INTRINSICALLY MOTIVATED DISCOVERY OF DIVERSE PATTERNS IN SELF-ORGANIZING SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

In many complex dynamical systems, artificial or natural, one can observe self-organization of patterns emerging from local rules. Cellular automata, like the Game of Life (GOL), have been widely used as abstract models enabling the study of various aspects of self-organization and morphogenesis, such as the emergence of spatially localized patterns. However, findings of self-organized patterns in such models have so far relied on manual tuning of parameters and initial states, and on the human eye to identify "interesting" patterns. In this paper, we formulate the problem of automated discovery of diverse self-organized patterns in such high-dimensional complex dynamical systems, as well as a framework for experimentation and evaluation. Using a continuous GOL as a testbed, we show that recent intrinsically-motivated machine learning algorithms (POP-IMGEPs), initially developed for learning of inverse models in robotics, can be transposed and used in this novel application area. These algorithms combine intrinsically-motivated goal exploration and unsupervised learning of goal space representations. Goal space representations describe the "interesting" features of patterns for which diverse variations should be discovered. In particular, we compare various approaches to define and learn goal space representations from the perspective of discovering diverse spatially localized patterns. Moreover, we introduce an extension of a state-of-the-art POP-IMGEP algorithm which incrementally learns a goal representation using a deep auto-encoder, and the use of CPPN primitives for generating initialization parameters. We show that it is more efficient than several baselines and equally efficient as a system pre-trained on a hand-made database of patterns identified by human experts.

1 INTRODUCTION

Self-organization of patterns that emerge from local rules is a pervasive phenomena in natural and artificial dynamical systems (Ball, 1999). It ranges from the formation of snow flakes, spots and rays on animal's skin, to spiral galaxies. Understanding these processes has boosted progress in many fields, ranging from physics to biology (Camazine et al., 2003). This progress relied importantly on the use of powerful and rich abstract computational models of self-organization (Kauffman, 1993). For example, cellular automata like Conway's Game of Life (GOL) have been used to study the emergence of spatially localized patterns (SLPs) (Gardener, 1970), informing theories of the origins of life (Gardener, 1970; Beer, 2004). SLPs, such as the famous glider in GOL (Gardner et al., 1983), are self-organizing patterns that have a local extension and can exist independently of other patterns. However, finding such novel self-organized patterns, and mapping the space of possible emergent patterns, has so far relied heavily on manual tuning of parameters and initial states. Moreover, the dependence of this exploration process on the human eye to identify "interesting" patterns is strongly limiting further advances.

We formulate here the problem of automated discovery of a *diverse* set of self-organized patterns in such high-dimensional, complex dynamical systems. This involves several challenges. A first challenge consists in determining a representation of patterns, possibly through learning, enabling to incentivize the discovery of diverse "interesting" patterns. Such a representation guides exploration by providing a measure of (di-)similarity between patterns. This problem is particularly challenging

Source code and videos at `https://intrinsically-motivated-discovery.github.io/`

in domains where patterns are high-dimensional as in GOL. In such cases, scientists have a limited intuition about what useful features are and how to represent them. Moreover, low-dimensional representations of patterns are needed for human browsing and the visualization of the discoveries. Representation learning shall both guide exploration, and be fed by self-collected data.

A second challenge consists in how to automate exploration of high-dimensional, continuous initialization parameters to discover efficiently "interesting" patterns, such as SLPs, with a limited budget of experiments. Sample efficiency is important to enable the later use of such discovery algorithms for physical systems (Grizou et al., 2019), where experimental time and costs are strongly bounded. For example, in the continuous GOL used in this paper as a testbed, initialization consists in determining the values of a real-valued, high-dimensional 256×256 matrix besides 7 additional dynamics parameters. The possible variations of this matrix are too large for a simple random sampling to be efficient. More structured methods are needed.

To address these challenges, we propose to leverage and transpose recent intrinsically motivated learning algorithms, within the family of population-based Intrinsically Motivated Goal Exploration Processes (POP-IMGEPs - denoted simply as IMGEPs below, Baranes & Oudeyer (2013); Péré et al. (2018)). They were initially designed to enable autonomous robots to explore and learn what effects can be produced by their actions, and how to control these effects. IMGEPs self-define goals in a goal space that represents important features of the outcomes of actions, such as the position reached by an arm. This allows the discovery of diverse novel effects within their goal representations. It was recently shown how deep neuronal auto-encoders enabled unsupervised learning of goal representations in IMGEPs from raw pixel perception of a robot's visual scene (Laversanne-Finot et al., 2018). We propose to use a similar mechanism for automated discovery of patterns by unsupervised learning of a low-dimensional representation of features of self-organized patterns. This removes the need for human expert knowledge to define such representations.

Moreover, a key ingredient for sample efficient exploration of IMGEPs for robotics has been the use of structured motion primitives to encode the space of body motions (Pastor et al., 2013). We propose to use a similar mechanism to handle the generation of structured initial states in GOL-like complex systems, based on specialized recurrent neural networks (CPPNs) (Stanley, 2006).

In summary, we provide in this paper the following contributions:

- We formulate the problem of automated discovery of diverse self-organized patterns in high-dimensional and complex game-of-life types of dynamical systems.
- We show how to transpose POP-IMGEPs algorithms to address the associated joint challenge of (learning to) represent interesting patterns and discovering them in a sample efficient manner.
- We compare various approaches to define or learn goal space representations for the sample efficient discovery of diverse SLPs in a continuous GOL testbed.
- We show that an extension of a state-of-the-art POP-IMGEP algorithm, with incremental learning of a goal space using a deep auto-encoder, is equally efficient than a system pretrained on a hand-made database of patterns.

2 Related Work

Automated Discovery in Complex Systems Automated processes have been widely used to explore complex dynamical systems. For example, evolutionary algorithms have been applied to search specific patterns or rules of cellular automata (Mitchell et al., 1996; Sapin et al., 2003). However, their objective is to optimize a specific goal instead of discovering a diversity of patterns. Another line of experiments represent active inquiry-based learning strategies which query which set of experiments to perform to improve a system model, i.e. a mapping from parameters to the system outcome. Such strategies have been used in biology (King et al., 2004; 2009), chemistry (Raccuglia et al., 2016; Reizman et al., 2016; Duros et al., 2017) and astrophysics (Richards et al., 2011). However, these approaches have relied on expert knowledge, and focused on automated optimization of a pre-defined target property. Here, we are interested to automatically discover and map a diversity of unseen patterns without prior knowledge of the system. An exception is the concurrent work of Grizou et al. (2019), which showed how a simple POP-IMGEP algorithm could be used to automate discovery of diverse patterns in oil-droplet systems. However, it used a low-dimensional input space, and a hand-defined low-dimensional representation of goal spaces, identified as a major limit of the system.

