UNIVERSITY
OF OSLO

Master's thesis

Automatically discovering patterns in Lenia

Cornelia Drougge Vassbotn

Informatics: Robotics and Intelligent Systems
60 ECTS study points

Department of Informatics
Faculty of Mathematics and Natural Sciences

Spring 2023

Cornelia Drougge Vassbotn

Automatically discovering patterns in Lenia

Supervisors:
Emma Stensby Norstein and Kyrre Glette

Abstract

Lenia is a 2D cellular automata, with continuous states, space, and time, and has the ability to generate lifelike and captivating virtual creatures. Lenia has been shown to be a great way for experimenting with evolution and artificial life, as life in Lenia is highly flexible and graphic, and shown to be capable of movement, growth, consumption, reproduction, self-repair, self-defense, and swarming, among others. Although Lenia demonstrates a variety of fascinating behaviors, it is a highly complex system, making it incredibly time-consuming to explore the vast space of potential Lenia patterns. This is due to the presence of numerous patterns that bears similarities to one another within the system. When encountering a large solutions space, a known optimization technique is the use of quality diversity, which we aim to utilize. This approach entails seeking a diverse array of solutions and behaviors, each showcasing a unique approach to solving the task at hand. By doing so, we can effectively explore a broad range of possibilities and uncover a multitude of distinct strategies for addressing the given problem. We want to exploit this advantage of quality diversity, with the aim of discovering a wide range of patterns in Lenia. In order to obtain a diverse range of patterns, it is necessary to distinguish between the various behaviors exhibited by each pattern. Through the automation of behavioral descriptor creation, we use a neural network to capture the essential aspects of each pattern that may not be explicitly described manually, and differentiates the patterns for us.

This thesis investigates whether automatically defined behavioral descriptors are helpful tools to achieve a wide range of patterns in Lenia. This was achieved by developing a new method for automatically searching through Lenia, by combining quality diversity and behavior descriptors. To better understand Lenia and achieve our goal, we focused on exploring pattern movement and used discovered patterns to pre-train behavioral descriptors. Using this method, we produced multiple collections of patterns with both animal and non-animal patterns. These collections were then used to analyze and evaluate the efficacy of our approach, by review of the state of the individuals, fitness, visualization of the patterns and archive, and inspecting the different groupings of patterns. This has enabled us to find strengths of our method, such as the method's ability discover intricate patterns and differentiate between complex textures, allowing a higher interpretation of behavior descriptors. We believe that the presented method can provide new insight into searching through the space of Lenia. Furthermore, the use of automatic behavioral descriptors assists in understanding and analyzing digital life more effectively, and that further in-depth studies could reveal new potential patterns in Lenia.

Acknowledgments

First and foremost, I would like to thank my supervisor Emma Stensby Norstein for her guidance, great ideas, and continuous feedback during this whole project. In addition, I would also like to thank co-supervisor Kyrre Glette for additional feedback and guidance throughout this project. Furthermore, I would like to thank the ROBIN research group for allowing me to use their resources in this project.

Further, I would also like to thank my family for all their support and guidance throughout this entire project. A special thanks goes out to my friends at IFI for engaging in stimulating discussions that enriched this project. Finally, I would like to thank Jørgen G. Russnes for his unconditional support, insightful ideas, and great patience during this project.

This work was conducted on the Fox supercomputer resource, owned by the University of Oslo Center for Information Technology.

Contents

List of Figures

List of Tables

Abbreviations

AE Autoencoders

AML AURORA Modified for exploring Lenia

AURORA Autonomous skill discovery with Quality-Diversity and Unsupervised Descriptors

BD Behavioral Descriptors

CA Cellular Automata

CVT Centroidal Voronoi Tessellation

CVT-MAP-Elites Centroidal Voronoi Tessellations MAP-Elites

DEAP Distributed Evolutionary Algorithms in Python

EA Evolutionary Algorithms

MAP - Elites Multi-dimensional Archive of Phenotypic Elites

NSLC Novelty search with local competition was developed

PCA Principal component analysis

QD Quality-Diversity

VAE Variational Autoencoders

Chapter 1

Introduction

1.1 Motivation

Nature has inspired our progress in humanity from the beginning of time. From birds giving us the idea to make planes, to artificial intelligence trying to replicate the brain, and artificial life trying to mimic the origin of life[1]. The study of artificial life began with the aim of understanding and replicating the principles of living systems through the creation of synthetic systems that exhibit life-like properties. It is therefore interesting to try to simulate how particle systems with simple rules, interact with each other and construct complex patterns. In 2001 Bedau et al.[2] summarized questions in artificial life to fourteen open problems with three major themes: The transition to life, the evolutionary potential of life, and the relation between life and mind and culture. These themes have all interesting sub-questions, but the transition to life, question three is of special interest: " 3. Determine whether fundamentally novel living organizations can exist."(Bedau et al.[2]).

If we are able to identify the characteristics that define a novel living entity, it could provide us with valuable insights into the requirements for creating life and the limitations that control life's existence[2].

Numerous methods have been employed to address this issue, including the development of particle systems such as swarm chemistry [3], cellular automata in Game of Life [4], and virtual creatures [5]. These particle systems have all been capable of replicating intricate lifelike behaviors. The incorporation of interactions among these particles has revealed the emergence of intricate patterns, much like those observed in the evolution of life on Earth.

To approach question three, B. Chan presented in 2019 *Lenia* [6], a 2D cellular automata with continuous space, time, and states. In the study of Lenia [6] B. Chan argues that Lenia is a strong contribution to the goal of artificial life, and shows similarities to biological life. To continue research with Lenia, and explore Lenia's potential, we might be able to get a stronger insight into the emergence of biological life and our universe.

While Lenia has the capability to generate a wide variety of patterns, searching through the space of patterns to find interesting ones can be time-consuming, given the high degree of similarity among many of the patterns. Effectively automating search through the solution space of Lenia is therefore essential when searching for new patterns. To search through the solution space of patterns, and with that different behaviors, we need a way of differentiating the different behaviors to achieve a wide array of solutions.

It is shown that automatically defined behavioral descriptors with the use of dimensional reduction algorithms can be a successful tool in Lenia [7]. Further, the combination of automatically defined behavioral descriptors and quality diversity has shown success to achieve a large number of possible behaviors without any prior knowledge or goal in the field of robotics [8]. However, to our knowledge using the combination of quality diversity and atomically defined behavior descriptors have not been attempted in Lenia. We believe that this approach has the ability to discovery of an even wider range of patters than has previously been possible in Lenia.

This thesis aims to explore Lenia and automate the search for novel and interesting creatures, with the use of a new method of combining evolutionary algorithms and automatically defined behavioral descriptors.

1.2 Research Goal

The aim of this thesis is to *investigate whether automatically defined behavioral descriptors are helpful tools to achieve a wide range of patterns in Lenia.*

In order to accomplish this, we have defined a few sub-aims. While Lenia has the ability to generate a wide variety of patterns, searching for the fascinating ones within the solution space can be a time-consuming task due to the numerous similarities among them. To discover many different solutions, and a wide array of behaviors, we want to use the approach of Quality Diversity in Lenia. Given the extensive parameter space in Lenia, it is not efficient to focus on distinguishing between individual parameters. Instead, our aim is to differentiate between the various behaviors exhibited by Lenia, by using behavior descriptors. This forms our first sub-aim of finding a method of automatically exploring Lenia, by combining quality diversity and automatically defined behavior descriptors. To create the behavior descriptors, we want to train a model that captures the most important information of each pattern. We also want to investigate whether already discovered patterns can be used to improve the search. This defines the second sub-aim, which involves investigating how the impact of pre-training the behavior descriptors affects the outcome. To further direct the search toward interesting patterns, we want to direct the search toward movement in Lenia patterns. This involves a comparison of the use of a movement-based fitness function and a random-based fitness function. This makes the third sub-aim directing the search toward the patterns movement, and assessing whether such movement facilitates the discovery of new and diverse patterns in Lenia. Finally, we want to analyze the resulting patterns, in order to evaluate our success with the method. We can condense the sub-aims as follows:

- Finding a method of automatically exploring Lenia by using a quality diversity algorithm with automatically defined behavior descriptors.
- Investigate what impact pre-training the behavior descriptors has on the outcome.
- Direct the search toward the pattern's movement, and assess whether such movement facilitates the discovery of new and diverse patterns in Lenia.
- Analyze the patterns, in order to evaluate the efficacy of our approach.

1.3 Contributions

This thesis has the following contributions:

A method of automatically exploring Lenia: AML

We present a method that involves the integration of Lenia into the existing AURORA method[8] with certain customizations tailored for our thesis. This approach enables the automated exploration of Lenia. We called this method **AML**; **A**URORA **M**odified for exploring **L**enia. The method includes using a quality diversity algorithm with automatically defined behavior descriptors. A method of centering the patterns to get more precise behavioral descriptors were developed.

Directing the search toward movement

We introduce a fitness function that directs the search in AML toward movement within the patterns. We conducted a comparison between the resulting patterns obtained with the directed search and patterns achieved without any guidance. Through this comparison, we observed that employing a movement-based fitness function to direct the search led to a higher percentage of animal patterns compared to a non-directed search. Consequently, our approach resulted in an archive containing an increased number of animal-patterns, along with fascinating patterns.

Using Discovered Animal Patterns for Pre-Training

In order to compare the impact of pre-training the behavior descriptor with discovered patterns, we conducted a comparison with an experiment where no pre-training was performed. Our findings revealed promising indications that incorporating discovered patterns during the pre-training phase improves both the quality and quantity of the patterns discovered.

Obtained a VAE Capable of Differentiate Complex Patterns

During our analysis of the obtained archives, we noticed a diverse range of similar patterns. Additionally, we examined the reconstructed patterns generated by the VAE and observed a notable improvement in the level of detail compared to the original VAE upon which our VAE is built. These observations provide evidence of AML's capability to distinguish intricate patterns.

1.4 Thesis Outline

Chapter 2: Background

This chapter presents an introduction to relevant research and related work. We focus on the fields of Lenia, Evolutionary algorithms, Quality Diversity, and Variational Auto Encoders. We also describe the AURORA algorithm [8], which our thesis is built upon.

Chapter 3: Implementation

Chapter 3 describes the implementation, and the choices made during designing of AML. Further, Chapter 3 describes the implementation of each element in AML. These are the algorithm we use to create and evolve individuals, the Lenia implementation, and the creation of behavior descriptors.

Chapter 4: Experiments

In chapter 4, we present the experiments, and the the relevant implementation for each experiment.

Chapter 5: Results

In this chapter we present the results for our experiments, which involves viewing of the state of the last generation, analysis of the fitness, visualization of the archive, inspection of activation mass of the population, visual inspection of the patterns, and reconstructed patterns of the VAE.

Chapter 6: Discussion

Chapter 6 contains discussions about the results found in the previous chapter.

Chapter 7: Conclusion and Future Work

The final chapter contains a summary of the results as well as the conclusion. Additionally, we discuss possible future work.

Appendix A

An appendix can be located after Chapter 7, with example patterns from each experiment.

Chapter 2

Background

2.1 Cellular Automata

Cellular Automata (CA) [9] was first presented in the 1940s by John Von Neumann and Stanislaw Ulam. CA is a mathematical system that consists of a grid of cells, each of which can be a finite number of states. CA is able to show how simple rules can lead to complex patterns and can simulate complex real-world scenarios [10]. The evolution of CA is based on an initial state, and the next state is generated based on the state before and a set of predefined rules.

Conway's Game of Life [11] is one example of two-dimensional CA. As described by Adamatzky [11], the rules are as follows :

- If a live cell is touching 2 or 3 live cells (called "neighbors"), then it remains alive the next generation, otherwise it dies.
- If a non-living cell is touching exactly 3 live cells, it comes to life next generation

With these rules, Conway was able to simulate interesting formations. Extended research from scientists has shown how Conway's Game of Life is Turing Complete [12], self-replicating patterns occur [13], and a huge variety of patterns have been found [11]. We can see an example of a generated state in Figure 2.1.

Since the first publication of CA, scientists have shown interesting research of cellular automaton in a variety of fields, such as biology [14], chemistry [15] and physics [16]. Within biology phenomenons, CA has been able to simulate biological processes. The study of S. Coombes [17] showed how the patterns of the geometry and pigmentation of seashells were able to be simulated using cellular automata.

2.2 Lenia

The cellular automata Lenia is a system of artificial life and was first introduced in 2018 by Chan [6]. Lenia is a two-dimensional CA with continuous space-time-state and generalized local rules [6].

Lenia started as a project of experimenting with Game of Life and resulted in an extension of Game of Life. In Lenia, we have a continuous space using arbitrary long-range neighborhoods and continuous time through arbitrary incremental updates, as well as continuous states using real numbers. In other words, Lenia neighborhoods are defined by smooth convolution kernels and a growth function.

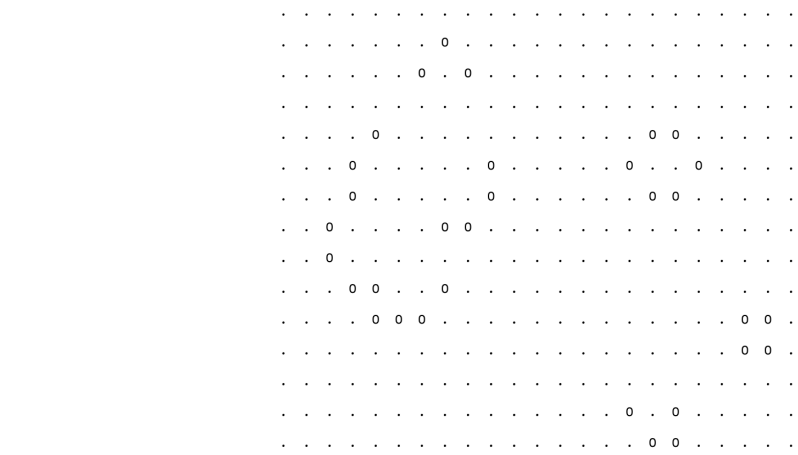


Figure 2.1: Example of a randomly generated game of life - board after 70 generations. The zeros are living cells, and the dots are dead. We can see that two static formations have been formed, the circle in the upper right and the square in the lower right. The left bigger formation and the lowest formation have not yet reached equilibrium.

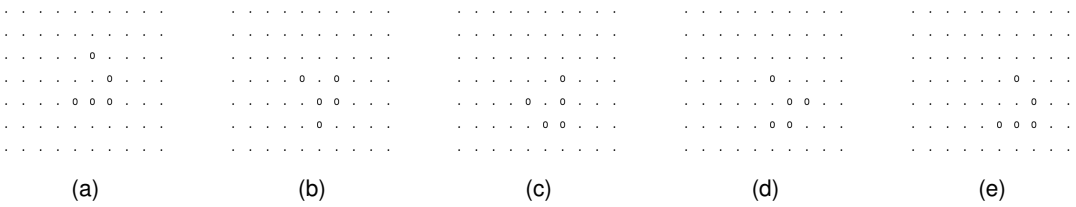


Figure 2.2: Examples of gliders, and its update path.

By comparing the formation of Game of Life in Figure 2.1, to Figure 2.6 of Lenia, we can see a resemblance to the Lenia figures in Game of Life. The static formation of Figure 2.6 (c), looks similar to the circle formed in Game of Life. By looking at Figure 2.2, we can see an example of a glider in Game of Life. The glider has shown to be an important discovery in cellular automata, as we can use gliders to build computational interactions and show the complexity of cellular automaton [18]. We can intuitively see the similarity between Figure 2.2 in Game of Life and Figure 2.6 (a) and (b) in Lenia. This shows the similarity in the cellular automata across different systems.

Research on the Lenia system has shown the ability to generate a high diversity of complex patterns and self-organized patterns. These self-organized and life-like patterns have a resemblance to the 'glider' in Game of Life, and some resemblance to microscopic organisms. Research has shown that these patterns can exhibit properties of biological systems, such as localized organization in space, movement in specific directions, rotation, as well as many others.

Researchers have divided the patterns into species and different animals. This has resulted in more than 400 species and a wide range of behaviors [6]. We can see in Figure 2.6 examples of interesting creatures found in Lenia. Both (a), (b), and (d) have a cycled generation, with a set structure sliding around the screen. In Figure 2.6 (c) we can see one example of Lenia having reached equilibrium. The cells still regenerate some cells, but these variations are not big enough to make new structures. The range of options in the search space is defined by all possible phenotypes and genotypes. We use a similar description of phenotype and genotype

as described by Chan [19]. With genotype, we mean a particular combination of parameter values and a start array. Whereas phonotype corresponds to the resulting configuration of the world array that is after applying all rules, in other words, the pattern.

2.2.1 Lenia Algorithm

We have from Chan [19] the algorithm of Lenia is as follows:

1. Take a 2D array (world A) of real values between 0 and 1, initialize 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.

In formula we can write these steps as:

$$A^{t+dt} = [A^t + dtG(K * A^t)]_0^1 \tag{2.1}$$

The Lenia algorithm can generate a wide variety of structures and behaviors. However, discovering new and interesting creatures and patterns within each simulation depends heavily on chance, as the algorithm uses a randomly generated 2D array (world A).

More often than not, when only randomness is involved, instead of generating interesting individuals as seen in 2.6, we get static patterns with little movement, as seen in Figure 2.3.

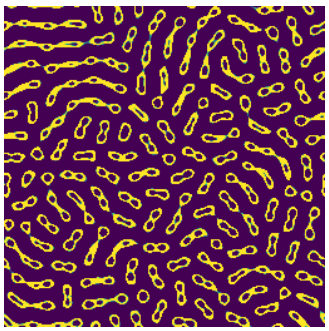


Figure 2.3: Example of a static pattern of Lenia

2.2.2 Statistical Measures

Reinke et al. [7] defined multiple statistical measures for patterns in Lenia. These are used to evaluate the patterns.

Activation mass

Activation mass, M_A measures the total sum over the activation of the Lenia pattern and normalizes it according to the size of the Lenia grid.

$$M_A = \frac{1}{L^2} \sum_x A \tag{2.2}$$

Activation volume

Activation volume, V_A, Measures the number of activated cells and normalizes it according to the size of the Lenia grid.

$$V_A = \frac{1}{L^2} |\{\forall x : A(x) > \epsilon \tag{2.3}$$

Activation density

Activation density measures how dense the activation is distributed on average over all active cells.

$$D_A = \frac{M_A}{V_A} \tag{2.4}$$

Activation mass center

$$(\bar{x}, \bar{y}) = \left(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right) \tag{2.5}$$

where

$$M_{pq} = \sum_x \sum_y x^p y^q A^t(x, y) \tag{2.6}$$

With M_{pq} is the image moment of order $(p + q)$ for $p, q \in \mathbb{N}$. The image moment is the weighted average, moment, of the pixels/ cells intensities.

Movement direction

With the center $(\bar{x}, \bar{y})$, we can define the movement direction of the activity center.

$$(m_x, m_y) = (\bar{x}, \bar{y})_t - (x_{mid}, y_{mid}) \tag{2.7}$$

with

$$(x_{mid}, y_{mid}) = \frac{L - 1}{2} \tag{2.8}$$

which defines the coordinates for the middle point of the grid.

Activation asymmetry

Activation asymmetry, A_A, measures the degree of symmetry in the distribution of activation along an axis that originates from the center of the pattern's activation mass and follows the direction of movement of this center.

$$A_A = \frac{1}{M_A}(M_A^{right} - M_A^{left}) \tag{2.9}$$

Where we define M_A^{right} and M_A^{left} splitting the pattern into two equal areas. The line separating right and left is defined by the line going through (x^{mid}, y^{mid}) of the centered pattern A, and goes through the final movement direction of the activity mass center (m_x, m_y). With M_A^{right} and M_A^{left} we can therefore compute the activity of each area, compare the activity, and compute the activation asymmetry A_A.

2.2.3 Dynamic parameters of Lenia

The dynamic parameters of Lenia control the behavior of each individual in Lenia. Therefore by changing these parameters, we change the outcome of the patterns in Lenia. We can see an example of this change in outcome in Figure 2.4, where all simulations have started with an equal start grid A. The difference in these parameters is therefore essential for the outcome. In Figure 2.4 (c), we can see a somewhat interesting pattern, but in future steps, all cells die. It is therefore crucial that the combination parameters match, and these result in a continuous living pattern.

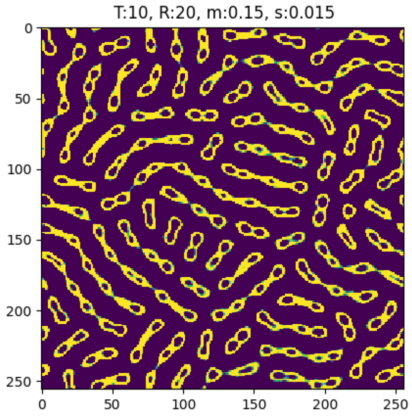
We have 4 dynamic parameters R, T, σ, μ. R controls the radius of the circle around a given cell, where the enclosed cells influence the activity in the next state of the given cell. T controls the growth update per time step and the strengthening of the update. The growth mapping is controlled by σ and μ.

2.2.4 Previous research within the field of Lenia

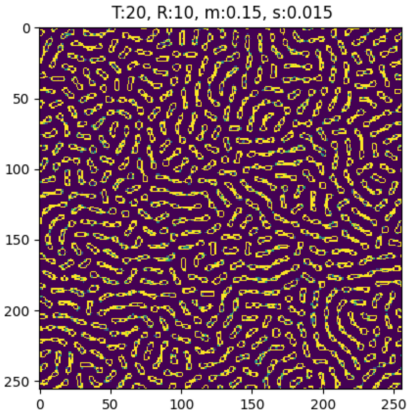
Since the release of Lenia, multiple research teams have focused on exploring the search space and testing multiple exploration and search algorithms within the space.

