IMPERIAL

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

# Discovering Diverse Artificial Life with Quality Diversity

*Author:*
Aidan Holmes

*Supervisor:*
Dr. Antoine Cully

Submitted in partial fulfillment of the requirements for the MSc degree in Artificial Intelligence of Imperial College London

September 2024

**Abstract**

This study enhances the Leniabreeder framework, a system for evolving diverse artificial life forms within the cellular automaton Lenia using Quality-Diversity algorithms. We investigate several approaches to increase creature diversity and complexity, including modifications to starting patterns, variation operators, and different VAE architectures for the AURORA QD algorithm.

Contrary to expectations, contrastive learning methods and Group Convolutional Neural Networks decrease repertoire diversity, revealing complex relationships between geometric invariance and evolutionary diversity. We introduce Lenia-specific mutation operators that are found to substantially increase creature diversity. Repertoire diversity is verified through new evaluation tools including interactive t-SNE visualizations, rotational diversity metrics and other qualitative methods. An automated discovery framework using extinction events and dynamic configurations enables the evolution of complex, self-organizing creatures previously difficult to obtain. The enhanced Leniabreeder framework demonstrates improved capabilities for generating and analyzing diverse artificial life forms, and represents a step towards open-ended creature discovery in Lenia.

## Acknowledgments

I extend my thanks to Dr. Antoine Cully for his guidance and insightful supervision throughout this project. Thanks also go to Maxence Faldor for his thoughtful feedback and creative suggestions, which significantly contributed to the depth and breadth of this research. Their combined expertise and support were instrumental in shaping the direction and success of this work.

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

To study emergent phenomena, such as complex self-organising patterns arising in rule based systems, we can use computational methods. An example method is cellular automata (CA), created by John von Neumann [1] and popularised by Gonway's Game of Life (GoL) [2]. While GoL is simple automaton, with its states determined by only four rules, systems of incredible complexity can arise in the form of self-organising patterns. More sophisticated automatons have been since created. Some examples of these include SmoothLife [3] and Lenia [4]. The patterns that emerge in these systems show certain properties, such as homeostasis [5] and reproduction, that are very alike what is seen in biological life.

When we discuss diversity in this paper, we specifically refer to the difference in the observed creatures in a repertoire. A repertoire refers to our collection of creatures. We want as many creatures of possible in a given repertoire to be unique.

CA patterns (which in the context of this paper may be referred to as solitons or creatures) are often found through manual or random search [4], though some recent works have explored the use of more sophisticated evolutionary methods [6] [7] [8]. The use of random and manual search is inefficient, and historically acted as a bottleneck in the discovery of new species [4]. Even when more sophisticated methods have been used, they often sacrifice creature diversity for optimality, we aim to prioritize diversity to understand if these systems can exhibit open-ended complexity in their patterns.

The aim of this research is to use a family of evolutionary algorithms, known as Quality-Diversity or QD algorithms, to efficiently discover a large host of dynamically stable and visually diverse creatures.

We specifically target the cellular automaton Lenia. It is unknown how many creatures can exist within Lenia, and currently automated discovery can be a challenging and time-consuming task. One of the "Five Grand Challenges of Lenia" is to be able to automatically discover Lenia creatures in an automated and efficient way [9]. This project aims to enhance a method already designed for this objective, Leniabreeder, by both improving the diversity of discovered creatures in a given repertoire of creatures, and introducing new ways to efficiently navigate the search space.

## 1.2 Project Goals

This research extends the work of Faldor and Cully's *Leniabreeder* [7], focusing on improving the diversity and complexity of artificial life forms evolved within the Lenia cellular automaton. The primary objective is to address one of the key challenges in Lenia research: the efficient and automated discovery of stable and diverse artificial creatures.

To achieve this, the project sets out to explore and implement several key approaches. First, it aims to investigate the impact of different starting patterns on evolutionary trajectories, seeking to understand how initial conditions influence the diversity of evolved creatures. The project also intends to understand the Variational Autoencoder (VAE) architecture used in AURORA, by experimenting with various hidden layer dimensions to improve both reconstruction quality and latent space diversity. We then further explore how the iso+lineDD [10] variation operator can be exploited to increase creature diversity. The research for this section will be conducted by tuning the original Leniabreeder work by Faldor and Cully, without making architectural changes.

A significant focus of the research is on developing methods in an attempt to achieve geometric invariance in creature representations, particularly rotational invariance. This involved implementing and evaluating contrastive learning approaches, such as NT-Xent and Triplet Margin Loss, as well as exploring group theory-based techniques like Group Convolutional Neural Networks (G-CNNs).

The project also aims to introduce novel, Lenia-specific mutation operators designed to enhance creature diversity. These include the Kernel Parameter Interpolation, Kernel Parameter Shift, and RGB Channel Swap mutations, with the goal of discovering new evolutionary pathways and expanding the range of possible life forms.

Another key objective was to develop enhanced evaluation tools and metrics for assessing creature diversity. This includes creating interactive visualization techniques and implementing new quantitative measures of diversity that account for rotational and temporal differences.

Finally, the project seeks to extend Leniabreeder's creature search capabilities by implementing an automated discovery process that can continuously generate new creatures through simulated extinction events and iterative evolution. This approach aims to explore a broader range of the Lenia morphospace and potentially discover entirely novel forms of artificial life.

Through these objectives, the project ultimately aims to advance our understanding of open-ended evolution in artificial life systems and contribute to the broader field of artificial life research.

## 1.3 Contributions

As the codebase was inherited from work done by Faldor and Cully, it is important to make clear the contributions we made through this project.

- We perform extended testing of the original Leniabreeder implementation, focused on the effects of hyperparameter tuning. We show that through certain tuning, we can induce a lot of diversity without major architectural changes.

- New starting creatures are introduced, facilitating the discovery of novel evolutionary lineages. These diverse initial forms evolve to occupy distinct ecological niches with varying efficiency and diversification levels.

- We develop novel evaluation metrics to quantify average pairwise diversity and self-diversity across orientations in creature repertoires. This enables large-scale analysis and cross-run comparisons without manual visualization of large archive subsections.

- We enhance Leniabreeder's visualization toolkit with interactive latent space visualizations, creature trajectory tracking, and click-to-view functionality. We also introduce latent space interpolation visualizations, both for entire repertoires and single creature trajectory.

- We implement two contrastive learning AURORA architectures in an attempt to generate more diverse creatures through rotation-invariant embeddings, addressing geometrically shifted creature duplication. We analyze their hyperparameters and evaluate their artificial life generation capabilities. We show this to actually be detrimental to the evolutionary process.

- We introduce AURORA variants using JAX-implemented G-CNN layers, again to try achieve rotational invariance. These are applied to both standard and contrastive AURORA architectures and evaluated against conventional benchmarks. Similarly, we show that this kind of constraint on the VAE does not allow for increased creature diversity.

- We introduce a VQ-VAE AURORA architecture to try and leverage discrete latent spaces for meaningful and diverse embeddings. We conduct tuning studies and evaluate its effectiveness in generating artificial life.

- We introduce four novel mutation operators tailored for Lenia, targeting specific genotype parameters and colour channels. These demonstrate superior efficacy in generating diverse artificial life repertoires. The fourth works by randomly choosing one of the other three at each generation, allowing us to reap the rewards of the others simultaneously

- We implement an automatic discovery process for Leniabreeder, iteratively introducing extinction events and then starting from new seeds with varying configurations. This allows for mass automatic discovery of new artificial life forms by removing evolutionary pressures.

We also include a markdown file[1] in the repository that makes clear what technical contributions/changes have been made as part of this project.

[1]The file is called contributions.md

## 1.4 Report Outline

This project explores diverse approaches to generate varied artificial life in Lenia while enhancing its visualization and evaluation toolkit. The report begins with background on Artificial Life and Quality Diversity, followed by an overview of the inherited Leniabreeder framework. Novel research using the existing tool demonstrates how different starting creatures evolve distinct evolutionary chains and how more aggressive variation operators significantly increase creature diversity.

The report then introduces new implementations to Leniabreeder, starting with toolkit additions for evaluating and visualizing generated repertoires. Four key approaches to generating diverse artificial life are presented. These include contrastive learning and group theory-based methods to encourage geometric invariance in embeddings, VQ-VAE implementation to leverage discrete latent spaces for more distinct feature capture, and novel Lenia-specific QD mutation operators that work directly on creature genotypes. These approaches aim to address the challenge of duplicate creatures in Leniabreeder, promote diversity in the latent space, and explore a wider range of distinct life forms. The report evaluates each method's effectiveness in generating diverse and interesting Lenia creatures. We finish by performing a critical evaluation of the project and offer further research areas for Leniabreeder, some of which we provide technical implementations towards.

## 1.5 Ethical Considerations

An ethics checklist is provided in the appendix. While typical computer science ethical issues like data misuse and privacy are not directly relevant here, exploring the nature of life and intelligence through systems like Lenia does raise some philosophical and ethical questions. If such systems exhibit open-ended evolution leading to emergent intelligence, we must consider the point ethical respect becomes warranted. Though current research has not reached that stage, proactively considering these issues is still interesting food for thought. Based on the research of this project, especially when considering the difficulty of evolving creatures beyind a certain size, we do not consider this to be currently an issue.[2]

[2]The Imperial DoC ethic checklist can be found completed at the following link: `https://docs.google.com/spreadsheets/d/1uwe8u6h8MSJaKWdpVZvE-yR88YqdQhKb/edit?usp=sharing&ouid=105467175375538712613&rtpof=true&sd=true`.

# Chapter 2

# Background

## 2.1 Artificial Life

### 2.1.1 Introduction to ALife

The emergence of life from inanimate matter and its apparent open-ended evolution has long intrigued researchers [11]. Advancements in computational technology, such as cellular automata, now enable modeling of these emergent properties within the field of Artificial Life (ALife). ALife focuses on developing computational models exhibiting lifelike properties such as self-organization, evolution, and adaptation [12]. By abstracting essential principles of living systems, researchers aim to understand the emergence of complexity and evolutionary dynamics.

Another significant focus of ALife is open-ended evolution, which, unlike the constrained evolution of genetic algorithms, aims to continuously generate novel forms and behaviors. This approach seeks to capture the creative potential of natural evolution, providing insights into the mechanisms driving perpetual innovation in biological systems. Recent work has made promising progress in realizing open-ended evolution [7][13][14].

### 2.1.2 Cellular Automaton Approaches to ALife

Cellular automata have long been been a foundational concept and widely utilized tool in the field of Artificial Life [15]. A cellular automaton is a computational model consisting of a grid of cells, each existing in a state determined by the model's dynamics. These dynamics typically involve updating cells based on the values of their neighbours [1].

By designing cellular automata with carefully crafted rules, researchers have aimed to create computational universes where novel forms and behaviors can continually emerge, mimicking the perpetual innovation and increasing complexity observed in natural evolutionary processes. These potentially open-ended cellular automata systems serve as virtual laboratories for studying the fundamental mechanisms driving the emergence of complexity and the potential for ongoing adaptation and diversification.

## 2.2 Lenia

Lenia, introduced by Bert Chan [4], represents a significant advancement in cellular automata, extending Conway's Game of Life into a continuous space-time-state system. Lenia incorporates multiple dimensions, kernels, and channels, providing a rich environment for exploring artificial life and complex, self-organizing patterns.

Expanded Lenia, developed by Chan in 2020 [16], further enhances the original concept. In this framework, the world state is represented by a $d$-dimensional lattice $\mathbf{A}^t$ with $c$ channels of real values between 0 and 1. This continuous state space allows for nuanced transitions and interactions, far surpassing the binary states of traditional cellular automata. We can see how pattern diversity has progressed from the Game of Life to smoothlife, an intermediate continuous CA [3], to Lenia in the Figure 2.1.

**Figure 2.1:** Evolution of gliders: Game of Life (left), SmoothLife (center), and Lenia (right), demonstrating increasing complexity.

At the heart of Expanded Lenia lies a complex update rule governed by multiple kernels. Each kernel $\mathbf{K}_k$ is characterized by a relative radius $r_k R$, a parameter $\beta_k \in [0,1]^B$, and a growth mapping $G_k$ with parameters $\mu_k$ and $\sigma_k$. The state evolution is described by the equation [16]:

$$\mathbf{A}_j^{t+\Delta t} = [\mathbf{A}j^t + \Delta t \sum_{i,k} \frac{h_k}{h} G_k(\mathbf{K}_k * \mathbf{A}_i^t)]_0^1$$

Here, $\mathbf{A}_j^t$ represents the state of channel $j$ at time $t$, $\Delta t$ is the time step, $h_k$ are weighting factors, $*$ denotes convolution, and $[\cdot]_0^1$ indicates clipping to the range $[0,1]$. This update mechanism allows for complex interactions between multiple channels, significantly increasing the potential for diverse and intricate pattern formation.

The construction of each kernel $\mathbf{K}_k$ involves combining an exponential core with a shell, defined by parameters $\phi_k$. The growth mapping $G_k : [0,1] \to [-1,1]$ is defined for this work as [7]:

$$G_k(u) = 2\exp\left(-\frac{(u-\mu_k)^2}{2\sigma_k^2}\right) - 1$$

The complexity of Expanded Lenia presents unique challenges in discovering stable, interesting patterns. Recent research has focused on advanced computational techniques to address these challenges. Quality-Diversity algorithms have shown promise in efficiently discovering diverse sets of high-performing solutions, overcoming the limitations of traditional search methods in Lenia's vast parameter space [7] [8] [17]. Interactive Evolutionary Computation has revealed a wide array of lifelike, self-organizing autonomous patterns, demonstrating

Lenia's potential for modeling complex biological phenomena.

Furthermore, recent work by Chan [18] explores Lenia's capacity for open-ended evolution, investigating its potential for continuous, unpredictable pattern generation and evolution. This research direction holds particular promise for understanding the fundamental principles of complex adaptive systems and the emergence of life-like behaviors.

## 2.3 Quality Diversity

### 2.3.1 Motivation

In optimization problems, the pursuit of singular optimal solutions often neglects the importance of exploring diverse solution spaces. Quality Diversity algorithms address this issue by emphasizing the simultaneous maximization of solution quality and diversity [19].

The motivation behind QD algorithms lies in recognizing that, in many real-world scenarios, a single optimal solution may be insufficient to address the complexity and variability of the problem. QD algorithms offer a systematic approach to exploring a broad range of high-quality solutions, thereby mitigating the risk of premature convergence and fostering the discovery of novel alternatives [20].

### 2.3.2 Procedure

Quality diversity algorithms typically consist of the following stages:

1. Search Space and Objectives
    - **Quality/Fitness Function:** We define a performance or fitness measure to evaluate solutions.
    - **Search Space:** We then Identify the dimensions of the search space and those along which fitness is measured.
2. Archive/Container Initialization
    - **Archive Structure:** An archive, also called a repertoire, is initialised where each cell represents a unique combination of descriptors.
    - **Initial Population:** We generate a random initial population of solutions, evaluate them, and assign them to the appropriate cells in the archive based on their descriptors.
3. Evaluate Initial Solutions
    - **Fitness/Diversity Evaluation:** We calculate the fitness score and descriptor space position for each solution in the archive and assign each solution to the corresponding cell defined by the solution descriptors.
    - **Handle Overlapping Solutions:** We retain only the highest fitness solution in cases where multiple solutions are assigned to the same cell.
4. Iterative Improvement

- **Selection:** We select parent solutions from the archive, either randomly or based on criteria such as fitness and diversity.
- **Variation:** Variation operators (e.g., mutation, crossover) are applied to generate offspring.
- **Evaluation:** We evaluate the quality and diversity of the offspring.
- **Archiving:** We place offspring into appropriate cells:
  - **Insertion:** Offspring are added to the archive if the cell is empty.
  - **Competition:** We replace the existing solution in the cell if the offspring has higher quality.

5. Repeat Until Completion

   - The iterative improvement steps are repeated until a predefined quality/diversity level is reached or for a set number of iterations.

In the context of Lenia, we can use properties of artificial creatures, such as their velocity or mass, as our measure of quality, or niche. We can then evolve new creatures to better fulfil the chosen niche, and in doing so we expect diverse differences in morphology.

### 2.3.3 MAP-Elites

MAP-Elites (Multi-dimensional Archive of Phenotypic Elites) is a supervised Quality-Diversity algorithm that uses a grid-based archive defined by feature dimensions [21]. It stores the best-performing (elite) solutions in each grid cell, enabling exploration and preservation of diverse, high-quality solutions. This process is visualised in Figure 2.2.

Compared to algorithms like Novelty-Search+Local Competition, MAP-Elites' grid representation facilitates better visualization and understanding of solution distribution across the feature space [21]. The grid's dimensions and resolution can be specified, allowing precise control over solution diversity. MAP-Elites will not be a focus of this project, but it is implemented within Leniabreeder and as such has a place in this projects contextual background.

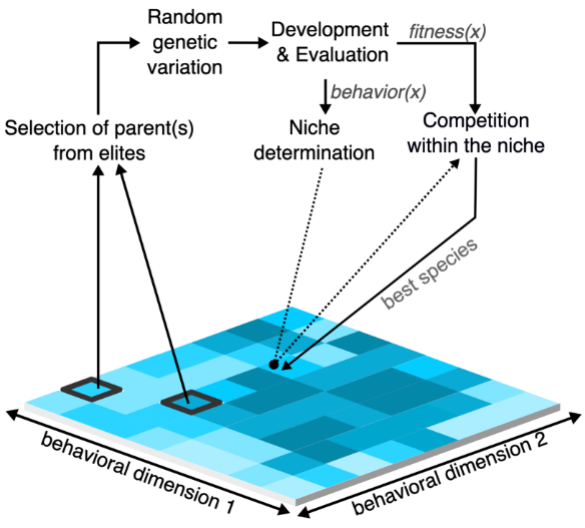


**Figure 2.2:** A visualisation of a 2x2 MAP-Elites grid's update procedure. [21]

### 2.3.4 AURORA

AURORA (AUtonomous RObots that Realize their Abilities) is an unsupervised Quality-Diversity algorithm combining QD principles with dimensionality reduction techniques [22]. Unlike supervised QD algorithms requiring manual behavioral descriptor specification, AURORA automates this process.

The algorithm generates a collection of solutions to train a dimensionality reduction algorithm, in the case of this project a variational autoencoder. The resulting latent representation serves as behavioral descriptors for the QD algorithm, eliminating manual definitions. This approach has shown comparable diversity to manually defined descriptors [22]. We note that the descriptor space changes over time, as AURORA refines the latent space.

AURORA incorporates sensory data and environmental information to enhance descriptor quality. It follows a standard QD process: random controller selection, mutation, evaluation, and archive placement. The key distinction lies in its automated behavioral descriptor generation using system sensory data.

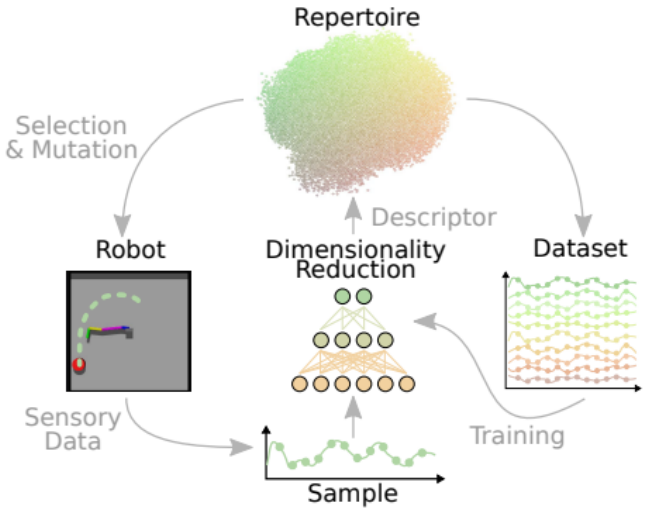


**Figure 2.3:** Diagram of the AURORA processs in the context of robot learning [22].

By automating descriptor definition and leveraging dimensionality reduction, AURORA streamlines the QD process, increases efficiency, and facilitates exploration of complex solution spaces. This approach can lead to the discovery of novel and diverse solutions, particularly in systems where manual descriptor definition is challenging or time consuming [22].

## 2.4 JAX

We use JAX for this project. JAX is a high-performance numerical computing library developed by Google Research, combining NumPy's familiar API with hardware acceleration and automatic differentiation [23]. It excels in machine learning and scientific computing tasks due to its key features. JAX employs Just-in-Time (JIT) compilation to optimize Python code for faster execution on CPUs and GPUs. Its automatic differentiation capabilities compute

derivatives of complex functions, crucial for gradient-based optimization in machine learning. JAX's efficient vectorization applies operations to arrays, enabling parallelization and improved performance. With seamless GPU and TPU support, JAX leverages hardware acceleration effortlessly. In our project, we harness these capabilities to implement and train our machine learning models efficiently, focusing on high-level algorithm design while JAX handles low-level performance optimizations.

## 2.5 Related Work

### 2.5.1 Automated Discovery of ALife

#### Leniax

Leniax, created by Morgan Giraud [8], is a library intended for large-scale Lenia simulations and creature searches that utilises JAX. Installable as a Python package, Leniax comes pre-built with the functionality to perform QD based evolution of creatures within Lenia. Furthermore, it is able to utilise GPU and TPU hardware, as a result of its underlying JAX framework. It is, however, more challenging to find definitive outcomes of this work, as it was made as a method to mint NFTs and there is limited discussion of it in academic research.

#### EvoJax and Lenia

EvoJax, developed by Tang et al. [6], is a scalable, general-purpose, hardware-accelerated neuroevolution toolkit. Built on top of the JAX library, EvoJax enables neuroevolution algorithms to work with neural networks running in parallel across multiple TPU/GPUs. The toolkit achieves high performance by implementing the evolution algorithm, neural network, and task all in NumPy, which is then compiled just-in-time using JAX to run on accelerators.

The LeniaEvoJAX example utilizes hardware acceleration to evolve complex cellular automata patterns. EvoJAX efficiently evolves Lenia creatures by running high-performance simulations in parallel on GPUs/TPUs. The integration of JAX allows for large-scale evolution, making it possible to explore diverse behaviors in Lenia's lifeforms. Through this setup, EvoJAX performs rapid neuroevolution cycles, enabling the automated discovery of unique, stable patterns and dynamic behaviors in Lenia. This implementation was not included in the original paper[6], and was added at a later date added by Bert Chan [17]. Some elements of the original Leniabreeder, such as the Lenia simulation and starting patterns, were taken from Chan's Evojax contribution.

#### Leniabreeder

Leniabreeder, similarly to Leniax, is a framework for the efficient discovery of artificial life within Lenia using Quality-Diversity algorithms and JAX [7]. The framework allows for the use of different QD algorithms, such as AURORA and MAP-Elites, to discover new creatures in large-scale simulations.

AURORA was shown to be especially effective at the discovery of new creatures, including those that are clearly different from the original Lenia creature bank. The work also indicates a step towards open-ended evolution within Lenia, as it shows vast repertoires of

diverse creatures can be evolved from a common ancestor [7].

This project aims to extend the Leniabreeder framework by implementing features such as new VAE architectures, evaluation toolkitting, and novel mutation operators that can be used to evolve a diverse host of artificial life. We present a more thorough introduction in the next chapter.

### 2.5.2 Open-Endedness

**An Overview of Open-Ended Evolution**

Open-ended evolution (OEE) directly refers to the continuous generation of novelty and increasing complexity in evolution. In the context of ALife, there are no clear guidelines as to how we can classify a system as being truly open-ended. In *An Overview of Open-Ended Evolution*, Packard et al [24] build on the ideas of the York OEE categories [25], that is that there are multiple different categories for OEE as opposed to a single definition. This paper proposes new categories, the Tokyo OEE categories, which are as follows:

- Interesting new kinds of entities.
- Evolution of evolvabillity.
- Major transitions.
- Semantic evolution.

