire board, and for each cell, the corresponding values in the kernel and surrounding cells are multiplied and summed to generate a new value for that cell. Lenia is a highly complex system that allows researchers to manipulate the mu and sigma values of the growth and kernel functions to create a diverse array of species. The discovery of new species is an active area of research, with efforts focused on using interactive evolutionary systems. The Gaussian function is used to calculate the values in the kernel, and it has many applications in various fields, including image processing. Gaussian filtering is a common technique used to smooth or blur images by convolving the image with a Gaussian kernel, with the size and shape of the kernel depending on the desired effect.

$$G(U) = \frac{1}{2\pi\sigma^2} e^{-\frac{(U-\mu)^2}{2\sigma^2}} \tag{1}$$

where σ is the standard deviation of the Gaussian function, μ is its mean, and U is the colvolved weighted sum of the kernel and board output. The Gaussian kernel is used for smoothing and blurring the simulation board, and it is a circular matrix that is centered around the current cell. The number of smooth rings or "shells" in the Gaussian kernel is determined by the "peaks" parameter, which specifies the amplitude of the peaks for each shell. The diameter of the kernel is specified by the user, and the parameter "R" is calculated as half of the diameter to generate a square grid of pixels around the current cell. The distance from the center of the kernel to each shell for every pixel is calculated using the Euclidean norm and stored in the "D" variable. The peak value for each pixel is determined by multiplying the peak value of the closest shell with the normalized distance from the center of the kernel to the closest shell for that pixel, and the Gaussian function is applied to each pixel using specified values of mu and sigma. The resulting smoothed kernel can be used to update the state of the simulation board. The growth function updates the simulation board state based on the neighborhood sum, which is calculated as the convolved value of the kernel over board. The Gaussian growth function is then applied to the neighborhood sum with two parameters: the mean or center of the curve m and the standard deviation or spread of the curve. This function provides a smooth and continuous function that gradually changes the board state in a more fluid-like manner. The smoothness of the Gaussian function prevents sudden and unrealistic patterns. By adjusting the mean and standard deviation parameters of the Gaussian function, each cell's growth or decay can be precisely controlled. This allows for a wide range of patterns, from simple geometric shapes to complex and intricate structures in the Lenia simulation. A detailed view of kernel and growth function can be seen in Figure 1

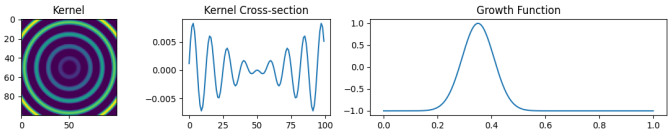

Fig. 1. Visualising Kernel, Gaussian Kernel Function and Gaussian Growth Function

A precise Lenia algorithm works as shown in Algorithm 1.

1) Take a 2D array (world A) of real values between 0 and 1, initialized with an initial pattern A_0.
2) Calculate weighted sums of A with a predefined array (kernel K), which is equivalent to calculate the convolution $K * A$; the kernel K has radius R, forming a ring or multiple concentric rings (parameter β = list of peak value of each ring).
3) Apply a growth mapping function G to the weighted sums; the growth mapping G is any unimodal function (parameters μ = growth center, σ = growth width).
4) Add a small portion dt of the values back to the array A.
5) Finally clip the states of A to between 0 and 1.
6) Repeat steps 2-5 for each time-step.

$$A_{t+dt} = [A_t + dt.G(K * A_t)]_{0,1} \tag{2}$$

In this formula, A_t and A_{t+dt} represent the values of the array world A before and after the update at time t and $t + dt$, respectively. The operator $*$ represents the convolution operation between the kernel K and the array A_t. The function $G(\cdot)$ represents the growth mapping function with parameters μ and σ, applied to the weighted sums of the convolution result. Finally, $[0, 1]$ represents the range of valid values for the array.

B. Lenia Taxonomy, Morphology and Behaviour Dynamics

The Lenia universe has a hierarchical taxonomy [3], [4] based on the morphology, behavior, and statistical properties of its lifeforms. Lifeforms within a species share the same morphology and behavior on both global and local scales, while a genus consists of species that share the same global morphology but differ locally. At the subfamily level, lifeforms occupy parallel niches of similar shapes, while a family is composed of subfamilies with the same architecture. At the highest level, a class is a grouping based on the arrangement of kernel, which influences behavior and morphology. Lenia morphology is characterized by a focus on symmetry, and new taxa must exhibit symmetry, with different types of symmetry associated with different modes of movement or behavior. Lenia behavior dynamics include stationary directional movement, solid, oscillating, or deviated local movement, random modes of movement, and particle reactions such as fission and fusion. These behaviors suggest that Lenia is highly adaptable

and capable of exhibiting a range of movement patterns and behaviors. For example, a known species show in Figure 2