The study of Reinke et al. [20] discusses the problem of automated discovery of diverse self-organized patterns in such high-dimensional complex dynamical systems, and creating a framework for testing and evaluation. They used an algorithm originally developed for learning inverse models in robotics. The algorithms intrinsically motivated goal exploration and unsupervised learning of goal space representations, and resulted in high performance when simulated in Lenia. They were able to identify more diverse patterns compared to random explorations. Reinke et al. [7] also developed an extension of this algorithm, POP-IMGEP, by using a deep auto-encoder to learn a goal representation, as well as the use of a compositional pattern-producing network for generating initialization parameters. The use of this extended algorithm showed to be more efficient than baselines, and as efficient as a model that was pre-trained on a hand-made database of patterns identified by human experts.

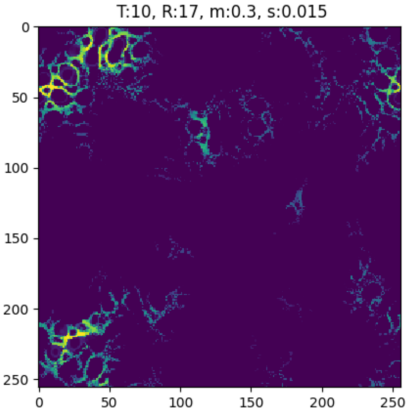
Reinke et Al. [20] evaluated each pattern after 200 time steps for all experiments, and experienced that 200 time steps were efficient for the pattern to stabilize. An un-stabilized pattern can be seen in Figure 2.5.



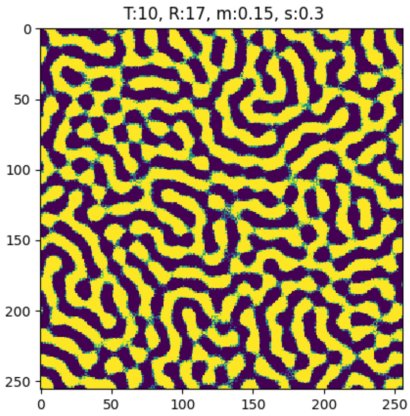
(a) T:10, R:20, m:0.15, s:0.015



(b) T:20, R:10, m:0.15, s:0.015



(c) T:10, R:17, m:0.3, s:0.015



(d) T:10, R:17, m:0.15, s:0.3

Figure 2.4: Examples of difference in dynamic parameters, all started with equal A at step 0.

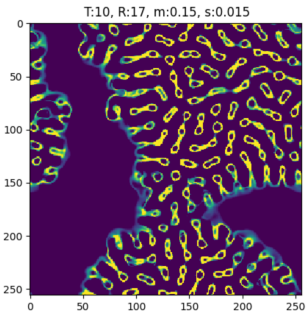


Figure 2.5: Example of a unstable pattern of Lenia

Etcheverry et al. [21] tried a similar approach to the problem with automated diversity-driven discovery of cellular automata, and also used the IMGEP algorithm, with the focus on autonomous exploration and unsupervised representation learning to describe "relevant" degrees of variations in the patterns. Here they argued that using a monolithic architecture for behavioral embedding design can lead to biased final discoveries that do not align with the needs of the end user. They also tried to address these issues, by introducing a novel dynamic and modular architecture that enables unsupervised learning of a hierarchy of diverse representations. This lead to a discovery engine that can adapt its diverse search toward the preferences of a user using only a very small amount of user feedback.

Studies involving modified Lenia

Plantec et al. [22] wanted a way to more easily find spatially localized patterns or life-like artificial creatures in Lenia. They proposed a method of adding a mass conservation constraint that they thought could facilitate the emergence of life-like artificial creatures. This resulted in an extension of Lenia, called Flow Lenia, which enables mass conservation. Flow Lenia allowed multi-species simulations, and they argued that Flow Lenia encourages more intrinsic evolution of self-organized artificial life forms within continuous cellular automata.

The studies demonstrate that Lenia, as a model of cellular automata, exhibits diversity and adaptability to tackle novel challenges, and has the capability to simulate the functionality of evolutionary and open-ended algorithms.

2.3 Evolutionary Computation

Taking inspiration from nature and biological evolution, computer scientists have used the principles of biological evolution to generate successful solutions and products. Evolutionary Computation is a problem-solving approach that imitates the natural process of evolution, creating a new set of solutions in each generation by building on the successful solutions of the previous generation.

2.3.1 Evolutionary Algorithm

Evolutionary algorithms (EAs) [23] form a subset of Evolutionary Computation, which are a type of meta-heuristic search algorithm inspired by biological evolution. The goal of EAs is to improve the population's fitness and in the last iteration have one or multiple adequate solutions. To achieve this goal, EAs have three main

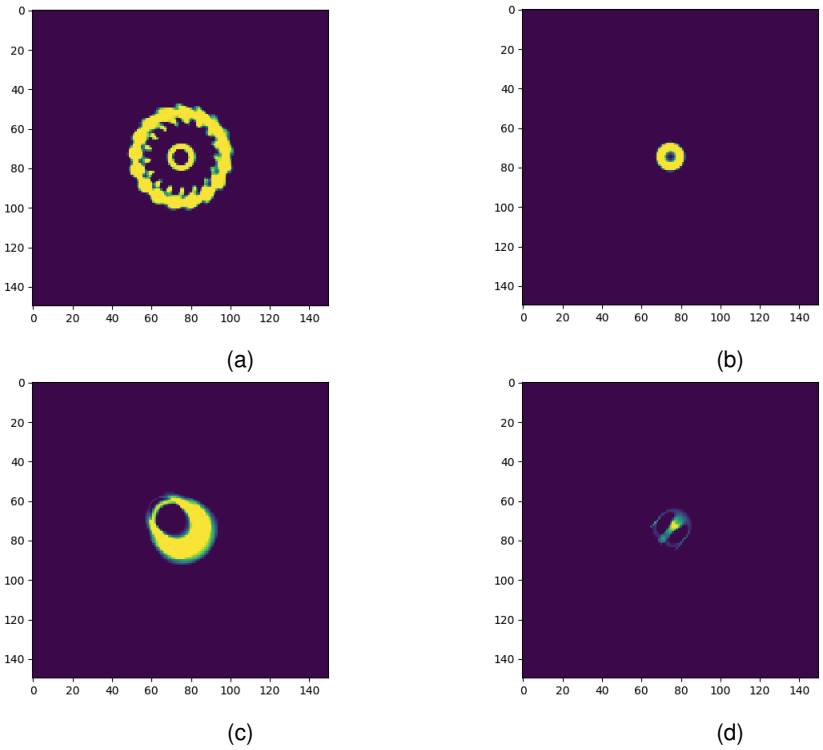


Figure 2.6: Examples of different families in Lenia.

components: replication, variation, and competition. When starting the process, a random population is used to initialize the algorithms, where each member, or individual, of the population represents a potential solution to the problem. As in nature, there are a limited number of resources and a limited number of individuals that survive for the next generation. To be able to compare the different individuals, the individuals are evaluated with the fitness function, and given a fitness score. The best-performing individuals are then selected as parents, and the combination of the two produces offspring, the replication. The offspring are created using mutation and recombination of the parents, variation. To summarize this process, the Evolutionary algorithm loop is found in Figure 2.7. This loop is run until a predefined criterion is met. A common challenge for researchers in evolutionary algorithms is to optimize the best fitness function and this result finds the best solution with the highest fitness.

2.3.2 Representations

When working with EAs, *genotype* and *phenotype* are two fundamental concepts used to describe the representation of solutions to a problem and their characteristics. Genotype refers to the digital genetic representation of an individual in EAs, such as DNA in nature. The genotype space is often represented as a string of values and is often referred to as the parameter space. The phenotype refers to the characteristics of the individual and is generated according to the genotype. The phenotype represents the solution in the environment, and it is what is optimized by the EA.

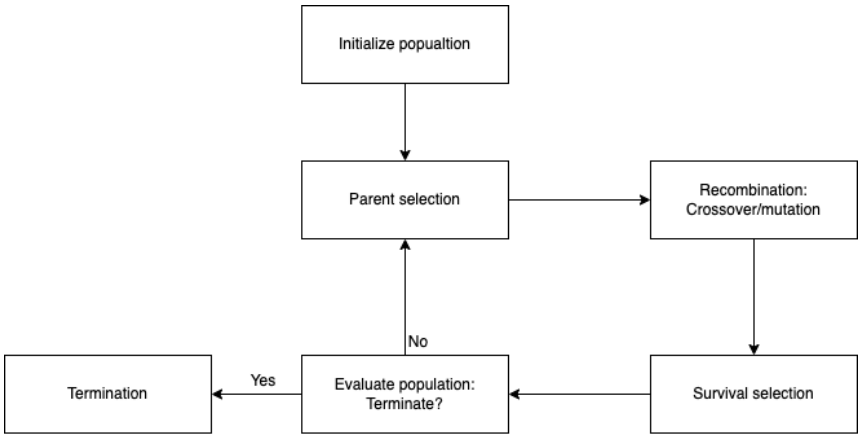


Figure 2.7: Example of a classic evolutionary algorithm loop.

2.3.3 Evolutionary Operators

Evolutionary Operators are used to simulate the mechanism of natural selection, mutation, and recombination, and are used to generate new individuals in each generation.

Selection

Selection is often divided into two categories: parent selection and survival selection. The operator selects individuals from the current population, often based on fitness.

Crossover

Crossover combines the two genotypes of two selected individuals to create new offspring. There are multiple approaches to performing crossovers, such as one-point crossover, n-point crossover, and uniform crossover [24].

Mutation

Mutation introduces small random changes to the genotypes of the individual. This helps to introduce new genetic material into the population that was not present before, which can help to explore new areas of the solution space.

The Fitness Function

Fitness is the standard measure used to evaluate individuals in EAs and is obtained by calculation with a fitness function. The fitness function measures the individual's capability and can give an indication of the mean quality of the population. As fitness indicates the quality of the individuals, it is a great advantage with a well-constructed fitness function to increase the chance of finding the best population. However, with complex tasks, designing the fitness function that matches the objective might become difficult [25]. The fitness landscape, as seen in Figure 2.8, is a metaphorical representation of how different genotypes are associated with different fitness.

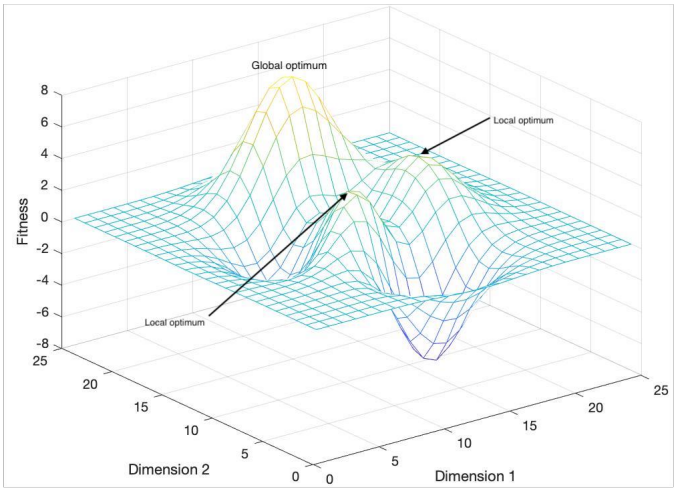


Figure 2.8: Visualisation of fitness landscape.

2.4 Autoencoder

Autoencoders (AE) are unsupervised learning models for representation learning and dimensionality reduction, and use the same original data as the target for training. Inside the neural network, the data are compressed and only relevant structure and data are kept. By using the same original data as the target for training, the neural net is trained to best reconstruct the data. Autoencoders have been used in many applications among learning features in images [26], facial recognition [27], and encoding of words [28]. Studies have also shown how autoencoders are able to generate new usable training data to be used in machine learning [27].

2.5 Variational Autoencoder

Variational Autoencoder (VAE), [29] shares its name and architecture similarities with autoencoders, however, a VAE has significant differences to an autoencoder. In contrast to autoencoders, VAE is a stochastic generative model that can give calibrated probabilities. Autoencoders on the other hand is a discriminative model that does not have a probabilistic foundation. VAE is built up by two connected neural networks, referred to as decoder and encoder. The decoder maps the input to a latent space, whereas the decoder does the opposite, mapping from the latent space to the input space to generate new data points.

Reinke et al. [20] used several deep VAE's to learn the latent representation of data, with the goal of capturing the most important features of the input data. This was used to define the goal space for the algorithm IMGEP they developed. The study showed promising results of using VAE learning the latent representation of Lenia, and used the lower dimensions of this data to define the goal space. The loss of the VAE Reinke et al. [7] started from approximately 40.000 and lowering to around 15.0000 over 2000 generations. From their results, we can see that the loss stabilized after around 1000 epochs.

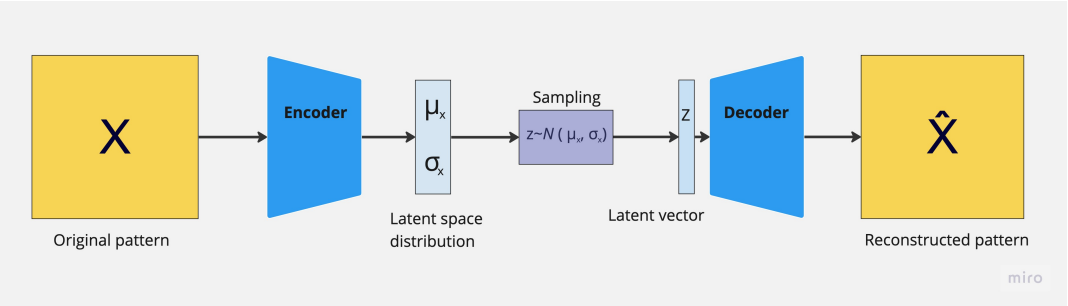
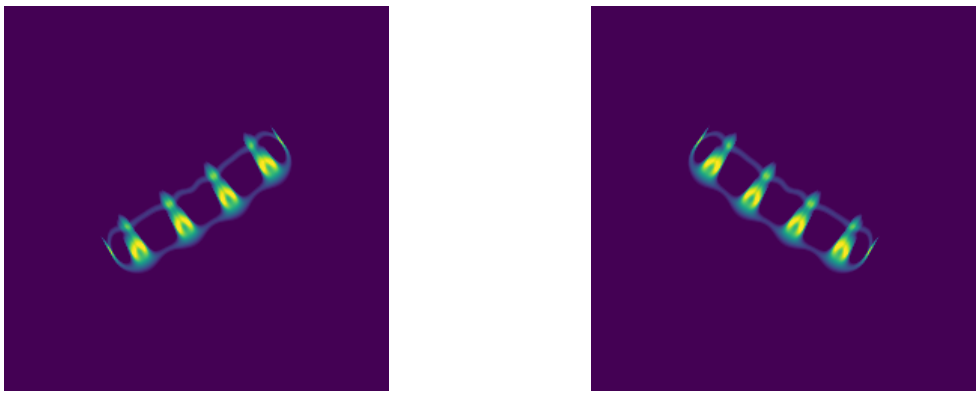


Figure 2.9: Example of a VAE

2.6 Data Augmentation

When working with deep neural networks, large data sets are often essential for getting satisfying results. However, large data sets are not always available, and to cope with this setback, data augmentation can be used. Data augmentation is the transformation of an input, x, such that $A(x)$ generates a similar but distinct different sample, thus expanding the training set [30]. Examples of data augmentation can be slight photometric deformations of a sample or random crop of a sample. As a consequence of data augmentation, the model often becomes more resistant to new unseen input. For example, when cropping or flipping an image input, as seen in Figure 2.10, the model is less dependent on the position of the object



(a) Original input

(b) Augmented input

Figure 2.10: Example of data augmentation with flipping along the horizontal axis.

2.7 Quality Diversity

An intriguing characteristic of biological evolution is its capability to generate a wide range of organisms that excel in their respective environments. In contrast, artificial evolution algorithm research has the last years often only focused on pure optimization, with the goal of only finding one superior soltuion. The process of evolution in nature is not as straightforward as pure optimization. It involves a more intricate process known as divergent search, wherein local optimization occurs within each niche while diversification happens simultaneously. This characteristic of being able to discover both *quality* and *diversity* during optimization, differs from many

traditional EAs and has become a new set of algorithms within machine learning, called Quality-Diversity Optimization. *Quality-Diversity* (QD) optimization is a problem-solving method that aims to generate a diverse set of high-quality solutions for a given task. This approach was first proposed in 2011 by Lehman & Stanley[31], with the introduction of the Novelty search. Lehman & Stanley argued that instead of pushing the population to seek an objective, it would be more effective to motivate the population towards new novel behaviors and abandon the objective. Lehman & Stanley also argued how this approach to optimizing a problem, shows parallels towards biological evolution, as biological evolution does not strive to produce any definitive species. Novelty search with local competition was developed (NSLC) by Lehman & Stanley[32], as a way of combining performance-driven search and novelty search. They argued that when rewarding individuals outperforming those most similar in morphology, resulted in more functional morphological diversity. Inspired by NSLC, Mouret & Clune developed Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) [33].

Studies involving QD optimization have shown success with using this optimization technique. Kim et al. [34] did a study involving a robot manipulating an object, and how to learn a wide range of behavior within the interaction with the object. They proposed and had success with a method to autonomously generate a diverse repertoire of simple object interaction behaviors in simulation.

Flageat et al. [35] and Salehi & Doncieux [36] suggested in 2022 benchmarking QD algorithms on neuroevolution for reinforcement learning, to compare and improve findings when working with QD algorithms. Flageat et al. [35] focused on QD benchmark within robot control, whereas Salehi & Doncieux [36] studied towards a more general benchmark within QD optimization.

2.7.1 Novelty Search

Earlier work on diversity in evolution has focused on diversity in the genetic search space, with the goal of finding a way around the local optima, with the idea of when finding a local optimum, try to explore away from the optimum and this may be enough to uncover the global optima. Lehman et al. [31] proposed a method of abandoning the objection function, and instead of explicitly seeking an objective or modeling natural evolution, to rather search for behavioral novelty. They argued that searching for novelty would lead to an increase in the complexity of solutions.

As we strive to replicate nature in evolutionary algorithms, we want to avoid converging to a single morphology and instead get various behaviors. One approach to avoid the local maximum is to use the method developed by Lehman and Stanley called Novelty Search [37]. In contrast to methods pushing the population toward one objective, we push the population toward behavioral diversity and rewards divergence. With this approach, we have the possibility to discover new unique solutions and discover a wide range of solutions.

One challenge with this approach is the goal of not only getting a wide range of solutions but also getting solutions with high quality.

Earlier studies with the use of novelty search have shown good results, as in Lehman [32]. The experiment showed that the use of novelty search with local competition used on locomoting virtual creatures resulted in more functional morphological diversity. They also discovered that the more functional creatures were found within a single run, rather than the model used with global competition.

2.7.2 MAP-Elites

Traditionally goal of search algorithms has been to return a single solution in the search space, with the highest performing score. Inspired by NSLC, Mouret & Clune developed Multi-dimensional Archive of Phenotypic Elites, MAP-Elites [33]. As an inspiration and contradiction to traditional search algorithms, MAP - Elites produce a holistic view of how high-performing solutions are spread through the search space [33]. The use of MAP-Elites gives a useful insight into how elements of solutions influence behavior.

MAP- Elites discretize the behavior space into cells, and attempt to find the highest-performing solution within each cell. With this implementation, MAP-Elites generate a highly diverse set of solutions, and with the highest-performing solution within each cell, we also get a quality-diverse population.

MAP-Elites have shown a wide range of cases, from generating diverse behavior of walking gaits on robots [38] and mixed-initiative game content generation [39], to exploring genetic programming systems [40]. The wide range of studies shows the large specter of usage of MAP-Elites and its property of getting diverse solutions and behaviors.

2.7.3 Centroidal Voronoi Tessellations

Centroidal Voronoi Tessellation (CVT) is a geometric partitioning of space into non-overlapping regions, where each region is defined as the set of points closest to a given generator point. In a CVT, the generated points are not placed arbitrarily but instead are iteratively moved to the center of mass, its centroid, of their respective Voronoi cells until a stable configuration is reached.

A Voronoi diagram is therefore a way to divide a space into regions based on the distance to points.

When dividing the space into CVT regions, there are multiple algorithms for executing this. One of these is Lloyds algorithms [41], and is widely used to create Voronoi Tessellations for 2D spaces. One disadvantage when using Lloyds algorithms in higher spaces than 2D, higher spaces involve more complex algorithms [42]. The study of Vassiliades et al.[43] therefore suggests using the Monte Carlo method to compute a close approximation to a CVT. The algorithms suggested work by first initializing k random centroids and generating K random points, with K always bigger than k. Uniformly distribute the K random points in feature space according to the bounds of each dimension. The algorithm then alternates between updating each centroid to be the mean of its corresponding point and updating each point to the closest centroid.

We get algorithm 1 as shown in the study of Vassiliades et al. [43] :

2.7.4 CVT-MAP-Elites

Centroidal Voronoi Tessellations MAP-Elites (CVT-MAP-Elites) is an extension of MAP-Elites, by introducing a constant pre-defined number of regions regardless of the dimensionality of the feature space[43].