When referencing OEE in this project, we focus on the first category as our metric, we want to continuously discover interesting new creatures.

# Chapter 3

# Leniabreeder Methods

## 3.1 Introduction

This chapter aims to introduce the methods of Leniabreeder. In particular, we focus on AURORA and the details of the evolutionary process. While MAP-Elites is included within the framework, it was not identified as an area of study for this project. This chapter does not intend to totally repeat the methodology of the original Leniabreeder paper, but rather to cover key aspects of it. For further detail, see *Toward Artificial Open-Ended Evolution within Lenia using Quality-Diversity* [7]. Unless stated otherwise, the aforementioned paper serves as the reference for content of this chapter. ‘

## 3.2 Evolution Process

### 3.2.1 Initialisation

The original Leniabreeder paper employed a specific world configuration for its experiments. This setup consisted of a 2D 128x128 array with 3 channels and utilized 15 kernels. The system's genotypes were defined by two key components: the seed (initial creature state) and the rule parameters. For each kernel, the rule parameters included $\mu_k$, $\sigma_k$, and $h_k$ which represent each kernel's growth function mean, standard deviation, and the weighting of the kernel, respectively. Othdf parameters were kept constant. The initial values for these parameters, as well as the starting cell states, were derived from an existing pattern known as “Aquarium”. We use a small repertoire size, at 256 and perform our evolution over a shorter timescale compared to the original work. This was done to improve computational efficiency.

### 3.2.2 Variation Operator

Genetic algorithms employ variation and mutation operators to generate new potential solutions. In the original Leniabreeder implementation, the variation operator used was iso+LineDD [10], which combines isotropic and directional variations to navigate the genotype solution space. Iso+LineDD can be expressed as:

$$x = x_1 + \sigma_1 \mathcal{N}(0, I) + \sigma_2 (x_2 - x_1) \mathcal{N}(0, 1)$$

Where $\sigma_1$ controls the intensity of the isotropic noise, and $\sigma_2$ controls the intensity of the directional noise. The iso+LineDD operator merges random movement in all directions with

targeted movement along the line between two parent solutions. The target movement aims to exploit promising search space. This dual approach enables both broad exploration of the solution space and focused exploitation of parental differences, aiming to enhance the diversity and quality of offspring solutions.

### 3.2.3 Phenotype Constraints

There are constraints on valid genotypes based on their resultant phenotypes. These constraints prohibit the acceptance of phenotypes that vanish, explode, or spread too widely across the Lenia world space into the repertoire. This approach encourages the development of stable creatures, as opposed to dissolving creatures or explosive patterns such as Turing patterns [26], [27].

### 3.2.4 VAE

AURORA employs a Variational Autoencoder (VAE) [28], an unsupervised learning technique that compresses the phenotypes into more manageable low-dimensional descriptors.

#### VAE Structure and Function

A standard VAE architecture is used:

- An encoder that compresses the input data
- A decoder that attempts to reconstruct the original input from the compressed representation

The VAE takes as input a $32 \times 32 \times 3$ slice of the Lenia world state, centered on the pattern's center of mass. This input is then encoded into an 8-dimensional latent vector, which serves as a compact representation of the pattern's key features.

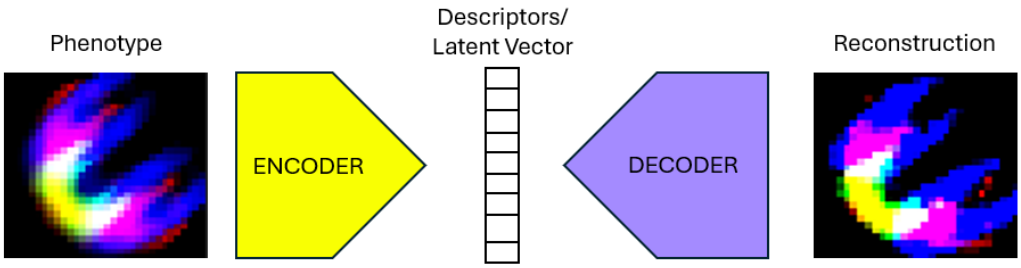


**Figure 3.1:** AURORA uses the dimensionality reduction of the VAE to generate descriptors. By training this VAE, the goal is to improve the quality of these descriptors. Note that this is a particularly poor reconstruction, used for illustrative effect.

We note that parameters, such as the latent vector dimensions, are configurable and experimentation with these parameters will be presented in the next chapter.

### 3.2.5 Evolution

The evolutionary process of Leniabreeder is visualized in Figure 3.2. It begins by selecting creatures from the repertoire and applying the variation operator discussed in section 3.2.1, and the expression of the new genotype yields a new phenotype. The resulting pattern is

then evaluated based on a defined fitness function.

This information is fed into the encoder part of the system, which compresses creature phenotypes into descriptor vectors. The decoder then attempts to reconstruct the original phenotype from this compressed representation. The VAE's encoder-decoder system is trained to improve its ability to represent and generate creatures.

Newly generated creatures compete with existing ones in the population. Successful new creatures are then added to the population, enhancing the diversity and quality of the repertoire.

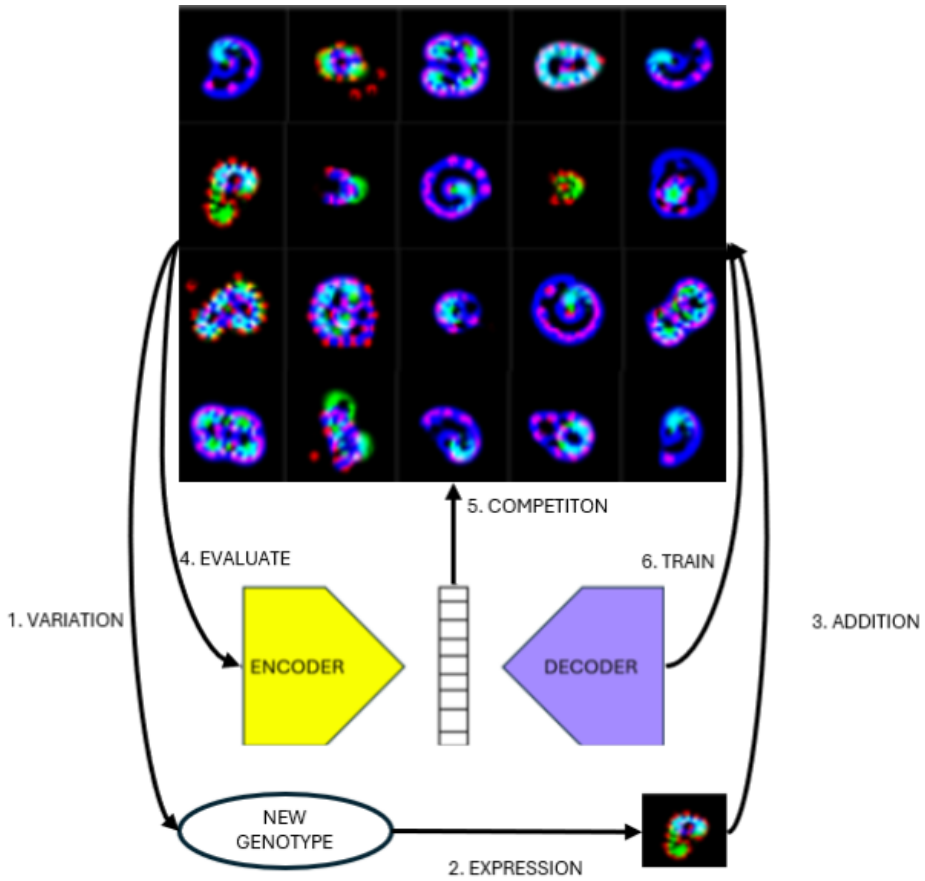


**Figure 3.2:** Leniabreeder process: New creatures are created through the variation operator, with VAE-generated descriptors. Creatures compete based on fitness for archive retention.

Unless specified otherwise, it can be assumed that all experiments presented in this report use a negative angle variance fitness function. This was identified as a promising fitness function for diversity in the original Leniabreeder implementation, and thus was chosen to serve as a benchmark.

A lot of the analysis presented in this report will be qualitative assessment of creatures. We provide here, in Figure 3.3, a large sub-section from repertoire of creatures generated with this standard configuration. These images are the final state of the phenotypes in the repertoire. This aims to provide the reader with a more substantial idea of the diversity of creatures we see when using the standard configuration and fitness function, as it will

make the future qualitative comparisons easier to understand. While there are a collection of different creatures, there is a substantial overlap. Furthermore, we notice a large favouritism for blue creatures, with some yellow, as well as very circular/multi-circular creatures. In our search for diversity, we aim to discover more interesting colourations and more complex structures. All qualitative results generated for this project are random sub-samples of a repertoire.

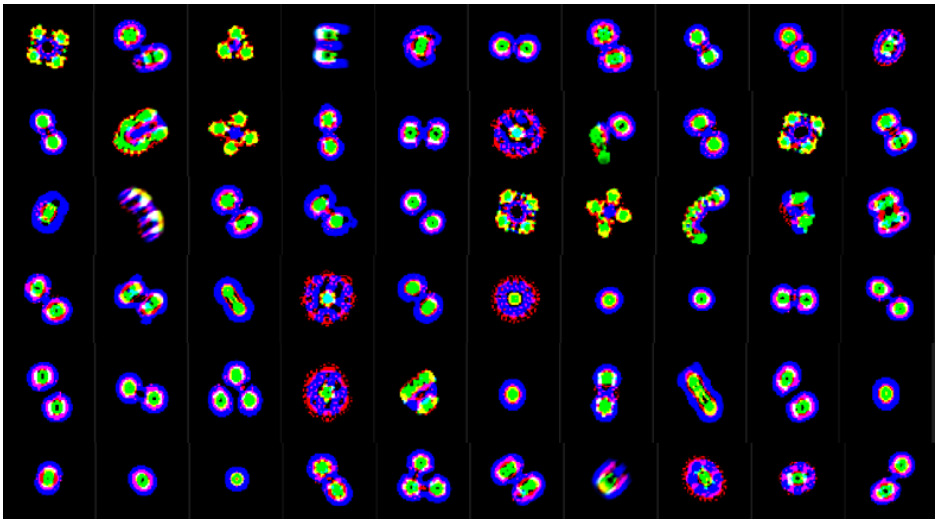

**Figure 3.3:** 60 Creatures randomly samples from a benchmark AURORA run. We do note that the benchmark settings may produce different results (when a different seed is used) that may offer more/less diversity in the repertoire.

## 3.3 Metrics and Evaluation

### 3.3.1 Unsupervised Descriptor and Fitness Calculation

The latent space representation is leveraged to define both the unsupervised descriptor and fitness for each pattern. The mean of the latent vectors over time is our descriptor. This average encapsulates the overall characteristics of the pattern throughout its evolution. The unsupervised fitness function is based on the stability of the pattern in latent space. We calculate it as the negative average distance between each latent vector and the mean vector, or effectively the variance. Patterns that maintain a consistent position in latent space (indicating a type stability) receive higher fitness scores. Note that this is not the only fitness function available in the original Leniabreeder, but it is directly related to the VAE.

### 3.3.2 QD Metrics

Leniabreeder calculates and logs a series of standard QD metrics. These QD metrics are formulated in the Table 3.1. These QD metrics are useful for monitoring the performance of Leniabreeder, and ensuring that new creatures of high fitness are continuously generated at each generation. It does not, however, provide information about the diversity of the solutions, which is a key area of interest. The coverage near always reaches 1 within a few generations, so we will typically have the maximum of solutions our repertoire can store.

| **QD Metric** | **Formula** |
|---|---|
| Coverage | $C_{filled}/C_{total}$ |
| Fitness | $f(x_i)$ |
| QD Score | $\sum_{i=1}^{N} f(x_i)$ |
| Max Fitness | $\max_{i=1}^{N} f(x_i)$ |
| Number of Elites | $N_i^{Elites}$ |

**Table 3.1:** Quality Diversity Metrics and their formulas. $f(x_i)$ is a variable function designated by the user. $C$ represents the number of cells in the repertoire. $N_i^{Elites}$ is the number of elites added at each generation

### 3.3.3 VAE Metrics

Various metrics are logged to monitor the VAE used in the AURORA variant of Leniabreeder.

| **VAE Metric** | **Formula** |
|---|---|
| Loss | $\mathcal{L} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{KL}}$ |
| Latent Variance | $\text{Var}(z) = \frac{1}{N}\sum_{i=1}^{N}(z_i - \bar{z})^2$ |

**Table 3.2:** VAE Metrics and Their Formulas

The VAE uses the standard loss function combination of a reconstruction loss and the Kullback-Leibler Divergence (KLD) [28]. These are formulated in the following two equations.

$$\mathcal{L}_{recon} = -\frac{1}{n}\sum i = 1^{n}[x_i \log(\hat{x}_i) + (1 - x_i)\log(1 - \hat{x}_i)] \tag{3.1}$$

Where $n$ is the number of elements in the input (e.g the cells of the phenotype), $x_i$ is the i-th element of the original input and $\hat{x}_i$ is the i-th element of the reconstructed output.

$$\mathcal{L}_{KLD} = -\frac{1}{2}\sum j = 1^{d}(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2) \tag{3.2}$$

Where $d$ is the dimensionality of the latent space, $\mu_j$ is the j-th element of the mean vector, $\log(\sigma_j^2)$ is the j-th element of the log-variance vector and $\sigma_j^2 = \exp(\log(\sigma_j^2))$ is the actual variance.

The loss value indicates the model's improvement rather than providing information about the generated creatures. Conversely, the variance of embedding vectors can suggest the diversity of creature descriptors, potentially indicating an improvement in repertoire diversity.

However, this metric has limitations due to the lack of interoperability of descriptor vectors. Proper visualization of creatures remains the preferred method for evaluating repertoire diversity. Additionally, this metric cannot be used to compare different VAE architectures in AURORA, as they may employ varying methods or weightings for embedding vectors in the latent space.

### 3.3.4 Fitness Metrics

Leniabreeder used different fitness metrics to guide the evolution of the initial creatures. These fitness metrics utilised statistical measures introduced in Chan's original Lenia paper. See the following table for the mathematical formulation of these measures.

| Statistical measure | Formula |
|---|---|
| Mass | $m = \sum_{x \in C} \sum_i A_i(x)$ |
| Center of mass | $\bar{x} = \frac{1}{m} \sum_{x \in C} x \sum_i A_i(x)$ |
| Velocity | $v = \frac{\Delta \bar{x}}{\Delta t}$ |
| Angle | $\alpha = \arg(v)$ |
| Linear velocity | $V = \lvert v \rvert$ |
| Angular velocity | $\omega = \frac{\Delta \alpha}{\Delta t}$ |
| Colour | $C = \frac{1}{m} \sum_{x \in C} A(x)$ |

**Table 3.3:** Statistical measures and their formulas. $A_i$ represents the Lenia world state at time step $i$.

We note that this colour measure serves not as a fitness function, but as a descriptor for MAP-Elites. The fitness function calculates the positive or negative mean, maximum, or variance for one of these statistics. While multi-fitness optimization is possible, it introduces complexity as users must consider how to weight different fitness metrics

## 3.4 Limitations

A primary limitation of Leniabreeder is its tendency to produce duplicates of the same creature, differing only by geometric shifts such as rotation or shearing. The exact cause was not immediately apparent, but two main hypotheses were proposed.

First, this duplication might result from latent embeddings lacking invariance to geometric shifts. A significant portion of this project's research, detailed in Chapters 6 and 7, addresses this hypothesis by implementing different architectures and data augmentations to teach the model geometric invariances, such as rotation invariance.

Second, the duplication could stem from evolutionary constraints imposed by Lenia dynamics, which might prohibit further evolution with the current variation operator. For instance, if a fitness function encourages large solitons that fill the phenotype space, further evolution becomes impossible as any larger creatures would be discarded. This is a non-trivial issue, as a larger limit on the phenotype space results in multi-creature systems, which is not the target area of study. Larger solitons are also more likely to be unstable, which is undesirable.

Leniabreeder also faces limitations in its evaluation tools, particularly in quantitatively assessing repertoire diversity. While strong visualization tools exist, such as functionality to display creatures based on fitness score or random selection, these methods are time-consuming and labor-intensive. Given that repertoires contain hundreds of creatures, manual diversity assessment becomes inefficient, though it is very effective.

# Chapter 4

# Further Leniabreeder Experimentation

## 4.1 Motivation

This chapter extends the existing research of the original Leniabreeder framework, focusing on uncovering novel evolutionary trees without major architectural changes. Our research also introduces new starting patterns, demonstrating their potential to generate unique evolutionary trajectories. We highlight how fitness functions' effectiveness as evolutionary guides varies among creatures.

Additionally, we present parameter scaling experiments of the original Leniabreeder framework, showing how factors like latent vector size and the variation operator parameters affect evolution and diversity.

These experiments serve three purposes: identifying promising avenues for future research within the broader project scope, elucidating the profound impact of diverse evolutionary patterns on artificial creatures' developmental pathways, and understanding how Leniabreeder parameters can be further exploited for diverse artificial life.

## 4.2 Experimental Method

Like the original work, our world is a 2D 128x128 array with 15 kernels. Our genotypes consist of the same 3072 cells in the 32x32x3 default phenotype slice, as well as the 45 different kernel parameters. We typically perform the Lenia update for 200 steps. We also use a repertoire size of 256 creatures. This is quite a small repertoire size, but it does mean that the process is significantly less computationally expensive, allowing experiments to be run on smaller GPU partitions, such as the 10GB partitions of Imperial's 100GB A100 GPUs on the SLURM cluster. An increased repertoire size would likely increase the number of individually diverse creatures presented (due to there being less of a bottleneck on optimally), however it is fair to extrapolate that if we increase diversity in a large way for a small repertoire, then this would follow on in an scaling experiment. We reason that a larger repertoire will loosen evolutionary pressures, allowing more diverse but lower quality solutions to exist. To reiterate, we prioritise diversity over quality for this project as we wish to discover as many

unique creatures as possible, and frame this in the context of potential open-endedness in Lenia.

A more detailed table of the complete baseline configuration, such as information about VAE learning rates and variation operator hyperparameter can be found in the appendix. We make clear any deviations from this original configuration for any given experiment, and all experiments are repeated multiple times to understand the uncertainties of any result. The number of repetitions depend on the computational cost of a given experiment, but typically we perform 3-5 runs for any configuration change.

This configuration is not only relevant for this chapter, but for the rest of this paper. New architectures will introduce new hyperparameters, and we present information about how these new parameters affect, or do not affect, the evolutionary process.

## 4.3 Initial Patterns

### 4.3.1 Introducing New Patterns

Leniabreeder initializes using pre-existing patterns, or creatures. The original paper exclusively explored the evolutionary trees of the pattern 5N7KKM, known as "Aquarium," using the implementation from Chan's contribution to EvoJax [6]. While EvoJax provides other patterns, they are primarily variations of Aquarium.

More unique patterns suitable for Leniabreeder were, as part of this project, found on Chan's personal GitHub page [29]. From this source, three new starting patterns were implemented for this study:

- 1MTFAY ("Self-Replicator")
- Y3CS55 ("MotherShip")
- G6G6CR ("Bizarre Cell")

We note that these specific patterns were chosen due to both their difference from 5N7KMM and each other. We visualise these creatures in Figure 4.1.

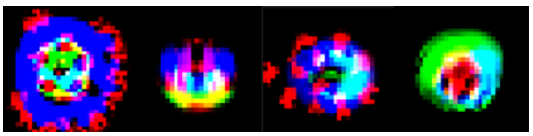

**Figure 4.1:** Creatures left to right: G6G6CR, 5N7KMM, Y3CS55, 1MTFAY.

We present evidence that with different starting creatures it is possible to evolve repertoires that are distinct from one another using the same evolutionary fitness function. See Figure 4.2 comparing a random sample of creatures from 5N7KMM, 1MTFAY, Y3CS55, and G6G6CR all evolved with a fitness function of negative angle variance.

While there is some overlap, we do see differences in the creatures produced using the different starting patterns. The seed 1MTFAY especially tends to evolve creatures that are very different compared to those evolved from others. From these results, we can conclude that different initial creatures will lead to different evolutionary chains. This was exploited in a later part of this project.

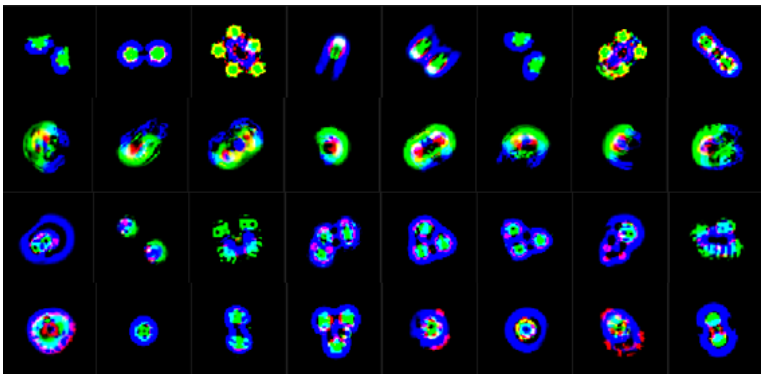

**Figure 4.2:** Rows 1-4: 5N7KMM, 1MTFAY, Y3CS55 and G6G6CR evolved with negative angle variance.

### 4.3.2 Different Creatures Fulfil Niches Differently

We present experiments using the four distinct starting patterns (5N7KMM, 1MTFAY, Y3CS55, and G6G6CR), and show how different fitness metrics can spawn different repertoires. Figures 4.3 and 4.4 illustrate randomly selected creatures evolved from each seed pattern, optimized for positive average linear velocity and positive mass variance respectively.

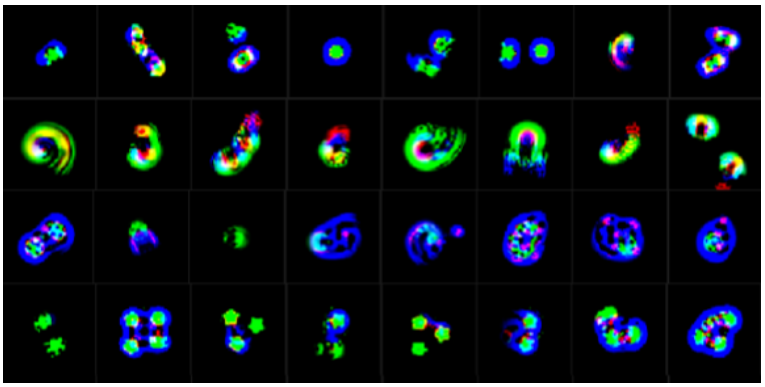

**Figure 4.3:** Rows 1-4: 5N7KMM, 1MTFAY, Y3CS55 and G6G6CR evolved with positive linear velocity.

We see in the previous figure that each initial pattern can produce diverse, and differing, results for each fitness metric. In this report, we can only present snapshots of these creatures. However, the visualisation tools within Leniabreeder do allow the user to illustrate the behaviour of these creatures over time. In doing so, we not only see diversity in the shape of creatures, but also in their behaviours. Being able to visualise the behaviours is majorly beneficial, as creatures can change significantly over the course of just a few Lenia steps, as shown in in Figure 4.5.