Intrinsically motivated learning Intrinsically-motivated learning algorithms (Baldassarre & Mirolli, 2013; Baranes & Oudeyer, 2013) autonomously organize an agent's exploration curriculum in order to discover efficiently a maximally diverse set of outcomes the agent can produce in an unknown environment. They are inspired from the way children self-develop open repertoires of skills and learn world models. Intrinsically Motivated Goal Exploration Processes (IMGEPs) (Baranes & Oudeyer, 2013; Forestier et al., 2017) are a family of curiosity-driven algorithms developed to allow efficient exploration of high-dimensional complex real world systems. Population-based versions of these algorithms, which leverage episodic memory, hindsight learning, and structured dynamic motion primitives to parameterize policies, enable sample efficient acquisition of high-dimensional skills in real world robots (Forestier et al., 2017; Rolf et al., 2010). Recent work (Laversanne-Finot et al., 2018; Péré et al., 2018) studied how to automatically learn the goal representations with the use of deep variational autoencoders. However, training was done passively and in an early stage on a precollected set of available observations. Recent approaches (Nair et al., 2018; Pong et al., 2019) introduced the use of an online training of VAEs to learn the important features of a goal space similar to the methods in this paper. However, these approaches focused on the problem of sequential decisions in MDPs, incurring a cost on sample efficiency. This problem is observed in various intrinsically motivated RL approaches (Bellemare et al., 2016; Burda et al., 2019). The approaches are orthogonal to the automated discovery framework considered here with independent experiments allowing the use of memory-based sample efficient methods. A related family of algorithms in evolutionary computation is novelty search (Lehman & Stanley, 2008) and quality-diversity algorithms (Pugh et al., 2016), which can be formalized as special kinds of population-based IMGEPs.

Representation learning We are using representation learning methods to learn autonomously goal spaces for IMGEPs. Representation learning aims at finding low-dimensional explanatory factors representing high-dimensional input data (Bengio et al., 2013). It is a key problem in many areas in order to understand the underlying structure of complex observations. Many state-of-the-art methods (Tschannen et al., 2018) have built on top of Deep *variational autoencoders* (VAE) (Kingma & Welling, 2013), using varying objectives and network architectures. However, studies of the interplay between representation learning and autonomous data collection through exploration of an environment have been limited so far.

3 Algorithmic Methods for Automated Discovery

3.1 Population-based Intrinsically Motivated Goal Exploration Processes

An IMGEP is an algorithmic process generating a sequence of experiments to explore the parameters of a system by targeting self-generated goals (Fig. 1). It aims to maximize the diversity of observations from that system within a budget of N experiments. In population-based IMGEPs, an explicit memory of the history $\mathcal{H}$ of experiments and observations is used to guide the process.

The systems are defined by three components. A parameter space Θ corresponding to the controllable system parameters θ. An observation space O where an observation o is a vector representing all the signals captured from the system. For this paper, the observations are a time series of images which depict the morphogenesis of activity patterns. Finally, an unknown environment dynamic D: $\Theta \rightarrow O$ which maps parameters to observations.

To explore a system, an IMGEP uses a goal space $\mathcal{T}$ that represents relevant features of its observations, computed using an encoding function $\hat{g} = \mathcal{R}(o)$. For the exploration of patterns, such features may describe their form or extension. The exploration process iterates N times through: 1) sample a goal from a goal sampling distribution $g \sim G(\mathcal{H})$; 2) infer corresponding parameter θ using a parameter sampling policy $\Pi = \Pr(\theta; g, \mathcal{H})$; 3) roll-out an experiment with θ, observe the outcome o, compute encoding $\mathcal{R}(o)$; 4) store $(\theta, o, \mathcal{R}(o))$ in history $\mathcal{H}$. Because the sampling of goals and parameters depend on a history of explored parameters, an initial set of N_{init} parameters are randomly sampled and explored before the intrinsically motivated goal exploration process starts.

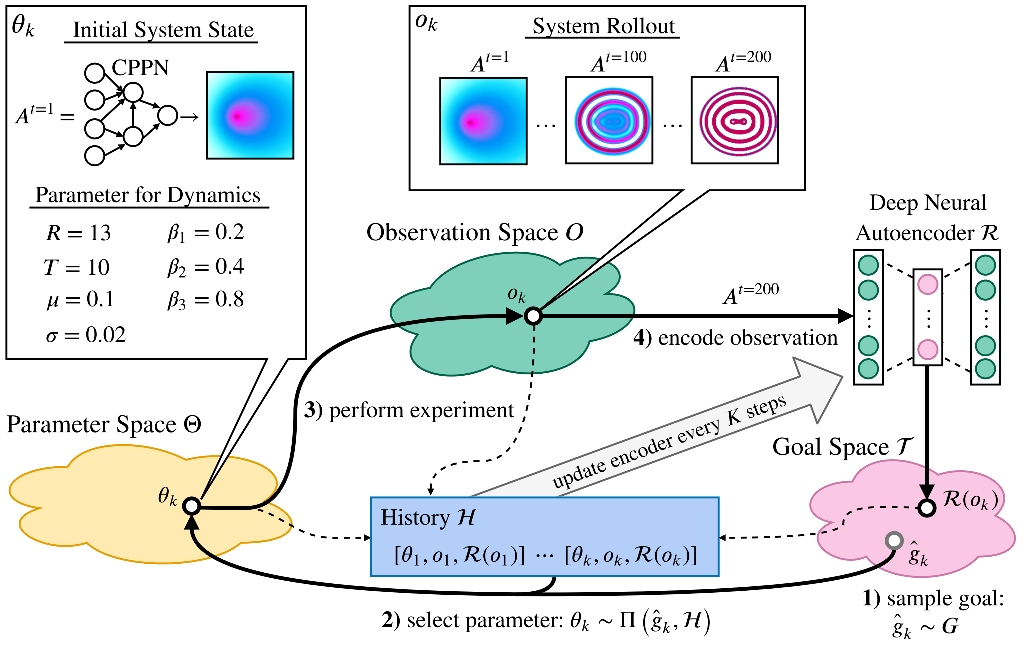


Figure 1: Population-based intrinsically motivated goal exploration process with incremental learning of a goal space (IMGEP-OGL algorithm) used to explore a continuous GOL model.

Different goal and parameter sampling mechanisms can be used within this architecture (Baranes & Oudeyer, 2013; Forestier & Oudeyer, 2016). In the experiments below, parameters are sampled by 1) given a goal, selecting the parameter from the history whose corresponding outcome is most similar in the goal space; 2) then mutating it by a random process. The goal sampling policy is a uniform distribution over a hypercube in $\mathcal{T}$ chosen to be large enough to bias exploration towards the frontiers of known goals to incentivize diversity.

3.2 Online Learning of Goal Spaces with Deep Auto-Encoders

For IMGEPs the definition of the goal space $\mathcal{T}$ and its corresponding encoder $\mathcal{R}$ are a critical part, because it biases exploration of the target system. One approach is to define a goal space by selecting features manually, for example by using computer vision algorithms to detect the positions of a pattern and its form. The diversity found by the IMGEPs will then be biased along these pre-defined features. A limit of this approach is its requirement of expert knowledge to select helpful features, particularly problematic in environments where experts do not know in advance what features are important, or how to formulate them.

Another approach is to learn goal space features by unsupervised representation learning. The aim is to learn a mapping $\mathcal{R}(o)$ from the raw sensor observations o to a compact latent vector $\mathbf{z} \in \mathbb{R}^d$. This latent mapping can be used as a goal space where a latent vector $\mathbf{z} = g$ is interpreted as a goal.