As the number of regions in MAP-Elites increases exponentially with the number of spaces, MAP-Elites struggle to scale the input to high-dimensional feature space. With high dimensions, MAP-Elites will grow exponentially with the number of added dimensions, creating computational problems. CVT-MAP-Elites however, introduce

Algorithm 1 Algorithm over CVT (from [43])

```
1: procedure CVT(k)
2:     C ← sample_points(k)
3:     S ← sample_points(K)
4:     for i = 0 → max_iter do
5:         I ← get_closest_centroid_indicies(S, C)
6:         C ← update_centroid(I)
7:     end for
8:     return centroid C
9: end procedure
```

the partition of high-dimensional space into evenly distributed geometric regions. With the help of Centroidal Voronoi Tessellations, the feature space is divided into a desired number of regions, and individuals are then placed into its closest region. If the region is already occupied, the individual with the highest fitness score is chosen. We can see in Figure 2.8 a visualization of the difference between MAP-Elites and CVT-MAP-Elites. In algorithm 2 we can see an overview of the CVT-MAP-Elites algorithm proposed in [43].

To address the computational demands, has CVT-MAP-Elite tried to handle, by using CVT [44], to partition the feature space into n homogeneous geometric regions, where n is the predefined number of regions. This results in individuals placed in the most relevant region, replacing a less fit one if the region is already occupied, and after n generations we have a wide range of behaviors. Results have shown that CVT-MAP-Elites has more precise control over the balance between diversity and performance than MAP- Elites[43].

Algorithm 2 Algorithm over CVT-MAP-Elites (from [43])

```
1: procedure CVT-MAP-Elites(k)
2:     C ← CVT(k)
3:     (X, P) ← create_empty_archive(k)
4:     for i = 0 → G do
5:         x ← random_solution()
6:         add_to_archive(x, X, P)
7:     end for
8:     for i = 0 → I do
9:         x ← selection(X)
10:        x' ← variation(x)
11:        add_to_archive(x', X, P)
12:    end for
13:    return archive (X, P)
14: end procedure
15: procedure ADD_TO_ARCHIVE(x, X, P)
16:    (p, b) ← evaluate(x)
17:    c ← get_index_of_closest_centriod(b, C)
18:    if P(c) = null or P(c) ≤ p then
19:        P(c) → p, X(c) → x
20:    end if
21: end procedure
```

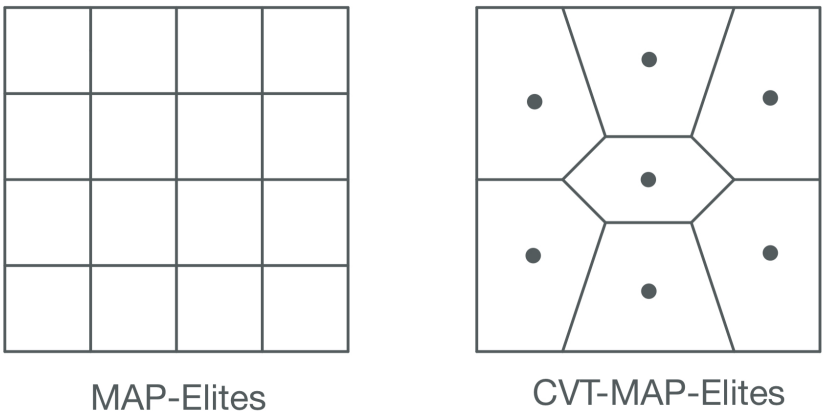


Figure 2.11: Visualisation of MAP-Elites vs CVT-MAP-Elites.

2.7.5 Behaviour Descriptors

As we attempt to understand and explore increasingly complex issues in the world, we frequently encounter complicated data when attempting to describe the problem. Moreover, there may be situations where we are uncertain about what is actually occurring. In these situations, the descriptors of the behaviors of the data become highly complex. In the search for diverse behaviors, it is necessary to understand and determine the characteristics of behavior. This is called *Behaviour Descriptors*, BD, which involves defining which results exhibit interesting behaviors and which do not.

Encouraging diversity of robot behaviors, as opposed to genotypes, has been a topic of research in the field of evolutionary robotics. With this, the question of how to compare different behavior has also been studied. The study of Mouret & Doncieux [45] raised this question, and review the main published approaches. They classified the different behavior characterizations into classes of multi-objective methods and generic, Hamming-based, behavioral distance. The experiments reviewed showed that encouraging diversity in behavior significantly improves the evolutionary process, regardless of genotype or task. Multi-objective methods were found to be the most effective among the benchmarked approaches. Additionally, the generic Hamming-based behavioral distance was shown to be at least as effective as task-specific behavioral metrics.

The classification of cellular automata has been approached in different ways. With 1D cellular automata, Wolfram presented a way of classification with subjective criteria, with the inversion to caption the complexity universality [46]. The suggestion was that all 1D cellular automata could be categorized into four different classes, with classification focusing on behavior. The same approach was also used on 2D cellular automata [47], with similar results. However, Eppstein later criticized Wolfram's classification with the complaint that the classification wrongfully defines the interesting rules [18].

2.7.6 Autonomous Skill Discovery with Quality-Diversity and Unsupervised Descriptors

The goal of QD - optimization is to have a repository of high-quality solutions. This approach often results in a high diversity of behaviors, however, QD -

optimization requires defining the behavior descriptors, BD. When working with complex problems, defining BD often requires a high level of expertise and prior knowledge. Autonomous skill discovery with Quality-Diversity and Unsupervised Descriptors (AURORA), is a QD algorithm introduced by A. Cully in 2019 [8]. AURORA was created as a proposal to solve the problem of having to manually define complex BD. Defining the correct BD is crucial when using novelty search, as it is important to define what separates two different solutions. To avoid manually defining BD, AURORA instead attempts to find BD using a Dimensionality Deduction, DR, algorithm. The utilization of a dimensionality reduction algorithm reduces the need for a high level of expertise, thereby minimizing the probability of human error. Instead, the latent representation generated by the dimensionality reduction algorithms acts as the BD. The procedure of Lenia can be categorized into three distinct parts: initialization, QD iteration, and the update of the DR algorithm. The subsequent subsections provide a detailed explanation of the algorithm's steps, followed by a summary.

Initialization procedure

AURORA starts with an initialization procedure. This procedure initializes the QD algorithm that contains a container consisting of all individuals. The initial data set is formed by randomly generated data. The DR algorithm is then trained on the first data set, to learn the latent and low dimensional representation of the data. The low-dimensional representation of the data is then used as BD.

QD iteration and update of DR algorithm

After the initialization procedure, the QD iteration takes place. In the QD- iteration, an individual is randomly selected, then mutated, and evaluated. After evaluation, the individual is placed into the collection, either as a new individual or as an improvement of an existing one. During the evaluation of the new mutated individual, the DR algorithms determine the corresponding BD. This is to determine where in the container the new individual should be placed. The QD iteration runs for a predefined number of rounds. During this procedure, the collection is gradually increased. Once the QD-iteration has completed the predetermined number of iterations, the DR algorithm is retrained. This improves the DR algorithm and allows us to further refine the latent representation. Individuals are often assigned new BD during this step of the algorithm. We now create a new container and attempt to place all individuals into the new container. To ensure that we do not store multiple individuals that have a very similar BD, we employ local competition. This ensures that only the best individual from an area is kept. This often causes the size of the collection to be reduced, since similar individuals are discarded.

AURORA Procedure

Initialization procedure of AURORA:

1. Random initialization of the QD algorithm

2. Sensory data of the randomly generated individuals are collected to form the first data set

3. Dimensionality Reduction, DR, is then trained on this first data set to learn a latent and low dimensional representation of the data.

4. Project the data into latent space

QD iteration:

1. An individual is randomly selected from the collection

2. Mutation of individual

3. The resulting new individual is evaluated

4. Attempted to be placed in the collection

Update of the DR algorithm:

1. Gather all individuals in the collection

2. Retrain the DR- algorithm on all individuals and collect new BD.

3. Attempt to store all individuals in the collection.

Chapter 3

Implementation

To enable exploration within Lenia we have implemented a system using a quality diversity algorithm with automatically defined behavior descriptors, called AML. The first part of this chapter describes AML as a whole, with the setup and implementation of AML. In the second part of this chapter, there is an in-depth exploration of the various components of AML, including detailed explanations of the variational autoencoder, Lenia, the evolutionary algorithm, and adapted CVT-MAP-Elites. The experiment code is open source, available on the GitHub repository [48].

3.1 AURORA Modified for Exploring Lenia

The AURORA Modified for exploring Lenia, AML, is a cyclic system, based on the AURORA method created by Cully[8]. Several distinctions exist between the original AURORA approach and our modified version. The original AURORA was created to enable robots to autonomously explore their capabilities by engaging with their surroundings. However, in our method, we are exploring Lenia's behavior space. Further, we use an adapted version of CTV-MAP-Elites as our container of individuals, which we refer to as the archive. Additionally, we incorporate Variational Autoencoder, VAE, as a method for dimensional reduction, DR.

Cully originally considered Principal component analysis (PCA) and AE, but as Reinke et al. [7] had promising results with their VAE, we decided to use this instead. The main advantage of using a VAE over AE, is the problem of the unregulated latent space in AE. This is resolved by VAE and makes us able to generate data from randomly sampled vectors from the latent space. Further, we did choose to not use Principal Component Analysis, as PCA is a linear transformation of data, and with a data size of $(256 * 256)$, this this would have been to computationally heavy for PCA. By the experiments of Cully [8], the results implied that PCA scored as well as AE, but these experiments were with a lower number of dimensions to reduce.

Given the central role of Lenia in the AML, it is necessary to repeat the definitions of genotype and phenotype as outlined in Section 2.2. The genotype in Lenia refers to the specific combination of parameters, which includes the starting grid A and the kernel parameters. The phenotype, on the other hand, denotes the pattern that emerges after 200 iterations. When we mention an individual, we are referring to a particular genotype along with its corresponding phenotype. An overview of AML can be found in Algorithm 3. We describe each procedure of AML in the following subsections.

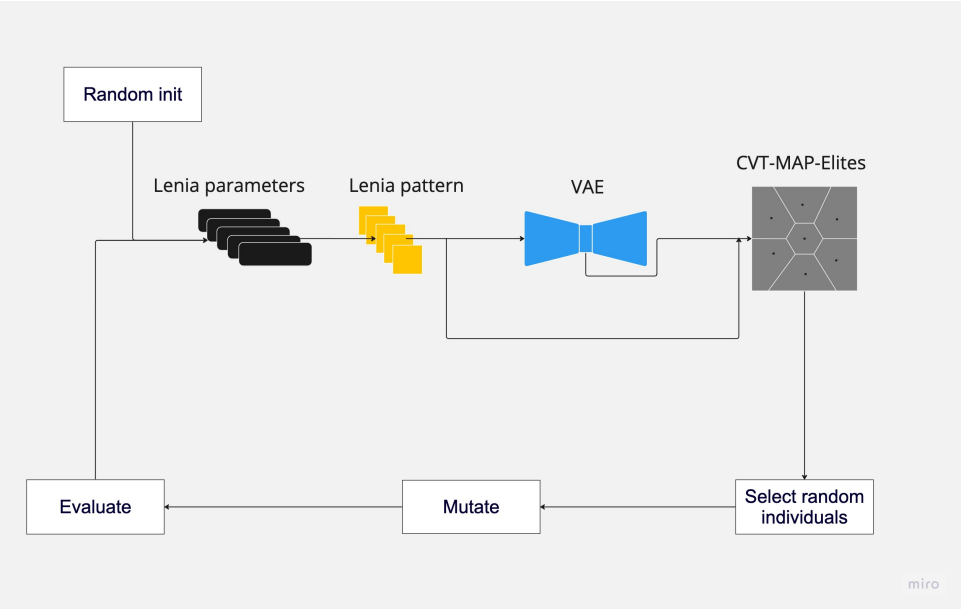


Figure 3.1: Visualisation of the AML, while in the exploration phase.

3.1.1 PRE-Init-Phase

Before we can start the cyclic phase of AML, we need to initialize the VAE. To achieve higher results, we pre-train the VAE. This is more deeply described in Section 3.2.

3.1.2 Initialization Procedure

The initialization procedure begins with the initialization of the archive, that is the adapted CVT-MAP-Elites. A more detailed explanation of the adapted CVT-MAP-Elites approach will be provided in Section 3.4. The population is initialized, with 100 individuals. All individuals have randomly generated grid A. At iteration 0, none of the individuals are mutated by the EA, and have all equal parameters of $R : 17, T : 10, \mu : 0.15, \sigma : 0.015$. To initialize the first generation into the archive, the population is sent through the VAE, to have their BD generated, and added to the archive.

3.1.3 Exploration Phase

The exploration phase is started by selecting 100 individuals from the archive, mutating the individuals, and evaluating their new fitness. In order to obtain their updated BD, they are passed through the encoder of the VAE, and we obtain the descriptors. The descriptors we get encoder are used to train the VAE. To train the VAE, the descriptors are sent through the decoder, and we obtain a reconstructed pattern. We construct a loss based on the original pattern and the reconstructed pattern, and use backpropagation to train the model.

All individuals are then attempted to be placed in the archive, based on their BD. This phase goes on for 800 generations, with the exception of a generation of BD updating, see Section 3.1.6 for more in-depth details.

Algorithm 3 AURORA Modified for exploring Lenia

procedure PRE-INITIALIZATION-PHASE
    Initialize the VAE
    Pre-train with already discovered augmented patterns
    Pre-train with random exploration
end procedure
procedure INITIALIZATION PROCEDURE
    Initialization adapted CVT-MAP-Elites archive
    Initialization of start population: 100 individuals
    Initialization epoch to add the population to the archive.
end procedure
procedure EXPLORATION PHASE
    Randomly select 100 individuals from the archive to make the new population
    Mutate population
    Evaluate population to get new fitness
    Send the population through VAE to get map placements/behavior descriptors
    Attempt adding all individuals to the archive
end procedure
procedure UPDATING THE BD
    Make a new empty archive
    Freeze the VAE
    Gather all individuals from the archive, send all through the VAE
    Try to add population to the archive
    Unfreeze the VAE
    Discharged the old archive
end procedure

Selection

For each generation, 100 individuals are randomly selected from the archive. We decided to use random selection, after testing by selecting the individuals with the highest fit. However, as one of the two fitness functions utilized is based on randomly generated fitness, as described in Section 3.1.5, selecting individuals based on fitness excludes a significant number of potentially valuable individuals. Additionally, we observed that when utilizing the fitness function based on movement, as described in Section 3.1.5, selecting the individuals with the highest fit resulted in a low number of individuals in our archive in the last generation. We therefore concluded with the use of random selection, to ensure the possibility of a wide range of behaviors.

Mutation

Lenia possesses a large and complex solution space that contains a high number of parameters. Moreover, numerous parameters yield similar patterns, thereby making it challenging to generate new individuals in the pursuit of diverse behaviors. The complexity arises due to the sheer number of parameters, and identifying the specific combination that leads to success proves to be a unique and arduous task. Nonetheless, in the original paper on Lenia by Chan [6], the author discusses a method of using discovered animals and manually mutating them to uncover new

animals. This implies that although locating the animals is difficult, there are happenings where they reside in close proximity within the feature space. The observation suggests that mutation, that is modification mutation, can serve as a means to discover new patterns. To take larger steps in the feature steps we have opted for replacement mutation, with the aim of not being trapped in a local optima. Therefore, we have chosen to employ both replacement and modification mutation in our approach to create and discover new patterns.

The offspring are thereby mutated using two types of mutation: replacement and modification. In replacement mutation, this involves randomly modifying the parameters of the kernel used to create the patterns. In this type of mutation, the changeable parameters of the kernel are completely replaced with brand new sampled random values. This can lead to a drastic change in the appearance of the pattern, as the new parameters may result in a significantly different output than the original parameters. We can see the mutation rate for each parameter in Table 3.4. In replacement mutation, the grid A is kept equal.

In modification mutation, the grid A has the possibility of having noise added. The mutation rate for A getting noise added is 0.1. The noise is created with a Gaussian distribution, with a mean of 0 and a standard deviation of 0.25. To ensure that $A \in [0, 1]$ any values above 1 are clipped to 1, and any values below 0 are clipped to 0.

3.1.4 Evaluation

Individuals are evaluated by examining the pattern produced after 200 iterations in Lenia, with the fitness score calculated using one of two fitness functions: movement-based or randomness-based. During the testing with fitness function, several dead patterns were observed, and to enhance the overall results, all such dead patterns were assigned a fitness score of -100. The following two subsections describe the two fitness functions.

3.1.5 Fitness Function Based on Movement

One of the primary obstacles we encountered during the designing of AML was determining the appropriate fitness function to use. Our goal was to employ a fitness function that would guide the search toward interesting patterns, potentially including animals. However, finding the right fitness function proved to be quite challenging. This was mainly due to the extensive array of pattern behaviors, and our reluctance to discount behaviors. Another challenge is that the neighbor grid is connected and formed as a ball, and patterns flow around it. Consequently, moving animals can have half the pattern on each side of the grid. To address these challenges, we opted to utilize the activation mass center as the core concept behind the movement, as they could indicate the animal's position within the image. However, this approach had its drawbacks. As an example, when a pattern is divided, with half of the active cells on each side of the grid, the image moment would be located at the center of the pattern. Despite this being an outlier, we decided not to give attention to this flaw. Instead, we focused on determining the length of movement by measuring the distance between the image moment at time steps 0, 50, and 100. This interval of time steps was decided after experimenting with both longer and shorter intervals, where 50 showed satisfying results. We could not assign a linear fitness score based solely on movement length, as it would disproportionately

favor large stochastic movements. We did not want small, steady movements to be a disadvantage, nor did we want circular motions, exhibited by already discovered animals, to be disadvantaged.

Therefore, we assigned patterns with consistent movement lengths across multiple time frames a high score. We also aimed to encourage significant movement lengths, as we wanted to store them in the archive and potentially mutate them to create improved animals. While small movements received a positive score, it was lower than that given to larger movements. Given the inherent difficulty in devising a clear fitness function due to the multitude of behavioral possibilities, we decided to introduce randomness at different intervals rather than enforcing a strict linear scoring approach.

In summary, the fitness function calculation involves using the mass center, that is image moment, as expressed in Equation 3.1.

$$(\bar{x}, \bar{y}) = \left(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}}\right) \tag{3.1}$$

The activation mass center, which represents the 'heaviest' point in the pattern, can be seen with a red dot in Figure 3.2a. By tracking the activation mass center at time step 1 and then again after 100 time-steps, we can determine the movement of the pattern, as demonstrated by the line in Figure 3.2b and Figure 3.2c.

Interestingly, we have observed that animals that are already discovered exhibit a consistent length of movement line. Building on this observation, we attempt to leverage it in our experiment by assigning a fitness score of 70 to patterns with a nearly constant length of movement line.

In order to emphasize movement, patterns with a movement line close to zero are assigned a fitness score of zero. Patterns with some degree of movement but lacking constancy, are assigned a randomly generated fitness score. The fitness function is summarized in Table 3.1.

Condition	**Fitness Score**
Dead	-100
a small movement close to zero	random(0, 25)
constant movement over multiple timeframes	90
big movement, but not constant	random(0,100)

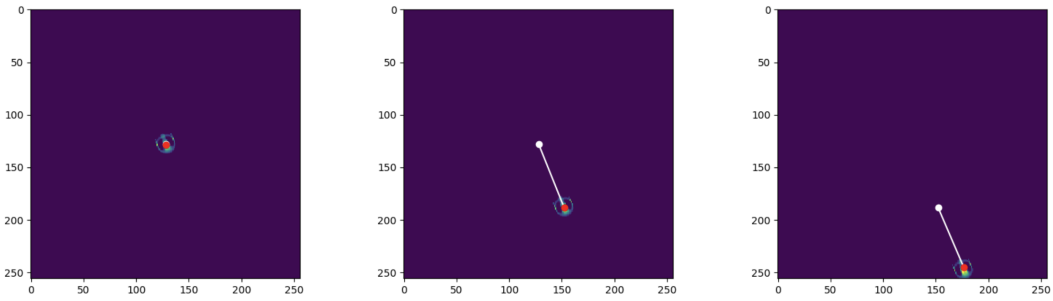
Table 3.1: Overview of the fitness function

Fitness Function Based on Random Score

The fitness function in this experiment is simple: if the pattern is not dead, get a random fitness between 0 and 100. The pattern is categorized as dead if all cells are 0 or 1, or if all cells contain the same value. If the pattern were dead, the individual would get a fitness of -100.

3.1.6 Updating the BD

Every n-th generation every individual in the population is reevaluated through the VAE to get new behavioral features. All individuals are then attempted to be added



(a) Glider at time step 0 (b) Glider at time step 100 (c) Glider at time step 200

Figure 3.2: Glider moving over three time steps

to the new archive, with their updated BD. The old archive is then discharged. For all experiments, we update the archive and all BD at generation: $(0, 50, 150, 350, 500)$. While doing the update we freeze the VAE. In Section 3.2.3, we will provide a more thorough explanation of the process of freezing the AVE. After the update of all BD, the VAE is unfrozen.

3.2 Variational Auto Encoder

The encoder gets Lenia patterns as inputs and used convolutional layers to compress the input to lower dimensions. The output of the encoder for each latent variable z_i the mean μ_i and log-variance $log(\sigma_i^2)$. As input for the decoder, the decoder samples for each latent variable $z_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$. The output of the decoder is the reconstructed pattern. As described in the background, Reinke et. al. [20] have promising results using VAE to learn lower representations of their input data. We thus decided to use the same VAE architecture for the VAE used in our study. The original code for the VAE can be found in the GitHub repository at [49], in the file at [50]. In Table 3.2, the overview of the encoder is presented. Further in Table 3.3, the overview of the decoder is presented.