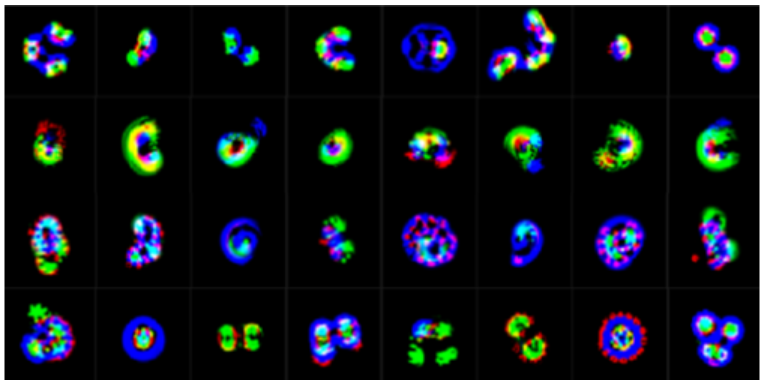

**Figure 4.4:** Rows 1-4: 5N7KMM, 1MTFAY, Y3CS55 and G6G6CR with positive mass variance.

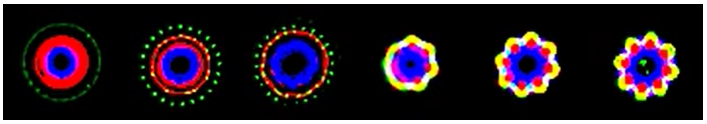

**Figure 4.5:** Two Lenia creatures' evolution over 500 Lenia timesteps, taking on new structures and colouration.

## 4.4 Phenotype Size

In some circumstances, such as when using mass as the fitness, the evolutionary trees can converge onto variations of the same, or very similar, creatures. See Figure 4.6 for an example.

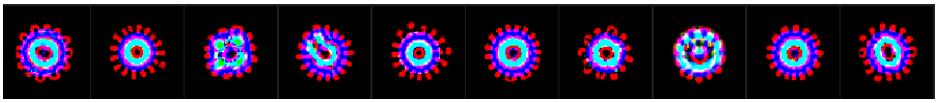

**Figure 4.6:** Creatures evolved from Y3CS55 using positive mass average fitness, with a phenotype size of 32x32.

While there are slight deviations between the different creatures, it is clear that there is a bottleneck on diversity. A suspected reason for this was that the phenotype size, 32x32x3, was too restrictive on certain fitness functions to split into diverse evolutionary trees. By putting such a constraint on fitness functions that favour large creatures, such as mass related metrics, there is not enough room in the phenotype for the creatures to evolve further. To test this, experiments were carried out using a phenotype size of 64x64. The remaining configuration remained the same. See the Figure 4.7 for the new host of creatures.

We see that when the phenotype size is increased, the evolutionary process instead favours multi-creature systems over individual creatures. This is not a desirable outcome, as these multi-creature systems do not represent true self-organisation, and therefore don't constitute true diversity. While it is interesting to see these multi-creature systems, they are not the intended target for this project.

We identified this as an area of interest early in the project, and is discussed later in the report. By using novel mutation operators and further tuning the current iso+lineDD variation

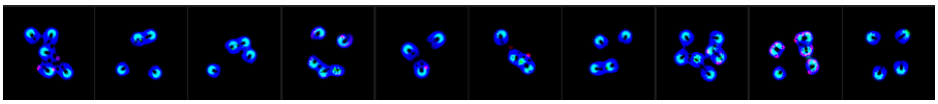

**Figure 4.7:** Random creatures taken from a repertoire evolved from Y3CS55 using positive mass average fitness, with a phenotype size of 64x64.

operator to encourage diverse life even with smaller phenotype sizes. We will also explore methods to iteratively evolve up to large phenotype size in Chapter 10.

## 4.5 Hyperparameter Experiments

### 4.5.1 Hidden Layer dimensions

AURORA uses the latent vector of the VAE as QD descriptors [22]. An area of interest for this project was understanding how the size of the descriptor vector affected the diversity of the repertoire and reconstruction quality of the VAE. We compare three identical VAEs, with the only difference being the size of the descriptor vector. These vectors have sizes 16, 32, and 64 respectively. We first compare the reconstructive capabilities of the VAE and how this changes when we change the hidden dimension size, see Figure 4.8. We use the negative angle variance fitness function.

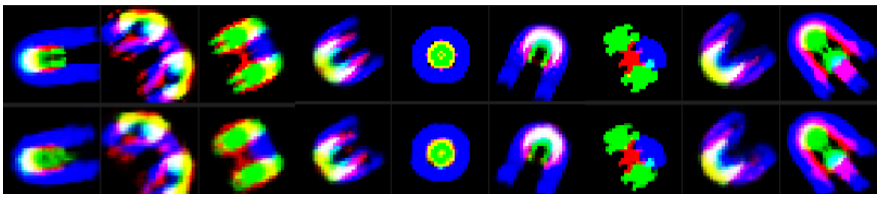

**Figure 4.8:** Row 1: Original soliton images. Row 2: Reconstructions. Columns 1-3: Hidden Dimension 16 VAE, columns 4-6: Hidden Dimension 32 VAE, columns 7-9: Hidden Dimension 64 VAE.

There is a clear jump in reconstruction clarity between the 16 and 32 hidden dimension VAEs, but the gap in quality between 32 and 64 is considerably smaller. This suggests that 16 dimensions may be insufficient to capture the complexity of Lenia patterns, while 32 dimensions provide a significant improvement. The smaller gap between 32 and 64 dimensions indicates diminishing returns, possibly because most relevant features are already captured by 32 dimensions. These findings do indicate that the original Leniabreeder paper's use of an 8 dimensional latent vector may be a limiting factor in the quality of its descriptors. We also consider the variance in the embedding space for each configuration, shown in Figure 4.9.

We can see that the larger hidden dimension configurations have a greater latent variance when compared with the smaller hidden dimension configuration. This increased variance in the latent space is a positive sign, as it indicates more diverse latent representations. Models with higher latent variance tend to generate a wider range of outputs as we have a wider probability distribution to draw from, potentially leading to more diverse Lenia patterns. While increased variance can sometimes come at the expense of reconstruction quality, Figure 4.8 shows that our reconstruction quality has also improved with scale. This suggests

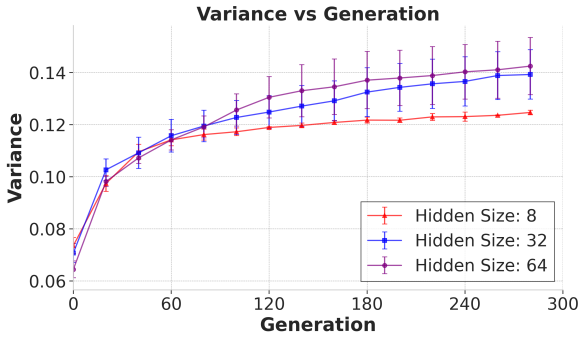


**Figure 4.9:** Benchmark latent variance over time with hidden dimension Size.

that the larger hidden dimensions are effectively capturing more nuanced features of the Lenia patterns without sacrificing the model's ability to accurately reconstruct them, potentially offering a better balance between diversity and fidelity in the generated creatures.

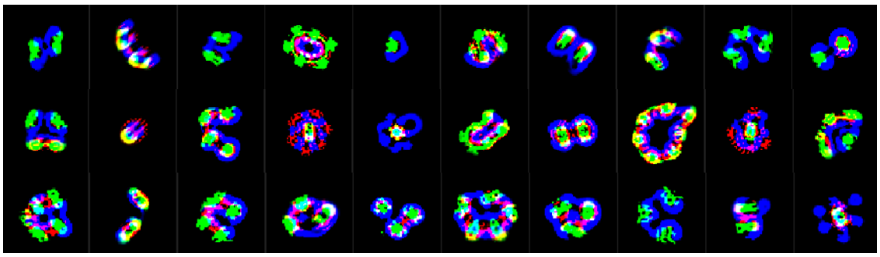

**Figure 4.10:** Row 1: Hidden dimension 8 VAE. Row 2: Hidden dimension 32 VAE. Row 3: Hidden dimension 64 VAE

We can see from the creatures in Figure 4.10 that we do observe more complex creatures as the descriptor vector size increases. The reason for this likely is that the larger hidden vector size allows for more detailed information capture. This increased capacity enables the VAE to encode and represent more intricate features and patterns within the Lenia creatures. Consequently, the model can generate and evolve more sophisticated and diverse organisms, potentially exploring a wider range of possible life-like behaviors and structures in Lenia. This observation aligns with our earlier findings on improved reconstruction quality and increased latent variance, suggesting that larger descriptor vectors contribute to both the fidelity and complexity of the evolved creatures.

### 4.5.2 Variation Operator Parameters

Leniabreeder uses the iso+LineDD variation operator, which has two hyperparameters: the isotropic sigma, $\sigma_{iso}$, and the directional (or line) sigma, $\sigma_{line}$ [10]. These parameters control the variance of the noise generated by the two Gaussian functions used in the variational operator. We investigate how these hyperparameters affect Leniabreeder's evolutionary process, demonstrating that appropriate parameter selection can overcome the evolutionary convergence observed in Section 4.3. We also note that $\sigma_{iso}$ is typically lower than $\sigma_{line}$, which is reflected in these studies.

We first present a study on the isotropic parameter, which controls the undirected Gaussian

noise added to the creature. We investigate values of 0.005, 0.025, 0.05, and 0.075. Using $\sigma_{iso} = 0.075$ resulted in total collapse of the repertoire, yielding no appropriate results. We present 20 creatures, randomly selected from the repertoire, for the $\sigma_{iso}$ of 0.005, 0.025, and 0.05 in Figure 4.11.

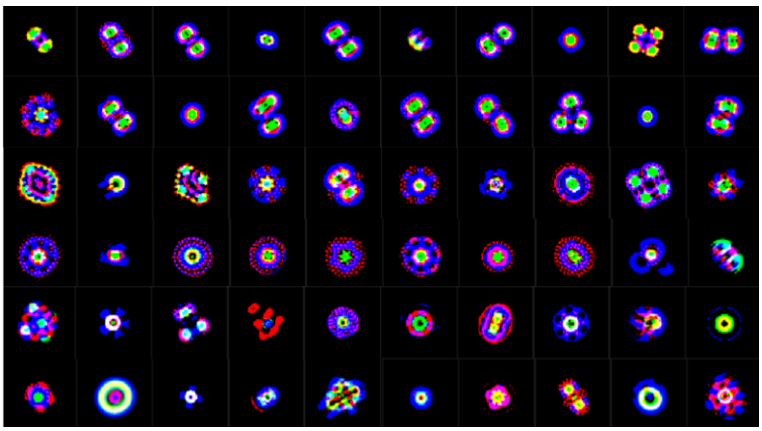

**Figure 4.11:** Rows 1-2: Creatures generated with $\sigma_{iso} = 0.005$. Rows 3-4: Creatures generated with $\sigma_{iso} = 0.025$. Rows 5-6: Creatures generated with $\sigma_{iso} = 0.05$.

Increasing $\sigma_{iso}$ reveals a clear impact on creature diversity, especially at the higher end. At 0.005, the value used in the original Leniabreeder framework, we observe considerable overlap and limited diversity. Raising the value to 0.025 introduces more unique creatures, though overlap persists. At 0.05, we see a substantial increase in diversity, seemingly overcoming the convergence barrier observed in initial experiments.

We also examine how the $\sigma_{line}$ value affects repertoire diversity, testing values of 0.05, 0.25, and 0.5. Unlike $\sigma_{iso}$, increasing $\sigma_{line}$ did not cause the creatures in the repertoire to vanish for any value. Again for each setup, we present 20 randomly selected creatures from the repertoire in Figure 4.12.

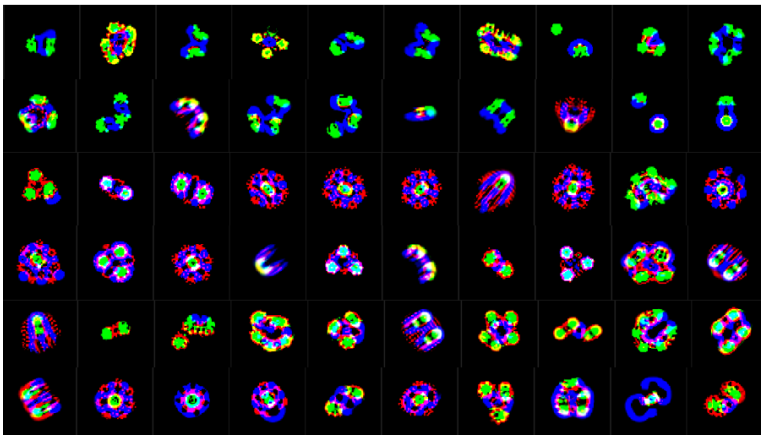

**Figure 4.12:** Rows 1-2: Creatures generated with $\sigma_{line}$ of 0.05. Rows 3-4: Creatures generated with $\sigma_{line}$ of 0.25. Rows 5-6: Creatures generated with $\sigma_{line}$ of 0.5.

Increasing $\sigma_{line}$ yields a modest increase in repertoire diversity, though less pronounced than changes to the $\sigma_{iso}$ at larger values. The line sigma experiments are more colourful, but when

we compare to the context of the rest of their repertoires the overall diversity is not increased.

Finally, we also present an experiment whereby we combine $\sigma_{line} = 0.5$ with $\sigma_{iso} = 0.05$. This combination of high values for both parameters aims to leverage the effects of each: the isotropic sigma's ability to encourage broader exploration and the line sigma's potential for fine-tuning along promising directions. We visualise a random sample of 50 creatures from this run in Figure 4.13.

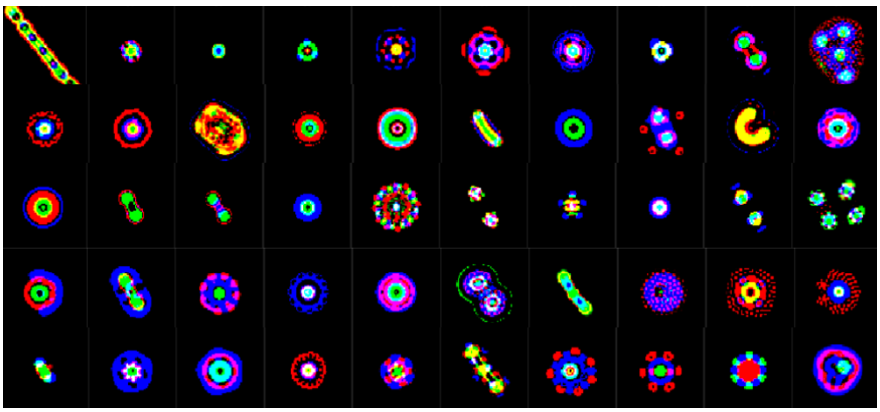

**Figure 4.13:** Random creatures generated with $\sigma_{iso} = 0.05$ and $\sigma_{line} = 0.5$. We observe rich diversity in both morphology and colouration.

We show in Figure 4.13 a massive increase in diversity of the discovered creatures. We see a wider variety of shapes, colours, behaviours. This indicates that the patterns we see repeated when using low isotropic and line noise values are indeed convergent towards certain creatures, and by adding additional pixel-wise noise to the creatures we are able to push them into further evolutionary chains. For future research, we propose experimentation using variation operator values that change over time, to see if a steady increase allows us to incrementally discover evolutionary chains that may be skipped when using a more aggressive approach to variation.

## 4.6 Chapter Summary

This chapter explored approaches to enhance diversity and complexity in discovered Lenia creatures without major architectural changes. We introduced new starting patterns, demonstrating their potential to generate unique evolutionary trajectories. Our investigations revealed that different initial patterns can lead to distinct repertoires, even under the same evolutionary pressures.

We examined the impact of phenotype size on evolutionary outcomes, finding that larger sizes can lead to undesirable multi-creature systems rather than true self-organising pattern diversity. Parameter scaling experiments on the VAE's hidden dimension sizes showed that increasing from 16 to 32 dimensions significantly improved reconstruction quality and latent space diversity, with diminishing returns beyond 32 dimensions.

Finally, we studied the effects of the variation operator parameters. Increasing the isotropic sigma parameter led to substantial improvements in creature diversity, while changes to the

line sigma parameter had more subtle effects. Combining large isotropic and line sigma values results in a massive uptick in repertoire diversity, but we acknowledge that such aggressive mutation may skip over some possible evolutionary trees. These findings highlight the crucial role of parameter tuning in balancing exploration and exploitation in Lenia's evolutionary process.

# Chapter 5

# Evaluation Tools

## 5.1 Motivation

A noteable challenge in this project was quantitatively evaluating diversity. To address this, we developed new evaluation and visualization tools, which were crucial to evaluating the effectiveness of different AURORA architectures. These new tools enabled a more comprehensive understanding of creature evolution across different models and parameters. By implementing diversity measures, interactive visualization techniques, and other tools we could more accurately track and compare creature diversity across various algorithmic configurations and evolutionary timescales, all while being mindful of any limitations to these approaches.

## 5.2 Interactive t-SNE Plotting

### 5.2.1 Repertoire

In our work with autoencoder models in AURORA, visualizing the embedding space was crucial to understand our behavioural descriptors. We employ t-SNE for dimensionality reduction to view the latent space, a common practice for autoencoders [30]. However, unlike supervised learning tasks where t-SNE plots can be colour-coded by class, our unsupervised approach lacked predefined labels for clear cluster separation.

To overcome this challenge, we leverage the image-based nature of our soliton data. We develop interactive 2D and 3D t-SNE plots where users can click on embeddings to view the corresponding input soliton. This tool enables effective cluster identification and provides insights into the model's interpretation of solitons, addressing the challenge of unlabeled data in the latent space. An example of this tool is shown in Figure 5.1.

The interactive t-SNE visualization proves crucial in evaluating new Leniabreeder architectures, especially when evaluating architectures aiming for geometric-invariance in its embeddings. It also offers a more efficient alternative to reviewing repertoire videos, allowing rapid assessment of a repertoire's diversity. This tool facilitates comprehensive analysis of soliton evolution and distribution across various algorithmic configurations.

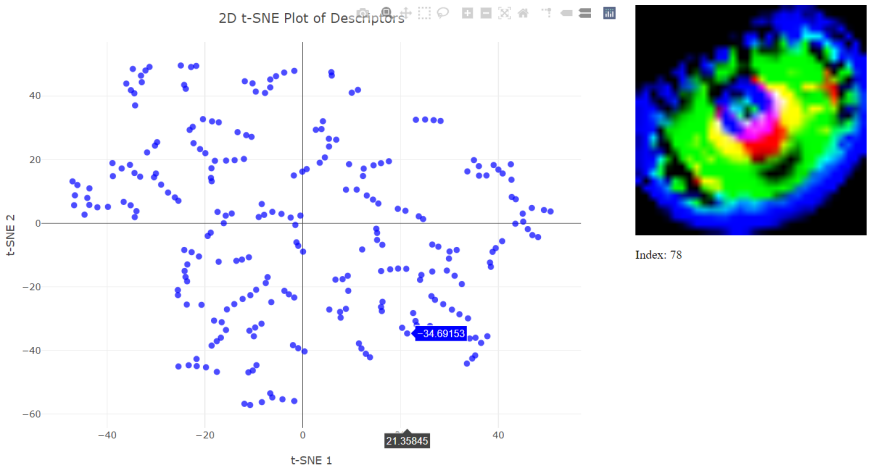


**Figure 5.1:** Interactive t-SNE plot. The user can click on any point in descriptor space, and will be presented with the soliton that corresponds to that embedding.

### 5.2.2 Latent Space Trajectory

We also develop a visualization tool that tracks a soliton's latent embedding evolution over time. This interactive plot expands on the previous work, offering a dynamic view of a single soliton's evolutionary journey. Users can click on any point to view the soliton's appearance at that moment. A colour gradient visually represents the sequence of time steps, providing an intuitive sense of developmental stages. This tool allows for a more detailed and interactive exploration of soliton evolution in the latent space.

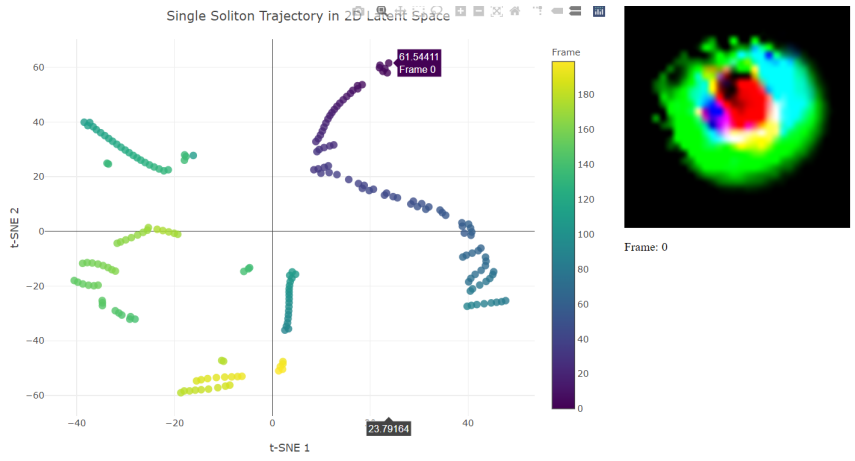


**Figure 5.2:** The latent trajectory of a creature as it evolves over 200 time-steps. We can see that there are large shifts in the descriptor space that happen very suddenly.

Plotting soliton trajectories across timesteps reveals that changes in the descriptor space are not necessarily continuous. Instead, we observe large jumps at certain points, despite minimal changes in the creature's form. This discontinuity highlights the sensitivity of our descriptor space to subtle changes in soliton characteristics. Such jumps could indicate moments of significant feature detection or transitions between distinct morphological states, demonstrating the model's ability to capture nuanced evolutionary changes.

### 5.2.3 Justifying t-SNE

While several dimensionality reduction techniques exist, t-SNE was chosen for its ability to preserve local structure and reveal clusters in high-dimensional data. Unlike Principal Component Analysis (PCA), which focuses on preserving global structure, t-SNE excels at maintaining relative distances between nearby points, making it ideal for visualizing how well the models embed similar creatures [30].

It's important to acknowledge t-SNE's limitations, such as its stochastic nature leading to different results across runs and its potential to sometimes exaggerate cluster separation [31]. However, these limitations were somewhat mitigated by our interactive approach, which allowed users to directly examine solitons corresponding to specific points, providing context and the ability to verify observed clusters.

Overall, t-SNE proves to be an effective choice for our visualization needs. Its ability to reveal intricate relationships in the latent space, combined with our interactive tools, provided invaluable insights into soliton evolution and diversity. This approach not only enhances our understanding of the embedding space but also offered an engaging way to explore the Lenia patterns.

## 5.3 Rotational Diversity Metrics

A key limitation of base Leniabreeder was the classification of rotated creatures as entirely new, hindering efficient quality assessment as they then took up many spaces in the repertoire. To address this, we develop macro-scale metrics to evaluate soliton diversity more effectively across the entire repertoire.

We introduce the rotational diversity score to quantify pattern uniqueness under various rotations. This metric aims to distinguish between genuinely novel patterns and rotational variants, providing a more accurate assessment of the soliton population's true diversity. This advancement improves our evaluation of evolved solitons' quality and diversity. It guides refinements to our evolutionary algorithms and selection criteria, aiming towards generating truly novel artificial life forms.

### 5.3.1 Algorithm Overview

The rotational diversity calculation involves the following steps:

1. Extract phenotypes from the repertoire
2. Calculate pairwise rotational diversities between phenotypes
3. Construct a diversity matrix
4. Compute diversity metrics based on the diversity matrix

### 5.3.2 Mathematical Formulation

#### Pairwise Rotational Diversity

For any two phenotypes $p_1$ and $p_2$, we define their rotational diversity as:

$$D(p_1, p_2) = \min_{\theta \in 0°, 15°, 30°, \ldots, 345°} \frac{1}{n_{\text{keep}}} \sum_{f=1}^{n_{\text{keep}}} \left| p_1^f - R_\theta(p_2^f) \right| \tag{5.1}$$

Here, $R_\theta(\cdot)$ is the rotation operator that rotates an image by angle $\theta$, and $p_i^f$ denotes the $f$-th frame of phenotype $i$. We use billinear interpolation to appropriately rotate at small angles [32]. The parameter $n_{\text{keep}}$ is the number of frames considered when calculating diversity. We average over these frames to account for temporal creature behavior in the diversity metric.