Previous IMGEP approaches already learned successfully their goal spaces with variational autoencoders (VAE) (Laversanne-Finot et al., 2018; Péré et al., 2018). However, the goal spaces were learned before the start of the exploration from a prerecorded dataset of observations from the target environment. During the exploration the learned representations were kept fixed. A problem with this pretraining approach is that it limits the possibilities to discover novel patterns beyond the distribution of pretraining examples, and requires expert knowledge.

In this paper we attempt to address this problem by continuously adapting the learned representation to the novel observations encountered during the exploration process. For this purpose, we propose an online goal space learning IMGEP (IMGEP-OGL), which learns the goal space incrementally during the exploration process (Algorithm 1). The training procedure of the VAE is integrated in the goal sampling exploration process by first initializing the VAE with random weights (Appendix E).

Algorithm 1: IMGEP-OGL

```
Initialize goal space encoder VAE $\mathcal{R}$ with random weights
for $i \leftarrow 1$ to $N$ do
    if $i < N_{init}$ then                    // Initial random iterations to populate $\mathcal{H}$
        Sample $\theta \sim \mathcal{U}(\Theta)$
    else                                      // Intrinsically motivated iterations
        Sample a goal $g \sim \mathcal{G}(\mathcal{H})$ based on space represented by $\mathcal{R}$
        Choose $\theta \sim \Pi(g, \mathcal{H})$
    Perform an experiment with $\theta$ and observe $o$
    Encode reached goal $\hat{g} = \mathcal{R}(o)$
    Append $(\theta, o, \hat{g})$ to the history $\mathcal{H}$
    if $i \mod K == 0$ then                   // Periodically train the network
        for E epochs do
            Train $\mathcal{R}$ on observations in $\mathcal{H}$ with importance sampling
        for $(\theta, o, \hat{g}) \in \mathcal{H}$ do    // Update the database of reached goals
            $\mathcal{H}[\hat{g}] \leftarrow \mathcal{R}(o)$
```

The VAE network is then trained every K explorations for E epochs on the observation collected in the history $\mathcal{H}$. Importance sampling is used to give more weight to recently discovered patterns.

3.3 Structuring the parameter space in IMGEPs: from DMPs to CPPNs

A key role in the generation of patterns in dynamical systems is their initial state $A^{t=1}$. IMGEPs sample these initial states and apply random perturbations to them during the exploration. For the experiments in this paper this state is a two-dimensional grid with 256×256 cells. Performing directly a random sampling of the 256×256 grid cells results in initial patterns that resemble white noise. Such random states result mainly in the emergence of global patterns that spread over the whole state space, complicating the search for spatially localized patterns. This effect is analogous to a similar problem in the exploration of robot controllers. Direct sampling of actions for individual actuators at a microscopic time scale is usually inefficient. A key ingredient for sample efficient exploration has been the use of structured primitives (dynamic motion primitives - DMPs) to encode the space of possible body motions (Pastor et al., 2013).

We solved the sampling problem for the initial states by transposing the idea of structured primitives. Indeed, "actions" consist here in deciding the parameters of an experiment, including the initial state. We propose to use compositional pattern producing networks (CPPNs) (Stanley, 2006) to produce structured initial patterns similar do DMPs. CPPNs are recurrent neural networks that allow the generation of structured initial states (Appendix B, Fig. 9) . The CPPNs are used as part of the parameters θ. They are defined by their network structure (number of neurons, connections between neurons) and their connection weights. They include a mechanism for random mutation of the weights and structure. The number of parameters in θ is therefore not fixed (yet starts small) and open-ended.

4 Experimental methods

We describe here the continuous Game of Life (Lenia) we use as a testbed representing a large class of high-dimensional dynamical systems, as well as the experimental procedures, the evaluation methods used to measure diversity and detect SLPs, and the used algorithmic baselines and ablations.

4.1 Continous Game of Life as a testbed

Lenia (Chan, 2018) is a continuous cellular automaton (Wolfram, 1983) similar to Conway's Game of Life (Gardener, 1970). Lenia, in particular, represents a high-dimensional complex dynamical system where diverse visual structures can self-organize and yet are hard to find by manual explo-

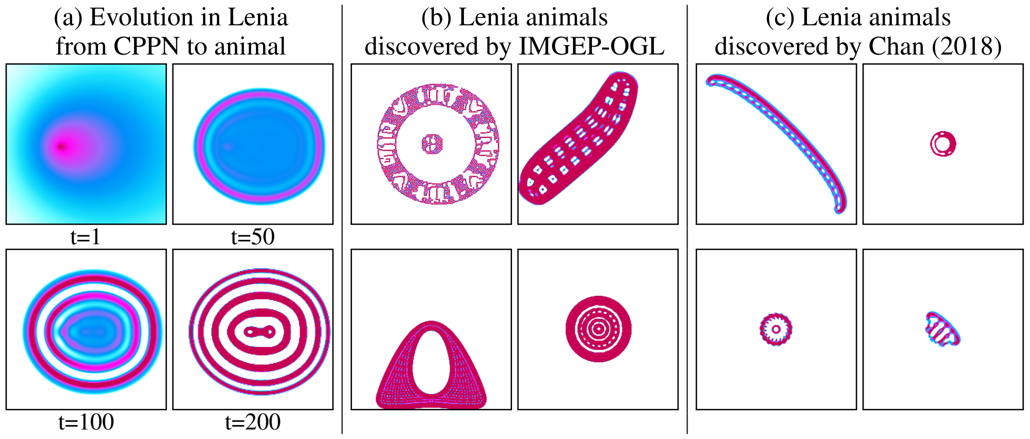

(a) Evolution in Lenia from CPPN to animal
t=1
t=50
t=100
t=200
(b) Lenia animals discovered by IMGEP-OGL
(c) Lenia animals discovered by Chan (2018)

Figure 2: Example patterns produced in the continuous GOL system (Lenia). Illustration of the dynamical morphing from an initial CPPN image to an animal (a). The automated discovery (b) is able to find similar complex animals as a human-expert manual search (c) by Chan (2018).

ration. It features the richness of Turing-complete game-of-life models. It is therefore well suited to test the performance of pattern exploration algorithms for unknown and complex systems. The fact that GOL models have been used widely to study self-organization in various disciplines, ranging from physics to biology and economics (Bak et al., 1989), also supports their generality and potential of reuse of our approach for discovery in other computational or wet high-dimensional systems.

Lenia consists of a two-dimensional grid of cells $A \in [0,1]^{256\times256}$ where the state of each cell is a real-valued scalar activity $A^t(x) \in [0,1]$. The state of cells evolves over discrete time steps t (Fig. 2, a). The activity change is computed by integrating the activity of neighbouring cells. Lenia's behavior is controlled by its initial pattern $A^{t=1}$ and several settings that control the dynamics of the activity change $(R, T, \mu, \sigma, \beta_1, \beta_2, \beta_3)$. Appendix A describes Lenia and its parameters in detail.