Encoder	
Input pattern A	256 x 256 x 1
Conv layer	32 kernels 4x4, stride 2, 1-padding + ReLU
Conv layer	32 kernels 4x4, stride 2, 1-padding + ReLU
Conv layer	32 kernels 4x4, stride 2, 1-padding + ReLU
Conv layer	32 kernels 4x4, stride 2, 1-padding + ReLU
FC layers	256 + ReLU, FC: 2x8

Table 3.2: Table over encoder of VAE

Loss function

The loss function for a batch that has been used is

$$Loss(x, \hat{x}, \mu, \sigma) = -a, +\beta \left(\sum_{i=1}^{d} b_i + \gamma Var([b_1, ..., b_d]) \right) \tag{3.2}$$

Decoder	
Input latent vector z	8 x 1
FC layers	256 + ReLU, 16x16x32 + ReLU
TransposeConv layer	32 kernels 4x4, stride 2, 1-padding + ReLU
TransposeConv layer	32 kernels 4x4, stride 2, 1-padding + ReLU
TransposeConv layer	32 kernels 4x4, stride 2, 1-padding + ReLU
TransposeConv layer	32 kernels 4x4, stride 2, 1-padding

Table 3.3: Table over decoder of VAE

where x are the input patterns, $\hat{x}$ are the reconstructed patterns, μ, σ are the outputs of the decoder network, and d is the number of latent dimensions.

The reconstruction accuracy a of the loss is given by a binary cross entropy with logits:

$$a = \frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{L^2}(x_{j,n} * log\sigma(\hat{x}_{j,n}) + (1 + x_{j,n}) * log(1 - \sigma(\hat{x}_{j,n}))) \tag{3.3}$$

where index j is for the single cells of the pattern and n for the data point in the current batch, N is the batch size, and $\sigma(x) = \frac{1}{1+e^{-x}}$. The KL divergence terms b_i are given by:

$$b_i = \frac{1}{2 * N}\sum_{n=1}^{N}\left(\sigma_{i,n}^2 + \mu_{i,n}^2 - log(\sigma_{i,n}^2) - 1\right) \tag{3.4}$$

Optimizer

As an optimizer, we used the Adam optimizer with learning rate $lr = 1e - 3$ and betas $\beta_1 = 0.9, \beta_2 = 0.999$, $eps = 1e - 8$, weight decay of $1e - 5$.

3.2.1 Pre-trianing of VAE and Data Augmentation

At the beginning of our training, the VAE starts with random weights in both the decoder and encoder. In the first epochs, the model is not trained, making it very bad at capturing behaviors. Reinke et al. [20] experimented with different variants of VAE, where one of the VAEs was trained before the exploration started. This showed promising results, and we also wanted to exploit this. The other VAE Reinke et al. [20] experimented with was online learned goal space VAE. This VAE is trained on Lenia patterns discovered during exploration, in a total of 2000 epochs. This version was somewhat better than the pre-trained VAE.

To exploit both of the findings of Reinke et al., we combined their approaches. We pre-trained the VAE with the already found pattern discovered by Chan et al. [51], as well as training the VAE on Lenia patterns discovered during exploration, without adding these patterns into our archive. During our experimentation with VAE and incorporating early found patterns into our archive, we frequently encountered outcomes where AML interpret very distinct patterns as very similar. This resulted in a full archive, stuck in a local optima. To address this issue, we decided to not incorporate the early-found patterns and considered this phase as purely VAE training.

Pre-training with already discovered pattern

Chan et al. have discovered several animals in Lenia, where 287 of these can be found in a GitHub repository at[51]. Since the number of animals already found was limited, it was difficult to pre-train the VAE using only these. We made the decision to utilize data augmentation in order to train the VAE, and attain higher results when making BD. We use four different setups to augment the input, resulting in four times the number of inputs to pre-train the weights, $287 * 4 = 1148$.

- Center pattern
- Center and transpose pattern
- Shift the pattern to the upper left
- Transposed and upper right

While we aim to leverage the benefits of pre-existing patterns, we also strive for our AML to autonomously identify all patterns. Therefore, we do not incorporate the discovered images into our dataset. To see if there are any advantages of pre-training the map with these already-found animals, we have experiments without the pre-training.

Pre-train with patterns discovered during exploration

After the VAE is trained on found patterns, the VAE is trained on patterns found during exploration. The VAE is trained for 1000 epochs with a population of 100 individuals. Further, the individuals discovered during this phase are not added to the archive.

3.2.2 Training the VAE

Training of the model happens during pre-train and during the exploration phase.

3.2.3 Freezing the VAE

During updating of the BDs, we stop the weights of the VAE from updating, thereby freezing the model. When the model is frozen, its existing weights are preserved and kept constant. During this time, all individuals from the archive are sent through the VAE, to get updated descriptors. Freezing the model reduces computational time and ensures consistent behavior during the updating phase. However, a drawback is that the fixed weights may not be optimal for new data, limiting the model's performance. Therefore, we avoid freezing the model during the exploration phase.

3.2.4 Centering of Patterns

When sending in patterns into the VAE, there are often occurrences where the VAE misplaces the patterns. The patterns in Lenia are moving through the grid at different time steps, see Figure 3.3, where we can see a glider at three different time steps. When the VAE receives these images, the VAE gives different behavioral descriptors, as it interprets the three patterns as different images. However, this is not ideal, and results in the archive full of the same pattern, but at a different time step. To prevent this from happening, we center all patterns with the help of the Activation

Mass Center Equation 2.5. Our goal is to have the mass center in the center of the image, that is at point $(128, 128)$. As all patterns are of equal size, $256 * 256$, when we have calculated the Activation Mass Center, we know in which quadrants of the image the mass center is in. We kneed to know which quadrant the mass center is, since this influence how we center the pattern. We have the advantage that the Lenia grid is connected as a ball, with north and south being connected, as well as east and west. It is therefore possible for us to split our patterns into parts, and put them back together.

We can see an example of how we split our pattern in Figure 3.4. We can see the original pattern in the solid square, with the mass center in the 4th quadrant. We can then split our pattern into four parts: A, B, C, and mass center. To calculate the size of our boxes, we take $mass\_center - 128$. We slice our pattern according to the right dimensions and reconstruct it according to our Figure 3.4.

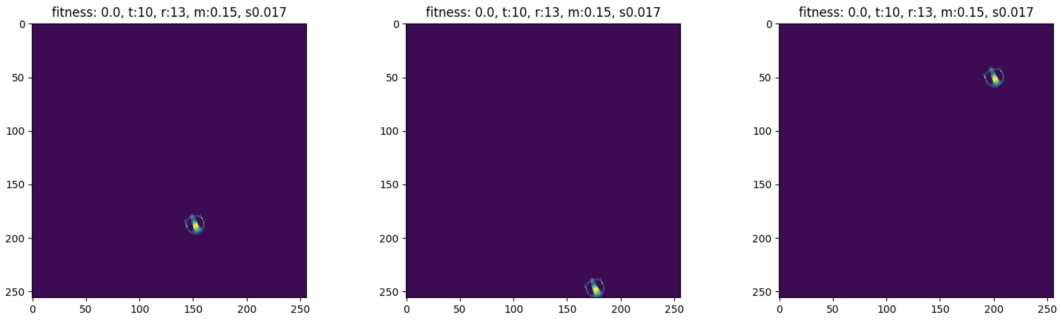


Figure 3.3: Example of the same animal, at different time step

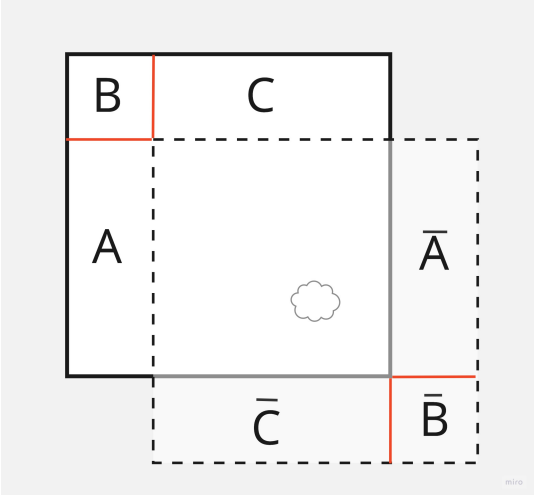


Figure 3.4: Centering of pattern, with mass center in 4th quantile.

3.3 Lenia

All experiments described were conducted using Lenia as a target system. The code used to implement Lenia was gathered from B. Chan[51] and Mordvintsev & VanderPlas[52] Lenia is also described in the Background, within Section 2.2. The

Lenia pattern is initialized as uniform random noise. As each iteration progresses, the pattern is convolved with a kernel, until a discernible pattern materializes.

3.3.1 Definitions

Lenia is a cellular automata, and by mathematical definition, we can define CA by five aspects: $\mathcal{A} = (\mathcal{L}, \mathcal{T}, \mathcal{S}, \mathcal{N}, \phi)$ [6]. Where $\mathcal{L}$ is the d dimensional grid, $\mathcal{T}$ is the timeline, $\mathcal{S}$ is the state set, $\mathcal{N}$ is the neighborhood of the origin, where $\mathcal{N} \subset \mathcal{L}$. ϕ is the local rule, where $\phi : S^{\mathcal{N}} \to \mathcal{S}$.

To translate this more directly into Lenia, we can define this into:

- A^t: The pattern at time step t, where $A^t : \mathcal{L} \to \mathcal{S}$. (The archive of states over the whole grid).
- $A^t(x)$: The state of a given cell x in the grid, where $x \in \mathcal{L}$.
- $A^t(\mathcal{N}_x) = A^t(n) : n \in \mathcal{N}_x$. Says what is the state in a given neighborhood in the grid.
- ϕ : The local rule: convolve the grid with kernel K.

We have our initial pattern A^0, and updated pattern A^{t+1}, according to the local rule ϕ over the hole grid, for each time-step Δt. This leads to the time evolution:

$$\phi^N(A^t) = A^{t+N\Delta t}$$

3.3.2 Lenia Implementation

As written above, Lenia consists of a two-dimensional grid of cells A^t, that construct our pattern. Every cell are in a state $A(x) \in [0, 1]$. We therefore get $A \in [0, 1]^{LxL}$, with $L = 256$ for all experiments. To further extend the space of Lenia, the neighborhood is extended to a discrete ball. In other words, the borders of the grid are connected, with north and south borders connected, as well as the east and west borders of the grid.

The local rule

To calculate the new state of all cells, we convolve the grid with a kernel $K : \mathcal{N} \to \mathcal{S}$ to find our potential distribution of a cell, $U^t(x)$, Equation 3.5.

$$U^t(x) = K * A^t(x) = \sum_{n \in \mathcal{N}(x)} K(N) A^t(x + n) \Delta x^2 \tag{3.5}$$

We feed the potential $U^t(x)$ into a growth mapping $G : [0, 1] \to [-1, 1]$ to get the growth distribution G^t, Equation 3.6.

$$G^t(x) = G(U^t(x)) \tag{3.6}$$

We update every site by taking a time-step and adding our growth. We need to clip the unit back into the range $[0, 1]$. We, therefore, get Equation 3.7.

$$A^{t+dt} = \left[A^t + dtG^t(x)\right]_0^1 \tag{3.7}$$

where $[n]_a^b = min(max(n, a), b)$ is the clip function.

The kernel

To define our kernel, we use a kernel core, K_C, and a kernel shell, K_S. The kernel core determines the kernel's detailed texture, and the kernel shell defines its frame/skeleton.

The kernel core, Equation 3.8, is a unimodal function, using polar distance as argument to create a uniform around the site.

$$K_C(r) = exp\left(\alpha - \frac{\alpha}{4t(1-r)}\right) \tag{3.8}$$

The kernel shell K_s, Equation 3.9, takes a parameter vector $\beta = (\beta_1, \beta_2, ..., \beta_B) \in [0,1]^B$, where β is kernel peaks of size B, the rank, and copies the kernel core into equidistant concentric rings with peak heights β_i:

$$K_S(r;\beta) = \beta_{[Br]} K_C(Brmod1) \tag{3.9}$$

To make sure $K * A \in [0,1]$, we normalize the kernel:

$$K_S(n) = \frac{K_S(||n||_2)}{|K_S|} \tag{3.10}$$

The growth mapping

To control how fast and wide our new pattern grows we use a growth function, with parameters $\mu, \sigma \in \mathbb{R}$ that controls the growth center and the growth width. We therefore get Equation 3.11.

$$G(x;\mu,\sigma) = 2exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) - 1 \tag{3.11}$$

When used in code

In total when updating the grid to the next state when using Lenia we have the Equation 3.12:

$$A^{t+dt} = \left[A^t + dtG\left(K * A^t\right)\right]_0^1 \tag{3.12}$$

We implement this with the help of discrete Fourier transform, according to the convolution theorem, and get Equation 3.13:

$$K * A^t = \mathcal{F}^{-1}\left\{\mathcal{F}\{K\}\mathcal{F}\left\{A^t\right\}\right\} \tag{3.13}$$

To help with the computational speed we use fast Fourier transform.

3.3.3 Sampling of Parameters

Lenia has multiple changeable parameters, that change the outcome of the final pattern. These parameters control the shape and the behavior of our kernel. In each generation, there is a given mutation rate for each parameter which we can see in Table 3.4. We can see in Section 2.2.3, Figure 2.4 the outcome of different parameters for an equal start pattern.

Parameter	min	max	mutation rate
R	1	20	0.5
T	2	10	0.5
μ	0	1	0.005
σ	0	0.3	0.01

Table 3.4: Overview of the parameter space in Lenia

3.3.4 Classification of Patterns

To differentiate the different behaviors of possible outcomes in Lenia, it is helpful to classify the outcomes. We use the same definition as [20], with one exception. The different classes are

- Animal: If the final Lenia pattern is a *finite* and *connected* pattern of activity. Furthermore, a pattern must exist for at least two time steps.
- Dead: All cells are either 0 or 1 after 200 iterations.
- Non-animal: Not animal and not dead.

Reinke et al. add one more constraint to their classification of animals. This addition is: "The cells of the connected pattern $P = x_1, ..., x_n$ must have at least 80% of all activation, i.e. $\sum_{x \in P} A(x) \geq 0.8 \sum_{\forall y} A(y)$." This means that potential animals that form when they are in the same grid as a non-animal, is not defined as an animal. We can see an example of this in Figure 3.5c, where we have a larger non-animal pattern on the left side of the grid. However, we also have an animal that has been formed on the right side of the grid, formed as a sausage. We have decided to count these patterns as animals, as the grid contains an animal.

Connected pattern

If cells x and y influence each other and are both active, they are considered to be connected as a pattern. Cells are active when they have a value over 0.1, in other words, ($A(x) \geq 0.1$ and $A(y) \geq 0.1$). Cells influence each other when they are within their radius of the kernel K.

Infinite Pattern

If the pattern covers the entire image and loops around the edges of the cell grid, it is considered to be infinite.

3.3.5 Distributed Evolutionary Algorithms in Python

Distributed Evolutionary Algorithms in Python (DEAP) is an evolutionary computation framework used to compute and test setups of evolutionary algorithms [53]. DEAP was introduced in 2012 with the goal of "evolutionary algorithms made easy". The base DEAP system is built up of a toolbox and a creator. The toolbox is a holder of all operators we want to use in EA. The creator module makes it possible for the user to build your own classes via both inheritance and composition. We use DEAP when implementing the evolutionary elements, such as the evolutionary algorithm, evolutionary operations, and initialization of the population.

(a) Infinite pattern

(b) Finite and connected pattern

(c) An infinite pattern to the left, and a finite, connected pattern to the right.

(d) Infinite non-animal where all cells of the grid are connected

Figure 3.5: Example of different types of patterns.

3.4 Adapted CVT - MAP - Elites

3.4.1 The use of Quality Diversity in Lenia

Lenia exhibits a vast collection of behaviors, making it challenging to design a single optimization algorithm that can effectively capture its diverse range of patterns. Previous studies on Lenia have revealed the difficulty in discovering animal-like patterns, with non-animal patterns emerging more frequently (Reinke et al., 2019; Chan et al., 2020). However, by employing quality diversity algorithms in Lenia, there is a promising prospect of uncovering all types of behaviors rather than solely converging on non-animal patterns. The essence of quality diversity algorithms lies in their ability to explore and search for various patterns, rather than being restricted to a single objective or pattern type.

Adapted CVT-MAP-Elites can be divided into two main phases: initialization and archive addition. We can see in Algorithm 4 the pseudo-code over adapted CVT - MAP - Elites. The primary distinction of this adapted CVT-MAP-Elites from the original lies in its dependence on the rest of the system to receive individuals, rather than generating them on its own. As a result, this adapted approach does not require the creation or variation of individuals, as well as evaluation. Given that an evolutionary algorithm is utilized to generate and assess new individuals, there is no need to create variations. This also means that the adapted CVT-MAP-Elites are dependent on AML. The code used for the adapted CVT-MAP-Elites, is based on the code collected from the GitHub repository of Mouret JB et al.[54], and created based on papers [33, 43, 55, 56].

The Initialization Phase

CVT-MAP-Elites starts by creating a CVT approximation, with the help of Monte Carlo methods to obtain a close approximation to a CVT feature space. In our experiments, this means that the 8-dimensional feature space was partitioned into k regions. After experimentation, it was decided to use $k = 5000$ regions in the CVT-MAP-Elites algorithm. Initially, 1000 regions were tried, but they were found to be insufficient. 10.000 regions were also attempted, but they were found to be too large, and the map remained very empty. The algorithm then creates an empty archive, C, with a capacity k. The empty archive store the χ genotypes, and $\mathcal{P}$ performance. In our case, this means descriptors and the fitness score. After this starts the phase of adding individuals to the archive.

Add individuals to the archive phase

This phase starts with the archive getting an individual, x. The individual consists of its genotype, χ, feature descriptor b, and fitness P. The algorithm then finds the individuals index, c, of the centroid in the archive, C, that is closest to the feature descriptor. In other words, the algorithm finds the region where the individual belongs, based on the individual's feature descriptor. If the region is available, the algorithm saves the individual in that region. On the other hand, if the region is already occupied, the individual with the highest fitness score, $\mathcal{P}$, is preserved.

Algorithm 4 Adapted CVT-MAP-Elites algorithm (adapted from [43])

procedure INITIALIZATION CVT-MAP-ELITES ADAPTED(k)
    $C \leftarrow CVT(k)$
    $(X, P) \leftarrow create\_empty\_archive(k)$
    return C, X, P
end procedure
procedure ADDING ARCHIVE-PHASE(x, C, X, P)
    $ADD\_TO\_ARCHIVE(x, X, P)$
    return *archive* (X, P)
end procedure
procedure ADD_TO_ARCHIVE(x, X, P)
    $c \leftarrow get\_index\_of\_closest\_centriod(b, C)$
    if $P(c) = null$ or $P(c) \leq p$ **then**
        $P(c) \rightarrow p$, $X(c) \rightarrow x$
    end if
end procedure

Chapter 4

Experiments

This chapter describes the experiments conducted in the thesis and the differences between each experiment. This involves the motivation for the experiments and the task spicific implementation setup. The experiment code is open source, available on GitHub [48]. An overview of the experiments can be seen in the following table.

	Random fitness	**Movement based fitness**
No pre-train with found patterns	Experiment 1	Experiment 2
Pre-train with found patterns	Experiment 3	Experiment 4

Table 4.1: Overview of the experiments

4.1 Experiment with Random Fitness function

Motivation

The main objective of conducting this experiment is to create a baseline for AML, and thoroughly test the performance of the entire system. The experiment consists of having a fitness function where the fitness is chosen at random, within the interval $[0, 100]$.

The goal is to observe the system's response to seemingly suboptimal solutions, based on their fitness score. As well as seeing if the system is capable of identifying and preserving these solutions as diverse individuals. With this experiment, we also hope to see if we can provide a better understanding of the system and the potential for future improvements.

While conducting the experiment's testing phase, we encountered instances where dead individuals would accumulate within the container due to their positive fitness. In order to avoid retaining dead patterns across multiple generations, we assigned negative fitness to them.

4.2 Experiment with Fitness Function Based on Movement

Motivation

A problem with searching through Lenia landscape is that interesting individuals are rare, and searching is time-consuming. It is hard to have clear definitions of what an interesting pattern is, as there are so many different types of patterns in Lenia. Moreover, the definition of an interesting pattern is subjective and difficult to define, given the numerous pattern types present in Lenia. However, a common trait in earlier discovered interesting patterns and animals is continuous movement. Therefore, the objective of this experiment is to observe how AML responds to a fitness function based on movement. To track movement we have we have chosen to use the mass center of the pattern, over multiple time frames. The patterns that have a constantly moving mass center over multiple time frames will achieve higher fitness. Patterns that are static will exhibit a movement score close to zero, whereas moving patterns will receive a positive movement score. A more thorough explanation of this fitness function can be found in Section 3.1.5.

4.3 Experiment without using Pre-Training with Discovered Patterns

Motivation

One of the benefits of using automatic behavioral descriptors is that there is no need for any prior behavioral knowledge prior to running the experiments. This is especially advantageous when working with complex systems, where the underlying relationships between variables may not be well understood, such as in Lenia. In order to further explore the potential benefits of this approach, we aim to examine the effectiveness of AML when trained solely during the exploration phase, without any prior knowledge. With this analysis, we hope to gain a deeper understanding of the performance of AML under these conditions, and the potential for utilizing automatic BDs in other applications. Furthermore, by evaluating the effectiveness of AML under varying levels of prior knowledge, we can identify the most suitable scenarios for utilizing this approach, and gain insights into how to optimize its performance. A more thorough explanation of how we train the BD, can be found in Section 3.2.1.