This metric, while termed rotational diversity, could also be called a rotational difference score. We use the minimum value across all rotations to compare with the closest match, ensuring a strict metric. A score of 1 indicates a completely unique creature, while 0 represents an identical match.We construct a diversity matrix $M$ where each element $M * ij$ is the pairwise rotational diversity between phenotypes $i$ and $j$: $M * ij = D(p_i, p_j)$. Typical values for this metric range between 0.1 to 0.2.

#### Diversity Metrics

From the diversity matrix, we compute the following metrics:

- Mean Diversity: For each phenotype, we calculate the average of the 5 most similar solutions (excluding self-comparison). The mean diversity is then computed as the average of these values across all phenotypes:

$$\bar{D} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{5} \sum_{j=1}^{5} \text{sort}(M_i)j \tag{5.2}$$

  where $M_i$ is the $i$-th row of the diversity matrix, and sort$(M_i)j$ is the $j$-th smallest value in that row (excluding the self-comparison).

- Mean Self-Diversity: For each phenotype, we compute its diversity relative to itself under rotation. This provides insight into the structure of the creatures, with low scores indicating highly symmetric creatures, or creatures that are stable over time, and high scores suggesting creatures with more complex structures, or that are instable over time. Note that for self-diversity we compute the pairwise rotational diversity different, and instead we use the mean score across all rotations, rather than the minimum (as using the minimum would always return 0).

These metrics provide a more nuanced evaluation of diversity, focusing on local neighborhoods in the phenotype space rather than global pairwise comparisons. We note that using the five nearest neighbours is a reasoned choice. Increasing the number of considered neighbours too much would start to compare very different creatures, potentially overestimating the repertoire's diversity. Conversely, choosing a smaller number could underestimate the true diversity. We argue it's better to err on the side of being punitive in the metrics, as we would rather underestimate the diversity of our repertoire than overestimate it. By focusing on a relatively small local neighborhood, we maintain a stricter standard for uniqueness, ensuring our diversity measures reflect meaningful differences between phenotypes.

### 5.3.3 Limitations

These metrics have limitations, particularly when multi-creature and multi-celled systems emerge. In these cases, duplicate creatures interact to create complex patterns that score well on rotational diversity, despite not representing truly new creatures. Therefore, we do not recommend total reliance on these metrics. They serve as a macro overview of a repertoire, but it's possible for the diversity score to be "cheated." This occurs when comparing two large creatures with very similar structures; due to the continuous nature of the pixel values, there can still be a high average difference between them. We also observe that these creatures exhibit significant temporal instability. While they may appear unchanged to the human eye over time, our metrics detect substantial variations. This discrepancy highlights the sensitivity of our quantitative measures compared to qualitative human observation. We can counter this by using qualitative evaluations when this is suspected. This is especially relevant for fitness functions that encourage mass growth, where some qualitative analysis is recommended. The advantage of having these metrics though, is that we need to perform less qualitative analysis to confirm their results. If a repertoire has a very low diversity score, then we can trust that. If it has a high diversity score, then we need only 20-30 creatures to verify if this diversity is true or "cheated". Therefore the metrics, despite their flaws, still massively improve the efficiency of repertoire analysis.

## 5.4 Leniabreeder Benchmarks

We present the average diversity and self-diversity of a repertoire every 20 generations, up to 300. We do this using the standard Leniabreeder configuration, except for the hidden dimension size. We present benchmarks for both 32 and 64 dimensional hidden vector VAEs, as Section 4.5.1 suggests these values return more complex creatures.

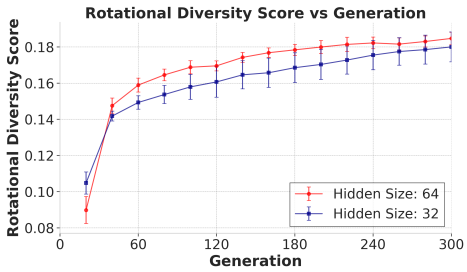


**Figure 5.3:** Rotational Diversity Score benchmark at every 20 generations.

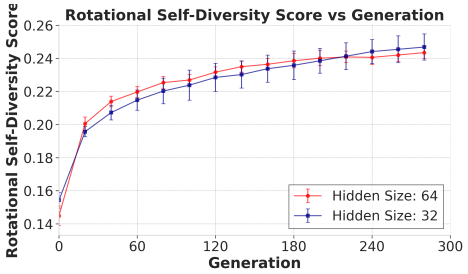


**Figure 5.4:** Rotational Self-Diversity score benchmark at every 20 generations.

As indicated in Section 4.5.1, a larged hidden dimension results in increased diversity in the repertoire, likely due to an increased creature complexity. We present the final values for both metrics, and their standard deviation in Table 5.1.

| Hidden Size | Diversity | | Self-Diversity | |
|---|---|---|---|---|
| | Mean | Std Dev | Mean | Std Dev |
| 32 | 0.180 | 0.008 | 0.245 | 0.008 |
| 64 | 0.185 | 0.003 | 0.244 | 0.004 |

**Table 5.1:** Comparison of Diversity and Self-Diversity measures across different Hidden Sizes

We also note that both the diversity and self-diversity scores are still increasing at generation 300. This will be used to compare the rate of diversity change for different approaches. While we present self-diversity scores through this paper, our focus will be near entirely on the repertoire diversity.

# Chapter 6

# Contrastive Learning in AURORA

## 6.1 Motivation

The primary objective of Leniabreeder is to generate a diverse array of creatures. However, it is observed that the repertoire often stores duplicated creatures, altered only by geometric transformations, as unique entities in the repertoire. This phenomenon, visualized in Figure 6.1, does not reflect true diversity.

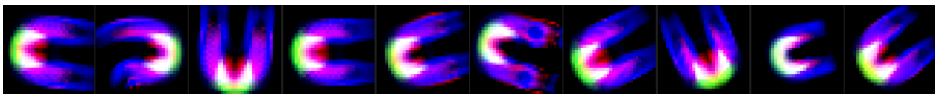

**Figure 6.1:** A collection of creatures from the same repertoire. It is clear that these are not diverse, and are instead (for the most part) rotated versions of each other

This issue of geometric duplicates highlights a significant challenge in accurately assessing the repertoire's diversity. By treating rotational or otherwise geometrically transformed versions of the same creature as distinct, the system potentially overestimates its success in generating novel forms. Addressing this limitation was crucial to developing a more accurate measure of the algorithm's performance in creating genuinely diverse and unique artificial life forms. We propose contrastive learning as an approach to teach certain geometric invariances to our VAE.

## 6.2 Geometric shifts

Before discussing the work done towards reaching different geometric invariances, it is worthwhile to consider some key geometric actions, and their relevance for Leniabreeder.

### 6.2.1 Rotation

Rotation in Leniabreeder involves the movement of creatures around their center. The original Leniabreeder implementation doesn't directly address rotational invariance, leading to rotated versions of the same creature being treated as distinct. This can artificially inflate the diversity in the repertoire, as essentially identical creatures in different orientations might be considered unique. Achieving rotational invariance is challenging, as it requires more complex processing. A significant portion of this chapter, and the next, will be dedicated to

methods aiming to address this problem.

### 6.2.2 Shearing and Scaling

Shearing and scaling are geometric transformations that can affect the apparent diversity of Lenia's creatures. Shearing distorts shape while preserving area, potentially arising from certain mutations or pattern-generating processes. Scaling primarily affects symmetric circular solitons, which can appear identical at different sizes. Often, these effects are a result of how creatures change over each time step, with their form often fluctuating in a way that appears stretched or somewhat resized when we take the sliced image for the VAE.

## 6.3 Data Augmentation

Leniabreeder employs data augmentation to enhance training data diversity through geometric transformations. These include rotations at 90-degree intervals from 0° to 360°, shearing applied at angles of ±0.2 radians following the formula $S(x, y) = (x + y \tan \alpha, y)$, and scaling using factors of 0.8 and 1.2, represented as $C(x, y) = (sx, sy)$. The shearing and scaling transformations are implemented using affine transformation matrices. The rotation choice balances rotational invariance with memory constraints, as the AURORA Leniabreeder already uses nearly 10GB of VRAM for small repertoires. While more fine-grained rotations could potentially benefit the model, the current approach strikes a balance between introducing rotational augmentations to be used for training towards rotational invariance and maintaining computational feasibility.

### 6.3.1 Impact of Data Augmentations

We observe that when data augmentation is used in the base Leniabreeder framework, there is a clear improvement in diversity. See Figures 6.2, and 6.3 for the diversity metrics over every 20 generations.

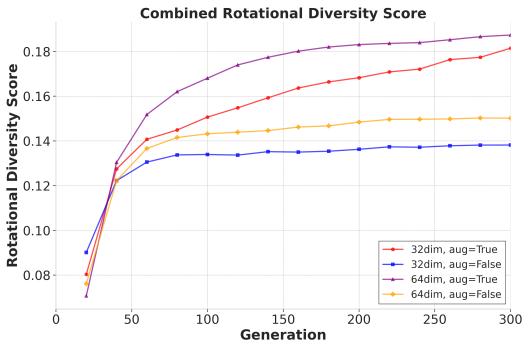


**Figure 6.2:** Rotational diversity score at every 20 generations with/without data augmentations.

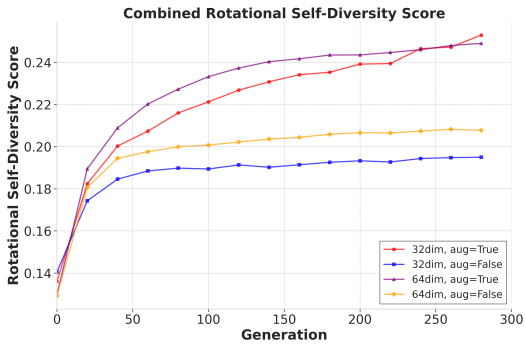


**Figure 6.3:** Rotational self-diversity score at every 20 generations with/without data augmentations.

We observe that data augmentation in the base Leniabreeder framework does substantially impact diversity, as shown in Figures 6.2, 6.3. Our diversity score not only reaches a much higher value, but it also appears that it is converging much slower, especially in the 32

hidden dimensional VAE. These augmentations are also crucial for enabling new AURORA architectures that utilize contrastive learning methods, which we hypothesised could lead to more diverse results.

We also extend the data augmention, adding scaling and shearing augmentations created through affline transformations.

## 6.4 Contrastive Learning

### 6.4.1 Theory

Contrastive learning is a technique that learns representations by comparing data samples, identifying related points as positive pairs and unrelated ones as negative pairs [33]. In Leniabreeder, this approach was applied with the aim to develop robust latent representations of Lenia creatures, training the model to recognize that various transformations of a single creature are similar, while other creatures are dissimilar.

The method aims to maximize similarity between positive pairs and minimize it between negative pairs in the learned representation space, typically using specialized objective functions like Triplet Margin [34] loss or NT-Xent loss [33]. For Lenia creatures, positive samples are created by applying transformations to a single creature, while negative samples are different creatures in the repertoire, encouraging learning of representations invariant to these transformations.

This study focuses on two objective functions: NT-Xent Loss and Triplet Margin Loss, both modulated by a parameter $\tau$ that controls the weight of the contrastive learning component within the Aurora framework. This parameter is called temperature for NT-Xent loss [33], and margin for triplet margin loss [34].

### 6.4.2 Data Augmentation Importance

Having augmented data was a necessity for the contrastive learning approaches taken for this project. Both methods required positive samples for each creature. However, the data used for this project was unlabelled, making this a challenge. The augmented data was held separate from the original data, but ordered the same. This means that the first creature in the original data would correspond to the first $N_{\text{Augs}}$ augmented samples. This made it very simple to draw positive samples for each creature. Negative samples were drawn from other creatures in the repertoire, though this could result in very similar creatures being selected as negatives.

### 6.4.3 NT-Xent Loss

#### Background

The NT-Xent (Normalized Temperature-scaled Cross Entropy) loss function's primary objective is to train an encoder that maximizes mutual information between an anchor sample's representation and that of its transformed (augmented) version, while minimizing mutual

information with other samples' representations in the batch [33]. From an information-theoretic perspective, we aim to maximize:

$$I(z; z^+) = \mathbb{E}_{p(z,z^+)} \left[ \log \frac{p(z|z^+)}{p(z)} \right] \tag{6.1}$$

Here, $z$ represents the anchor sample's encoding, and $z^+$ denotes the encoding of its augmented version.

The NT-Xent loss approximates the mutual information maximization objective by framing it as a classification task. Given a representation $z$, the model must identify its augmented pair $z^+$ among a set of negative samples. This is achieved through a softmax-based formulation:

$$\mathcal{L}_{\text{NT-Xent}} = -\log \frac{\exp(sim(z, z^+)/\tau)}{\exp(sim(z, z^+)/\tau) + \sum_{i=1}^{N} \exp(sim(z, z_i)/\tau)} \tag{6.2}$$

In this equation $z_i$ represents the encodings of negative samples, $sim()$ denotes a similarity function (cosine similarity in this implementation) and $\tau$ is a temperature parameter that controls the concentration of the distribution. Cosine similarity was chosen as it appears frequently in the literature [33][35][36], but a euclidean similarity version is also provided in the project code repository.

This formulation effectively pushes the representation of an anchor sample closer to its augmented version in the latent space. Unlike Triplet Margin loss [34], NT-Xent loss does not use negative samples to directly push dissimilar samples away, but rather does so indirectly. It achieves this indirect behaviour by increasing the similarity of positive pairs relative to all other pairs, which implicitly reduces the similarity to negative samples. The temperature parameter $\tau$ modulates the strength of the attractions, allowing for fine-tuning of the learned representations' properties [33]. The process can be seen in Figure 6.4.

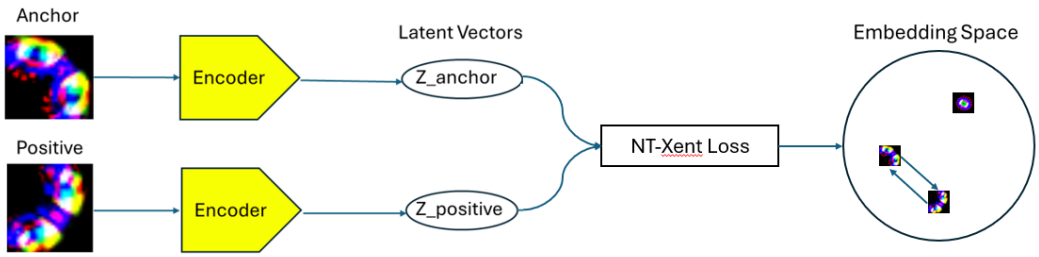


**Figure 6.4:** Diagram showing how NT-Xent encourages the model to cluster similar creatures. An anchor and positive sample are provided, and the model is rewarded by the NT-Xent loss function for pulling their embeddings closer together. Negatives are indirectly pushed away in this method.

### Implementation

The NT-Xent loss implementation follows a structured process. Representations are first normalized to facilitate cosine similarity calculations. Similarity matrices are then computed for both positive pairs (original and augmented views) and all pairs within the batch. A temperature scaling factor is applied to control the concentration of the resulting distribution. The process continues with the calculation of positive and negative similarities, carefully

excluding self-similarity to prevent trivial solutions. Finally, the loss is computed using a log-sum-exp formulation, which ensures numerical stability. This approach allows for efficient batch processing and robust handling of the high-dimensional representations typical in Lenia pattern analysis. By excluding self-similarity, the model is forced to learn meaningful features that distinguish between different Lenia creatures while recognizing various transformations of the same creature.

### 6.4.4 Triplet Margin Loss

#### Background

The Triplet Margin Loss is a contrastive learning technique rooted in metric learning. It aims to learn embeddings by optimizing a distance metric that clusters similar samples while separating dissimilar ones in the embedding space [34]. The core principle is to satisfy the constraint:

$$d(f(x_a), f(x_p)) + \alpha < d(f(x_a), f(x_n)) \quad \forall (x_a, x_p, x_n) \in \mathcal{T} \tag{6.3}$$

Where $f(\cdot)$ is the embedding function, $(x_a, x_p, x_n)$ is a triplet of anchor, positive, and negative samples, $d(\cdot, \cdot)$ is the distance metric, $\alpha$ is the margin, and $\mathcal{T}$ is the set of all possible triplets in the training set. In our implementation $f(\cdot)$ refers to the variational autoencoder. The standard Triplet Margin Loss is defined as:

$$\mathcal{L}_{\text{triplet}} = \max(0, d(f(x_a), f(x_p)) - d(f(x_a), f(x_n)) + \alpha) \tag{6.4}$$

This formulation is a hinge loss, meaning it only penalizes violations of the desired constraint [37]. It's scale-invariant to the embedding function, which means it's not affected by uniform scaling of the embeddings. The loss encourages relative ordering of distances rather than absolute values, focusing on the relationships between samples rather than their specific locations in the embedding space. This approach makes the model more robust to variations in the scale of the input data and helps it learn more generalisable representations of the Lenia creatures. This approach can be visualised in Figure 6.5.

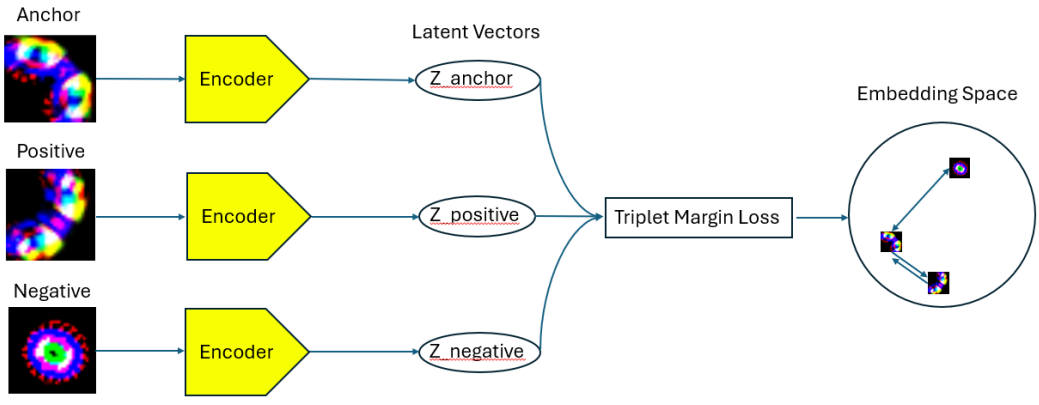


**Figure 6.5:** Diagram showing how Triplet Margin loss encourages the model to cluster similar creatures and push back differing creatures. An anchor, positive, and negative sample are provided, and the model is rewared by the loss function for pulling the anchor and positive embeddings together, as well as pushing away the negative embedding

### Implementation

The implementation proceeds as follows: Original and augmented embeddings are provided separately to the loss function. Anchors, positives, and negatives are selected. Augmented samples are formatted for easy selection corresponding to each original sample. Squared Euclidean distances are calculated between the selected samples. The triplet loss is computed using these distances and the specified margin. The mean loss across all triplets in the batch is returned.

This approach creates a representation space where similar Lenia creatures cluster together, potentially enhancing pattern recognition and generation capabilities in the Leniabreeder context. We do acknowledge that there is a slight potential for the negative sample to be another soliton in the repertoire that is a duplicate of the original creature, which may cause the model some confusion. We believe that the chance of this happening is sufficiently low for Triplet Margin loss to remain a valid approach.

## 6.5 Results

### 6.5.1 NT-Xent Loss

We first present the diversity metrics for the NT-Xent objective function architecture, show in Figures 6.6 and 6.7.

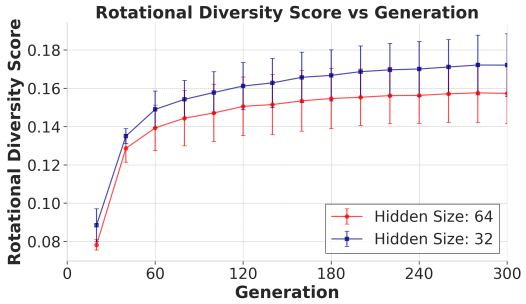


**Figure 6.6:** Rotational diversity score at every 20 generations using NT-Xent loss.

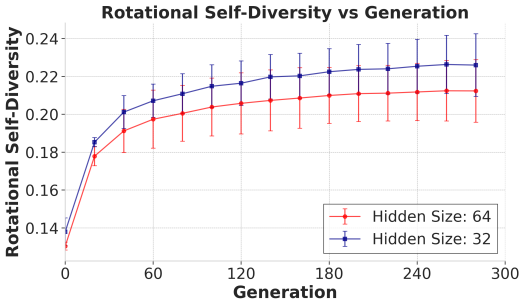


**Figure 6.7:** Rotational self-diversity score at every 20 generations using NT-Xent loss.

Interestingly, these metrics suggest a dissimilar relation between hidden vector size and diversity compared to what was observed in Chapter 4. Contrary to our earlier findings, here we see that a larger latent vector appears to create less diverse creatures. This unexpected result could be due to overfitting with larger hidden vectors, increased focus on fine details rather than broader features, or unique interactions between the NT-Xent loss and larger latent spaces. We perform qualitative analysis of these results, presenting them in Figure 6.8.

Upon examination of Figure 6.8, we observe that while there isn't a substantial difference in overall diversity between the two hidden vector sizes, the structural characteristics of the creatures do vary noticeably. This suggests that the improved metric readings for the 32-dimensional latent vector may reflect differences in individual creature structure rather than repertoire diversity.

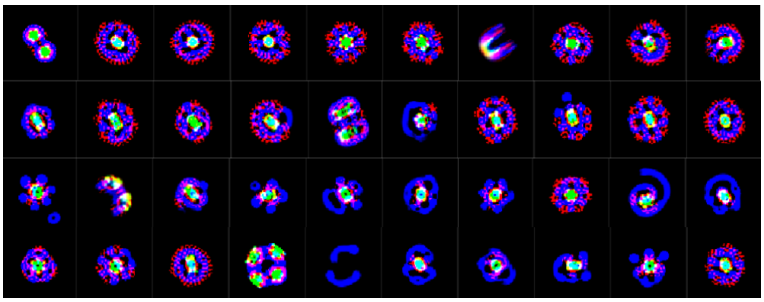

**Figure 6.8:** Row 1-2: 20 Random creatures generated using NT-Xent Loss and a hidden dimension size of 32. Row 3-4: 20 Random creatures generated using NT-Xent Loss and a hidden dimension size of 64.

Notably, large creatures occupying most of the phenotype space, as seen in rows 1 and 2, tend to score well in both inter-creature and self-comparisons. When we visualise these creatures overtime, we observe them to be highly chaotic with their colour distribution, switching between different allocations of blue,red, and yellow. We hypothesise that these creatures score well because of this chaotic nature; enough differences occur at each comparison point to allow a reasonable score. This phenomena was discussed in Chapter 5, and as stated there we only need to visualise a very small subset of the repertoire to see whether this is the case.

We now examine the impact of the NT-Xent loss temperature parameter on the architecture's performance. This parameter governs the model's sensitivity to hard negative examples, with lower temperatures producing sharper distinctions between positive and negative pairs. Our original implementation used a high temperature value of 0.9, theoretically allowing for more diverse embeddings by introducing flexibility in learned representations.

To investigate this effect, we conduct experiments using temperature values of 0.25, 0.5, and 0.75. These values allow us to observe how different levels of distinction between positive and negative pairs affect the diversity and quality of generated creatures. The results are visualised in Figures 6.9, 6.10, and 6.11.