Lenia can be understood as a self-organizing morphogenetic system. Its parameters for the initial pattern $A^{t=1}$ and dynamics control determine the development of morphological patterns. Lenia can produce diverse patterns with different dynamics (stable, non-stable or chaotic). Most interesting, spatially localized coherent patterns that resemble in their shapes microscopic animals can emerge (Fig. 2, b, c). These pattern types, which we will denote "animals" as a short name, are a key reason scientists have used GOL models to study theories of the origins of life (Gardener, 1970; Beer, 2004). Therefore, in our evaluation method based on measures of diversity (see below), we will in particular study the performance of IMGEPs, and the impact of using various approaches for goal space representation, on finding a diversity of *animal* patterns. We implemented for this purpose different pattern classifiers to analyze the exploration results (Appendix A.2). Initially we differentiate between dead and alive patterns. A pattern is dead if the activity of all cells are either 0 or 1. Alive patterns are separated into animals and non-animals. Animals are a connected areas of positive activity which are finite, i.e. which do not infinitely cross several borders. All other patterns are non-animals whose activity usually spreads over the whole state space.

4.2 Evaluation based on the diversity of Patterns

The algorithms are evaluated based on their discovered diversity of patterns. Diversity is measured by the spread of the exploration in an *analytic behavior space*. This space is externally defined by the experimenter as in previous evaluation approaches in the IMGEP literature. For example, in Péré et al. (2018) is the diversity of discovered effects of a robot that manipulates objects measured by binning the space of object positions and counting the number of bins discovered. A difference here is that the experimenter does not have access to an easily interpretable hand-defined low-dimensional representation of possible patterns, equivalent to the cartesian coordinate of rigid objects. The space of raw observations O, i.e. the final Lenia patterns $A^{t=200}$, is also too high-dimensional for a meaningful measure of spread in it. We constructed therefore an external evaluation space. First, a latent representation space was build through the training of a VAE to learn the important features over a very large dataset of Lenia patterns identified during the many experiments over all evaluated algorithms. This large dataset enabled to cover a diversity of patterns orders of magnitude larger than what could be found in any single algorithm experiment, which experimental budget was order of magnitude smaller. We then augmented that space by concatenating hand-defined features (the same as for the HGS algorithm). See Appendix C for more information.

For each experiment all explored patterns were projected into the analytic behavior space. The diversity of the patterns is then measured by discretizing the space into bins of equal size by splitting each dimension into 7 sections (results were found to be robust to the number of bins per dimension, see C). This results in 7^{13} bins. The number of bins in which at least one explored entity falls is used as a measure for diversity.

We also measured the diversity in the space of parameters Θ by constructing an *analytic parameter space*. The 15 features of this space consisted of Lenia's parameters (R, T, μ, σ, β_1, β_2, β_3) and the latent representation of a VAE. The VAE was trained on a large dataset of initial Lenia states ($A^{t=1}$) used over the experimental campaign. This diversity measures also used 7 bins per dimension.

4.3 ALGORITHMS

The exploration behaviors of different IMGEP algorithms were evaluated and compared to a random exploration. The IMGEP variants differ in their way how the goal space is defined or learned. Appendices D and E provide details and hyperparameters.

Random exploration: The IMGEP variants were compared to a random exploration that sampled randomly for each of the N exploration iterations the parameters θ including the initial state $A^{t=1}$.

IMGEP-HGS - Goal exploration with a hand-defined goal space: The first IMGEP uses a hand-defined goal space that is composed of 5 features used in Chan (2018). Each feature measures a certain property of the final pattern $A^{t=200}$ that emerged in Lenia: 1) the sum over the activity of all cells, 2) the number of activated cells, 3) the density of the activity center, 4) an asymmetry measure of the pattern and 5) a distribution measure of the pattern.

IMGEP-PGL - Goal exploration with a pretrained goal space: For this IMGEP variant the goal space was learned with a VAE approach on training data before the exploration process started. The training set consisted of 558 Lenia patterns: half were animals that have been manually identified by Chan (2018); the other half randomly generated with CPPNs, see Section 4.4.

IMGEP-OGL - Goal exploration with online learning of the goal space: Algorithm 1.

IMGEP-RGS - Goal exploration with a random goal space: An ablated IMGEP using a goal space based on the encoder of a VAE with random weights.

4.4 EXPERIMENTAL PROCEDURE AND HYPERPARAMETERS

For each algorithm 10 repetitions of the exploration experiment were conducted. Each experiment consisted of $N = 5000$ exploration iterations. This number was chosen to be compatible with the application of the algorithms in physical experimental setups similar to Grizou et al. (2019), planned in future work. For IMGEP variants the first $N_{\text{init}} = 1000$ iterations used random parameter sampling to initialize their histories $\mathcal{H}$. For the following 4000 iterations each IMGEP approach sampled a goal g via an uniform distribution over its goal space. The ranges for sampling in the hand-defined goal space (HGS) are defined in Table 5 (Appendix D). The ranges for the VAE based goal spaces (PGL, OGL) were set to $[-3, 3]$ for each of their latent variables. Then, the parameter θ_k from a previous exploration in $\mathcal{H}$ was selected whose reached goal $\hat{g}_k$ had the minimum euclidean distance to the current goal g within the goal space. This parameter was then mutated to generate the parameter θ that was explored.

The parameters θ consisted of a CPPN (Section 3.3) that generates the initial state $A^{t=1}$ for Lenia and the settings defining Lenia's dynamics: $\theta = [\text{CPPN} \rightarrow A^{t=1}, R, T, \mu, \sigma, \beta_1, \beta_2, \beta_3]$. The CPPNs were initialized and mutated by a random process that defines their structure and connection weights as done by Stanley (2006). The random initialization of the other Lenia settings was done by an uniform distribution and their mutation by a Gaussian distribution around the original values. The meta parameters to initialize and mutate the parameters were the same for all algorithms (Appendix B). They were manually chosen without optimizing them for a specific algorithm.

5 Results

We address several questions evaluating the ability of IMGEP algorithms to identify a diverse set of patterns, and in particular diverse "animal" patterns (i.e. spatially localized patterns).

Does goal exploration outperform random parameter exploration? In robotics/agents contexts where scenes are populated with rigid objects, various forms of goal exploration algorithms outperform random parameter exploration (Laversanne-Finot et al., 2018). We checked whether this still holds in continuous GOL which have very different properties. Measures of the diversity in the analytic behavior space confirmed the advantage of IMGEPs with hand-designed (HGS) or learned goal spaces (PGL/OGL) over random explorations (Fig. 3, b). The organization resulting from goal exploration is also visible through the diversity in the space of parameters. IMGEPs focus their exploration on subspaces that are useful for targeting new goals. In contrast, random parameter exploration is unguided, resulting in a higher diversity in the parameter space (Fig. 3, b).