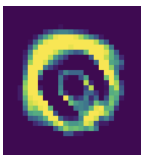

Fig. 2. A known specie with mu 0.24, sigma 0.029

C. Evolutionary Cellular Automata

Evolutionary cellular automata (CA) [9] can be used to evolve Lenia, a digital life form with complex behaviors and morphology. The search for useful CA rules is challenging due to the difficulty of predicting the outcome of a given rule set from a starting configuration. To overcome this challenge, an evolutionary search is a natural choice for exploring CA with desired properties. However, traditional CA are difficult to evolve naively, and variants of the framework are often used to improve evolvability. For example, representations inspired by genetic programming have been used for the transition function. While traditional CA have a consistent transition function for all cells, Lenia experiments have explored the use of homogeneous growth functions while mutating the kernel using genetic algoithm. The ability to evolve CA rules allows for the exploration of new and interesting behaviors and structures is same in Lenia considering rules analog to kernel and growth functions, leading to a better understanding of its complex and diverse universe.

D. Measuring Complexity

Measuring complexity in Cellular Automata has been widely studied before [10] and [11]. Two of the metrics that we use in our project are described follows:

1) Variation Over Time: Measuring complexity in Lenia can be a challenging task, as the board can exhibit a wide range of dynamic behaviors over time, ranging from simple repetitive patterns to highly complex and unpredictable behaviors. One way to quantify complexity is to measure the variation in the state of the board over time. One approach is to measure the number of active cells on the board at each time step. The number of active cells can provide a rough estimate of the complexity of the system, as more active cells typically indicate a higher degree of complexity. However, it is important to note that the number of active cells can vary significantly between different time steps, and may not be a reliable measure of complexity in all cases. To account for this variability, mathematical models can be used to measure the variation in the state of the board over time. It is the deviation based measure of complexity, which calculates the differences of the distribution of active cells on the board at certain time steps. Deviation is a measure of the randomness or differences in states of a system, and can be used to quantify the variation of complexity in a system over time. An overview of the approach is shown in Figure 3

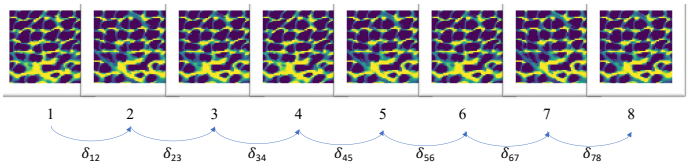

Fig. 3. Variation over time approach to measure complexity

2) Compression based: Another approach, uses encoder-decoder network to compress the input images. Encoder-decoder networks have been proposed as a way to measure the complexity of cellular automata such as Lenia. In this approach, the encoder network is trained to encode the current state of the Lenia board into a lower-dimensional feature space, while the decoder network is trained to decode the features back to the original state. The key idea is that the amount of information that is lost during the encoding and decoding process can be used as a measure of the complexity of the Lenia board. The calculated Reconstruction loss (using a distance meausure) will then decide whether the input is complex or ordered. Hypothesis is, if the input is complex and radnom, it is challenging for decoder to reconstruct the input and hence it will result higher reconstruction loss while in the ordered behaviour the reconstruction loss would be small. To train the encoder-decoder network, a dataset of Lenia board states is first generated by running the simulation for a certain number of steps. The encoder network is then trained to map each board state to a set of features that capture the key patterns and structures in the board. The decoder network is trained to reconstruct the board state from the features, with the goal of minimizing the reconstruction error. An overview of the approach is shown in Figure 4

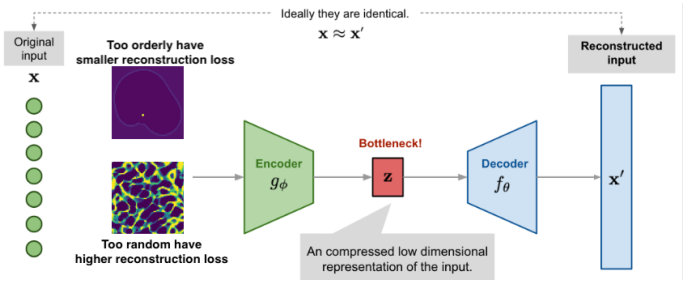

Fig. 4. Auto-encoder approach to measure complexity

III. Methodology

This paper aims to simulate a mathematical model of Lenia using Python. The methodology section presents the code that is implemented to achieve this objective. To achieve this goal, various libraries are imported such as NumPy, Matplotlib, and SciPy. These libraries help to perform mathematical computations, plot graphs, and apply convolution on arrays. The code uses the class Lenia, which initializes with a kernel and board. The kernel is defined as a Gaussian function with parameters mu and sigma, and a growth function. The board is defined as a

numpy array, which holds the state of the Lenia at a given time. The $show\_board()$ function creates a plot of the board and displays it on the screen. The $animate()$ function is used to create an animation of the board. It runs for a specified number of frames with a frame interval of 50ms. The $animate\_step()$ function updates the board for each frame and renders it on the plot. The $save\_animation()$ function saves the animation in the GIF format. The $normalise\_kernel()$ function normalizes the kernel such that the sum of all its elements is equal to one. The $plot\_kernel\_info()$ function plots the kernel, its cross-section, and the growth function. The $run\_simulation()$ function initiates the simulation process by calling the $animate()$ function and saving the output animation in a specified directory with a file name of output.gif.