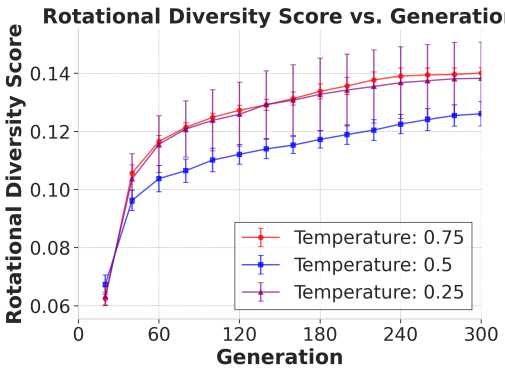


**Figure 6.9:** Rotational diversity score at every 20 generations using different NT-Xent temperatures.

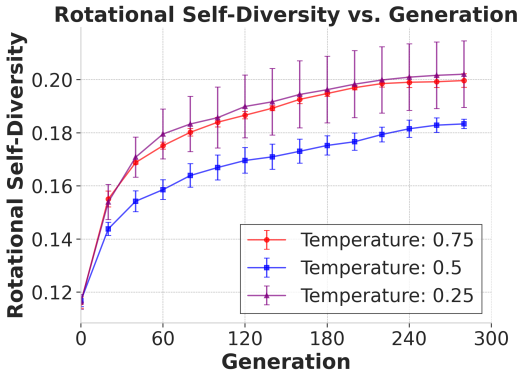


**Figure 6.10:** Rotational self-diversity score at every 20 generations using different NT-Xent temperatures.

The impact of temperature on diversity in our NT-Xent objective AURORA architecture re-

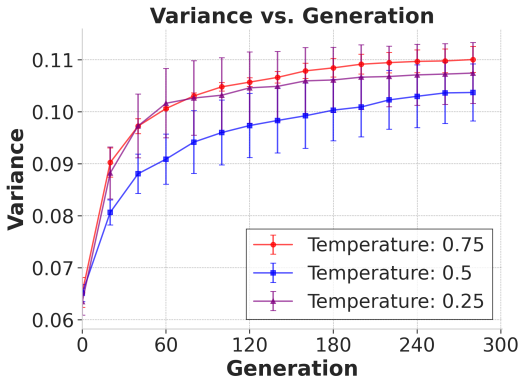


**Figure 6.11:** Latent variance at every 20 generations using different NT-Xent temperatures.

veals an intriguing pattern. We observe a significant decrease in diversity metrics at a balanced temperature of 0.5, evident in both quantitative measures and qualitative analysis of generated creatures (Figure 6.12).

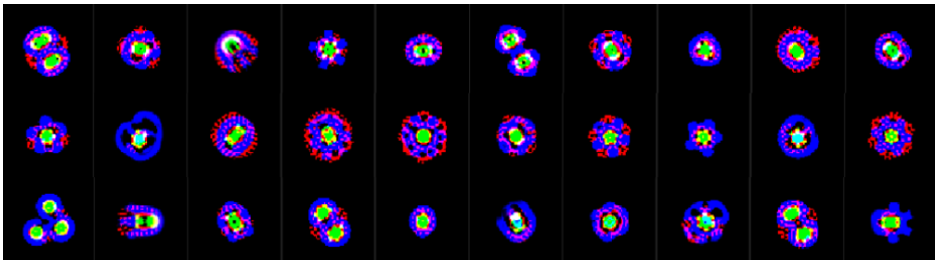

**Figure 6.12:** Creatures generated using NT-Xent AURORA at different temperatures. Row 1: $\tau = 0.25$, Row 2: $\tau = 0.5$, Row 3: $\tau = 0.75$.

Lower temperatures (0.25) yield sharper distinctions between positive and negative pairs, emphasizing unique features and potentially increasing diversity. Conversely, higher temperatures permit more flexible representations, allowing for a wider range of feature combinations. The unexpected diversity dip at 0.5 may result from an optimization landscape conducive to local minima or a narrowed focus on specific feature ranges. Ultimately though, the results are indicative that none of these setups outperform the original AURORA architecture.

While further investigation into this phenomenon could provide valuable insights for NT-Xent loss tuning, we chose to redirect our resources elsewhere. This decision was based on consistent indicators suggesting that the NT-Xent objective AURORA architecture does not significantly enhance creature diversity, prompting us to explore more promising avenues in our research.

### 6.5.2 Triplet Margin Loss

We begin by examining the diversity metrics for the Triplet Margin loss objective function, as illustrated in Figures 6.13 and 6.14. The results reveal a trend similar to that observed with the NT-Xent objective function: larger latent vector sizes appear to negatively impact diversity metrics. This consistent pattern across different contrastive learning approaches

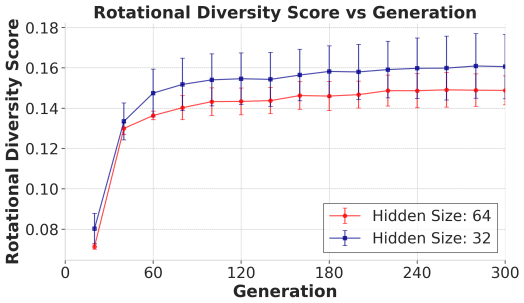


**Figure 6.13:** Rotational Diversity Score at every 20 generations using Triplet Margin loss.

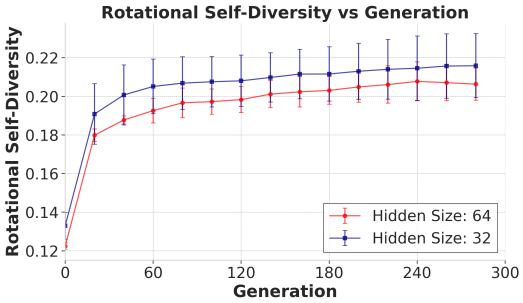


**Figure 6.14:** Rotational Self-Diversity score at every 20 generations using Triplet Margin loss.

suggests that the relationship between latent space dimensionality and creature diversity may be more complex than initially anticipated. It's possible that increased dimensionality, while theoretically offering more capacity for diverse representations, may in practice lead to overfitting or a focus on fine-grained details that don't contribute meaningfully to overall creature diversity in the context of Lenia, at least when a contrastive learning approach is applied.

While the quantitative metrics indicate comparable performance between Triplet Margin and NT-Xent losses, a qualitative analysis of the generated creatures (Figure 6.15) reveals subtle but significant differences. These distinctions highlight the importance of combining quantitative metrics with visual inspection to fully assess the effectiveness of different objective functions in generating diverse and interesting Lenia creatures.

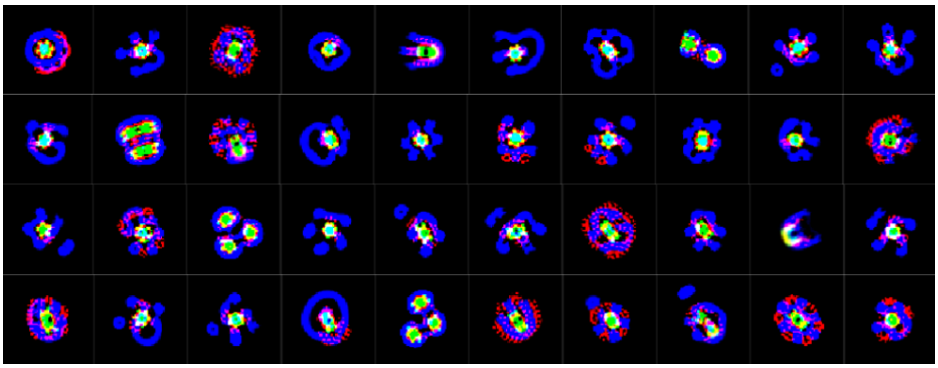

**Figure 6.15:** Row 1-2: Creatures generated with a 32 dimensional latent vector and Triplet Margin AURORA. Row 3-4: Creatures generated with a 64 dimensional latent vector and Triplet Margin AURORA.

Figure 6.15 reveals a noticeable increase in creature diversity when using triplet margin loss compared to NT-Xent loss. However, this improvement does not outperform the standard AURORA architecture. To further investigate potential limiting factors, we conducted experiments with different margin values.

The results presented in Figures 6.16, 6.17, and 6.18 consistently show that a smaller margin value produces the most diverse repertoire. To provide a qualitative perspective on these

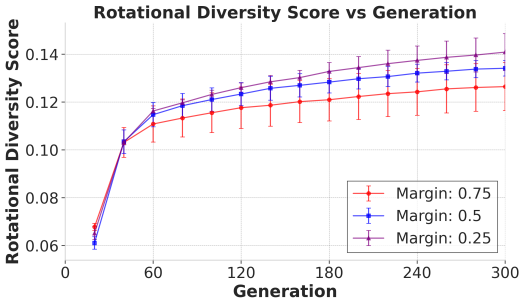


**Figure 6.16:** Rotational diversity score at every 20 generations using Triplet Margin Loss with different margins.

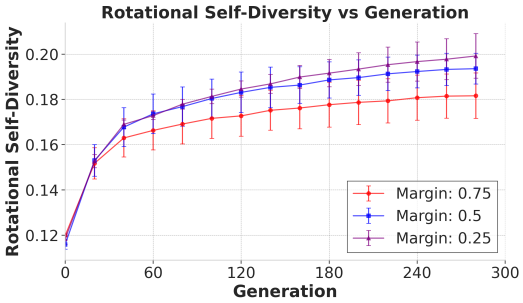


**Figure 6.17:** Rotational Self-Diversity score at every 20 generations using Triplet Margin Loss with different margins.

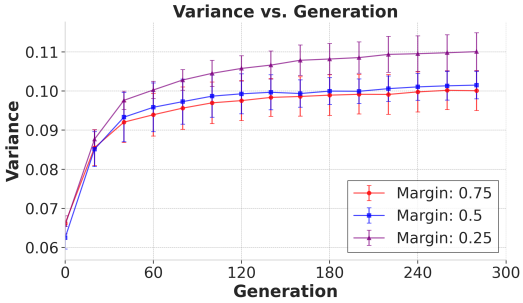


**Figure 6.18:** Latent Variance at every 20 generations using Triplet Margin Loss with different margins.

findings, we visualize representative creatures from each margin setting in Figure 6.19.

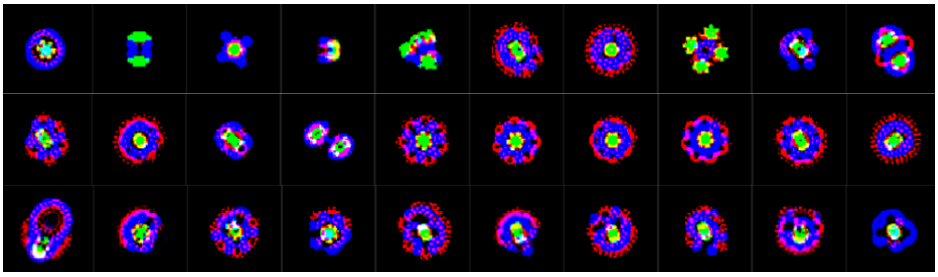

**Figure 6.19:** Row 1: Creatures generated with a margin of 0.25. Row 2: Creatures generated with a margin of 0.5. Row 3: Creatures generated with a margin of 0.75.

We see the diversity increase as the margin value decreases. This could be because a smaller margin allows for more flexibility in the embedding space, encouraging the model to learn more nuanced representations. A reduced constraint on separation between positive and negative pairs potentially allows for more subtle differences between creatures. With less pressure to create large separations, the model might focus on capturing finer details that contribute to creature diversity. A smaller margin might also allow the model to explore a wider range of possible creature configurations, as it's not forced to create large gaps between different types. This approach might strike a better balance between maintaining similarity for related creatures and encouraging diversity across the repertoire.

### 6.5.3 Overall

To conclude this results section, we present formally in Table 6.1 the quantitative comparisons between the baseline NT-Xent and Triplet Margin approaches to the standard benchmarks set in Section 5.4.

| Objective Function | Hidden Size | Diversity | | Self-Diversity | |
|---|---|---|---|---|---|
| | | Mean | Std Dev | Mean | Std Dev |
| Benchmark | 32 | 0.180 | 0.008 | 0.245 | 0.008 |
| | 64 | 0.185 | 0.003 | 0.244 | 0.004 |
| NT-Xent | 32 | 0.172 | 0.016 | 0.226 | 0.017 |
| | 64 | 0.157 | 0.016 | 0.212 | 0.017 |
| Triplet Margin | 32 | 0.161 | 0.016 | 0.216 | 0.017 |
| | 64 | 0.149 | 0.007 | 0.206 | 0.008 |

**Table 6.1:** Comparison of Diversity and Self-Diversity measures across different Hidden Sizes for multiple Objective Functions

The scores for both contrastive learning approaches are much lower than the benchmark, being over a standard deviation away. This reinforces the qualitative results wherein we observe substantially less diversity in the creatures within the repertoire. We have also shown that tuning the new contrastive learning hyperparameters is not sufficient to bring contrastive learning on par with the original benchmark. This indicates that adding the additional constraints imposed by contrastive learning is too much for a diverse evolutionary process. We acknowledge the high standard deviation value observed for the contrastive learning approaches. Again, we believe this is a result of chaotic temporal behaviours in the creatures we observe, which in some cases "trick" the metric. This problem was uniquely observed when working with the contrastive learning methods.

## 6.6 Chapter Summary

We have implemented contrastive learning AURORA architectures. We show that these methods do not necessarily offer improvements to the diversity of the repertoire of creatures we evolve. Triplet Margin loss appears to be more effective than NT-Xent loss, though both are less so than the benchmark. We have also explored the impacts of each CL architecture's unique hyperparameters, and have discussed how these directly impacted the generated diversity in the Lenia creatures.

We hypothesise that by adding contrastive learning we are in effect adding a bottleneck to the learning process, so that the model is not able to effectively generate diverse embeddings when having to accommodate the added CL objectives. Attempts at using a larger VAE were undertaken, with the hope that added features could help to capture these added requirements, but this was not successful.

Despite contrastive learning being unsuccessful in this use-case, successfully implementing rotational invariance still seemed like a promising approach. As such, we also tried to implement this by taking approaches grounded in group theory.

# Chapter 7

# Group Theoretic Approaches to Rotation Invariance

## 7.1 Group Theory

We also propose group-theoretic approaches aimed at achieving rotation-invariant embeddings. A group $(G, \cdot)$ is a set $G$ equipped with a binary operation $\cdot$ that satisfies four axioms [38]:

- Closure: For all $a, b \in G$, the result of the operation, $a \cdot b$, is also in $G$.
- Associativity: For all $a, b, c \in G$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- Identity: There exists an element $e \in G$ such that for every element $a \in G$, $e \cdot a = a \cdot e = a$.
- Invertibility: For each $a \in G$, there exists an element $b \in G$ such that $a \cdot b = b \cdot a = e$.

In the context of geometric transformations, group theory offers an elegant way to describe and analyze the symmetries of objects, in this context Lenia creatures. A symmetry group is a group whose elements are symmetries of a given mathematical object. For instance, the group of rotations in 2D space, SO(2), is defined as [39]:

$$\mathrm{SO}(2) = \left\{ \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} : \theta \in \mathbb{R} \right\} \tag{7.1}$$

The concept of geometric invariance is intrinsically linked to group theory. An object or property is said to be invariant under a group of transformations if it remains unchanged when any transformation from the group is applied. Mathematically, for a group $G$ acting on a set $X$, a function $f : X \to Y$ is $G$-invariant if:

$$f(g \cdot x) = f(x) \quad \forall g \in G, x \in X \tag{7.2}$$

The power of group theory lies in its ability to provide a formal framework for understanding and constructing invariant representations. By identifying the relevant symmetry groups for Lenia creatures, we can design feature extraction methods or neural network architectures that respect these symmetries.

For example, convolutional neural networks (CNNs) inherently respect translation equivariance, meaning that a translation of the input results in a corresponding translation of the

feature maps. This property can be expressed group-theoretically as:

$$\text{CNN}(T_t \cdot x) = T_t \cdot \text{CNN}(x) \tag{7.3}$$

where $T_t$ represents a translation by $t$.

By extending this concept to the rotational groups, we can develop more general group equivariant models. These models would be capable of learning representations that are inherently invariant (through the use of global pooling operations) or equivariant to the relevant transformations, potentially leading to more efficient and robust analysis and generation of Lenia creatures.

## 7.2 Group-CNN Approach

Group Convolutional Neural Networks (G-CNNs) extend conventional CNNs by incorporating group theory to achieve equivariance to certain group actions such as rotations [40]. In our implementation, we leverage this property by pooling over the equivariant layers to learn an invariant representation.

The fundamental principle of G-CNNs is the replacement of the standard convolution operation with a group convolution. For a group $G$ (in our case, the group of 2D rotations), the group convolution is defined as:

$$f \star_G \psi = \sum_{h \in G} f(h)\psi(g^{-1}h) \tag{7.4}$$

where $f$ represents the input feature map, $\psi$ denotes the filter, and $g, h$ are elements of the group $G$.

For discrete rotations (0°, 90°, 180°, 270°), which form the cyclic group $C_4$, we can express this more concretely. Let $R_\theta$ denote a rotation by angle $\theta$. Then our group convolution becomes:

$$[f \star_{C_4} \psi](x, \theta) = \sum_{i=0}^{3} \sum_{y} f(R_{i\pi/2}y)\psi(R_{-\theta}(x - y)) \tag{7.5}$$

This operation ensures that the network's response is equivariant to rotations of the input. The input is transformed to each possible rotation, and then a conventional convolution is applied to these transformations with the same filter, creating four feature maps each corresponding to a different rotation of the original input [41], see Figure 7.1. When we rotate the input, the resulting feature maps are simply rotated versions of what we would have obtained from the original, non-rotated input. The network responds consistently to features regardless of their orientation in the input. This can be implemented by rotating the inputs or the filters. We chose the input approach, due to being a less complex implementation, despite being slightly less efficient.

By maintaining this property through the layers of the network, we ensure that the geometric structure of the data is preserved. This consistent behavior across rotations is precisely what we mean by equivariance. We note that rather than rotate the image, this same effect can be achieved by keeping a constant image and rotating the filter.

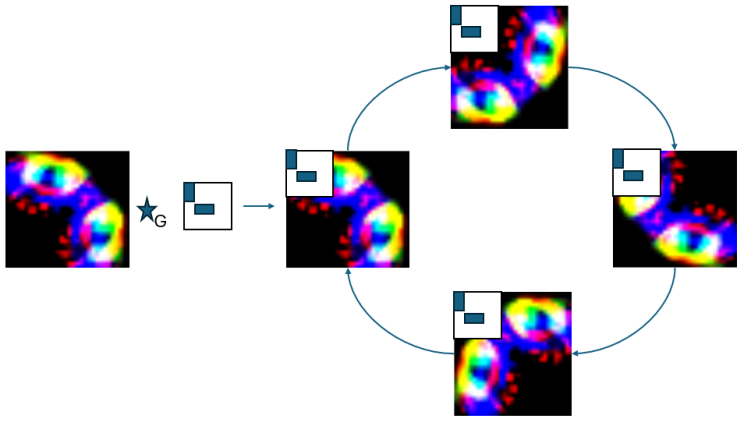

**Figure 7.1:** Figure showing a group convolution using image rotation. We apply the same filter to multiple 90 degree rotations of the same image. Diagram inspired by [41].

To achieve rotation invariance from this equivariant representation, we introduce a pooling operation over the group [40]. For our discrete rotation group $C_4$, this pooling operation can be expressed as:

$$\Phi(f)(x) = \max_{\theta \in \{0, \pi/2, \pi, 3\pi/2\}} [f \star_{C_4} \psi](x, \theta) \tag{7.6}$$

where $\Phi$ is our invariant mapping. The pooling operation results in rotation invariance because:

1. For any rotation $R_\alpha$ of the input, the equivariance property of the group convolution gives us:
$$[R_\alpha f \star_{C_4} \psi](x, \theta) = [f \star_{C_4} \psi](R_{-\alpha}x, \theta - \alpha) \tag{7.7}$$

2. Applying the max-pooling operation to this rotated input:

$$\begin{aligned} \Phi(R_\alpha f)(x) &= \max_\theta [R_\alpha f \star_{C_4} \psi](x, \theta) && (7.8)\\ &= \max_\theta [f \star_{C_4} \psi](R_{-\alpha}x, \theta - \alpha) && (7.9)\\ &= \max_{\theta'} [f \star_{C_4} \psi](R_{-\alpha}x, \theta') && (7.10)\\ &= \Phi(f)(R_{-\alpha}x) && (7.11) \end{aligned}$$

   Here, we've used the change of variables $\theta' = \theta - \alpha$, and the fact that taking the maximum over all rotations is invariant to a fixed rotation.

3. This demonstrates that rotating the input results only in a spatial transformation of the output, but the values remain unchanged:
$$\Phi(R_\alpha f) = R_\alpha \Phi(f) \tag{7.12}$$

We use manually defined JAX G-CNN layers to construct a replacement Encoder-Decoder architecture for the VAE of AURORA. We then perform experiments using this new architecture to assess the effectivenss of the G-CNN layer.

## 7.3 Results

We split the result sections into three parts. The first will discuss the impact of G-CNN layers when used with the standard AURORA VAE from the original paper, and the other two will look at how G-CNN layers impact the NT-Xent and Triplet Margin CL VAEs.

### 7.3.1 Standard VAE + G-CNN

We start by visualising the diversity metrics of the standard VAE architecture using G-CNN layers in the encoder and decoder. See Figures 7.2, 7.3.

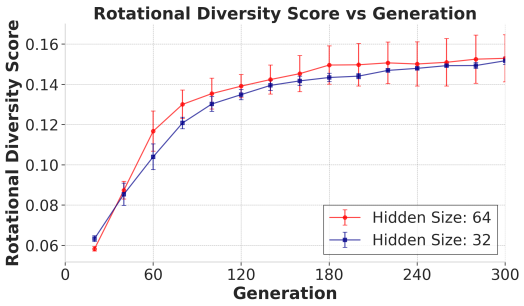


**Figure 7.2:** Rotational diversity score at every 20 generations using the standard VAE with G-CNN layers.

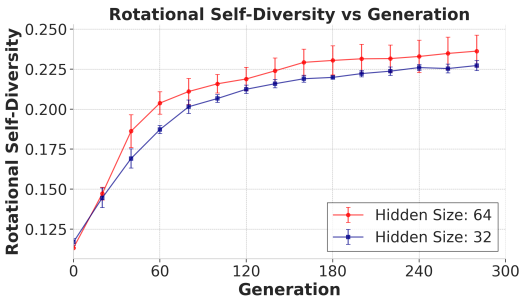


**Figure 7.3:** Rotational Self-Diversity score at every 20 generations using the standard VAE with G-CNN layers.

Similar to what was observed with the contrastive learning approaches, by trying to teach rotational invariance into the model, we seem to hamper the diversity of our outputs. However, when we perform qualitative analysis, we do see some interesting results.
Figure 7.4 shows us that while there are many duplicated creatures in the repertoire, the overall distribution is different compared to the benchmark repertoire. We shift away from the earlier dominance of blue in the soliton colour, instead seeing a lot more yellow. However, we still struggle to reach other colours beyond this. While these creatures may look reasonably complex, it is important to note that many of them are the same creatures who have just been manipulated slightly different over the Lenia timesteps. As our diversity metric takes the minimum value over multiple frames, it is likely that the metric is picking up that many of these creatures are the same. Furthermore, many of these creatures have large open areas, or are not too large in size. These creatures in general tend to score slightly less

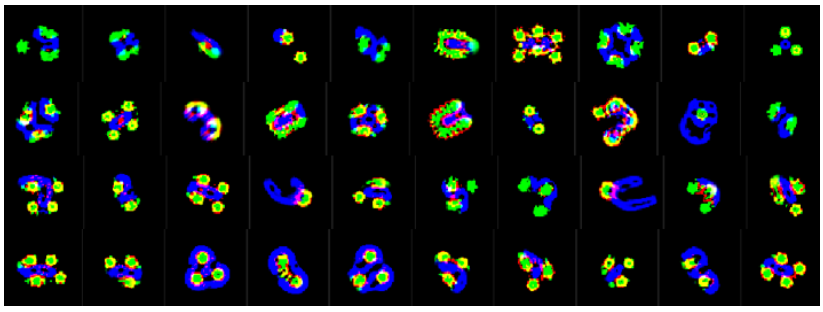

**Figure 7.4:** Rows 1-2: Standard VAE+G-CNN creatures with 32 dimensional hidden vector. Rows 3-4: Standard VAE+G-CNN creatures with 64 hidden dimensional vector

well on the diversity metrics, a known limitation of them.