What is the impact of learning a goal space vs. using random or hand-defined features? We compared also the performance of random VAE goal spaces (RGS) to learned goal spaces (PGL/OGL). For reinforcement learning problems, using intrinsic reward functions based on random features of the observations can result in a high or boosted performance (Burda et al., 2018; 2019). In our context however, using random features (RGS) collapsed the performance of goal exploration, and did not even outperform random parameter exploration for all kinds of behavioural diversity (Fig. 3). Results also show that hand-defined features (HGS) produced significantly less global diversity and less "animal" diversity than using learned features (PGL/OGL). However, HGS

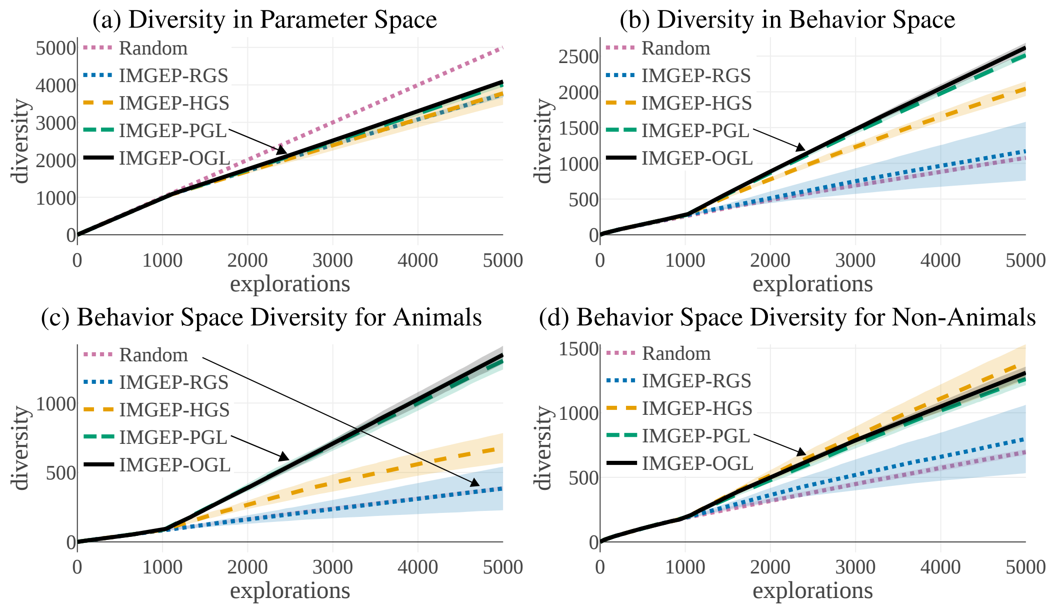


Figure 3: (a) Although random explorations reach the highest diversity in the analytic parameter space, (b) IMGEPs reach a higher diversity in the analytic behavior space (except when using random representations). (c) IMGEPs with a learned goal space discovered a larger diversity of animals compared to a hand-defined goal space. (d) Learned goal spaces are as efficient as a hand-defined space for finding diverse non-animals patterns. Overall, IMGEPs with unsupervised learning of goal features are efficient to discover a *diversity of diverse patterns*. Depicted is the average diversity ($n = 10$) with the standard deviation as shaded area (for some not visible because it is too small). See Fig. 27, 28, 29, 30 and 31 in Annex for a qualitative visual illustration of these results.

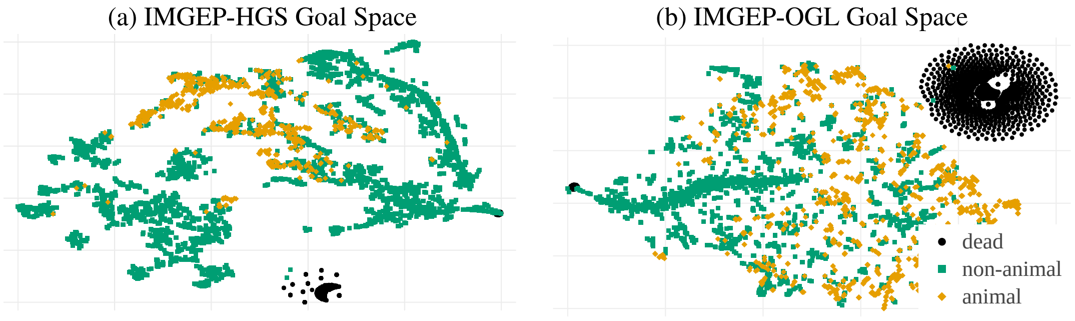


Figure 4: (a) Hand-defined and (b) learned goal spaces have major differences shown here by a t-SNE visualization. The different number and size of clusters of animals or non-animals can explain the differences in their resulting diversity between the algorithms (Fig. 3).

found an equal diversity of "non-animals". These results show that in this domain, the goal-space has a critical influence on the type and diversity of patterns discovered. Furthermore, unsupervised learning is an efficient approach to discover a *diversity of diverse patterns*, i.e. both efficient at finding diverse animals and diverse non-animals.

Is pretraining on a database of expert patterns necessary for efficient discovery of diverse animals? A possibility to bias exploration towards patterns of interest, such as "animals", is to pretrain a goal space with a pattern dataset hand-made by an expert. Here PGL is pretrained with a dataset containing 50% animals. This leads PGL to discover a high diversity of animals. However, the new online approach (IMGEP-OGL) is as efficient as PGL to discover diverse patterns (Fig. 3, b,c,d). Taken together, these results uncover an interesting bias of using learned features with a VAE architecture, which strongly incentivizes efficient discovery of diverse spatially localized patterns.

How do goal space representations differ? We analyzed the goal spaces of the different IMGEP variants to understand their behavior by visualizing their reached goals in a two-dimensional space. T-SNE (Maaten & Hinton, 2008) was used to reduce the high-dimensional goal spaces. It puts points that were nearby in the high-dimensional space also close to each other in the two-dimensional visualization.

The hand-defined (HGS) and learned (OGL) goal spaces show strong differences between each other (Fig. 4). We believe this explains their different abilities to find either a high diversity of non-animals or animals (Fig. 3, c, d). The goal space of the IMGEP-HGS shows large areas and several clusters for non-animal patterns (Fig. 4, a). Animals form only few and nearby clusters. Thus, the hand-defined features seem poor to discriminate and describe animal patterns in Lenia. As a consequence, when goals are uniformly sampled within this goal space during the exploration process, then more goals are generated in regions that describe non-animals. This can explain why IMGEP-HGS explored a higher diversity of non-animal patterns but only a low diversity of animal patterns. In contrast, features learned by IMGEP-OGL capture better factors that differentiate animal patterns. This is indicated by the several clusters of animals that span a wide area in its goal space (Fig. 4, b).

We attribute this effect to the difficulty of VAEs to capture sharp details (Zhao et al., 2017b). They therefore represent mainly the general form of the patterns but not their fine-grained structures. Animals differ often in their form whereas non-animals occupy often the whole cell grid and differ in their fine-grained details. The goal spaces learned by VAEs seem therefore better suited for exploring diverse sets of animal patterns.

6 Conclusion

We formulated a novel application area for machine learning: the problem of automatically discovering self-organized patterns in complex dynamical systems with high-dimensions both in the action space and in the observation space. We showed that this problem calls for advanced methods requiring the dynamic interaction between sample efficient autonomous exploration and unsupervised representation learning. We demonstrated that population-based IMGEPs are a promising algorithmic framework to address this challenge, by showing how it can discover diverse self-organized patterns in a continuous GOL. In particular, we introduced a new approach of learning a goal space representation online via data collected during the exploration process. It enables sample efficient discovery of diverse sets of animal-like patterns, similar to those identified by human experts and yet without relying on such prior expert knowledge (Fig. 2). We also analyzed the impact of goal space representations on the diversity and types of discovered patterns.

The continuous game of life shares many properties with other artificial or natural complex systems, explaining why GOL models have been used in many disciplines to study self-organization, see Bak et al. (1989). We therefore believe this study shows the potential of IMGEPs to automated discovery in other systems encountered in physics, chemistry or even computer animation. In further work, we aim to apply this approach in roboticized wet experiments such as the one presented in Grizou et al. (2019) and addressing fundamental understanding of how proto-cells can self-organize.