Task Specific Implementation

Perform pre-training during the pre-initialization. That is, in the pre-training, there is no training of the VAE with earlier found patterns. However, there is exploration for 1000 generations to train the VAE, but without adding the found individuals into the container. After the 1000 generations, the population will be reset. The reason for this is that we want to avoid any local optima that the algorithm might have stumbled upon, and got stuck in, during this phase.

4.4 Experiment with Pre-Training Using Discovered Patterns

Motivation

We also wanted to explore whether pre-training the VAE would help the performance of AML. To compare if AML is able to perform without having any prior knowledge of patterns before the algorithm is run, we also set up an experiment with training the VAE with already discovered patterns. By comparing the results of this experiment with those obtained when the algorithm is not pre-trained, we can gain a better understanding of the benefits of pre-training the VAE and its impact on AML's performance.

Task specific implementation

In the pre-initiation phase of the system, the pre-found patterns are fed through the VAE. These pre-found patterns will be processed after pre-training with exploration, and similar to pre-training with exploration, they we not be added to the container. Furthermore, the pre-found patterns will undergo data augmentation to enhance the variability of the dataset.

4.5 Experiments Overview

As described, these four different sections are summarized into four experiments that we can see in Table 4.1.

Experiment 1

Experiment with random fitness function and without pre-training with already discovered patterns.

Experiment 2

Experiment with movement-based fitness function, and without pre-training with already discovered patterns.

Experiment 3

Experiment with random fitness function and with pre-training with already discovered patterns.

Experiment 4

Experiment with movement-based fitness function and with pre-training with already discovered patterns.

Chapter 5

Results

This chapter presents the results from the experiments described in the previous chapter. While designing AML, there were some challenges in determining the definition of an interesting pattern. As all patterns have quite different qualities, and outcomes, there is no golden rule to explain what a good result is. We experimented with the movement in image moment as a fitness function, as earlier discovered interesting patterns, often had this trait. As there is no exact standard for evaluating the outcomes, we have broken down the evaluation process into multiple assessments. These assessments are:

- Overview of the state of individuals in the last generation
- Visualisation of Fitness of Individuals
- Visualization of the archive
- Inspection of activation mass of the population
- Visual inspection of patterns
- Reconstructed patterns from the VAE

Visual inspection provides us with insights that other forms of analysis do not offer. This is due to the difficult task of automatically differentiating between interesting and not interesting patterns. If we only differentiate with fitness we might miss interesting patterns, but with low fitness.

The resulting archives from each experiment, can be found at the GitHub repository at [48].

5.1 Overview of the State of Individuals in the Last Generation

When evaluating the state of individuals in the last generation, we categorized the patterns into three classes: animal, dead, and non-animal. The definitions of the three classes are below. A more in-depth explanation be found in Section 3.3.4. All categorizations were done after 200 iterations of the initial pattern. To categorize each pattern, visual analysis was performed.

- Animal: If the final Lenia pattern is a finite and a connected pattern of activity. Furthermore, a pattern must exist for at least two time-steps.

- Dead: All cells are either 0 or 1 after 200 iterations.
- Non-animal: Not animal and not dead

The state of individuals in the final archive, that is the last generation, can be found in Figure 5.1. It is worth noting that we chose to represent the numbers in percentages, this is due to variations in the size of the final archive across different experiments.

In Experiment 1 there were no dead patterns, while 4.1% were animals. From Figure 5.1 we can see that in all experiments, the majority of states in the last generation were non-animal. In the Experiment 1 archive there were no dead patterns, while 4.1% were animals. In the Experiment 2 archive there were 0.9% dead patterns, while 4.8% of the archive were animals. Experiment 3 showed the lowest number of animals, with only 0.5% of the archive registered as animals. Within the Experiment 4 archive, there were 17.8% dead patterns, and 22.8% of the archive were animal patterns. This was the highest percentage of dead and animal patterns when comparing to the other experiments.

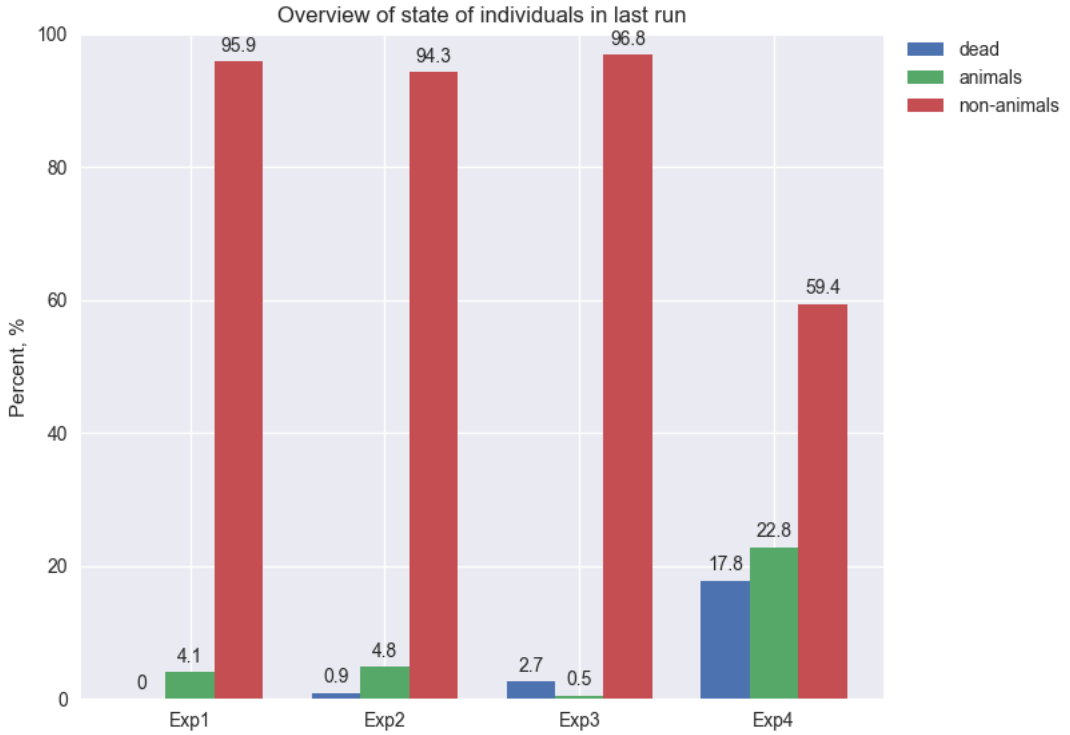


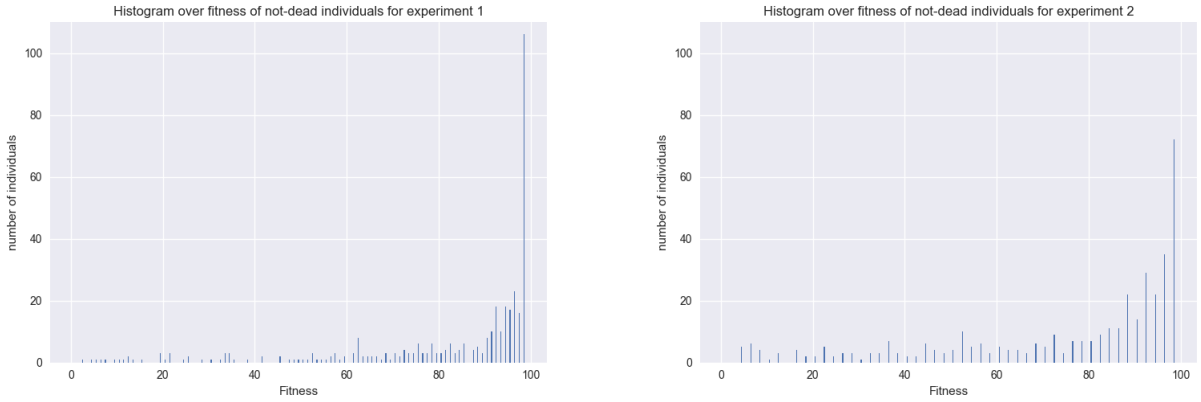
Figure 5.1: Overview over state of all patterns in the final archive.

The total regions of the archive were 5000 in all experiments. There is one pattern per region, and the size of the filled regions in the different archives was:

- Experiment 1: 527 patterns
- Experiment 2: 688 patterns
- Experiment 3: 2099 patterns
- Experiment 4: 883 patterns

5.2 Visualisation of Fitness of Individuals

From Figure 5.2 and Figure 5.3, histograms of the fitness score in each experiment can be observed, providing a visual representation of the distribution of fitness scores in the data set. The dead patterns were not included. The x-axis represents the fitness score, and the y-axis represents the number of individuals. Each fitness score is represented as a bin, that is one line bar. These histograms enable us to observe and analyze the frequency and distribution of fitness scores across the population.

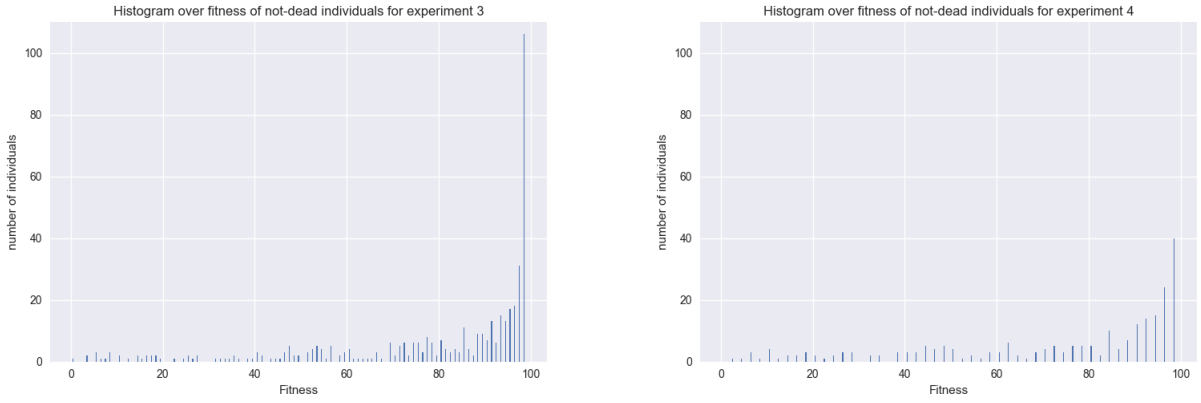


(a) Experiment 1: Map with random fitness score

(b) Experiment 2: Map with moving fitness

Figure 5.2: Histogram of fitness score for experiment 1 and 2

In Figure 5.2a the histogram for Experiment 1 with random fitness scores is presented. The histogram for Experiment 2 with movement-based fitness score is presented in Figure 5.2b. It is clear from both figures that the distribution of fitness scores is skewed toward higher fitness. Moreover, the figures display an overweight fitness score of 100 in both histograms, showing that a significant number of individuals in the population achieved the best possible fitness.



(a) Experiment 3: Map with random fitness

(b) Experiment 4: Map with moving fitness

Figure 5.3: Histogram of fitness score for experiment 3 and 4

In Figure 5.3 the histogram of experiments using pre-training with previously discovered patterns is presented, with random fitness score in Figure 5.3a, and

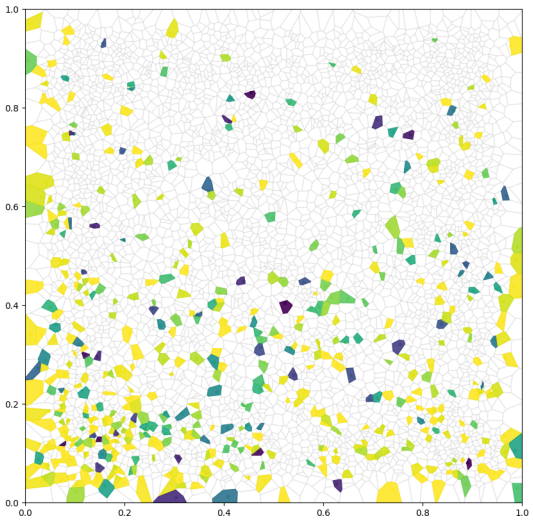
movement-based fitness score in Figure 5.3b. As with Figure 5.2, we can see that the histograms are skewed towards the higher fitness values.

The histograms based on random fitness function, both have a larger number of individuals with high fitness scores. In all histograms, we can see that individuals with a lower fitness score have been retained by the system. This indicates that AML was able to retain individuals with lower fitness scores. This is as expected since Experiments 1 and 3 used random fitness scores. However, Figure 5.2b displays that there are more individuals with high fitness scores in the archive from Experiment 2, than in the archive from Experiment 4 in Figure 5.3b. When comparing Figure 5.1 with Figure 5.3a and Figure 5.3b, and the percentage of animals with high fitness scores, we cannot here see any immediate connection.

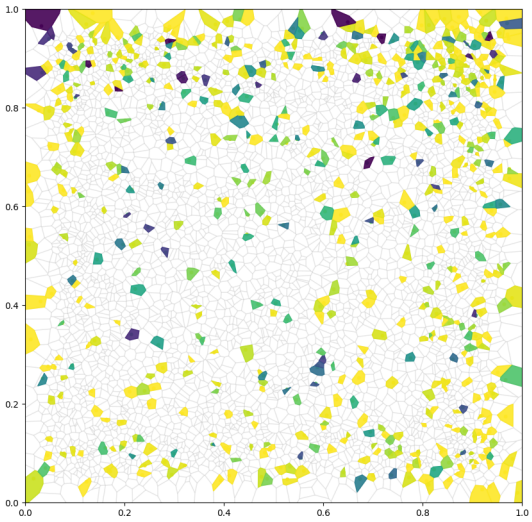
5.3 Visualization of the Archive

To help visualize the individuals from the archive, a 2D map was created. The code for the map was collected from the GitHub repo of Mouret JB et. al[54], and created based on papers [33, 43, 55, 56]. The purpose of the code is to make a 2-dimensional visualization, a map, to see how the archive is filled. Each region has the possibility to be occupied by one individual. The map is based on the 8-dimensional archive and transforms the data into a visual 2-dimensional map. It is possible that there is a loss in the quality of the data during the transformation, and we use the map mainly to see how the archive is filled. From Figure 5.5 and Figure 5.7 the visualization of the archive from each experiment can be seen. The color represents fitness, as seen in Figure 5.4 and Figure 5.6, and the x-axis and y-axis indicate the position of each individual.

Figure 5.4: The fitness color bar is used to visualize the archive.



(a) Experiment 1: Map with random fitness



(b) Experiment 2: Map with moving fitness

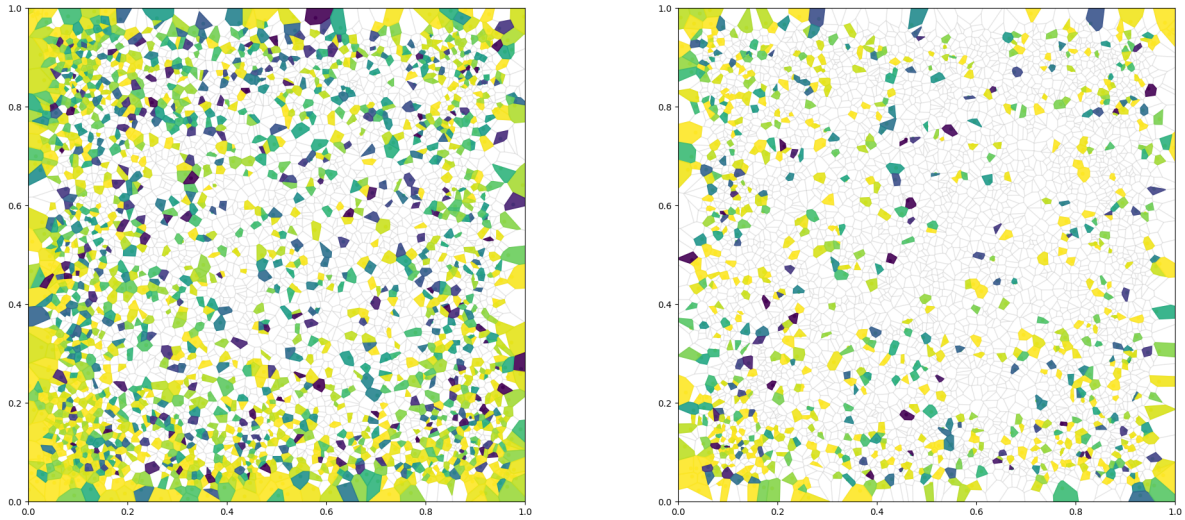
Figure 5.5: Visualization of CVT-MAP-elites - archive for experiment 1 and 2

Figure 5.5a visualizes the map of the archive belonging to experiment 1, which involved random fitness. Based on the figure, it is clear that a significant portion of the map remains unfilled or unpopulated, as there are many blank regions. We can also see that there is an overweight of high fitness values, which coordinates with the corresponding histogram. Upon observation of the map, it appears that the regions closer to the borders are more filled in comparison to the middle regions. One likely explanation for this observation is that there are fewer, larger regions near the edges of the map compared to the central areas.

Figure 5.5b visualizes the map of the archive belonging to experiment 2, which

involved moving-based fitness. The map is partially filled, with numerous empty regions. When compared to Figure 5.5a, the two maps exhibit a partially opposing pattern in terms of their filled regions. Specifically, Figure 5.5b is more filled in the upper right corner, while Figure 5.5a has a skewed filled region in the lower left corner. The dimensions of the map are equal to the dimensions of the BD, but represent different elements in each map. This difference arises due to the individuality of the VAE in each experiment, after its individual training. Consequently, establishing correlations between the various maps and linking different regions of the map with diverse patterns becomes challenging. However, as an optimal map has fulfilled close to all its regions, we can assume that the algorithm's performance was not optimal.

Figure 5.6: The fitness color bar is used to visualize the archive.



(a) Experiment 3: Map with random fitness

(b) Experiment 4: Map with moving fitness

Figure 5.7: Visualization of CVT-MAP-elites - archive for experiment 3 and 4

Figure 5.7a visualizes the map of the archive belonging to experiment 3, which involved random fitness and pre-training. In comparison to Figure 5.5a and Figure 5.5b, it is evident that the map presented in this figure is significantly more densely populated. However, there are still a large number of regions on the map that are empty. The increased density within the map indicates that the algorithm, guided by the use of pre-training using previously identified patterns and random fitness scores, successfully discovers and contains a broader range of patterns.

Figure 5.7b visualizes the map of the archive belonging to Experiment 4, which involved moving-based fitness and pre-training scores. Contrasting with Experiment 2 displayed in Figure 5.5b, which did not involve pre-training using previously identified patterns, we observe that the map displayed in Figure 5.7b exhibits a somewhat higher level of the filling. Furthermore, the filled regions are more evenly

distributed across the entire map. However, when comparing Figure 5.7b and Figure 5.7a, the experiment with random fitness score has more regions filled.

5.4 Inspection of Activation Mass of the Population

To evaluate the proportion of different patterns within the archive, statistical measures presented in Section 2.2.2 are used. We mainly focus on Activation Mass, which is the normalized sum of the activation of the Lenia pattern. We evaluate all patterns within the archive, calculate the pattern's activation mass, and make a histogram based on this. In the histogram, the x-axis represents the activation mass score, while the y-axis represents the count of individuals with each score. The histograms consist of 40 bins, categorizing each pattern into one of the 40 bars based on its score.

Given that closely comparable patterns will achieve similar activation mass scores, a high percentage of a particular class in the archive suggests a significant presence of nearly identical patterns. To analyze the most substantial grouping in each experiment, we select the largest group within each archive as identified by the histogram. Thereafter, we examine four representative patterns from this group for visual analysis.

5.4.1 Experiment 1

The histogram of the activation mass score for Experiment 1 can be observed in Figure 5.8. Multiple bars in the histogram exhibit a significant number of patterns with similar scores, particularly in the area of $[0.53, 0.56]$ where over 100 instances share this score.

By visually examining the pattern with the score within the interval $[0.529, 0.535]$, we can see that these patterns appear quite similar. Examples of four patterns with this score can be seen in Figure 5.9. Figure 5.9b and Figure 5.9c are especially similar, while Figure 5.9a looks like Figure 5.9b zoomed in. However, when examining Figure 5.9d, we can see that this differentiates from the other three. The pattern appears more detailed, with a smaller grainier pattern. Nevertheless, Figure 5.9d appears to have a similar structure as the other three figures, but more zoomed out.

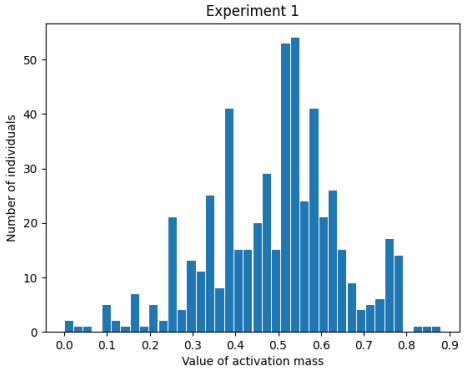


Figure 5.8: Histogram of activation mass of the archive in Experiment 1

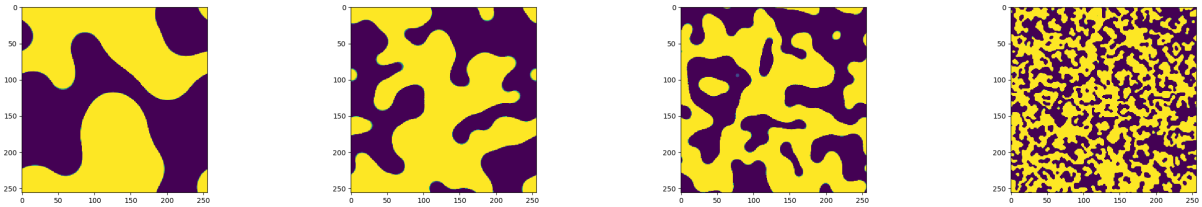


Figure 5.9: Experiment 1 pattern with activation mass within the interval of $[0.529, 0.535]$

.