This trade-off between rotational invariance and output diversity suggests a fundamental tension in the learning process. As the model becomes more adept at recognizing rotated versions of the same input as equivalent, it may be compressing its latent space in a way that reduces the variety of distinct outputs it can generate. This could indicate that achieving both high rotational invariance and high output diversity simultaneously may require more sophisticated architectural or training strategies.

### 7.3.2 NT-Xent VAE + G-CNN

Combining the G-CNNs with the NT-Xent objective causes a further decline in the diversity in generated output descriptors, which we show in Figures 7.5 and 7.6.

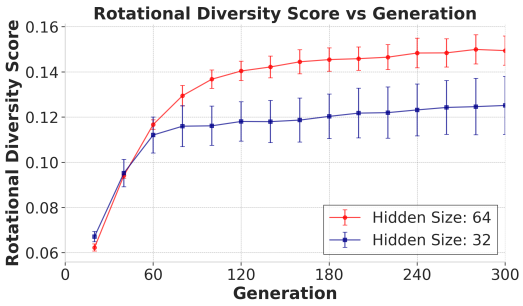


**Figure 7.5:** Rotational diversity score at every 20 generations using the NT-Xent VAE with G-CNN layers.

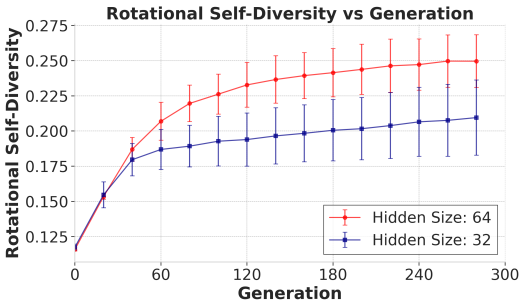


**Figure 7.6:** Rotational self-diversity score at every 20 generations using the NT-Xent VAE with G-CNN layers.

We see this effect is especially prevalent for the smaller hidden vector size of 32 with it performing worse than the hidden size of 64, which is interesting as Figure 6.7 showed us that without G-CNN layers the smaller hidden vector was more effective. This reversal in performance between hidden sizes suggests a complex interaction between the NT-Xent objective, G-CNN layers, and the dimensionality of the latent space. We visualise example creatures from these experiments, so that we can qualitatively analyse the repertoire.
As we would expect from our metrics, and previously seen NT-Xent behaviour, we lack any meaningful diversity in these samples. It becomes clear why the hidden vector size of 32 setup scored so poorly; not only do we mostly have duplicated creatures but they are also

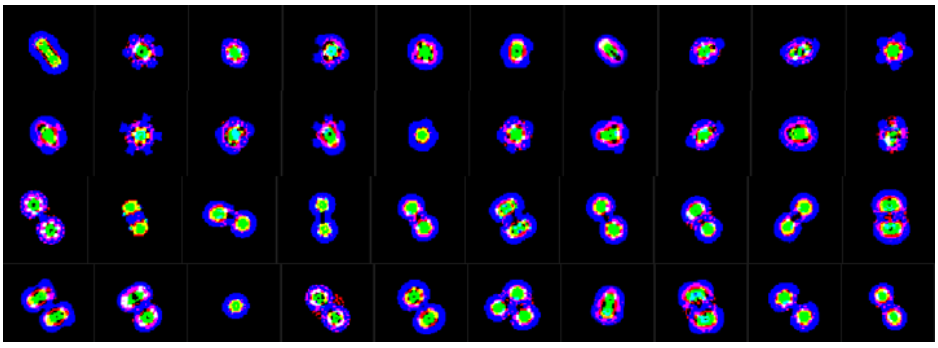

**Figure 7.7:** NT-Xent VAE+G-CNN creatures with 32 dimensional hidden vector. Rows 3-4: NT-Xent VAE+G-CNN creatures with 64 hidden dimensional vector

very small and, as we have presented already, smaller creatures do tend to perform a little worse on these metrics. The 64 hidden dimension run is a bit better, but it is still non-comparable to the standard VAE with and without G-CNN layers

It appears that the combination of NT-Xent and G-CNN might be overly constraining for smaller latent spaces, potentially leading to a collapse in the diversity of learned representations. The larger hidden size of 64 may provide enough capacity to maintain some level of diversity while still accommodating the rotational invariance imposed by the G-CNN layers and the contrastive learning from NT-Xent. This observation highlights the delicate balance required in designing architectures that aim to incorporate multiple learning objectives and structural biases.

### 7.3.3 Triplet Margin VAE + G-CNN

Similar to what was seen with NT-Xent, the G-CNNs with the Triplet Margin objective causes a further decline in the diversity in generated output descriptors, which we show in Figures 7.8 and 7.9.

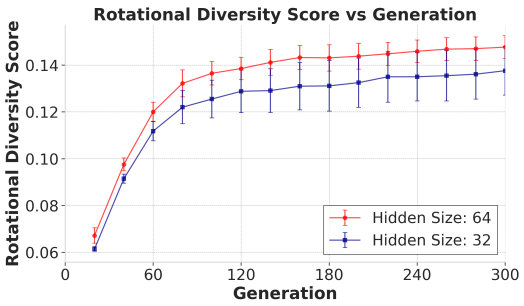


**Figure 7.8:** Rotational diversity score at every 20 generations using the Triplet Margin VAE with G-CNN layers.

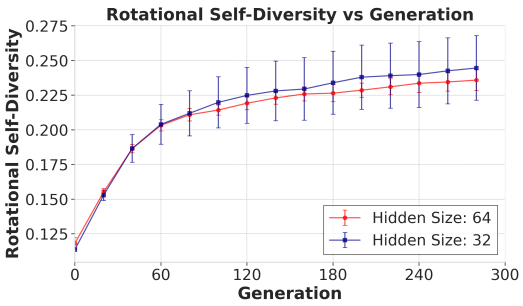


**Figure 7.9:** Rotational self-diversity score at every 20 generations using the Triplet Margin VAE with G-CNN layers.

We can presume similar interactions occur here as they do with the NT-Xent loss, and by simultaneous constraining the VAE using two different methods we bottleneck the ability to generate diverse descriptors in the latent space. In Figure 7.10 we visualise some resulting creatures.

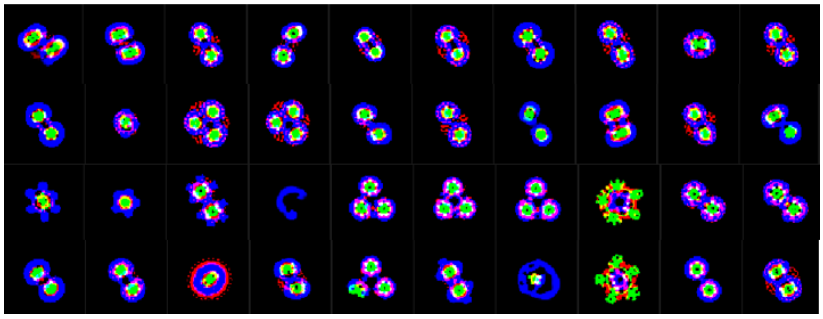

**Figure 7.10:** Rows 1-2: Triplet Margin VAE+G-CNN creatures with 32 dimensional hidden vector. Rows 3-4: Triplet Margin VAE+G-CNN creatures with 64 hidden dimensional vector

As we would expect, we see a collapse in diversity that is especially prominent for the hidden size 32 VAE. Interestingly, the creatures the algorithm has seemingly converged to are

different than the ones seen in Chapter 6, likely related to the new method of mapping key features of input data. The larger hidden vector does show more diversity in shape and colour, but we do see some near exact replicates in the repertoire (creatures 5, 6, 7 and then 9, 10) which will no doubt drag the average score of the repertoire down.

### 7.3.4 Overall

We conclude this result section by presenting direct comparisons of diversity scores of all the G-CNN approaches with the original benchmark set earlier in this project.

| Objective Function | Hidden Size | Diversity | | Self-Diversity | |
|---|---|---|---|---|---|
| | | Mean | Std Dev | Mean | Std Dev |
| Benchmark | 32 | 0.180 | 0.008 | 0.245 | 0.008 |
| | 64 | 0.185 | 0.003 | 0.244 | 0.004 |
| Standard + G-CNN | 32 | 0.152 | 0.002 | 0.227 | 0.003 |
| | 64 | 0.153 | 0.002 | 0.236 | 0.010 |
| NT-Xent + G-CNN | 32 | 0.125 | 0.013 | 0.210 | 0.027 |
| | 64 | 0.149 | 0.006 | 0.250 | 0.019 |
| Triplet Margin + G-CNN | 32 | 0.138 | 0.011 | 0.245 | 0.023 |
| | 64 | 0.148 | 0.005 | 0.236 | 0.008 |

**Table 7.1:** Comparison of Diversity and Self-Diversity measures across different Hidden Sizes for multiple Objective Functions using G-CNN layers

Our results demonstrate that G-CNN approaches consistently failed to improve diversity metrics in the generated repertoire, a finding corroborated by qualitative analysis. This outcome suggests that enforcing rotational invariance in embeddings constrains the latent space, impeding the generation of diverse outputs. These findings reveal a fundamental tension in generative model design for artificial life: a trade-off between invariance and diversity.

These results, while initially disappointing, offer valuable insights for future research. They underscore the need for novel architectures capable of more effectively balancing invariance and diversity, and suggest that alternative approaches to achieving increasing diversity, beyond rotation invariance, warrant exploration. In the context of artificial life, these findings highlight the complexity of designing systems that generate diverse yet structurally consistent forms, representing a fundamental challenge in the field.

## 7.4 Chapter Summary

This chapter explored the implementation of Group Convolutional Neural Networks (G-CNNs) in the context of generating diverse artificial life forms. G-CNNs extend conventional CNNs by incorporating group theory to achieve equivariance to certain group actions, such as rotations. The approach replaces standard convolution operations with group convolutions, ensuring that the network's response is equivariant to rotations of the input. To achieve rotation invariance, a pooling operation over the group was introduced, allowing the model to recognize and generate Lenia creatures regardless of their orientation.

The results of implementing G-CNNs, both standalone and in combination with contrastive learning objectives (NT-Xent and Triplet Margin), revealed a consistent pattern. Contrary to expectations, G-CNN approaches failed to improve diversity metrics in the generated repertoire. This finding reinforces a fundamental tension in generative model design for artificial life: a trade-off between invariance and diversity, first speculated in the contrastive learning section. The enforcement of rotational invariance in embeddings appeared to constrain the latent space, impeding the generation of diverse outputs.

# Chapter 8

# Discrete Latent Space for AURORA

## 8.1 Motivation

By implementing a discrete latent space, we hoped to enhance the quality of the unsupervised QD descriptors, which we anticipated would lead to a more diverse and meaningful range of solutions. Furthermore, we expected that this structure would naturally define neighborhoods and transitions between soliton behaviors, potentially facilitating more efficient exploration of the solution space.

Through these hypothesized benefits, we aimed to enhance AURORA's ability to discover, represent, and optimize a wide range of soliton behaviors. We hoped that the discrete latent space could lead to more structured and meaningful behavioral descriptors, driving more diverse and interesting discoveries.

## 8.2 Theoretical Concepts

### 8.2.1 Vector Quantization

Vector Quantization is a key concept in VQ-VAEs, which involves mapping continuous vectors to a discrete set of embedding vectors. Given an input vector $z_e(x)$, which is the output of the encoder, the vector quantization operation, denoted by $q(z)$, maps this input to the nearest embedding vector in the codebook (the embedding space) [42]. Mathematically, this operation is defined as:

$$q(z) = e_k, \quad \text{where } k = \text{argmin}_j \|z_e(x) - e_j\|_2 \tag{8.1}$$

In this equation, $k$ is the index of the closest embedding vector in the codebook, found by minimizing the Euclidean distance between the input vector $z_e(x)$ and each embedding vector $e_j$.

This process effectively discretizes the latent space, offering two advantages over continuous spaces [42]:

- It enables the model to learn more structured and interpretable data representations.
- It acts as a regularizer, helping to mitigate the issue of posterior collapse, a common problem in VAEs where the latent variables ignore the input data.

### 8.2.2 Discrete Latent Space

The discrete latent space in VQ-VAEs differs fundamentally from the continuous latent space in standard VAEs. It comprises a finite set of vectors (the codebook) rather than a continuous distribution. The posterior distribution in VQ-VAEs is expressed as:

$$q(z|x) = \begin{cases} 1 & \text{for } z = k \\ 0 & \text{otherwise} \end{cases} \tag{8.2}$$

where $k = \text{argmin}_j \|z_e(x) - e_j\|_2$

### 8.2.3 Mapping to the Discrete Latent Space

The process of mapping inputs to the discrete latent space in VQ-VAEs involves the following steps:

1. **Encoding**: The input $x$ is first passed through an encoder network to produce a continuous latent representation $z_e(x)$. The configuration files allow users to choose between a standard CNN encoder and a G-CNN encoder.

2. **Vector Quantization**: The continuous representation $z_e(x)$ is then quantized by mapping it to the nearest embedding vector in the codebook:

   $$z_q(x) = e_k, \quad \text{where } k = \arg\min_j \|z_e(x) - e_j\|_2$$

   Here, $k$ is the index of the closest embedding vector, found by minimizing the Euclidean distance between $z_e(x)$ and each embedding vector $e_j$.

3. **Straight-Through Estimator**: To allow gradients to propagate through the non-differentiable argmin operation, VQ-VAEs use the straight-through estimator. During the forward pass:

   $$z_q(x) = z_e(x) + \text{sg}[e_k - z_e(x)]$$

   where $\text{sg}[\cdot]$ denotes the stop-gradient operator, ensuring that gradients are not back-propagated through $e_k$.

4. **Gradient Computation**: In the backward pass, the gradients of the loss with respect to $z_q(x)$ are directly copied to $z_e(x)$:

   $$\frac{\partial L}{\partial z_e(x)} = \frac{\partial L}{\partial z_q(x)}$$

   This step enables gradient flow through the encoder despite the quantization step.

This mapping procedure allows VQ-VAEs to learn a discrete latent representation while maintaining end-to-end differentiability. The use of the straight-through estimator ensures that gradients from the decoder can flow back to the encoder, facilitating effective training of the entire model [42].

### 8.2.4 VQ-VAE Loss Functions

To effectively train VQ-VAEs and ensure proper learning of both the encoder and the codebook, two additional loss terms are introduced:

**Commitment Loss**

The commitment loss is defined as:

$$\mathcal{L}_{\text{commit}} = \|z_e(x) - \text{sg}[e_k]\|_2^2 \tag{8.3}$$

where sg$[\cdot]$ denotes the stop-gradient operator. This operator prevents gradients from flowing back into the codebook vectors during backpropagation, treating them as constants when computing gradients. Consequently, only the encoder parameters are updated based on this loss. The commitment loss encourages the encoder output to align with codebook vectors, preventing arbitrary growth of the encoder's output space.

**Codebook Loss**

The codebook loss is given by:

$$\mathcal{L}_{\text{codebook}} = \|e_k - \text{sg}[z_e(x)]\|_2^2 \tag{8.4}$$

This loss term updates the codebook vectors to better match the encoder outputs. Here, the stop-gradient operator is applied to the encoder outputs, ensuring that only the codebook vectors are updated based on this loss. The objective is to refine the codebook vectors to more accurately represent the encoder outputs, thereby making the codebook more representative of the underlying data distribution.

This combined loss, along with vector quantization, enables VQ-VAEs to learn effective discrete latent representations while maintaining high reconstruction quality. The interplay between these components allows VQ-VAEs to capture complex data structures in a discrete latent space, offering unique advantages in generative modeling applications.

## 8.3 VQ-VAE Hyperparameters

### 8.3.1 Number of Embeddings

The number of embeddings in a VQ-VAE, often denoted as K, represents the size of the codebook or the number of discrete codes available for encoding. This hyperparameter directly affects the model's capacity to represent different patterns in the data. A larger number of embeddings allows for more fine-grained representations but increases computational complexity and may lead to underutilization of some codes. Conversely, too few embeddings might result in loss of important details.

### 8.3.2 Commitment Cost

The commitment cost, typically represented by $\beta$, is a hyperparameter that balances the learning of the encoder and the codebook. It penalizes the encoder for producing embeddings that are far from the nearest codebook vector, encouraging the encoder to commit to codebook entries. A higher commitment cost forces the encoder outputs to stay closer to the codebook vectors, potentially leading to more stable training but possibly at the cost of reconstruction quality. Conversely, a lower commitment cost allows for more flexibility in the encoder but might result in less effective use of the codebook.

## 8.4 Perplexity

Perplexity is a measure of the effective codebook size. It is computed as:

$$\text{perplexity} = \exp\left(-\sum_{i=1}^{K} p_i \log p_i\right) \tag{8.5}$$

where $p_i$ is the probability of using the $i$-th codebook vector. A higher perplexity indicates that the model is using more of its codebook vectors effectively [43]. Beyond the standard VQ-VAE loss function, we also include a negative perplexity component, encouraging the model to use its codebook vectors more effectively.

## 8.5 Results

### 8.5.1 Reconstruction Quality

Our implementation of the VQ-VAE produces less clear reconstructions compared to the original Leniabreeder VAE, as shown in Figure 8.1. This reduction in clarity likely stems from several factors inherent to discrete latent representations. Vector quantization may struggle to capture fine details of complex Lenia patterns due to its limited codebook. The process of mapping continuous values to discrete codes can result in loss of subtle features. Additionally, VQ-VAE training challenges such as codebook collapse could impact reconstruction quality. The current hyperparameters may also not be optimally suited for Lenia's unique characteristics. To improve reconstruction fidelity, future work could focus on refining the VQ-VAE architecture specifically for Lenia systems, addressing these limitations through targeted modifications and optimizations.

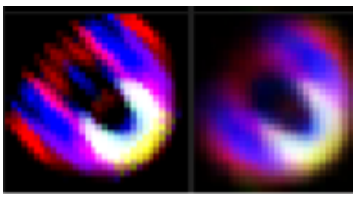

**Figure 8.1:** Creature reconstruction using the VQ-VAE. Image 1: Original input. Image 2: reconstruction.

### 8.5.2 Ability to Generate Diverse Life

In this section, we provide a qualitative analysis of the VQ-VAE's capacity to generate diverse life forms. Figure 8.2 displays image snapshots of the generated creatures. However, it is important to note that random samples from a larger dataset may not fully capture the complete diversity of the generated repertoire. Additionally, while we can assess structural diversity through random samples, behavioral diversity is not directly represented.

The VQ-VAE does generate a variety of different creatures, but it appears that its qualitative performance is quite similar to the benchmark, with perhaps slightly less diversity in colour. This observation is based on a qualitative assessment and comparison of random subsets of a broader sample. Despite these findings, the VQ-VAE still shows potential, as the codebook usage ratio is notably low, indicating that the model may not be fully leveraging the diversity

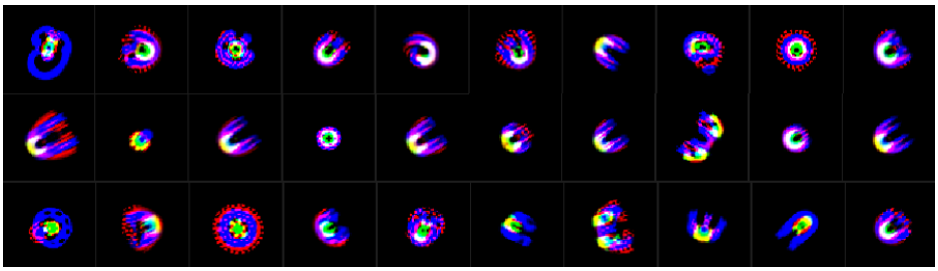

**Figure 8.2:** A selection of random creatures generated using the VQ-VAE, optimized for negative angle variance.

of its embeddings.

| Objective Function | Hidden Size | Diversity | | Self-Diversity | |
|---|---|---|---|---|---|
| | | Mean | Std Dev | Mean | Std Dev |
| Benchmark | 32 | 0.180 | 0.008 | 0.245 | 0.008 |
| | 64 | 0.185 | 0.003 | 0.244 | 0.004 |
| VQ-VAE | 32 | 0.150 | 0.010 | 0.228 | 0.011 |
| | 64 | 0.151 | 0.009 | 0.228 | 0.012 |

**Table 8.1:** Comparison of Diversity and Self-Diversity measures across different Hidden Sizes for the VQ-VAE and benchmark. We take the peak diversity values over the generations.

There is potential for improvement if the VQ-VAE's hyperparameters/architecture are optimized to encourage more diverse codebook usage. Therefore, while the current results do not demonstrate an advantage over the original implementation, there is potential for higher performance with further hyperparameter and architectural experimentation. It could also be interesting to try and directly incorporate the VQ-VAE's discrete latent space as a replacement for the archive of AURORA, but that could be done outside of the context of Lenia.

## 8.6 Chapter Summary

This chapter examines the implementation of a discrete latent space in VQ-VAEs to enhance unsupervised QD descriptors for cellular automata. Vector Quantization maps continuous vectors to a discrete set of embedding vectors, aiming to improve interpretability and mitigate posterior collapse.

The VQ-VAE architecture involves encoding, vector quantization, and a straight-through estimator for gradient computation. Training uses a composite loss function including commitment, codebook, and reconstruction losses. Key hyperparameters are the number of embeddings and commitment cost, with codebook usage ratio and perplexity as primary metrics.

We show that the model can generate some diversity in the creatures it discovers, but it does not outperform the original Leniabreeder implementation.

# Chapter 9

# Lenia-Specific Mutation Operators

## 9.1 Motivation

We present three novel mutation operators that leverage the properties of Lenia to further increase diversity in our repertoires. We also present a fourth operator, that acts as a method to use the other three operators simultaneously. For each generation, we perform the mutation on half of the repertoire creatures, and the previously discussed iso+lineDD variation operator on the other half. We do use an increased isotropic value, of $\sigma_{iso} = 0.05$ as we know this also improves diversity, but we do not change $\sigma_{line}$. We choose not to increase these values further, as we want to be able to compare the results of this chapter to those in chapter 4, as well as the standard repertoire in chapter 3.

## 9.2 Kernel Parameters

We remind the reader of the non-fixed parameters in Lenia's genotype. The weighting of each kernel, $h_k$, the mean of Lenia's growth function, $\mu_k$, and the standard deviation of Lenia's growth function, $\sigma_k$. Here $k$ denotes that each kernel has its own non-fixed parameters. We explore mutation operators that act on these parameters, and therefore the rules of Lenia.