REFERENCES

Per Bak, Kan Chen, and Michael Creutz. Self-organized criticality in the'game of life. *Nature*, 342 (6251):780, 1989.

Gianluca Baldassarre and Marco Mirolli. *Intrinsically motivated learning in natural and artificial systems*. Springer, 2013.

Philip Ball. *The self-made tapestry: pattern formation in nature*, volume 198. Oxford University Press Oxford, 1999.

Adrien Baranes and Pierre-Yves Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013.

Randall D Beer. Autopoiesis and cognition in the game of life. *Artificial Life*, 10(3):309–326, 2004.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pp. 1471–1479, 2016.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in $beta$-vae. *arXiv preprint arXiv:1804.03599*, 2018.

Scott Camazine, Jean-Louis Deneubourg, Nigel R Franks, James Sneyd, Eric Bonabeau, and Guy Theraula. *Self-organization in biological systems*. Princeton university press, 2003.

Bert Wang-Chak Chan. Lenia-biology of artificial life. *arXiv preprint arXiv:1812.05433*, 2018.

Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, pp. 2610–2620, 2018.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

Vasilios Duros, Jonathan Grizou, Weimin Xuan, Zied Hosni, De-Liang Long, Haralampos N Miras, and Leroy Cronin. Human versus robots in the discovery and crystallization of gigantic polyoxometalates. *Angewandte Chemie*, 129(36):10955–10960, 2017.

Sébastien Forestier and Pierre-Yves Oudeyer. Modular active curiosity-driven discovery of tool use. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3965–3972. IEEE, 2016.

Sébastien Forestier, Yoan Mollard, and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*, 2017.

Martin Gardener. Mathematical games: The fantastic combinations of john conway's new solitaire game" life,". *Scientific American*, 223:120–123, 1970.

Martin Gardner, Martin Gardner, Martin Gardner, and Martin Gardner. *Wheels, life, and other mathematical amusements*, volume 86. WH Freeman New York, 1983.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Jonathan Grizou, Laurie J Points, Abhishek Sharma, and Leroy Cronin. Exploration of self-propelling droplets using a curiosity driven robotic assistant. *arXiv preprint arXiv:1904.12635*, 2019.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv preprint arXiv:1901.05534*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, volume 3, 2017.

I. T. Jolliffe. *Principal Component Analysis and Factor Analysis*, pp. 115–128. Springer New York, New York, NY, 1986. ISBN 978-1-4757-1904-8. doi: 10.1007/978-1-4757-1904-8_7.

Stuart A Kauffman. *The origins of order: Self-organization and selection in evolution.* OUP USA, 1993.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

Ross D King, Kenneth E Whelan, Ffion M Jones, Philip GK Reiser, Christopher H Bryant, Stephen H Muggleton, Douglas B Kell, and Stephen G Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247, 2004.

Ross D King, Jem Rowland, Stephen G Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N Soldatova, et al. The automation of science. *Science*, 324(5923):85–89, 2009.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pp. 4743–4751, 2016.

Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. *arXiv preprint arXiv:1711.00848*, 2017.

Adrien Laversanne-Finot, Alexandre Pere, and Pierre-Yves Oudeyer. Curiosity driven exploration of learned disentangled goal spaces. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto (eds.), *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 487–504. PMLR, 29–31 Oct 2018.

Joel Lehman and Kenneth O Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pp. 329–336, 2008.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Melanie Mitchell, James P Crutchfield, Rajarshi Das, et al. Evolving cellular automata with genetic algorithms: A review of recent work. In *Proceedings of the First International Conference on Evolutionary Computation and Its Applications (EvCA96)*, volume 8. Moscow, 1996.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pp. 9191–9200, 2018.

Peter Pastor, Mrinal Kalakrishnan, Franziska Meier, Freek Stulp, Jonas Buchli, Evangelos Theodorou, and Stefan Schaal. From dynamic movement primitives to associative skill memories. *Robotics and Autonomous Systems*, 61(4):351–361, 2013.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Alexandre Péré, Sébastien Forestier, Olivier Sigaud, and Pierre-Yves Oudeyer. Unsupervised Learning of Goal Spaces for Intrinsically Motivated Goal Exploration. In *ICLR2018 - 6th International Conference on Learning Representations*, Vancouver, Canada, April 2018.

Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40, 2016.

Paul Raccuglia, Katherine C Elbert, Philip DF Adler, Casey Falk, Malia B Wenny, Aurelio Mollo, Matthias Zeller, Sorelle A Friedler, Joshua Schrier, and Alexander J Norquist. Machine-learning-assisted materials discovery using failed experiments. *Nature*, 533(7601):73, 2016.

Brandon J Reizman, Yi-Ming Wang, Stephen L Buchwald, and Klavs F Jensen. Suzuki–miyaura cross-coupling optimization enabled by automated feedback. *Reaction chemistry & engineering*, 1(6):658–666, 2016.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Joseph W Richards, Dan L Starr, Henrik Brink, Adam A Miller, Joshua S Bloom, Nathaniel R Butler, J Berian James, James P Long, and John Rice. Active learning to overcome sample selection bias: application to photometric variable star classification. *The Astrophysical Journal*, 744(2):192, 2011.

Matthias Rolf, Jochen J Steil, and Michael Gienger. Goal babbling permits direct learning of inverse kinematics. *IEEE Transactions on Autonomous Mental Development*, 2(3):216–229, 2010.

Emmanuel Sapin, Olivier Bailleux, and Chabrier Jean-Jacques. Research of a cellular automaton simulating logic gates by evolutionary algorithms. In *European Conference on Genetic Programming*, pp. 414–423. Springer, 2003.

Kenneth O Stanley. Exploiting regularity without development. In *Proceedings of the AAAI Fall Symposium on Developmental Systems*, pp. 37. AAAI Press Menlo Park, CA, 2006.

Kenneth O Stanley and Risto Miikkulainen. Efficient evolution of neural network topologies. In *Proceedings of the 2002 Congress on Evolutionary Computation.*, volume 2, pp. 1757–1762. IEEE, 2002.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.

Michael Tschannen, Olivier Bachem, and Mario Lucic. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*, 2018.

Stephen Wolfram. Statistical mechanics of cellular automata. *Reviews of modern physics*, 55(3): 601, 1983.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017a.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Towards deeper understanding of variational autoencoding models. *arXiv preprint arXiv:1702.08658*, 2017b.

A Target System: continuous Game of Life (Lenia)

The Lenia model is a particular implementation of continuous Game of Life models (Chan, 2018). It was used as the target system for all exploration experiments. The following section describes Lenia and the parameters to control its behavior in detail. It is followed by a description of the classifiers used to categorize dead, animal and non-animal Lenia patterns. Finally, statistical measures about the patterns are introduced which were used to define goal and analytic spaces.