5.4.2 Experiment 2

The histogram of the activation mass score for Experiment 2 can be observed in Figure 5.10. Figure 5.10 indicates that there exists a notable grouping within the archive, consisting of over 100 patterns that have attained an activation mass score of 0.816. This group significantly surpasses all others in size. We have studied this group, and selected four example patterns from the collection, illustrated in Figure 5.11. By examining the patterns in Figure 5.11, it becomes apparent that they are very close in appearance. Given the activation mass score of 0.816, a substantial number of cells exhibit high levels of activity, which is readily apparent in the patterns. Among the four patterns, Figure 5.11c stands out, as it exhibits three regions with blue-gray colors. These areas arise because the pattern has not yet reached its final pattern, and needs more than 200 iterations to reach a stable pattern. Given approximately 20 additional iterations, the pattern would resemble the other three patterns. However, these patterns with blue-gray colors will result in a moving image moment.

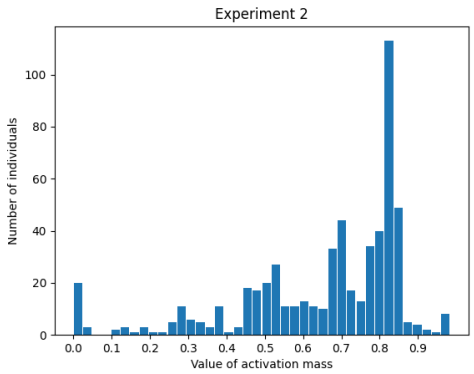


Figure 5.10: Histogram of activation mass of the archive in Experiment 2

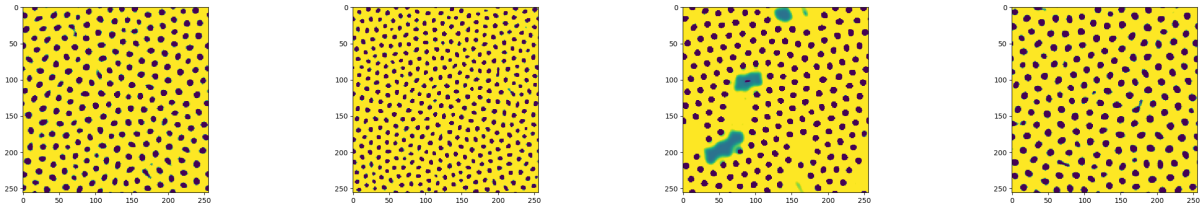


Figure 5.11: Experiment 2 patterns with activation mass within the interval of $[0.815, 0.817]$

.

5.4.3 Experiment 3

We can see in Figure 5.12, a histogram of Activation Mass for the archive in Experiment 3. Figure 5.13 indicates that there are two large groupings of similar patterns with an activation mass score close to 0.3. The two groups together consist of approximately 200 patterns. By visualizing four patterns with activation mass close to 0.3 in Figure 5.13, we can see that the patterns have a very similar composition. The patterns are all non-animals and consist of multiple dots of various sizes. Figure 5.13b displays larger dots compared to the other patterns in the figure, making it easier to distinguish from the rest. This observation highlights that the activation mass serves as a mere indication of the grouping of similar patterns, rather than a definitive determination of similarity.

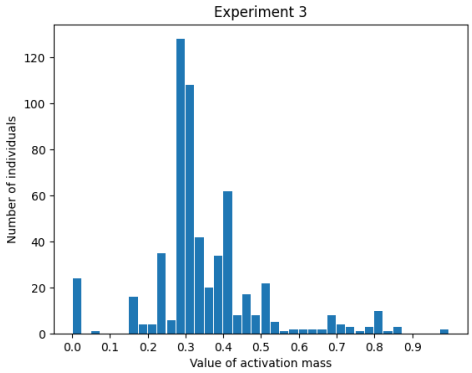


Figure 5.12: Histogram of activation mass of the archive in Experiment 3

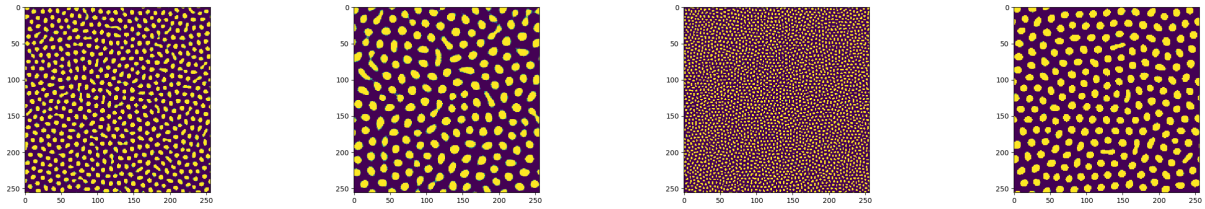


Figure 5.13: Experiment 3 pattern, with activation mass within the interval of $[0.25, 0.3]$

.

5.4.4 Experiment 4

We can see in Figure 5.14, a histogram of Activation Mass for the archive in Experiment 4. It can be observed that the histogram is skewed toward a lower activation mass score. This corresponds with the high number of animals found in the archive, seen in Figure 5.1, as animal patterns are finite patterns that have a lower number of active cells. The most significant grouping occurs within the range of $[0, 0.2]$ This observation aligns with Figure 5.1, where a notable proportion of the animals in the archive are classified as dead. This outcome was expected, considering that dead patterns receive an activation mass score of zero. We can see an example of a dead pattern in Figure 5.15c. However, there is also a sizable group with scores close to zero. Illustrations of these patterns can be found in Figure 5.15a, 5.15b, and 5.15d. These patterns are relatively small in size, which contributes to their very low activation mass scores. Despite their small size, some of these animals are categorized as animals, as they are finite and have a connected pattern of activity. However, a significant group of these patterns do not survive two time steps, they are non-animals.

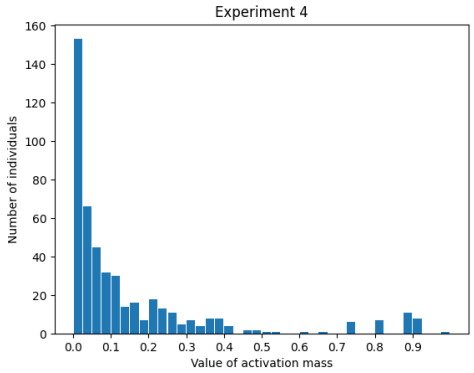
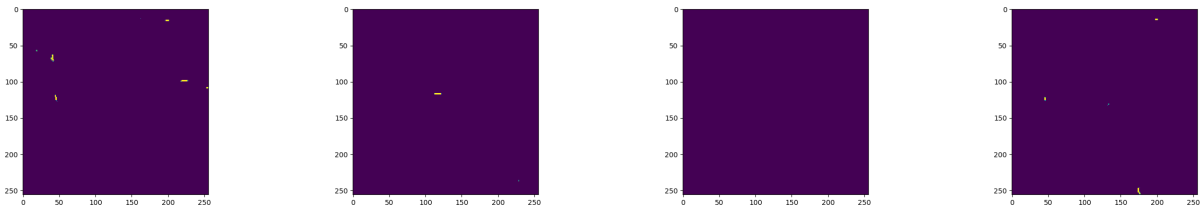


Figure 5.14: Histogram of activation mass of the archive in Experiment 4



(a) (b) (c) (d)

Figure 5.15: Experiment 4 pattern, with activation mass within the interval of $[0, 0.02]$

.

5.5 Visual Inspection of Patterns

There are 4 patterns that are highlighted in each experiment. These patterns are a small portion of the data from each experiment, where we have tried to capture both patterns that often are repeated and unique patterns of each experiment. There is a larger selection of patterns van bee seen in Figure A.1, A.2, A.3, and A.1 in Appendix A. In this section, do we try to capture if there is something unique with each archive, and what animals and non-animals have been discovered.

5.5.1 Experiment 1

We can see in Figure 5.16, the visualization of 8 patterns from Experiment 1. As we saw in the barplot of the percentage of dead, animal, and non-animals in Figure 5.1, there were very few animals in this archive. The most recurring patterns within this collection were variations of Figure 5.16a, Figure 5.16b, Figure 5.16c, and Figure 5.16d. With variations, we mean patterns that are qualitatively similar but are identified as different by the VAE due to being zoomed in or out compared to each other. The patterns are non-animals, as they are dependent of each other (if we remove one of the dots, the other dots will spread and recreate the pattern). In Figure 5.16e and Figure 5.16f, we can see patterns that also were recurrent within the archive, but rarer. We can see examples of animals found in this archive, in Figure 5.16g and Figure 5.16h.

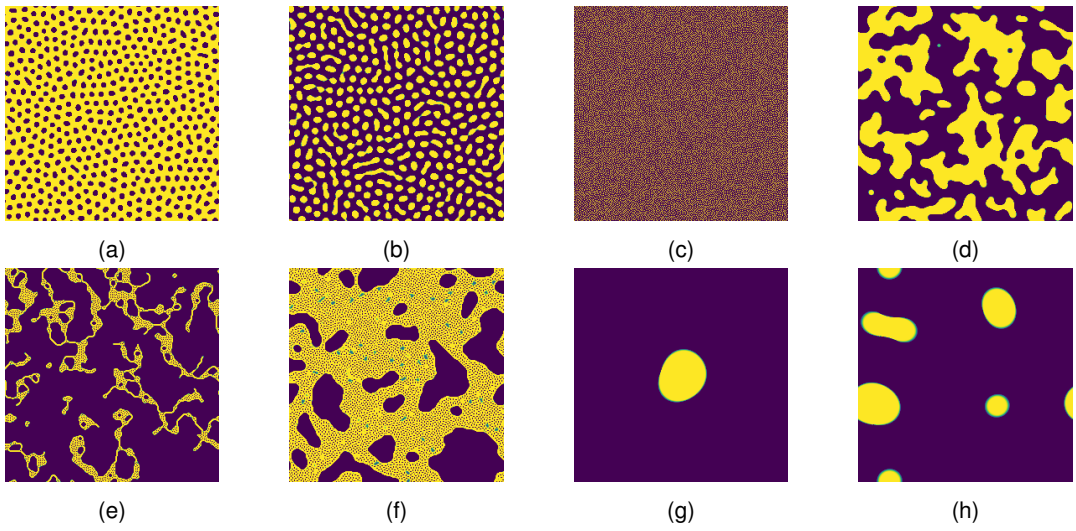


Figure 5.16: Experiment 1 patterns

5.5.2 Experiment 2

We can see in Figure 5.17, the visualization of 8 patterns from Experiment 2. In this archive, there was a high portion of recurring patterns that were variants of Figure 5.17a, Figure 5.17b, and Figure 5.17c. That is non-animals, which are large patterns, that are slightly moving. Especially in Figure 5.17b we can see an example of a non-animal that will over multiple time frames achieve a moving image moment. We also saw the high grouping of large patterns, that are slightly moving in the histogram in Figure 5.10. Additionally, in this experiment, we got a higher number of patterns that contain animals. We can see animals in Figure 5.17d, Figure 5.17e, Figure 5.17f, Figure 5.17g and Figure 5.17h. There are multiple animals that go over the edge of the grid. However, as the grid wraps around the edges, forming a ball-shaped neighborhood, this still counts as an animal

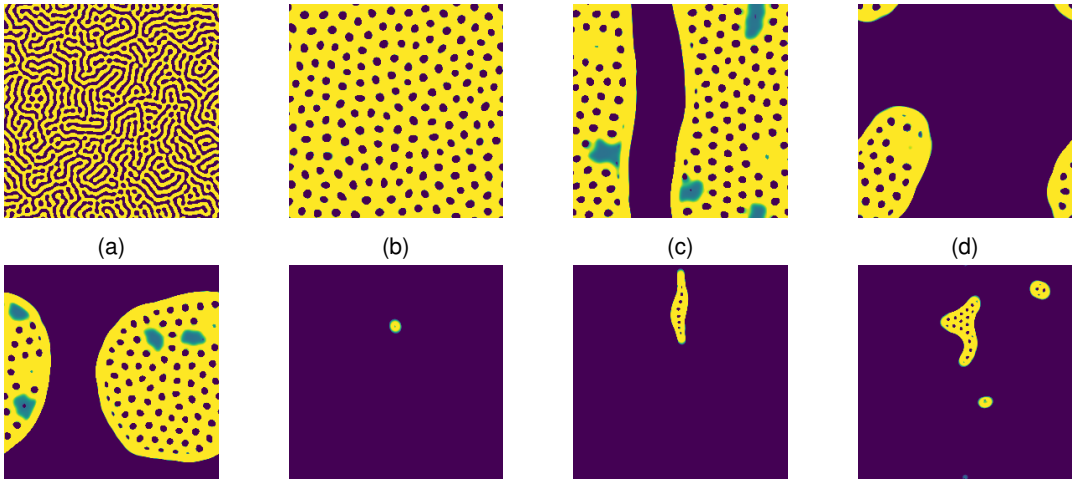


Figure 5.17: Experiment 2 patterns

5.5.3 Experiment 3

Figure 5.18 presents the visualization of eight patterns derived from Experiment 3. As shown in Figure 5.7a, the archive is densely populated with a wide variety of patterns. However, upon closer examination of the archive, we observe that these different patterns bear a striking resemblance. In other words, AML was able to find a lot of different patterns, however, the patterns had a high resemblance. This similarity is noticeable in examples such as Figure 5.18a and Figure 5.18b, where despite being distinct patterns, they still exhibit notable similarities in appearance. A notable distinction of this archive, when compared to others, is the presence of new pattern types illustrated in Figures 5.18e, 5.18f, 5.18g, and 5.18h. These patterns illustrate a more intricate structure characterized by the composition of multiple elements. Figure 5.18f is of special interest, as we can see an animal within the overall pattern, highlighting unique characteristics within the archive. When comparing this archive to the others, we find a wider range of diverse non-animal patterns. This is as expected when looking at the visualization of the archive in Figure 5.7a.

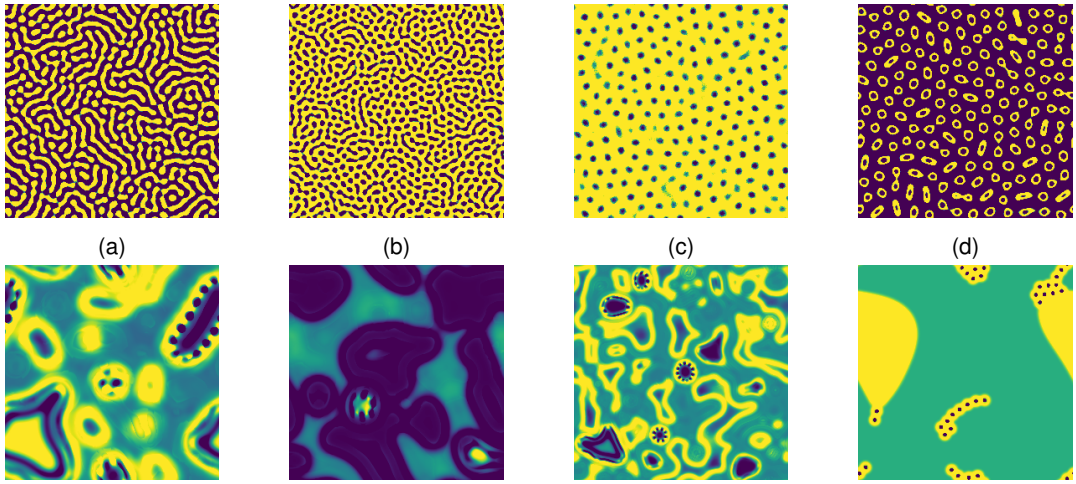


Figure 5.18: Experiment 3 patterns

5.5.4 Experiment 4

Figure 5.19 illustrates the visualization of eight patterns derived from Experiment 4. This archive contained the highest number of animals, as observed in Figure 5.1. The archive also contained a high number of dead patterns, that we have chosen to not illustrate. Examples of animals can be seen in Figure 5.19b, 5.19c, 5.19d, and 5.19f. Additionally, Figure 5.19g and 5.19h depict intriguing examples of non-animal formations, which exhibit infinite patterns. The archive contained numerous variations of these slender, infinite patterns. These patterns were not found in any of the other archives. Figure 5.19e was also of interest, as this is a pattern that shows a large difference from other non-animals.

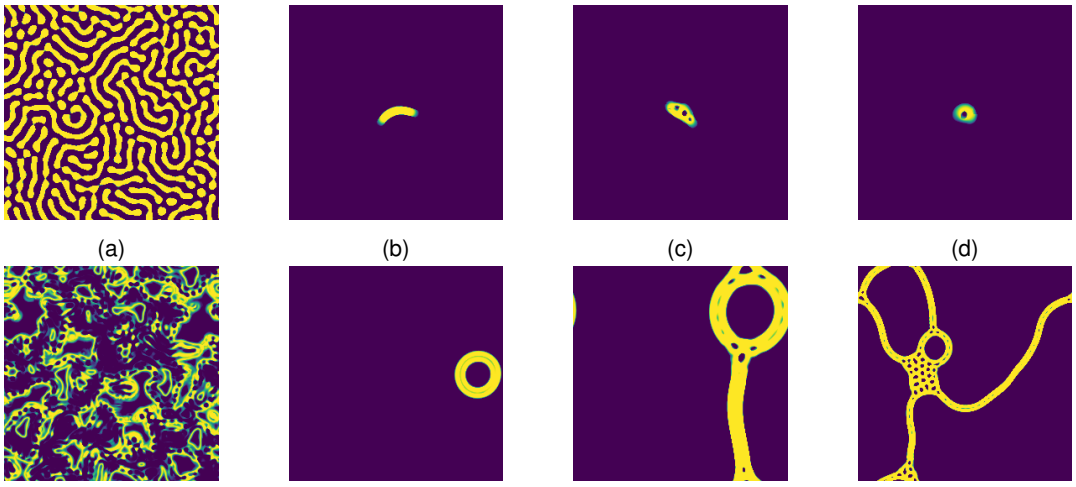
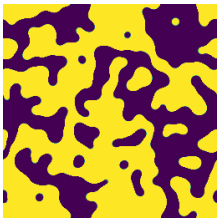


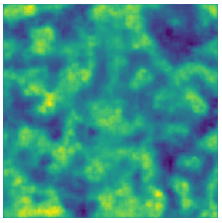
Figure 5.19: Experiment 4 patterns

5.6 Reconstructed Patterns from the VAE

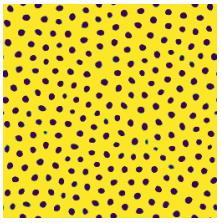
In Figure 5.20, a pattern from each archive is displayed along with its corresponding reconstruction generated by the VAE. The BD serves as input to the decoder, which produces the reconstructed pattern. These reconstructions play a crucial role in training the VAE and ensuring the adequacy of the BD. Upon closer examination of the patterns, it becomes apparent that the reconstructions bear resemblance to the original patterns. However, there are noticeable variations in all reconstructions. This outcome is to be expected since the reconstruction is based on an 8-dimensional vector, resulting in some loss of information during the encoding and decoding process. In both Figure 5.20a and 5.20b, and Figure 5.20c and 5.20d we can recognize the original pattern in the reconstructed image. The decoder manages to reconstruct the complex patterns to a certain extent. One large difference is observed in Figure 5.20e and 5.20f. The original pattern is consisting of all dead cells, whereas in the reconstruction t can be observed that there are still active elements present. In Figure 5.20g and 5.20h we can also see that there are some variations to the original pattern. Active elements are present in the reconstruction where they were not expected. However, the overall pattern is still recognizable, and the VAE successfully reconstructs the hole on the left side of the pattern.



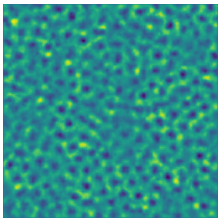
(a) Experiment 1 original



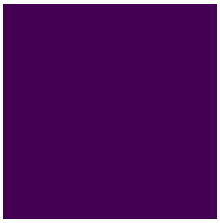
(b) Experiment 1 reconstructed



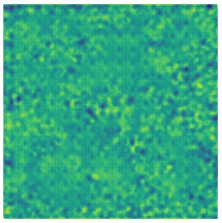
(c) Experiment 2 original



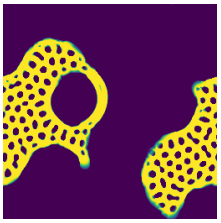
(d) Experiment 2 reconstructed



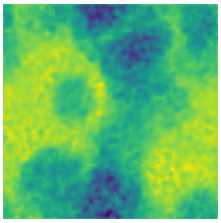
(e) Experiment 3 original



(f) Experiment 3 reconstructed



(g) Experiment 4 original



(h) Experiment 4 reconstructed

Figure 5.20: Reconstruction examples from the VAE

Chapter 6

Discussion

In this chapter, we discuss the choices made in the design of AML, as well as the results presented. We also discuss how our results are relevant and answer the aims as well as the sub-aims of this thesis. The main aim of this thesis was to investigate whether automatically defined behavioral descriptors are helpful tools for exploring Lenia. The sub-aims of the thesis were as follows: 1) We wanted to find a method of automatically exploring Lenia by using a quality diversity algorithm with automatically defined behavior descriptors 2) We wanted to investigate what impact pre-training the behavior descriptors has on the outcome. 3) We wanted to direct the search toward the patterns movement, and assess whether such movement facilitates the discovery of new and diverse patterns in Lenia. 4) We wanted to categorize the patterns, analyze, and evaluate the efficacy of our approach.