## 9.3 Kernel Interpolation Mutation

The kernel Interpolation mutation aims to introduce diversity in the Lenia patterns by exchanging information between different kernels. This operator selects two random kernels and interpolates their parameters, creating a smooth transition between their characteristics. The interpolation factor is randomly chosen for each parameter, allowing for varied levels of information exchange. This approach maintains the overall structure of the kernels while introducing novel combinations of their properties. We initially wished to propose this as a direct kernel swap, but we found this to be too unstable because it led to creatures vanishing. Figure 9.1 shows, even in a small sample size, much greater diversity in creature morphology and colour than we could extract using just the variation operator. We see varying sizes, and while there is still a bias towards blue/ yellow creatures we see other colours, such as pink, arise. We note that these creatures, and those that follow in this chapter, are totally random samples from the repertoire.

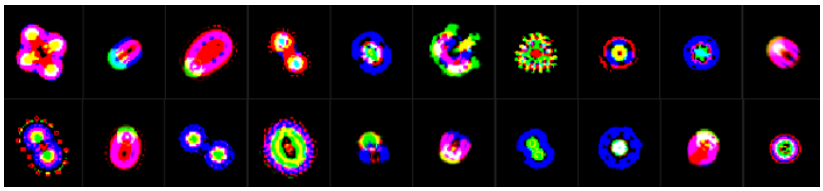

**Figure 9.1:** 20 creatures discovered using the Kernel Swap mutation operator.

## 9.4 Kernel Shift Mutation

This operator applies a partial, parameter-wise shift to the kernel parameters. The shift amount is randomly determined for each parameter, ensuring a conservative modification that preserves some original kernel structure. This is similar in effect to the kernel interpolation mutation, but means that kernel parameter values can also move away from each other. This mutation allows for the exploration of slightly altered spatial configurations while maintaining the core characteristics of the original pattern.

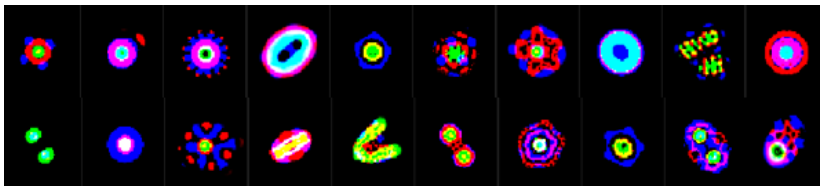

**Figure 9.2:** 20 creatures discovered using the Kernel Shift mutation operator.

Figure 9.2 shows a major increase in observed repertoire diversity compared to our original benchmarks. We see many different structures and colours. Even in this small set of 20 creatures, we see huge novel diversity, showing that we can use this mutator to generate new creatures.

## 9.5 RGB Channel Swap Mutation

The RGB channel swap mutation enhances pattern diversity in Leniabreeder by addressing the convergence of colour schemes in evolved Lenia patterns. This mutation randomly permutes the red, green, and blue channels of the seed pattern in each individual's genotype. By preserving spatial structure while introducing colour variations, it uncovers new pattern dynamics and behaviors, expanding the range of evolvable artificial life forms.

This approach balances exploration and exploitation, allowing the Quality-Diversity algorithm to generate patterns that may look structurally similar but behave differently due to altered colours.

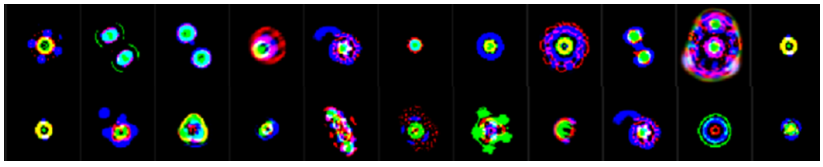

**Figure 9.3:** 20 creatures discovered using the RGB Channel Swap mutation operator.

The effectiveness of this mutation may be attributed to its ability to encourage pattern reorganization without directly adding mass to the creature, unlike the gaussian component of the Iso+LineDD variation operator. This allows for the evolution of a wider array of persistent creatures, avoiding convergence to large, unevolvable forms.

## 9.6 Random (Adaptive) Mutation Operator

The Random Mutation Operator dynamically combines the Kernel Interpolation, Kernel Shift, and RGB Channel Swap. For each individual assigned to the mutation operator every generation, it randomly selects one of these three mutation types. This approach leverages the unique benefits of each mutation: kernel parameter interpolation, spatial reconfiguration, and colour permutation. By dynamically adjusting the mutation strategy, it complements the increased isotropic sigma in the variation operator, enhancing pattern diversity and exploration of the Lenia morphospace. This flexible meta-strategy aims to produce more diverse and robust repertoires of Lenia creatures while balancing structural modifications and colour variations.

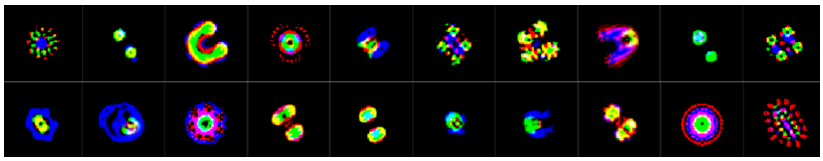

**Figure 9.4:** 20 creatures discovered using the Random mutation operator.

Figure 9.4 shows that while we do have diverse shapes and colours, this approach does not necessarily generate more diversity than using the mutation operators individually. We do, however, anticipate this operator being the most effective when experimenting at scale. We hypothesise that each individual mutator may eventually converge to a set of extremely high quality solutions. However, if we combine the mutators we will hopefully explore many other different evolutionary trees before we reach that point.

## 9.7 Chapter Summary

To conclude this chapter, we have shown that through the use of novel mutation operators we are able to discover extremely diverse repertoires of new creatures. These findings suggest that initial beliefs that the VAE in AURORA constituted the bottleneck were incorrect, and rather the evolutionary bottlenecks were actually byproducts of Lenia's own dynamics.

We also address that we have not used much quantitative analysis in this section. We believe that the qualitative results show significant diversity to demonstrate the efficacy of these methods. Furthermore, all of the creatures displayed in the appendix were generated using these operators, so for more examples we direct the reader there.

# Chapter 10

# Automated Discovery

## 10.1 Motivation

Leniabreeder serves as a stepping stone in fulfilling one of the main challenges of Lenia, efficient and automatic creature discovery. The original implementation could discover up to N amount of creatures, where N is the maximum size of the QD repertoire. Compared to existing methods, such as using random search to discover Lenia creatures, this is very efficient. Even in this project, without the addition of any new mutation operators or any other new additions, we have shown that the original Leniabreeder implementation can have its parameters tuned to return highly diverse results. However, in any given run we could at a maximum discover N new creatures. We propose an extension whereby we can discover N*M new creatures, where N is the size of the repertoire and M is the number of iterations of the new method. We show that each iteration will not just yield more creatures, but these creatures will be continuously diverse.

## 10.2 Continuous Discovery

We propose an extension whereby we exploit the knowledge that different starting patterns can result in highly diverse evolutionary trees. Rather than end the process after a given number of generations, we save all discovered creatures' genotype information and then wipe the repertoire in an extinction event. We then randomly choose one of these new creatures and start the process again, and in doing so we show that we can discover novel new creatures in mass. To reiterate, this novelty comes from the unique evolutionary trees offered by different creatures. This process can be repeated an infinite, or defined, number of times. While this would be computationally expensive, we hypothesise it could be one of the most efficient ways to navigate the search space of possible Lenia creatures. We visualise this process in Figure 10.1.

### 10.2.1 Extinction Events

We observe at times that in evolutionary algorithms, the evolutionary pressure of the system can become overwhelming and as such massively constrain the diversity in new creatures [44]. Leniabreeder is a QD algorithm, meaning there is a focus on both optimality and diversity. However, for this particular research we are arguably more interested in diversity than optimality, and we believe it is possible that the optimality component constrains the

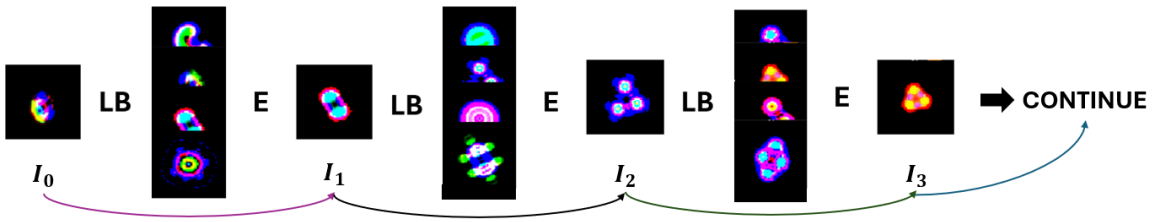


**Figure 10.1:** At each iteration, $I_i$, we perform the Leniabreeder process, LB, to generate a repertoire of creatures. We then perform an extinction event, E, to restart the repertoire with only one survivor. This process is then continuously repeated.

diversity objective.

By performing an extinction event, we clear the repertoire of any existing evolutionary pressures or constraints that existed in the previous iterations. This allows new creatures to appear that would have otherwise never survived in the original repertoire. These new creatures could in theory then evolve into descendents that would have survived in the previous repertoire as well. This mimics phenomena observed in biological life; humans are the dominant species of our era, but we likely would have struggled in a world dominated by other evolutionary pressures exerted by other creatures, such as dinosaurs.

### 10.2.2 Iterative adaption

We stress that this process is not simply rerunning Leniabreeder with a new starting pattern, but rather a true iterative process that allows for the Leniabreeder configuration to be adapted on-the-fly. At each iteration, we can dynamically change the fitness metric to cover new search space or change the phenotype size. The latter is highly useful, as it allows us to further investigate the phenomena observed in Section 4.4. As a reminder, this was the investigation into how phenotype sizing affects creature complexity. We find that starting with a large phenotype results in multi-creature systems, as opposed to single or multi-celled individual creatures. Using Automatic-Leniabreeder, we can slowly increase the phenotype size at each iteration, allowing for a more stable evolutionary process. In using an iterative process, we also define a clear evolutionary trajectory, and we do observe that each subsequent iteration's repertoire presents clear traits of its initial ancestor.

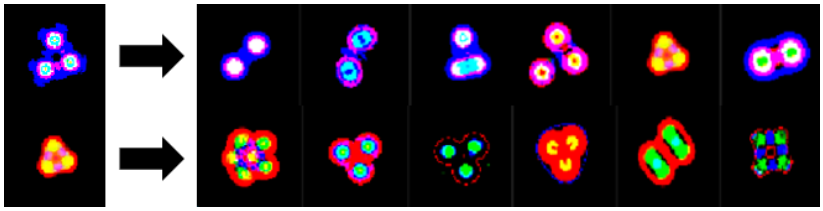

**Figure 10.2:** We observe that evolved repertoires often exhibit characteristics of their ancestor, such as shape or colour. We do see some variance though, which is important for further novel evolution.

## 10.3 Returning to Phenotype Size

In section 4.4, we showed that using the original Leniabreeder implementation, it was hard to evolve large self-organising creatures when we used a large phenotype. Instead we went on to show that multi-creature systems consisting of duplicates of a single small creature was more common. We postulated that if we could instead incrementally increase the phenotype, after starting from a smaller size, then we could see the emergence of large self-organising creatures. With this increase in size, we would then hope that there would be more space for diverse creatures. Incorporating this into the original Leniabreeder architecture presents a non-trivial challenge, as updating the phenotype size mid-way through the QD process would require re-instantiation of elements such as the VAE. While transfer learning could be employed to preserve information between updates, it would remain a challenging endeavor.

Automatic Leniabreeder makes this much simpler. As we iterate around the entire Leniabreeder process, we can increment the phenotype at the end of each iteration without having to reinitialise everything additional times. We present here some results from this iterative process, where we start with a phenotype slice of 32x32x3 and increment this up by 4 at each time step (i.e. iteration 2 will have a size of 40x40x3). We randomly choose the creature to survive the extinction event to be evolved at a new size. We start at iteration 0, as this effectively acts as an initialisation step for the process.

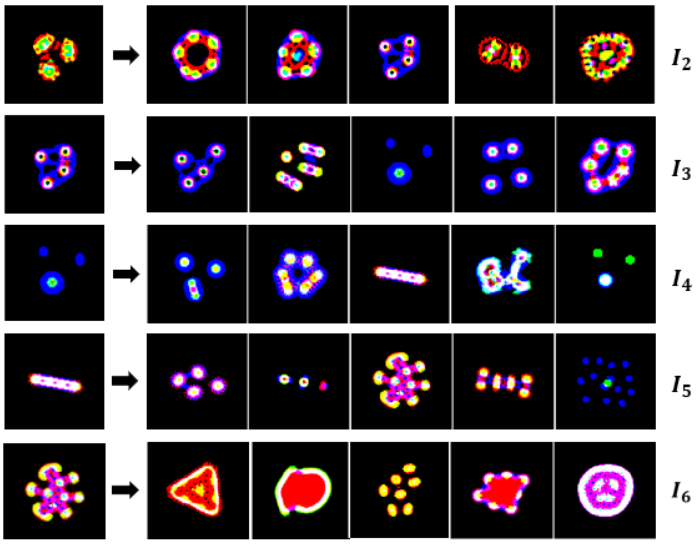


**Figure 10.3:** Iterative evolution at iterations $I_2$ to $I_6$ of the process. Phenotype sizes range from 40x40x3 to 56x56x3. Each iteration increases phenotype height and width by 4. We note that the world slice taken for each image is double the phenotype, which is why all soliton/multi-soliton snapshots look to be the same scale.

Figure 10.3 displays how slowly increasing the complexity can reveal the discovery of new, large self-organising creatures. We do, however, see cases whereby we have unconnected creatures and not true self-organisation to the correct scale, such as the second last image of $I_3$ and the last image of $I_5$. Nonetheless, this approach does clearly unlock levels of complexity in individual creatures not observed when simply starting with a large phenotype size. This justifies the use of an Automatic Leniabreeder, as it means we can more stably explore new areas of the possible creature morphology space.

We also note that the ability to self-organise seems very dependent on the ancestor phenotype. Should this phenotype be multiple solitons, or a long and thin creature, we tend to observe that its offspring will mostly be multi-creature systems. Of course, the benefit of randomly selecting our surviving creature means that we may select an outlier creature. Consider again Figure 10.3, we see that $I_5$ has mostly multi-creature systems, but we selected a large singular self-organising pattern to persist. As a result, we observe many large self-organising creatures that do display an element of increasing complexity.

As a final point of interest, larger phenotype creatures also at times show much more vibrant colour transitions than smaller images, as seen in Figure 10.4.

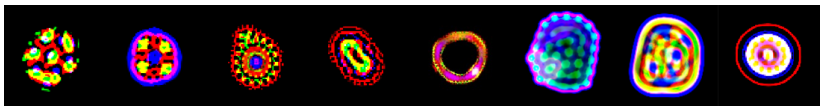

**Figure 10.4:** Creatures 1-4: Phenotype size $\leq$ 44x44x3. Creatures 5-8: Phenotype size $>$ 44x44x3. We show larger creatures tend to have smoother colour boundaries.

The emergence of more vibrant color transitions in larger phenotypes further underscores how increased spatial resolution can lead to richer, more complex patterns. This scale-dependent behavior echoes the principles of emergence in complex systems, where new properties and behaviors arise at different levels of organization.

## 10.4 Possible Limitations

This approach is not without potential limitations/pitfalls. It is theoretically possible that due to the random selection of surviving solitons, we could choose very similar solitons at each timestep. This could mean that over the course of all the iterations, each new repertoire does not have the sort of diversity we would otherwise hope to achieve. We can think of this as kind of a diversity collapse, though we could change the fitness function at each iteration to avoid this, and in doing so search different evolutionary spaces. We do propose a slightly different structure, show in Figure 10.5, for Automatic Leniabreeder that we believe could negate this risk.

Rather than iteratively choose a soliton from the previous iteration to persist after the extinction event, we create a bank of all creatures ever discovered and draw from here. This makes it less likely that we converge onto evolutionary paths over time, or at least that it takes substantially longer to do so. We note that we do not implement the approach described in Figure 10.5, and instead highlight it as future work for Leniabreeder.

## 10.5 Open-Endedness

We revisit the Open-Ended Evolution (OEE) category we identified at the beginning of this paper, specifically the emergence of interesting new types of creatures. We recognize our Automatic, adaptable Leniabreeder as am advancement toward this goal. Our research demonstrates that this algorithm, by incorporating extinction events and adaptive configuration changes, is capable of discovering new and diverse artificial creatures on an even

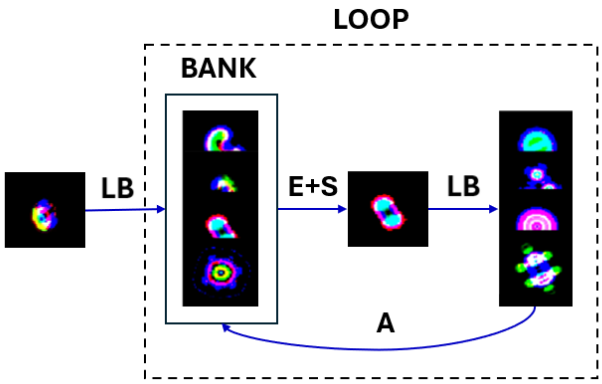


**Figure 10.5:** Rather than iteratively evolve from the last repertoire, we store discovered creatures in a bank (A), and draw from the collectively discovered samples at each time step after extinction (E+S). This approach sacrifices directional evolution for more long-term stability.

larger scale than the original method, especially over long time scales.

While we cannot definitively claim that this algorithm will discover an infinite variety of new stable creatures, we believe it exhibits a high degree of open-endedness. Proving that the algorithm would converge over time would be extremely challenging due to the time and compute required. This is particularly true when considering the alternative implementation outlined in Figure 10.5.

Our findings suggest that the Automatic, adaptable Leniabreeder could represent a substantial step forward in the field of artificial life and open-ended evolution, opening up new avenues for further research and exploration.

# Chapter 11

# Project Evaluation, Conclusion, and Future Work

## 11.1 Key Results

The key results of this project include:

- **Diverse evolutionary trees**: We showed that using different starting creatures allows us to explore novel evolutionary trees. Furthermore, we show that these ancestors can have varying suitability for different niches.

- **Enhanced diversity through parameter tuning:** We demonstrated that appropriate tuning of the variation operator parameters can significantly enhance creature diversity without major architectural changes.

- **Novel mutation operators:** We introduced three Lenia-specific mutation operators - Kernel Swap, Kernel Shift, and RGB Channel Swap - which substantially increased repertoire diversity without compromising creature quality. An adaptive mutation operator combining these three also showed promising results.

- **Improved evaluation tools:** We developed enhanced evaluation metrics, including rotational diversity scores and interactive t-SNE visualizations, enabling more comprehensive analysis of creature diversity across different models and evolutionary timescales.

- **Insights on geometric invariance:** Our experiments with contrastive learning approaches (NT-Xent and Triplet Margin Loss) and Group Convolutional Neural Networks revealed that enforcing rotational invariance can actually constrain diversity in evolved creatures, contrary to initial expectations.

- **VQ-VAE implementation:** While not outperforming the original VAE, our Vector Quantized VAE implementation was able to generate diverse creature repertoires. By making better use of the codebook, we could see more diverse descriptors to be used in AURORA.

- **Automated discovery process:** We developed an iterative, automated discovery process incorporating extinction events and adaptive configurations, allowing for the exploration of larger phenotype sizes and potentially more complex creatures.

- **Progress towards open-ended evolution:** Our Automatic Leniabreeder demonstrated a high degree of open-endedness, capable of discovering diverse artificial creatures on a larger scale than the original method.

These results collectively represent significant advancements in the Leniabreeder framework, offering new insights into artificial life evolution within Lenia and opening avenues for further research in open-ended evolution and complex system modeling.

## 11.2 Critical Evaluation

This project has made enhancements to Leniabreeder, but it's important to critically assess both the achievements and limitations.

The enhanced evaluation metrics and interactive visualizations greatly improved our ability to analyze evolved creatures. Yet, the acknowledgment that these metrics can sometimes be "cheated" highlights an ongoing challenge in quantitatively assessing diversity in complex systems like Lenia. This suggests that more robust, cheat-proof metrics could be a valuable focus for future work. Ultimately, it is likely that research like this favours qualitative work.

The implementation of an iterative, automated discovery process with extinction events shows promise for exploring larger phenotype sizes and more complex creatures. However, its computational intensity and time demands could limit practical applicability. Nonetheless, it represents a unique tool in exploring vast areas of the search space.

Despite significant effort, the project was unable to fully resolve the issue of rotational invariance. Attempts with new data augmentations, contrastive learning and G-CNNs actually led to decreased diversity, suggesting that alternative approaches to achieving geometric invariance should be explored, or its importance in this context reconsidered. Indeed, later experiments in the project with features such as the mutation operators imply that the VAE was never the bottleneck despite the initial hypothesis.

The VQ-VAE implementation, while a novel addition to the framework, did not outperform the original VAE. This outcome indicates that discrete latent spaces may not be inherently superior for representing Lenia creatures.

While the project demonstrated a commendable breadth of experimentation, this broad focus may have come at the expense of deeper exploration in certain areas. For instance, more time could have been devoted to investigating aspects more closely related to Lenia itself, such as mass conserving Flow Lenia or VQ-VAE fine tuning.

## 11.3 Conclusion

This project has extended Leniabreeder, yielding valuable insights into artificial life evolution within Lenia. Our exploration of varied starting patterns and hyperparameter tuning has illuminated key factors influencing the diversity and complexity of evolved creatures. While our pursuit of geometric invariance through contrastive learning and group theory-inspired approaches did not fully resolve rotational duplication challenges, it has laid impor-

tant groundwork for future research in this area.

The development of novel, Lenia-specific mutation operators stands as a particular success of this work. The Kernel Swap, Kernel Shift, and RGB Channel Swap mutations, especially when combined in the Adaptive Mutation Operator, have demonstrably enhanced our ability to generate diverse and interesting creatures without compromising quality. This breakthrough suggests new avenues for expanding diversity in artificial life systems.

A significant advancement in this project is the introduction of the Automatic Leniabreeder. This iterative process, incorporating extinction events and adaptive configurations, has shown great potential for discovering diverse and complex creatures on a larger scale than previously possible. By allowing for dynamic changes in fitness metrics and phenotype sizes between iterations, the Automated Leniabreeder enables the exploration of evolutionary trajectories that were previously inaccessible. This approach not only increases the efficiency of creature discovery but also represents a step towards achieving open-ended evolution, under the definition of being able to create new and interesting creatures, within Lenia.

Our enhanced evaluation tools, including interactive visualizations and new diversity metrics, have not only improved our ability to analyze results but also deepened our understanding of diversity within Lenia. However, the limitations of these metrics highlight the ongoing challenge of quantitatively assessing diversity in complex systems.

The implementation of the VQ-VAE architecture, while not outperforming the original VAE, has opened up new possibilities for representation learning in Lenia. This, along with our experiments in automated discovery and iterative evolution, points towards promising directions for future research.

In summary, this project has expanded the capabilities of the Leniabreeder framework and opened exciting new paths for research in open-ended evolution and artificial life. While challenges remain, particularly in achieving geometric invariance and optimizing computational efficiency, the advancements made here provide a solid foundation for future explorations in this fascinating field. As a final note, we include a curated zoo of some of the most interesting creatures from the project in the appendix for the readers viewing.

## 11.4 Future Research

### 11.4.1 Mass Conservation/Flow Lenia

Flow Lenia is an extension of Lenia that introduces mass conservation principles to create new dynamics and enhance the diversity of evolved patterns. In Flow Lenia, cell states represent concentrations of matter, and the system ensures that the total mass is conserved over time. This mass-conservation feature helps prevent the instability issues often seen in the original Lenia, such as patterns vanishing or growing uncontrollably.