A.1 Implementation Details and Parameters

Lenia (Chan, 2018) is a cellular automaton (Wolfram, 1983). It consists of a two-dimensional grid of cells $A \in [0,1]^{L \times L}$ with $L = 256$ for all experiments. The cell grid is similar to the surface of a ball. Cells on the north border are neighbors to the south border cells. The east and west border are also connected. The state of each cell is a real-valued scalar activity $A(x) \in [0,1]$. The states of cells A^t evolve over discrete time steps $t = [1, \dots, M]$ with $M = 200$ for all experiments. The activity change of a cell is computed by integrating the previous activity of its neighbouring cells:

$$A^{t+1}(x) = \left[A^t(x) + \Delta t \, G\left(K * A^t(x)\right)\right]_0^1 ,$$

where G is the *growth mapping*, K is the *kernel*, $\Delta t = \frac{1}{T}$ with $T \in \mathbb{N}$ is the *time step* and $[n]_b^a = \min(\max(n,a),b)$ is the clip function. For all experiments an exponential growth mapping was used:

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

with $\mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}$ being perimeters that control its shape.

The kernel integrates the activity of the current cell x and its neighbours by a convolution with a *kernel function* $K(n)$:

$$K * A^t(x) = \sum_{n \in \mathcal{N}(x)} K(n) A^t(x+n) \Delta x^2, \tag{1}$$

where $\mathcal{N}$ is the *neighborhood* around the cell x and $\Delta x = \frac{1}{R}$ with $R \in \mathbb{N}$ is the site distance. The neighborhood is defined by a circle around x with radius R: $\mathcal{N}(x) = \{y : ||x-y||_2 \leq R\}$. The kernel is constructed by a *kernel core* function $K_C : [0,1] \mapsto [0,1]$ and a *kernel shell* function $K_S : [0,1] \mapsto [0,1]$. The kernel core creates a ring around the center coordinate and is defined by an exponential:

$$K_C(r) = \exp\left(\alpha - \frac{\alpha}{4r(1-r)}\right) \text{ with } \alpha = 4 .$$

The kernel shell K_S takes a vector parameter $\beta = (\beta_1, \beta_2, \beta_3) \in [0,1]^3$ and copies the kernel core into concentric rings. The rings are of equal thickness with peak heights β_i:

$$K_S(r; \beta) = \beta_{\lfloor Br \rfloor} K_C(Br \; mod \; 1) .$$

Finally, the kernel is normalized:

$$K(n) = \frac{K_S(||n||_2)}{|K_S|}.$$

In total 8 parameters $\theta = \{A^{t=1}, R, T, \mu, \sigma, \beta_1, \beta_2, \beta_3\}$ controlled the behavior of Lenia for all experiments. $A^{t=1}$ is the starting pattern of the system. R is the radius of the circle around a cell x whose enclosed cells influence the activity of x. T controls the growth strength update per time step. The growth mapping is controlled by μ and σ. The form of the kernel function is controlled by $\beta_1, \beta_2, \beta_3$.

We based our Python implementation of Lenia on the code provided by `https://github.com/Chakazul/Lenia`.

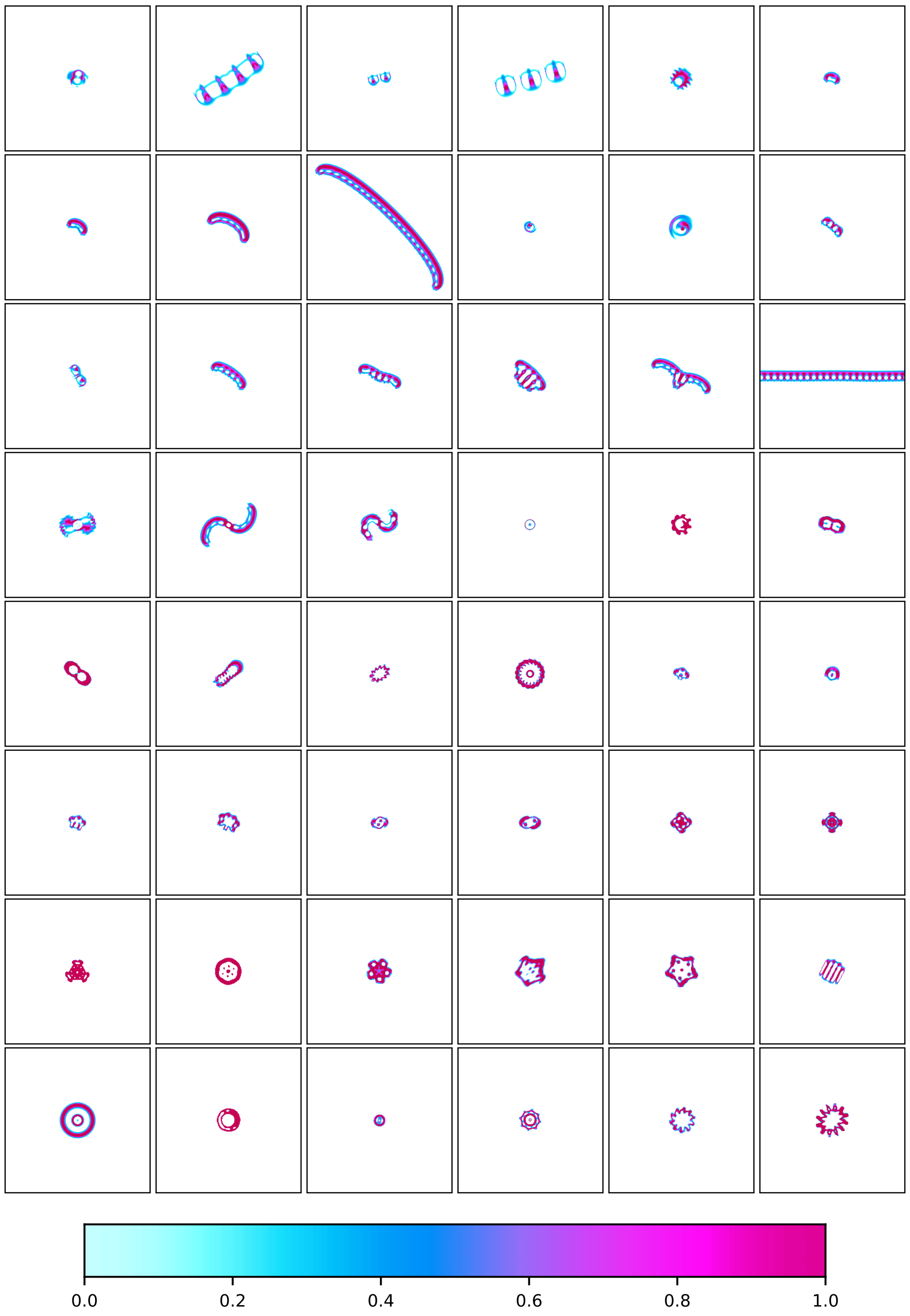


Figure 5: Patterns identified by a human-expert (Chan, 2018). The activity of each cell is mapped from $[0, 1]$ to the color scheme visualized at the bottom. An activity of 0 is white. This color scheme is used for all illustrated patters throughout the paper.

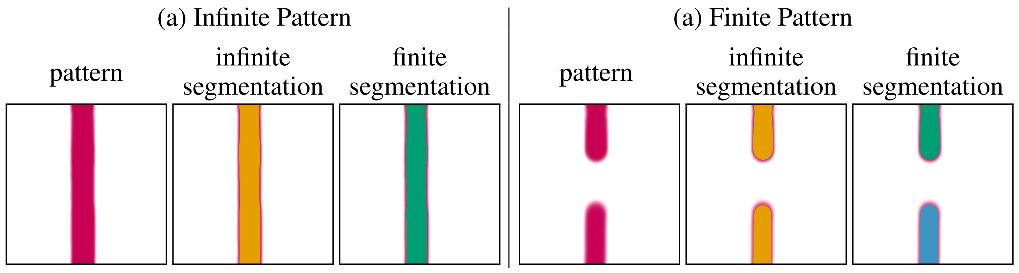


Figure 6: Classification of Lenia patterns into finite and infinite patterns. Infinite patterns form loop between the image borders which are identified if a segment is connected between two borders in the infinite and finite segmentation. Finite patterns form no loops. They have connected segments between borders in the infinite but not finite segmentation. Segments are colorized in yellow, green and blue.