6.1 Automatically Exploring Lenia

To be able to automatically explore Lenia we combined QD and BD. We decided to use QD in our method, as we want to find a wide array of different patterns. QD has shown great success in generating a repertoire of good solutions, and with that leading to creative solutions[57]. By exploiting this characteristic of QD, we aimed to achieve a large archive of patterns, and with that a large archive of different and creative behaviors of the patterns. In the study where Cully presented AURORA[8], we saw that he successfully applied QD in a robotics application. We wanted to investigate if this would be sufficient in Lenia as well. Further, we opted to use methods from evolution to create new individuals. We desired to adopt this approach based on the achievements and inspiration drawn from the study of Chan[6], and his success in generating new animal patterns. In the following subsections, we will first discuss our success in using QD in our method and the flaws of our method within QD. Further, we will discuss our approach to generating new individuals, with a focus on the creation of the start grid and mutating individuals.

6.1.1 Quality Diversity

As described in the Results chapter, AML demonstrated its ability to identify and retain a wide range of diverse patterns. Further, we observed the capability of AML to retain both animals and non-animals, as well as its ability in recognizing intricate textures. When designing AML, we noticed enhanced performance when focusing on centering the patterns, as elaborated in Section 3.2.4. This approach encouraged the

AML's comprehension that two animals were equivalent, despite their occurrence at different time intervals. However, we also observed some occasions when this was not successful, as in Figure 5.17d. This is due to the fact that Lenia's grid is structured as a ball, where the southern and northern edges, as well as the western and eastern edges, are connected. This sometimes resulted in patterns that were halfway on each edge, with the image moment in the middle. These images were not centered, as the image moment interpreted the patterns as centered. There is also a possibility for patterns to be mirrored versions of each other in different directions. These patterns will be interpreted as different and will result in different BDs.

One of the larger challenges with AML was large groupings of very similar patterns, as we saw in Section 5.4. In this section, we saw that in all archives there were one or more major groups that had a very similar activation mass. Activation mass is only an indication of a similar pattern and very different patterns may have the same activation mass. However, large groupings of similar non-animal patterns were seen in Section 5.5. This confirmed our suspicions that there were large groupings of close to equal patterns within all archives. By visual inspection of the patterns, we saw that the indication of large groupings of similar patterns was right. As none of the patterns were identical, they achieved different behavioral descriptors and were placed in different regions on the map. The AML believes that the patterns are different, and saves them as different. These large groupings are a weakness of AML, which goes against our desire for a more diverse outcome. By minimizing these clusters, we could potentially enhance the effectiveness of algorithms, resulting in higher quality and more varied results. In Section 6.2 we will discuss further underlying reasons for this occurrence.

6.1.2 The Initialization of Start Grid

In our study, we initialized the start grid randomly. This proved to work, and we observed multiple different animals and non-animals. There were also a surprisingly low number of dead pattern in Experiment 1, 2, and 3. However, in the related study conducted by Reinke et al.[7], they employed neural networks to generate the start array and achieved promising results. Their approach demonstrated the ability to generate a larger quantity of animals and a greater diversity of animals compared to our study. This discrepancy in results could potentially be attributed to the differing approaches in generating the start grid. Given that the grid plays a crucial role in the discovery of animals, it is plausible that our simplistic method of randomly generating the grid may have been inadequate in capturing the necessary complexity and structure for optimal animal identification.

6.1.3 Mutation of the Pattern

In our study, we opted to mutate the grid by adding noise. However, since we did not discover a significant variety of animal types, there is an opportunity for improvement in our mutation method within AML. In the study of Chan [6] they demonstrated the discovery of new animals through manual mutation of previously discovered animals. Their approach involved a single-side flipping operation, which can be considered a form of manual crossover operation. Drawing inspiration from Chan's success with manual mutation, it might be worthwhile to explore the implementation of a crossover operation at random positions within the grid or consider performing a form of crossover after centering the pattern. These alternative approaches to mutation

could potentially yield interesting results and enhance the diversity of discovered patterns within AML.

6.2 Effects of Pre-Training

We conducted two versions of pre-training in our thesis. In the first version, we had pre-training for 1000 generations, during which Lenia generated its own data in the exploration phase. The second version involved pre-training before the 1000 generations, where we introduced discovered patterns from Lenia. The reason for the testing with pre-training of discovered patterns, was that Reinke et al. [7] conducted experiments involving various versions of VAEs, one of which was trained on discovered patterns, and one was trained during exploration. The aim was to exploit their findings by combining the two techniques.

Our results indicate that the experiments showed improvements when using discovered patterns during pre-training, but in different domains. The experiment with random fitness showed a larger archive, while the experiment with movement-based fitness showed a higher percentage of animals. However, it is important to note that we were unable to validate these results through multiple runs of each experiment, as this was not conducted in our thesis. This was due to limited time and resources.

6.2.1 Experiment 1 and 3

When comparing the experiments conducted with and without pre-training using a random fitness function, a notable difference is observed in the level of archive filling. Specifically, Experiment 3, which involved pre-training, exhibited a significantly higher degree of archive filling compared to Experiment 1, which lacked pre-training. A possible explanation for this substantial difference lies in the ability of the VAE to generate high-quality BDs at an earlier stage when pre-training is applied. Initially, even with pre-training, the VAE may not be as optimal at generating BDs. Consequently, patterns could be erroneously placed in incorrect regions within the map. To reduce this issue, a new map is generated and all patterns from the previous map are passed through the VAE to obtain updated BDs at specific intervals, that is at $[50, 150, 350, 500]$ generations. Following each BD update, a shrinkage in the archive size is observed. This occurs because patterns with initially distinct BDs are reassigned to similar BDs and subsequently placed within the same region. This starts local competition, resulting in the preservation of only the best individual. Considering the significant difference in archive sizes, this is a strong indication that pre-training with discovered animals aided AML in generating BDs and differentiating patterns from each other at an earlier stage. Consequently, the substantial shrinkage effect observed in the absence of pre-training was reduced, leading to a larger archive in the final generation.

When comparing the percentages of discovered animals in each archive, we observed that the archive with pre-training exhibited a higher percentage of animals. This finding was somewhat unexpected, considering the significantly larger size of the pre-training archive. After a closer inspection of the pattern grouping based on their activation mass score, we observed a large grouping within the archive in Experiment 3. There were significant clusters of patterns that exhibited notable similarity. The two largest groups contained over 100 patterns each, sharing a similar composition.

This observation suggests that although the BDs generated in Experiment 3 were of considerable quality, they separated patterns to such an extent that even slight variations in pattern led to their placement in separate regions. Thus, patterns that were nearly identical and would be perceived as having equivalent behavior by human eyes were categorized as different. This insight indicates that the QD of AML did not achieve the desired level of optimization. Patterns that were closely similar were classified as different due to the excessive separation imposed by the BDs.

The study of Reinke et al. [7] argued that their VAE were able to reconstruct the general form of the pattern, however, they observed a significant level of blurriness in the reconstructed patterns. They argued that this indicated that the VAE was not able to encode the finer details and textures, and that was the reason for their success in identifying more animals, as the animals often have a difference in form, rather than texture. This contradicts our belief that the VAE was able to differentiate the different finer patterns, resulting in the archive with variants of non-animals. However, when comparing the reconstructed images from our VAE in Figure 5.20, to their reconstructed patterns, we see that our training has achieved some higher texture in the reconstruction. This difference is particularly high when comparing the non-animal patterns, where we noticed that our training process managed to capture higher levels of texture in the reconstruction. This observation suggests that our claim regarding the VAE's ability to separate non-animal patterns with intricate details may be accurate.

One way of trying to be able to handle the high occurence of similar pattern may be to try to use a different sizing of the map and run AML for more generations. Further, there is the possibility to modify the VAE, and train it to not differentiate the nearly identical patterns. In our thesis, we were not able to do this, due to limited resources and time.

6.2.2 Experiment 2 and 4

When comparing Experiments 2 and 4, we observed that the number of the discovered pattern in the archive of Experiment 4 was higher than of Experiment 2. This finding aligns with our observations in Experiments 1 and 3, although the difference in archive sizes is not as significant. This suggests that pre-training with discovered patterns may not have as been pronounced an impact as initially anticipated when considering Experiments 1 and 3.

On the other hand, Experiment 4, which involved pre-training, exhibited a higher percentage of animals compared to Experiment 2. This implies that pre-training with discovered animals proved beneficial. This phenomenon could be attributed to similar factors as observed in Experiments 1 and 3, where the BD performed better at an earlier stage.

It is also possible that the substantial difference between experiments is influenced by random variations that occur in each run. This underlines the importance of conducting multiple runs of the same experiment to ascertain clear distinctions. Ideally, we would have pursued this approach, but we did not have the time to do this.

6.3 The Fffects of Movement as a Fitness Function

We examined the fitness scores of patterns stored in the archive from Experiments 2 and 4, as shown in Figure 5.2b and Figure 5.3b.

The histogram reveals a significant number of patterns with high fitness scores. However, there is still a considerable portion of patterns with lower fitness scores, indicating a low rate of patterns exhibiting constant movement length over multiple time frames. The presence of individuals with fitness scores of 100 suggests the existence of patterns characterized by high movement lengths, which were randomly assigned a score of 100. Additionally, the high number of individuals with lower scores indicates the prevalence of patterns with smaller movements. We also observed in Experiment 2 the presence of a large grouping of slightly moving, pulsing patterns. Furthermore, Experiments 2 and 4 exhibited the highest percentage of animals within the archive. We also observed in Experiment 2 the presence of a large grouping of slightly moving, pulsing patterns. This observation suggests that the use of a more directed search with movement contributed to AML's ability to discover more intriguing patterns, as well as animals. However, the limited occurrence of individuals displaying constant movement length indicates the potential for improvement in this aspect of the fitness function.

The study of David & Bongard[18] tried a strategy with evolving patterns in Lenia by selecting for mobility and conservation of mean cell value. They designed the fitness function with a positive reward for displacement in the center of mass, penalized for changes in average cell values, and severely penalized for patterns that disappear entirely. This fitness function is similar to ours, however, they preserved average cell values. This approach contributes to stable animals, that do not disappear over multiple time steps, or enlarge and change to a non-animal. We did not use this approach in our fitness function, as we would solely focus on movement and how this influenced AML. However, implementing this feature might help AML to not get dominated by non-animals, as we saw in Experiment 3. A possible drawback to the algorithm by David & Bongard is that it penalizes the changes in average cell values. This makes it prone to lose potential interesting non-animal patterns, as well as animals that exhibit size variations during different time steps.

6.4 Classification of Patterns

When we compare our results to Reinke et al. [7], we see that Reinke in all experiments was able to find a higher percentage of animals. In their random exploration, they found 8.1% animals, whereas our results with random exploration were at 4.1% and 0.5% animals. Their score for dead patterns was at 34.6%, whereas our score with random exploration was at 0% and 2.7%. This could indicate that with the use of our evolutionary algorithm, we were able to stop generating dying patterns. As they had a larger exploration time, with 5000 generations, this could likely influence the outcome.

We used a similar categorization as Reinke et al.[7], which involved finite connected patterns that needed to persist for at least two time steps. This classification allowed for the identification of animals that survived for two time steps but perished in the fourth. Additionally, there were patterns that initially resembled animals but evolved into non-animals after ten time steps. This categorization of

animals, made it possible for animals that survived two time steps, but would die in the fourth. Patterns that also would start as an animal, but after 30 time steps would evolve into a non-animal. To improve this categorization, and enable the identification of more stable animals, we could consider incorporating a definition based on cyclical behavior. In other words, a pattern would need to progress for X number of time steps and then revert back to its original formation.

This categorization could be improved, to enable more stable animals. One definition that could be used is cyclical behavior, that is a pattern must go X timesteps, and then return to the same formation.

6.5 Discussion on the Ffficacy of Training the VAE

One significant drawback of AML in the thesis is the requirement for training the VAE, which consumes considerable time and resources, yet it is crucial for achieving satisfactory BDs. The total running time for AML was individual for each experiment and was in the interval of approximately three to five days. Of this total time, the pre-training took approximately two days. The long training time is caused by the initialization of the VAE with random weights in the neural networks, as the VAE is constructed by two convolutional neural networks with multiple layers that need to be sufficiently trained. All our experiments took a considerable amount of time to run and we consider these factors as the main reason for the long training time. It would be beneficial to reduce this pre-training time, both to be more efficient and spare resources. However, if we reduce the training time, there is a risk that the resulting BDs may not be as precise.

With the use of transfer learning, there is a possibility of reducing the training time while still attaining satisfactory BDs. Transfer learning is an approach used to leverage knowledge acquired from one task, to effectively solve another task. In other words, if we use networks that are already pre-trained, and then fine-tune the network to specialize it towards our problem, we might be able to reduce training time. On the other hand, one benefit of only training the VAE with Lenia patterns, is not introducing any inherited biases from another training set. Additionally, one crucial aspect of automatic BD generation is minimizing the reliance on extensive prior information. This means that AML should not require any additional information, such as a pre-trained network.

The survey of Ribani & Marengoni[58] brought up the question of negative transfer when examining transfer learning in convolutional nets, where transfer learning does not lead to any improvements. Circumstances, where this was observed, were cases with insufficient relevance between the source and target tasks, or the transfer method could not leverage the relationship between the two tasks. This could be the case of using transfer learning in AML if we do not use the appropriate pre-trained net.

The use of transfer learning the VAE within AML might increase the efficiency and decrease the training time and resources used to train the VAE. However, the choice of which pre-trained network is crucial, as we do not want a negative transfer to occur.

6.6 Discussion on the Analysis of the Results

By conducting multiple replications of the experiments, a more robust evaluation would be formed, enabling a clearer comprehension of the patterns and trends. This approach serves to lessen the impact of random variations and generates stronger evidence regarding the validity and reliability of the observed outcomes. It enables the identification of potential biases or outliers and allows for the assessment of overall consistency in the findings. This increased level of replications of the experiments, enhances the confidence in the conclusions drawn from the analysis, ensuring that the reported results are not artifacts of chance occurrences. This would provide more reliable and reproducible findings, reinforcing the validity of the observed phenomena and enhancing the overall scientific rigor of the research. However, due to time constraints, this was not done in this thesis.

On the other side, we observed significant variations in the size of archives after pre-training, and notable improvements in the discovery of animals when utilizing the movement-based fitness function in our experiments. As there were so large findings, this suggests that incorporating pre-training with discovered patterns yielded enhancements in performance. Furthermore, an extensive range of patterns was present across all archives. In Section 5.6, we saw examples of original and reconstructed patterns of the VAE with success. This observation indicates that the utilization of VAE aids in recognizing and distinguishing between different patterns and behaviors. The consistent presence of this phenomenon across all experiments strengthens the indication of the VAE's usefulness as a BD.

Chapter 7

Conclusion & Future Work

7.1 Conclusion

The primary goal of the thesis was to investigate whether automatically defined behavioral descriptors are helpful tools to achieve a wide range of patterns in Lenia. This was achieved by developing AML, a method for automatically exploring Lenia, by using a quality diversity algorithm with automatically defined behavior descriptors. We directed the search toward the pattern's movement, by using the mass moment of the pattern to give the pattern a fitness score. Additionally, we investigated how the impact of pre-training the behavior descriptors had on the outcome. In order to assess the effectiveness of the movement-based fitness function and pre-training approach, we conducted four experiments with the following set-ups: 1) Random fitness values and no pre-training, 2) Movement-based fitness function and no pre-training, 3) Random fitness values and pre-training, and 4) Movement-based fitness function and pre-training.

Comparing the results from each of the experiments allowed us to evaluate the performance of the movement and pre-training method. For each experiment, we discovered a wide range of patterns containing both animals and non-animals. Further, we found indications that using a search directed by movement encourages the discovery of diverse patterns in Lenia, as well as the development of animals. Additionally, we saw indications that incorporating discovered patterns during the pre-training phase of the VAE resulted in improved outcomes in terms of both size of the archive and the quality of the patterns. However, it is challenging to make definitive conclusions due to the need for additional testing and further training of AML In other words, we have indications that using automatically defined behavioral descriptors is a helpful tool when discovering patterns in Lenia.

7.2 Future Work

Since this study has shown indications that it is possible to explore Lenia with the use of automatic BD, with interesting results, this opens up many new ideas for further studies.

7.2.1 Initialization of the Start Array

In our study, we developed the initialization array by random generation. However, the study of Reinke et al. [7] demonstrated promising results by utilizing neural networks to generate the start array. Incorporating this approach into AML could potentially enable a more intentional and targeted exploration within Lenia.

7.2.2 A Larger Population and a Higher Number of Generations

In our results, we observed that none of the archives were filled. We believe that one of the reasons for this was the need for more generations with a larger population. Due to resource limitations in this thesis, it would be intriguing to explore the results of conducting longer and larger-scale experiments.

7.2.3 Exploring Alternative Behavioral Descriptors in AML

In our experiment, we observed favorable outcomes utilizing the Variational Auto Encoder (VAE), which we selected based on the success demonstrated by Reinke et al. [7]. However, we did not experiment with different types of BDs in AML, as this was out of scope. Thus, it would be intriguing to investigate the performance of AML by employing alternative behavioral descriptors, such as autoencoders or handcrafted descriptors, and assess whether these alternatives yield improved results.

7.2.4 Test other Fitness Functions

We have seen indications that using movement to calculate the fitness score leads to more animal patterns. It would have been interesting to see how AML reacted to different fitness functions, such as the study of David & Bongard[18], where they had success by selecting for mobility and conservation of mean cell value.

7.2.5 Pre-Training with a Larger Number of Discovered Patterns

We have observed indications that incorporating discovered patterns during the training of the VAE produces better results. As a next step, it would be intriguing to explore AML's response to larger datasets. This investigation would provide valuable insights into the performance of AML when dealing with increased data volumes.

7.2.6 Different Sizing of the Archive

In Experiment 3, we observed an archive characterized by a significant number of filled regions but an absence of animals. These regions were filled with many similar patterns, that the VAE separated in behavior, which the human would not differentiate. This detailing in behavior led us to consider the possibility that the archive was too detailed in some regions, Different sizing of the archive could potentially address this issue.

7.2.7 Using Transfer Learning to Train the VAE

As written in Section 6.5, we discussed the possibility of incorporating transfer learning within our archive. This could possibly shorten the total training time of AML, and lead to a higher percentage of animals within the archive.

7.2.8 Classification of Patterns

One of the more difficult aspects of our thesis was the classification of patterns as dead, non-animal, or animal. As we wrote in Section 6.4, both Reinke et al.[7] and our classification of patterns was not optimal. A general rule for what defines an animal pattern within Lenia Lenia could help in the search for new animal patterns and more diverse non-animal patterns.

7.2.9 Mutation of the Pattern

As discussed in Section 6.1.3, there might be more optimal ways of mutation the pattern, such as a form of crossover mutation.

Bibliography

1. Langton, C. G. Studying artificial life with cellular automata. *Physica D: Nonlinear Phenomena* **22,** 120–149 (1986).
2. Bedau, M. A. *et al.* Open problems in artificial life. *Artificial life* **6,** 363–376 (2000).
3. Sayama, H. Swarm chemistry. *Artificial life* **15,** 105–114 (2009).
4. Gardner, M. MATHEMATICAL GAMES. *Scientific American* **223,** 120–123. ISSN: 00368733, 19467087. http://www.jstor.org/stable/24927642 (2023) (1970).
5. Sims, K. Evolving 3D morphology and behavior by competition. *Artificial life* **1,** 353–372 (1994).
6. Chan, B. W.-C. Lenia-biology of artificial life. *arXiv preprint arXiv:1812.05433* (2018).
7. Reinke, C., Etcheverry, M. & Oudeyer, P.-Y. Intrinsically motivated discovery of diverse patterns in self-organizing systems. *arXiv preprint arXiv:1908.06663* (2019).
8. Cully, A. *Autonomous skill discovery with quality-diversity and unsupervised descriptors* in *Proceedings of the Genetic and Evolutionary Computation Conference* (2019), 81–89.
9. Wolfram, S. Statistical mechanics of cellular automata. *Reviews of modern physics* **55,** 601 (1983).
10. Chopard, B. & Droz, M. Cellular automata. *Modelling of Physical* (1998).
11. Adamatzky, A. *Game of life cellular automata* (Springer, 2010).
12. Rendell, P. & Rendell, P. Game of Life Universal Turing Machine. *Turing Machine Universality of the Game of Life,* 71–89 (2016).
13. Kari, J. Theory of cellular automata: A survey. *Theoretical computer science* **334,** 3–33 (2005).
14. Ermentrout, G. B. & Edelstein-Keshet, L. Cellular automata approaches to biological modeling. *Journal of theoretical Biology* **160,** 97–133 (1993).
15. Kier, L. B., Seybold, P. G. & Cheng, C.-K. *Modeling chemical systems using cellular automata* (Springer Science & Business Media, 2005).
16. Vichniac, G. Y. Simulating physics with cellular automata. *Physica D: Nonlinear Phenomena* **10,** 96–116 (1984).
17. Coombes, S. The geometry and pigmentation of seashells. *Nottingham: Department of Mathematical Sciences, University of Nottingham* (2009).

18. Davis, Q. T. & Bongard, J. Selecting Continuous Life-Like Cellular Automata for Halting Unpredictability: Evolving for Abiogenesis. *arXiv preprint arXiv:2204.07541* (2022).