A key innovation in Flow Lenia is the introduction of a flow field, which controls the movement of matter between cells. This flow is influenced by the original Lenia update rules and includes a diffusion mechanism to prevent excessive concentrations. As a result, Flow Lenia tends to produce more stable and complex patterns, which could lead to the evolution of

more diverse and intricate creatures.

Flow Lenia also opens the door to multi-species simulations, where different update rules can coexist within the same environment. This could lead to the discovery of complex ecosystems with multiple interacting species, expanding the potential of evolutionary exploration in Lenia.

Although we have made initial attempts to implement Flow Lenia, there are still some unknown errors that need to be addressed. Further investigation and refinement will be necessary to fully integrate this approach into the Leniabreeder platform.

### 11.4.2 Totally random initialisation

Another direction for future research is the exploration of totally random initialization. While the current approach relies on pre-existing patterns or creatures as evolutionary starting points, adopting a completely randomized initialization strategy could potentially unlock even greater diversity in evolved creatures within Lenia.

This approach would involve generating random initial states for the Lenia world, rather than beginning with known stable patterns. Additionally, kernel parameters would be randomly initialized instead of using pre-defined configurations. By starting from this blank slate, we could reduce the bias inherent in using pre-existing patterns, allowing for more open-ended evolution. This method might enable the discovery of new stable patterns that are not easily reached from known starting points, and the increased diversity in the initial population could lead to more varied evolutionary trajectories.

However, implementing totally random initialization is not without challenges. Many randomly initialized patterns might quickly dissolve or explode, necessitating the development of efficient filtering mechanisms to identify promising candidates. The search space would also be significantly larger, potentially requiring more computational resources. Moreover, evolving stable, complex patterns from completely random starting points may prove more difficult than building upon known, stable configurations. We did attempt to do this as part of this project, but were not successful. Our approach involved a random world state, whereby we used our phenotype centering function to focus on the center of mass after N time steps. We were unable to find a stable starting pattern this way, and as such more effective ways to snapshot stable patterns after initialisation present an avenue for future research.

To address these challenges, future research could focus on developing constraints or heuristics to guide random initialization towards more promising starting points. Creating adaptive mechanisms to quickly identify and cultivate potentially interesting random patterns would also be crucial. Another approach could involve combining random initialization with the existing method, perhaps using a mixed population of known and random patterns to balance exploration and exploitation.

# Bibliography


[1] John Von Neumann, Arthur W Burks, et al. Theory of self-reproducing automata. *IEEE Transactions on Neural Networks*, 5(1):3–14, 1966. pages 2, 6

[2] Martin Gardner. The fantastic combinations of john conway's new solitaire game "life". *Sc. Am.*, 223:20–123, 1970. pages 2

[3] Stephan Rafler. Generalization of conway's "game of life" to a continuous domain-smoothlife. *arXiv preprint arXiv:1111.1567*, 2011. pages 2, 7

[4] Bert Wang-Chak Chan. Lenia-biology of artificial life. *arXiv preprint arXiv:1812.05433*, 2018. pages 2, 7

[5] Oxford Reference. Homeostasis. `https://www.oxfordreference.com/view/10.1093/oi/authority.20110803095942859`, 2024. Accessed on September 7, 2024. pages 2

[6] Yujin Tang, Yingtao Tian, and David Ha. Evojax: Hardware-accelerated neuroevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 308–311, 2022. pages 2, 11, 20

[7] Maxence Faldor and Antoine Cully. Toward artificial open-ended evolution within lenia using quality-diversity. *ALIFE*, 2024. pages 2, 3, 6, 7, 11, 12, 13

[8] Morgan Giroud. Leniax. `https://leniax.readthedocs.io/en/latest/`, 2024. Accessed: 20/05/2024. pages 2, 7, 11

[9] Bert Chan and Will Cavendish. Open science and open source projects for geometric automata. `https://openlenia.github.io/`, 2024. Accessed: 2024-09-01. pages 2

[10] Vassiiis Vassiliades and Jean-Baptiste Mouret. Discovering the elite hypervolume by leveraging interspecies correlation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 149–156, 2018. pages 3, 13, 24

[11] John Maynard Smith and Eors Szathmary. *The major transitions in evolution*. OUP Oxford, 1997. pages 6

[12] Mark A Bedau. Artificial life: organization, adaptation and complexity from the bottom up. *Trends in cognitive sciences*, 7(11):505–512, 2003. pages 6

[13] Erwan Plantec, Gautier Hamon, Mayalen Etcheverry, Pierre-Yves Oudeyer, Clément Moulin-Frier, and Bert Wang-Chak Chan. Flow-lenia: Towards open-ended evolution in cellular automata through mass conservation and parameter localization. In *Artificial Life Conference Proceedings 35*, volume 2023, page 131. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , 2023. pages 6

[14] Joel Lehman and Kenneth O Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 211–218, 2011. pages 6

[15] Christopher G Langton. Studying artificial life with cellular automata. *Physica D: nonlinear phenomena*, 22(1-3):120–149, 1986. pages 6

[16] Bert Wang-Chak Chan. Lenia and expanded universe. In *Artificial Life Conference Proceedings 32*, pages 221–229. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , 2020. pages 7

[17] Bert Chan. Lenia evojax example. `https://github.com/google/evojax/blob/main/examples/notebooks/LeniaEvoJAX.ipynb`, 2022. Accessed: 2024-09-01. pages 7, 11

[18] Bert Wang-Chak Chan. Towards large-scale simulations of open-ended evolution in continuous cellular automata. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pages 127–130, 2023. pages 8

[19] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015. pages 8

[20] Antoine Cully and Yiannis Demiris. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22(2):245–259, 2017. pages 8

[21] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015. pages 9

[22] Antoine Cully. Autonomous skill discovery with quality-diversity and unsupervised descriptors. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 81–89, 2019. pages 10, 23

[23] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. pages 10

[24] Norman Packard, Mark A Bedau, Alastair Channon, Takashi Ikegami, Steen Rasmussen, Kenneth O Stanley, and Tim Taylor. An overview of open-ended evolution: Editorial introduction to the open-ended evolution ii special issue. *Artificial life*, 25(2):93–103, 2019. pages 12

[25] Tim Taylor, Mark Bedau, Alastair Channon, David Ackley, Wolfgang Banzhaf, Guillaume Beslon, Emily Dolson, Tom Froese, Simon Hickinbotham, Takashi Ikegami, et al. Open-ended evolution: Perspectives from the oee workshop in york. *Artificial life*, 22(3):408–423, 2016. pages 12

[26] Alan Turing. The chemical basis of morphogenesis. *Bulletin of mathematical biology*, 52:153–197, 1990. pages 14

[27] Sean T Vittadello, Thomas Leyshon, David Schnoerr, and Michael PH Stumpf. Turing pattern design principles and their robustness. *Philosophical Transactions of the Royal Society A*, 379(2213):20200272, 2021. pages 14

[28] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. pages 14, 17

[29] Bert Wang-Chak Chan. Lenia-ce. `https://github.com/Chakazul/chakazul.github.io/tree/master/lenia-CE`, 2023. Accessed on 26/08/2024. pages 20

[30] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. pages 28, 30

[31] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-sne effectively. *Distill*, 1(10):e2, 2016. pages 30

[32] Cambridge in Colour. Understanding digital image interpolation. `https://www.cambridgeincolour.com/tutorials/image-interpolation.htm`, 2020. Accessed: 2024-09-07. pages 31

[33] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. pages 36, 37

[34] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015. pages 36, 37, 38

[35] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020. pages 37

[36] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29, 2016. pages 37

[37] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009. pages 38

[38] David S Dummit and Richard M Foote. *Abstract algebra*. John Wiley & Sons, Hoboken, NJ, 3 edition, 2004. pages 45

[39] Brian C Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*, volume 222 of *Graduate Texts in Mathematics*. Springer, Cham, 2 edition, 2015. pages 45

[40] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016. pages 46, 47

[41] Casper van Engelenburg. Geometric deep learning: Group equivariant convolutional networks. `https://medium.com/swlh/geometric-deep-learning-group-equivariant-convolutional-networks-ec687c7a7b41`, 2020. Accessed: 26/08/2024. pages 46, 47

[42] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. pages 53, 54

[43] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019. pages 56

[44] Joel Lehman and Risto Miikkulainen. Enhancing divergent search through extinction events. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, page 951–958, New York, NY, USA, 2015. Association for Computing Machinery. pages 61

# Appendix

## Configurations

Find below a table of the standard configuration for each implementation of Leniabreeder.

### Standard Leniabreeder

| Aurora Configuration I | |
|---|---|
| *General* | |
| seed | 2 |
| pattern_id | ”5N7KKM” |
| R, T | 13, 10 |
| *Simulation* | |
| world_size | 128 |
| world_scale | 1 |
| n_step | 200 |
| *Genotype & Phenotype* | |
| n_params_size | 3 |
| n_cells_size | 32 |
| phenotype_size | 32 |
| center_phenotype | True |
| record_phenotype | True |
| adaptive_phenotype | False |
| switch | False |
| *QD Parameters* | |
| qd.n_generations | 300 |
| qd.log_interval | 20 |
| qd.batch_size | 256 |
| qd.repertoire_size | 256 |

| Aurora Configuration II | |
|---|---|
| *GA Emitter* | |
| qd.iso_sigma | 0.005 |
| qd.line_sigma | 0.05 |
| qd.kernel_mutate | False |
| qd.mutation_type | None |
| qd.variation_percentage | 1 |
| *Evaluation* | |
| qd.fitness | neg_angle_var |
| qd.secondary_fitness | null |
| qd.secondary_fitness_weight | 1.0 |
| qd.n_keep | 16 |
| *Auto-Encoder* | |
| qd.features, qd.hidden_size | 128, 8 |
| qd.dropout | 0.8 |
| qd.train_ratio | 16 |
| qd.lr_init_value, qd.lr_end_value | 5e-4, 0.0 |
| qd.ae_batch_size | 64 |
| qd.n_keep_ae | 64 |
| qd.use_data_augmentation | True |
| qd.group_cnn | False |

**Table 11.1:** Default configuration for AURORA Leniabreeder

### Contrastive Learning Leniabreeder

| **Aurora Configuration I** | |
|---|---|
| *General* | |
| seed | 2 |
| pattern_id | ”5N7KKM” |
| R, T | 13, 10 |
| *Simulation* | |
| world_size | 128 |
| world_scale | 1 |
| n_step | 200 |
| *Genotype & Phenotype* | |
| n_params_size | 3 |
| n_cells_size | 32 |
| phenotype_size | 32 |
| center_phenotype | True |
| record_phenotype | True |
| adaptive_phenotype | False |
| switch | False |
| *QD Parameters* | |
| qd.n_generations | 300 |
| qd.log_interval | 20 |
| qd.batch_size | 256 |
| qd.repertoire_size | 256 |

| **Aurora Configuration II** | |
|---|---|
| *GA Emitter* | |
| qd.iso_sigma | 0.005 |
| qd.line_sigma | 0.05 |
| qd.kernel_mutate | False |
| qd.mutation_type | None |
| qd.variation_percentage | 1 |
| *Evaluation* | |
| qd.fitness | neg_angle_var |
| qd.secondary_fitness | null |
| qd.secondary_fitness_weight | 1.0 |
| qd.n_keep | 16 |
| *Auto-Encoder* | |
| qd.features, qd.hidden_size | 128, 8 |
| qd.dropout | 0.8 |
| qd.train_ratio | 16 |
| qd.lr_init_value, qd.lr_end_value | 5e-4, 0.0 |
| qd.lr_transition_steps | 0 |
| qd.lr_transition_begin | 0 |
| qd.ae_batch_size | 64 |
| qd.n_keep_ae | 64 |
| qd.use_data_augmentation | True |
| qd.contrastive_learning | True |
| qd.contrastive_loss_fn | 0 |
| qd.group_cnn | False |
| qd.beta | 1.0 |
| qd.gamma | 1.0 |
| qd.delta | 0 |
| qd.temperature | 0.8 |
| qd.margin | 0.8 |

**Table 11.2:** Default configuration for AURORA CL Leniabreeder

### VQ-VAE Leniabreeder

| **VQ-VAE AURORA Configuration I** | |
|---|---|
| *Init pattern* | |
| pattern_id | ”5N7KKM” |
| R | 13 |
| T | 10 |
| *Simulation* | |
| world_size | 128 |
| world_scale | 1 |
| n_step | 200 |
| *Genotype* | |
| n_params_size | 3 |
| n_cells_size | 32 |
| *Phenotype* | |
| phenotype_size | 32 |
| center_phenotype | True |
| record_phenotype | True |
| adaptive_phenotype | False |
| switch | False |
| *QD General* | |
| qd.n_generations | 300 |
| qd.log_interval | 20 |
| qd.batch_size | 256 |
| *QD Repertoire* | |
| qd.repertoire_size | 256 |

| **VQ-VAE AURORA Configuration II** | |
|---|---|
| *QD GA Emitter* | |
| qd.iso_sigma | 0.005 |
| qd.line_sigma | 0.05 |
| qd.kernel_mutate | False |
| qd.mutation_type | None |
| qd.variation_percentage | 1 |
| *QD Evaluation* | |
| qd.fitness | neg_angle_var |
| qd.secondary_fitness | null |
| qd.secondary_fitness_weight | 1.0 |
| qd.n_keep | 16 |
| *QD Auto-Encoder* | |
| qd.features | 256 |
| qd.hidden_size | 32 |
| qd.dropout | 0.8 |
| qd.num_embeddings | 128 |
| qd.commitment_cost | 0.25 |
| qd.train_ratio | 16 |
| qd.lr_init_value | 5e-4 |
| qd.lr_end_value | 0.0 |
| qd.lr_transition_steps | 0 |
| qd.lr_transition_begin | 0 |
| qd.ae_batch_size | 64 |
| qd.n_keep_ae | 64 |
| qd.use_data_augmentation | True |
| qd.use_gcnn | False |

**Table 11.3:** Default configuration for VQ-VAE AURORA Leniabreeder

### Automatic Leniabreeder

| **AURORA Configuration I** | |
|---|---|
| *General* | |
| seed | 2 |
| *Init pattern* | |
| pattern_id | ”5N7KKM” |
| R | 13 |
| T | 10 |
| *Simulation* | |
| world_size | 128 |
| world_scale | 1 |
| n_step | 200 |
| *Genotype* | |
| n_params_size | 3 |
| n_cells_size | 32 |
| *Phenotype* | |
| phenotype_size | 32 |
| center_phenotype | True |
| record_phenotype | True |
| adaptive_phenotype | False |
| phenotype_size_increment | 0 |
| switch | False |
| max_iterations | 10 |
| *QD General* | |
| qd.n_generations | 300 |
| qd.log_interval | 20 |
| qd.batch_size | 256 |
| *QD Repertoire* | |
| qd.repertoire_size | 256 |

| **AURORA Configuration II** | |
|---|---|
| *QD GA Emitter* | |
| qd.iso_sigma | 0.005 |
| qd.line_sigma | 0.05 |
| qd.kernel_mutate | False |
| qd.mutation_type | None |
| qd.variation_percentage | 1 |
| *QD Evaluation* | |
| qd.fitness | neg_angle_var |
| qd.secondary_fitness | null |
| qd.secondary_fitness_weight | 1.0 |
| qd.n_keep | 16 |
| qd.adaptive_fitness | False |
| *QD Auto-Encoder* | |
| qd.features | 128 |
| qd.hidden_size | 8 |
| qd.dropout | 0.8 |
| qd.train_ratio | 16 |
| qd.lr_init_value | 5e-4 |
| qd.lr_end_value | 0.0 |
| qd.lr_transition_steps | 0 |
| qd.lr_transition_begin | 0 |
| qd.ae_batch_size | 64 |
| qd.n_keep_ae | 64 |
| qd.use_data_augmentation | True |
| qd.group_cnn | False |

**Table 11.4:** Default configuration for Automatic Leniabreeder

### Interesting creatures discovered with Automatic Leniabreeder

This appendix section aims to highlight some interesting creatures discovered using Automatic Leniabreeder. We include them here purely to give the reader a deeper understanding as to the extent we can discover diverse creatures.

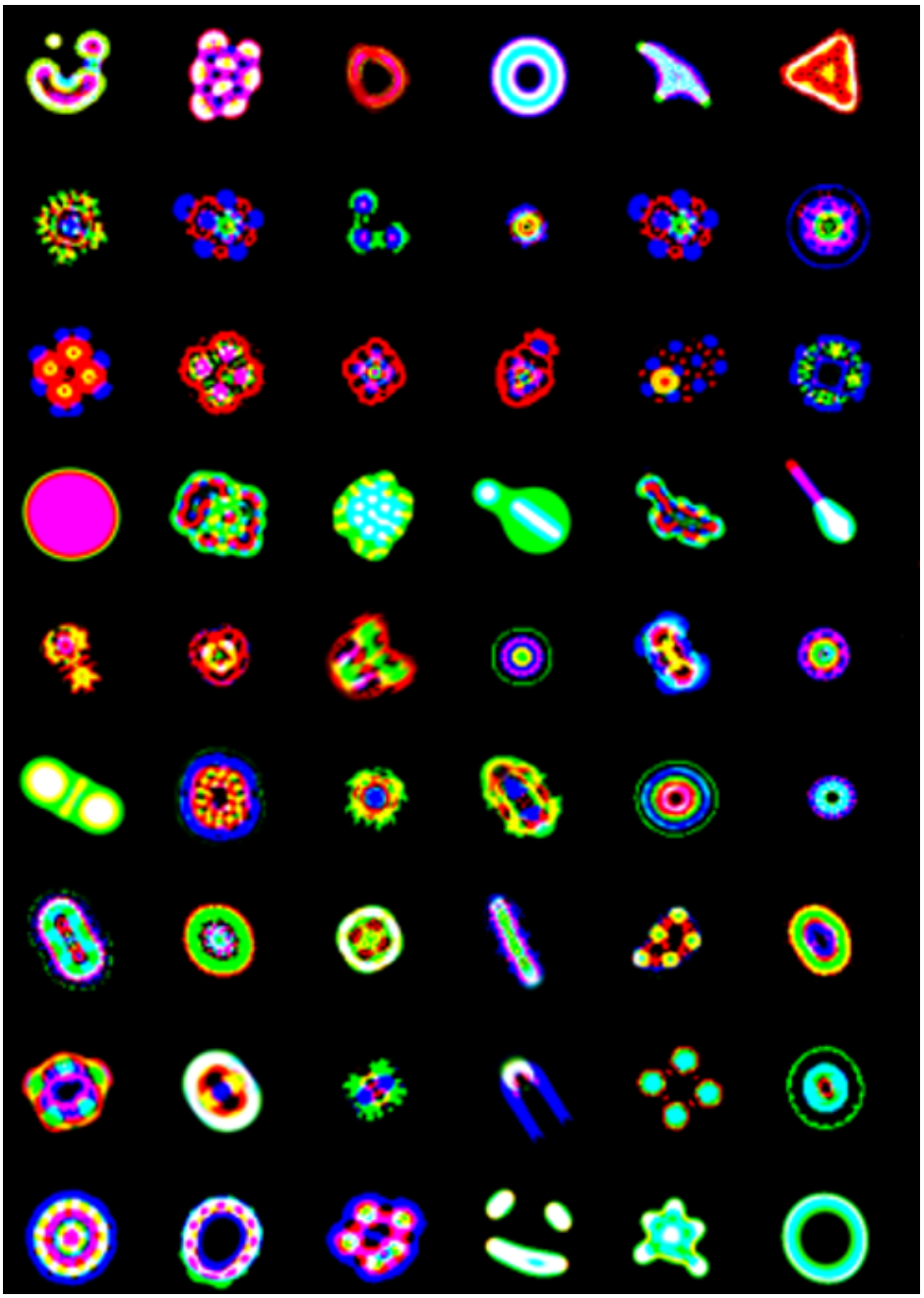

**Figure 11.1:** Part 1 of curated zoo of creatures discovered using Automatic Leniabreeder.

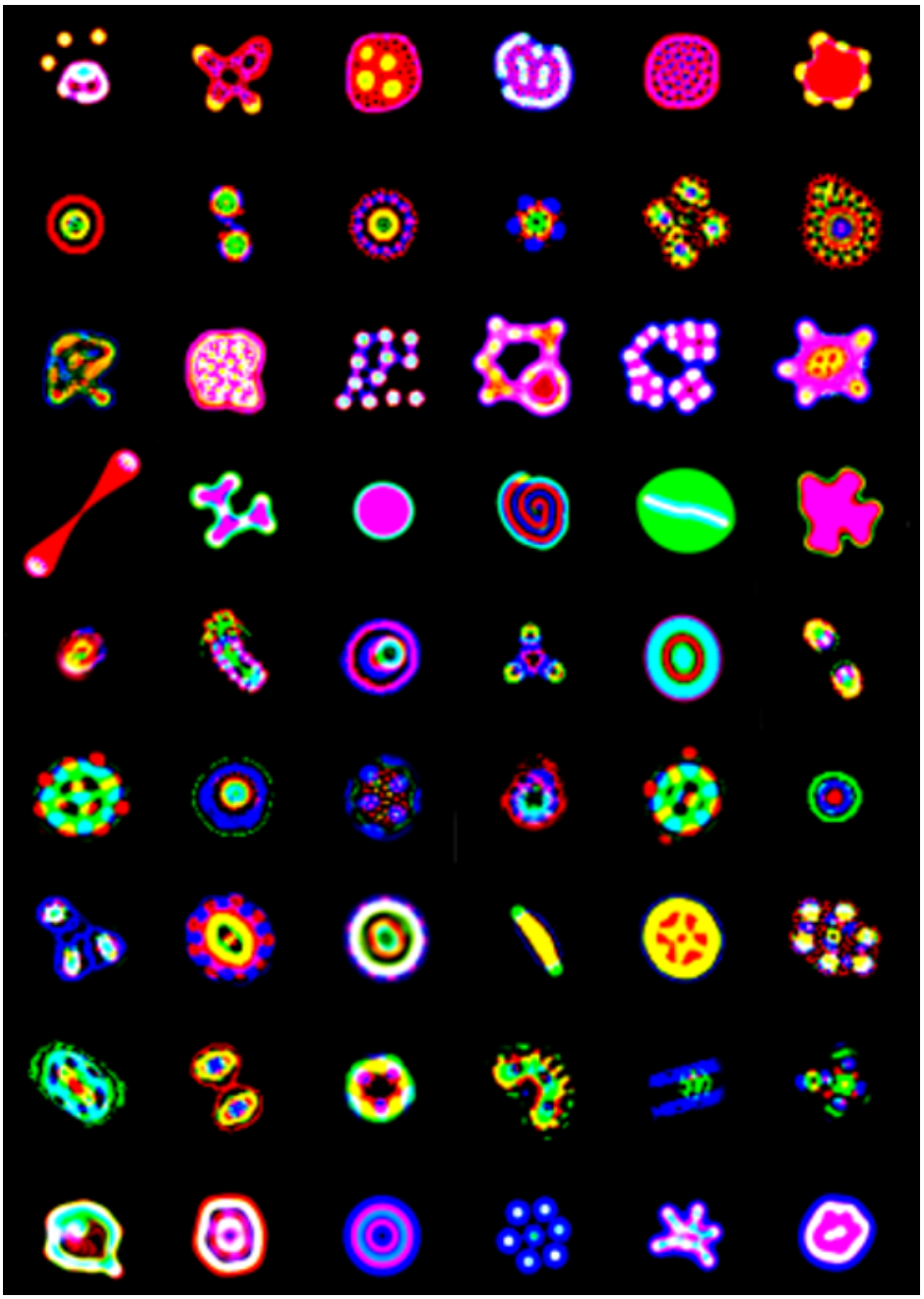

**Figure 11.2:** Part 2 of curated zoo of creatures discovered using Automatic Leniabreeder.