The methodology of the study involves implementing a genetic algorithm to optimize a Lenia kernel for a given board size to produce non-boring behaviour. The following steps were carried out to achieve this:

1) Random Kernel and Board Generator: Two functions were created to generate a random kernel and a random board. The kernel generator function randomly generates a square matrix of size $kernel\_sizexkernel\_size$ with random values ranging between 0 and 1, rounded off to 3 decimal places. The board generator function generates a square matrix of size $board\_sizexboard\_size$ with random values ranging between 0 and 1, rounded off to 3 decimal places.
2) Chromosome: An Individual class was created, which consists of the kernel generated from the random kernel generator and its corresponding fitness value. The fitness value is determined by calculating the number of live cells in the board after running the Lenia simulation using the generated kernel.
3) Population: A Population class was created, which consists of a list of Individuals generated from the random kernel generator. And the board generated from the random board generator is used for all individuals in the population.
4) Genetic Algorithm: A Genetic Algorithm class was created, which consists of a static method for mutating a population of individuals. The mutation rate was set at 0.01, which means that there is a 1% chance of mutating each value in the kernel. For each gene in the kernel, the code checks if a random number generated by $random.random()$ is less than the mutation rate. If so, the gene is mutated by replacing it with a new random number generated by $np.random.rand()$, rounded to three decimal places using $np.round()$.
5) Run Genetic Algorithm: A run_ga function was created to run the genetic algorithm for a specified number of generations. The function starts by generating a population of individuals, then for each generation, the fitness of each individual is calculated using the board generated from the random board generator. The individuals are then sorted in descending order based on their fitness values. The top individual is selected as an elite individual and retained for the next generation, and the rest of the population is mutated using the Genetic Algorithm class. A new population is then created by selecting mutated individuals randomly from the population. This process is repeated for the specified number of generations. The final population of individuals sorted in descending order of their fitness values is printed at the end of the run_ga function.

IV. Implementation

V. Results

VI. Conclusions

References

[1] K. O. Stanley, J. Lehman, and L. Soros, "Open-endedness: The last grand challenge you've never heard of," While open-endedness could be a force for discovering intelligence, it could also be a component of AI itself, 2017.

[2] K. Gregor and F. Besse, "Self-organizing intelligent matter: A blueprint for an ai generating algorithm," arXiv preprint arXiv:2101.07627, 2021.

[3] B. W.-C. Chan, "Lenia-biology of artificial life," arXiv preprint arXiv:1812.05433, 2018.

[4] B. W.-C. Chan, "Lenia and expanded universe," arXiv preprint arXiv:2005.03742, 2020.

[5] A. Variengien, S. Nichele, T. Glover, and S. Pontes-Filho, "Towards self-organized control: Using neural cellular automata to robustly control a cart-pole agent," arXiv preprint arXiv:2106.15240, 2021.

[6] G. Hamon, M. Etcheverry, B. W.-C. Chan, C. Moulin-Frier, and P.-Y. Oudeyer, "Learning sensorimotor agency in cellular automata," 2022.

[7] E. Plantec, G. Hamon, M. Etcheverry, P.-Y. Oudeyer, C. Moulin-Frier, and B. W.-C. Chan, "Flow Lenia: Mass conservation for the study of virtual creatures in continuous cellular automata," arXiv preprint arXiv:2212.07906, 2022.

[8] Alexander Mordvintsev, Eyvind Niklasson, Ettore Randazzo, Google December 23, 2022 Particle Lenia and the energy-based formulation (no date). Available at: https://google-research.github.io/self-organising-systems/particle-lenia/ (Accessed: March 6, 2023).

[9] D. Medernach, T. Kowaliw, C. Ryan, and R. Doursat, "Long-term evolutionary dynamics in heterogeneous cellular automata," in Proceedings of the 15th annual conference on Genetic and evolutionary computation, 2013, pp. 231–238.

[10] H. Cisneros, J. Sivic, and T. Mikolov, "Evolving structures in complex systems," in 2019 IEEE Symposium Series on Computational Intelligence (SSCI), 2019, pp. 230–237.

[11] H. Cisneros, J. Sivic, and T. Mikolov, "Visualizing computation in large-scale cellular automata," arXiv preprint arXiv:2104.01008, 2021.