19. Chan, B. W.-C. Lenia and expanded universe. *arXiv preprint arXiv:2005.03742* (2020).

20. Reinke, C., Etcheverry, M. & Oudeyer, P.-Y. Intrinsically motivated exploration for automated discovery of patterns in morphogenetic systems. *arXiv preprint arXiv:1908.06663* (2019).

21. Etcheverry, M., Moulin-Frier, C. & Oudeyer, P.-Y. Hierarchically organized latent modules for exploratory search in morphogenetic systems. *Advances in Neural Information Processing Systems* **33,** 4846–4859 (2020).

22. Plantec, E. *et al.* Flow Lenia: Mass conservation for the study of virtual creatures in continuous cellular automata. *arXiv preprint arXiv:2212.07906* (2022).

23. Neri, F., Cotta, C. & Moscato, P. *Handbook of memetic algorithms* (Springer, 2011).

24. Eiben, A. E. & Smith, J. E. *Introduction to evolutionary computing* (Springer, 2015).

25. Baresel, A., Sthamer, H. & Schmidt, M. *Fitness function design to improve evolutionary structural testing* in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation* (2002), 1329–1336.

26. Hinton, G. E., Krizhevsky, A. & Wang, S. D. *Transforming auto-encoders* in *International conference on artificial neural networks* (2011), 44–51.

27. Géron, A. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems* (" O'Reilly Media, Inc.", 2019).

28. Liou, C.-Y., Cheng, W.-C., Liou, J.-W. & Liou, D.-R. Autoencoder for words. *Neurocomputing* **139,** 84–96 (2014).

29. Kingma, D. P. & Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

30. Zhang, A., Lipton, Z. C., Li, M. & Smola, A. J. Dive into deep learning. *arXiv preprint arXiv:2106.11342* (2021).

31. Lehman, J. & Stanley, K. O. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation* **19,** 189–223 (2011).

32. Lehman, J. & Stanley, K. O. *Evolving a diversity of virtual creatures through novelty search and local competition* in *Proceedings of the 13th annual conference on Genetic and evolutionary computation* (2011), 211–218.

33. Mouret, J.-B. & Clune, J. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909* (2015).

34. Kim, S., Coninx, A. & Doncieux, S. From exploration to control: learning object manipulation skills through novelty search and local adaptation. *Robotics and Autonomous Systems* **136,** 103710 (2021).

35. Flageat, M. *et al.* Benchmarking Quality-Diversity Algorithms on Neuroevolution for Reinforcement Learning. *arXiv preprint arXiv:2211.02193* (2022).

36. Salehi, A. & Doncieux, S. Towards QD-suite: developing a set of benchmarks for Quality-Diversity algorithms. *arXiv preprint arXiv:2205.03207* (2022).

37. Lehman, J., Stanley, K. O. *et al. Exploiting open-endedness to solve problems through the search for novelty.* in *ALIFE* (2008), 329–336.

38. Cully, A., Clune, J., Tarapore, D. & Mouret, J.-B. Robots that can adapt like animals. *Nature* **521,** 503–507 (2015).

39. Alvarez, A., Dahlskog, S., Font, J. & Togelius, J. *Empowering quality diversity in dungeon design with interactive constrained map-elites* in *2019 IEEE Conference on Games (CoG)* (2019), 1–8.

40. Dolson, E., Lalejini, A. & Ofria, C. in *Genetic Programming Theory and Practice XVI* 1–16 (Springer, 2019).

41. Du, Q., Emelianenko, M. & Ju, L. Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations. *SIAM journal on numerical analysis* **44,** 102–119 (2006).

42. Aurenhammer, F. & Klein, R. Voronoi Diagrams. *Handbook of computational geometry* **5,** 201–290 (2000).

43. Vassiliades, V., Chatzilygeroudis, K. & Mouret, J.-B. Using centroidal voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm. *IEEE Transactions on Evolutionary Computation* **22,** 623–630 (2017).

44. Du, Q., Faber, V. & Gunzburger, M. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM review* **41,** 637–676 (1999).

45. Mouret, J.-B. & Doncieux, S. Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evolutionary computation* **20,** 91–133 (2012).

46. Wolfram, S. Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena* **10,** 1–35 (1984).

47. Packard, N. H. & Wolfram, S. Two-dimensional cellular automata. *Journal of Statistical physics* **38,** 901–946 (1985).

48. icvassbo. *master-thesis-code* version 1.0.0. May 2023. https://github.com/icvassbo/master-thesis-code.

49. *Automated Discovery of Patterns in Lenia* original-date: 2019-08-04T08:56:35Z. 19th Apr. 2023. https://github.com/flowersteam/automated_discovery_of_lenia_patterns (2023).

50. *Automated Discovery of Patterns in Lenia* original-date: 2019-08-04T08:56:35Z. 19th Apr. 2023. https://github.com/flowersteam/automated_discovery_of_lenia_patterns/blob/97cc7cde2120fa95225d1e470e00b8aa8c034e97/autodisc/autodisc/representations/static/pytorchnnrepresentation/pytorchnnarchitectures.py (2023).

51. Chan, B. *Visit Lenia portal for more information* original-date: 2018-02-25T05:51:46Z. 21st Apr. 2023. https://github.com/Chakazul/Lenia (2023).

52. *Lenia Tutorial* original-date: 2021-07-15T16:38:31Z. 10th Mar. 2023. https://github.com/OpenLenia/Lenia-Tutorial (2023).

53. Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M. & Gagné, C. DEAP: Evolutionary Algorithms Made Easy. *Journal of Machine Learning Research* **13,** 2171–2175 (July 2012).

54. *Python3 Map-Elites* original-date: 2019-04-12T08:27:07Z. 28th Apr. 2023. https://github.com/resibots/pymap_elites (2023).

55. Vassiliades, V. & Mouret, J.-B. *Discovering the elite hypervolume by leveraging interspecies correlation* in *Proceedings of the Genetic and Evolutionary Computation Conference* (2018), 149–156.

56. Mouret, J.-B. & Maguire, G. *Quality diversity for multi-task optimization* in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (2020), 121–129.

57. Lehman, J. *et al.* The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. *arXiv preprint arXiv:1803.03453* (2018).

58. Ribani, R. & Marengoni, M. *A survey of transfer learning for convolutional neural networks* in *2019 32nd SIBGRAPI conference on graphics, patterns and images tutorials (SIBGRAPI-T)* (2019), 47–57.

Appendix A

Figures

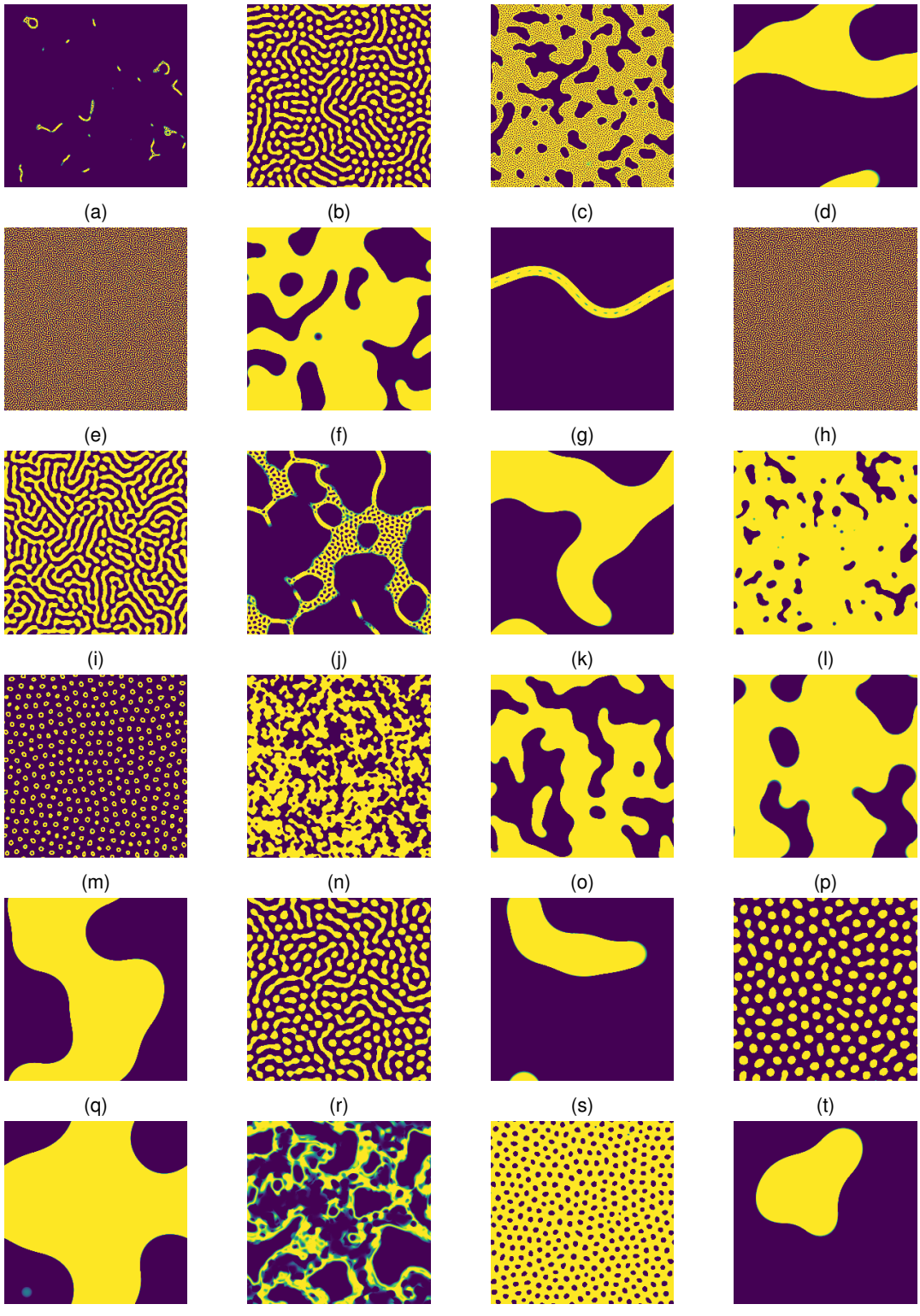


Figure A.1: Archive 1 patterns

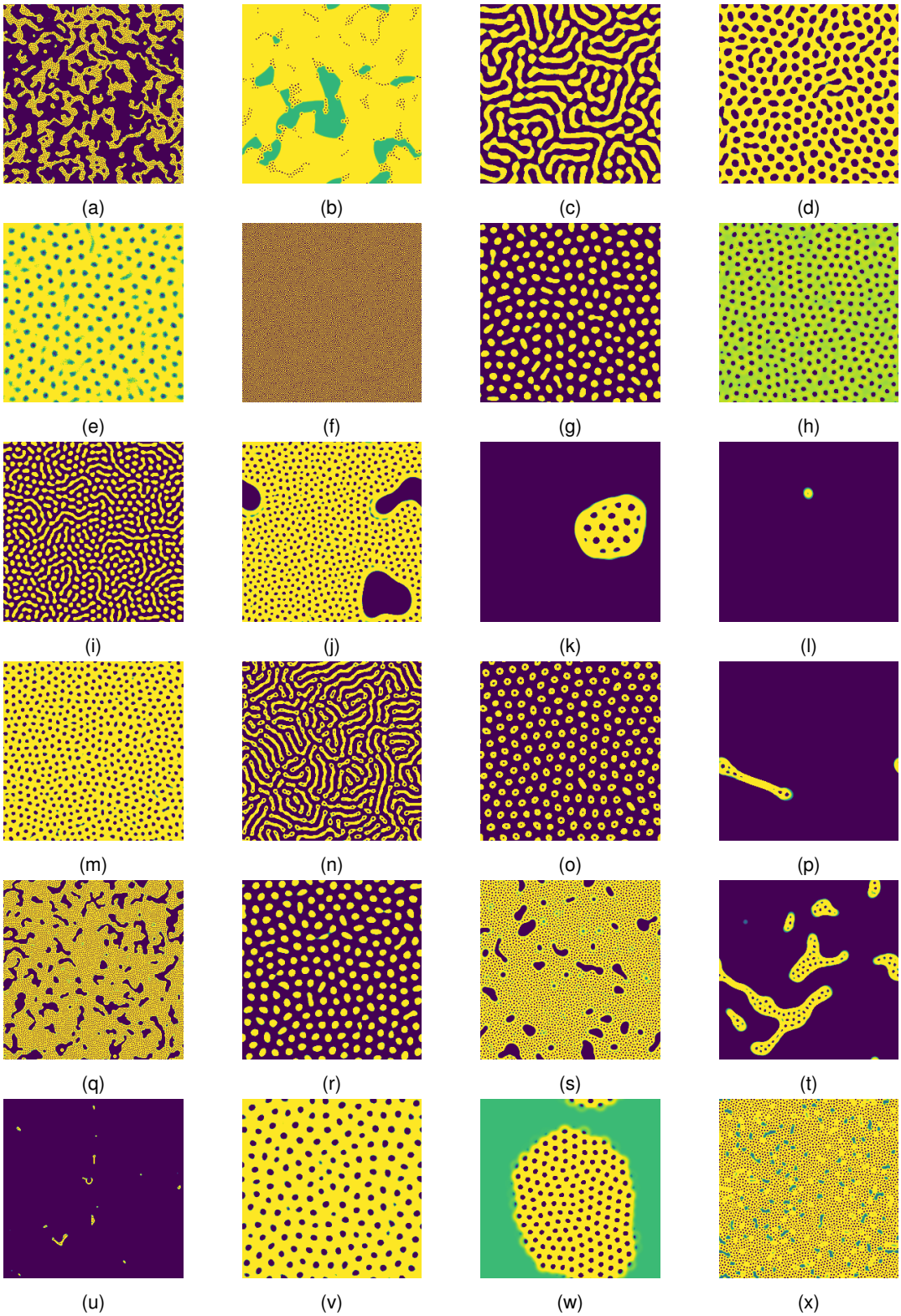


Figure A.2: Archive 2 patterns

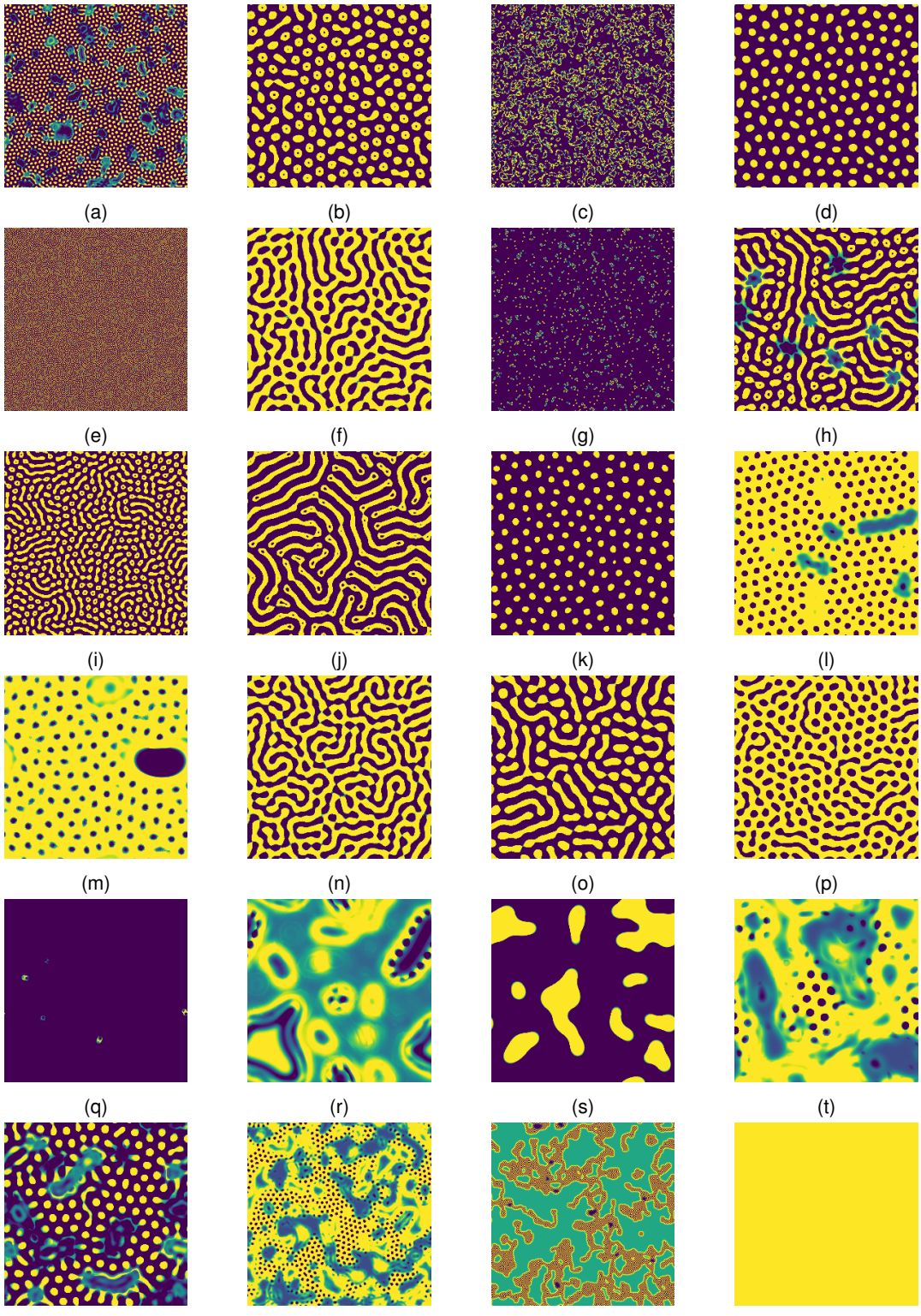


Figure A.3: Archive 3 patterns

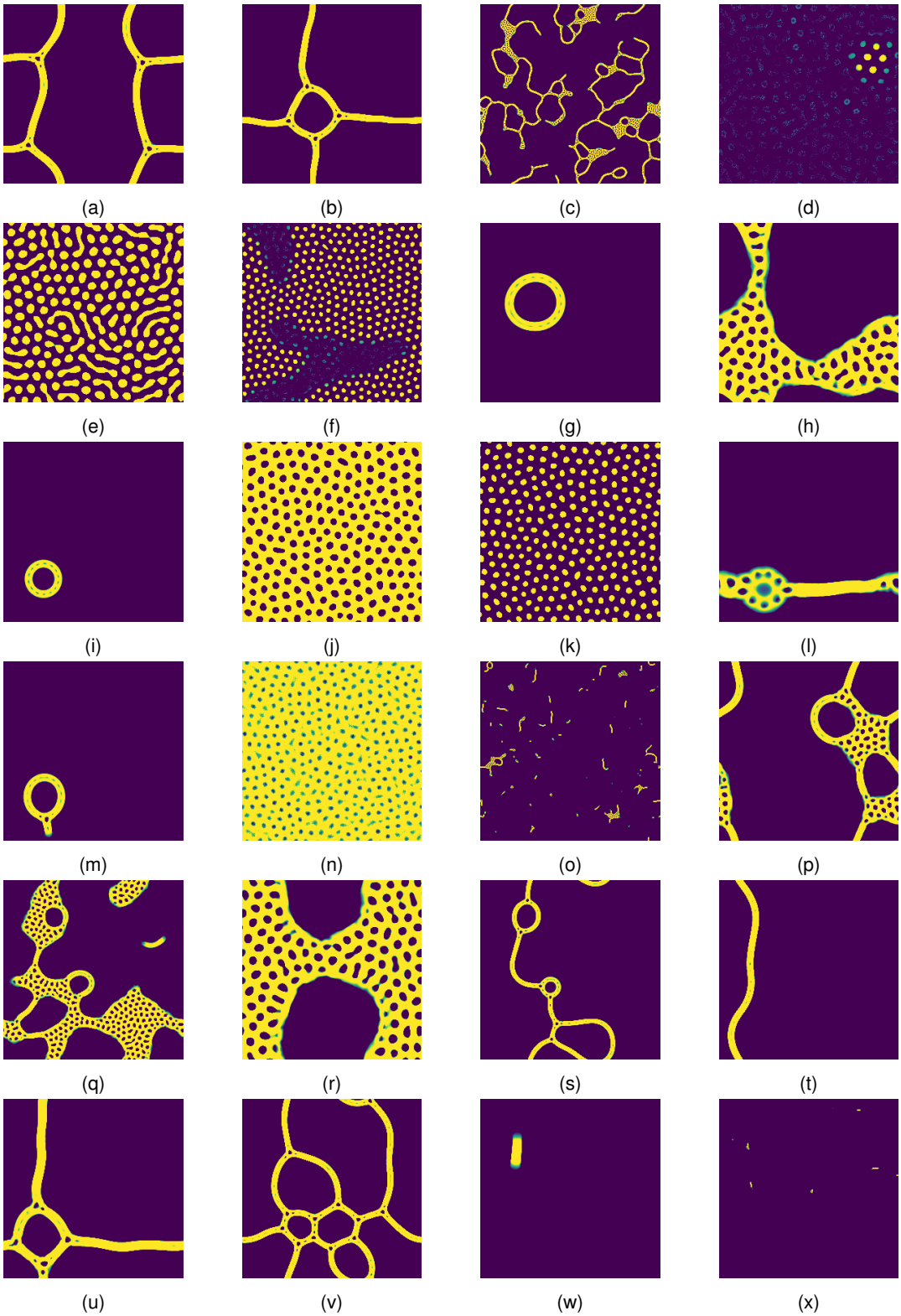


Figure A.4: Archive 4 patterns