A.2 Classifier

We categorized 3 types of patterns that are observed in Lenia. The categories were used to analyze if the exploration algorithms showed differences in their exploration behaviors by identifying different types of patterns. The 3 categories are dead, animals and non-animals. For each class is a classifier defined. The classifiers only classify the final pattern in which the Lenia system morphs after $M = 200$ time steps.

Dead Classifier: For dead patterns is the activity of all cells either 0 or 1 in the last time step.

Animal Classifier: The final Lenia pattern is classified as an animal if it is a *finite* and *connected* pattern of activity. Cells x, y are connected as a pattern if both are active ($A(x) \geq 0.1$ and $A(y) \geq 0.1$) and if they influence each other. Cells influence each other when they are within their radius of the kernel K as defined by the parameter R (Eq. 1).

Furthermore, the connected pattern must be finite. In Lenia finite and infinite patterns can be differentiated because the opposite borders of Lenia's cell grid are connected, so that the space is similar to a ball surface. Thus, a pattern can loop around this surface making it infinite. We identify infinite patterns by the following approach. First, all connected patterns are identified for the case of assuming an infinite grid cell, i.e. opposite grid cell borders are connected. Second, all connected patterns for the case of a finite grid cell, i.e. opposite grid cell borders are not connected, are identified. Third, for each border pair (north-south and east-west) it is tested if cells within a distance of R from both borders exists, that are part of a connected pattern for the infinite and finite grid cell case. If such a pattern exists than it is assumed to be infinite, because it loops around the grid cell surface of Lenia (Fig. 6, a). All other patterns are considered to be finite (Fig. 6, b). Please note that this method has a drawback. It can not identify certain infinite patterns that loop over several borders, for example, if a pattern exists that connects the north to east and then the west to south border (Fig. 7).

Moreover, there are two additional constraints that an animal pattern must fulfill. First, the cells of the connected pattern $P = \{x_1, \ldots, x_n\}$ must have at least 80% of all activation, i.e. $\sum_{x \in P} A(x) \geq 0.8 \sum_{\forall y} A(y)$. Second, a pattern must exists for the last two time steps ($t = M$ and $t = M - 1$).

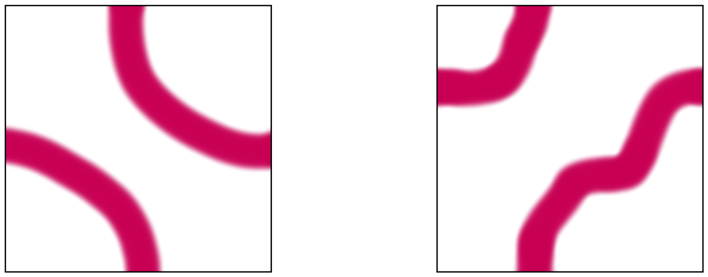


Figure 7: Examples of infinite patterns that are misclassified as a finite patterns.

Both constraint are used to avoid that too small patterns or chaotic entities which change drastically between time steps are classified as animals. See Fig. 5, 27, 29, 30 and 31 for examples of animal patterns.

Non-Animal Classifier: We also classified non-animal patterns which are all entities that were not dead and not an animal. These patterns spread usually over the whole state space and are connected via borders. See Fig. 27, 29, 30 and 31 for examples of non-animal patterns.

A.3 Statistical Measures for Lenia Patterns

We defined five statistical measurements for the final patterns $A^{t=M}$ that emerge in Lenia. The measures were used as features for hand-defined goal spaces of IMGEPs and to define partly the analytic behavior space in which the results of the exploration experiments were compared.

Activation mass M_A: Measures the sum over the total activation of the final pattern and normalizes it according to the size of the Lenia grid:

$$M_A = \frac{1}{L^2} \sum_x A^{t=M}(x) \; ,$$

where $L^2 = 256 \cdot 256$ is the number of cells of the Lenia system.

Activation volume V_A: Measures the number of active cells and normalizes it according to the size of the Lenia grid:

$$V_A = \frac{1}{L^2} \left| \left\{ \forall x : A^{t=M}(x) > \epsilon \right\} \right| \text{ with } \epsilon = 10^{-4}.$$

Activation density D_A: Measures how dense the activation is distributed on average over all active cells:

$$D_A = \frac{M_A}{V_A}.$$

Activation asymmetry A_A: Measures how symmetrical the activation is distributed according to an axis that starts in the center of the patterns activation mass and goes along the last movement direction of this center. This measure was introduced to especially characterize animal patterns such as shown in Fig. 5. The center of the activity mass is usually also the center of the animals and analyzing the activity along their movement axis measures how symmetrical they are.

As a first step, the center of the activation mass is computed for every time step of the Lenia simulation and the Lenia pattern recentered to this location. This ensures that the center is all the time correctly computed in the case the animal moves and reaches one border to appear on the opposite border in the uncentered pattern. The center $(\bar{x}, \bar{y})_t$ for time step t is calculated by:

$$(\bar{x}, \bar{y})_t = \left(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right) \text{ with } M_{pq} = \sum_x \sum_y x^p y^q A^t(x, y) \; ,$$

where M_{pq} measures the image moment (or raw moment) of order $(p+q)$ for $p, q \in \mathbb{N}$.

Based on the center $(\bar{x}, \bar{y})_t$ the pattern A^t is recentered to A^t_C by shifting the x and y indexes according to the center:

$$A^t_C(x, y) = A^t((x - \bar{x}) \bmod L, (y - \bar{y}) \bmod L) \; , \tag{2}$$

where L is width and length of the Lenia grid and the indexing is $x, y = 0, \ldots, L-1$. After each time step the center is recomputed and the pattern recentered:

$$A^{t=1} \underset{\text{recenter}}{\longrightarrow} A^{t=1}_C \underset{\text{Lenia step}}{\longrightarrow} A^{t=2} \underset{\text{recenter}}{\longrightarrow} A^{t=2}_C \underset{\text{Lenia step}}{\longrightarrow} \ldots \; .$$

Please note, the simulations and all figures of patterns in the paper are done with the uncentered pattern. The centered version is only computed for the purpose of statistical measurements.

The recenter step by $(\bar{x}, \bar{y})_t$ defines also the movement direction of the activity center:

$$(m_x, m_y)_t = (\bar{x}, \bar{y})_t - \left(x^{\text{mid}}, y^{\text{mid}} \right) = \left(\bar{x} - x^{\text{mid}}, \bar{y} - y^{\text{mid}} \right) \; ,$$

where $x^{\mathrm{mid}}, y^{\mathrm{mid}} = \frac{L-1}{2}$ are the coordinates for the middle point of the grid. A line can be defined that starts in the midpoint $\left(x^{\mathrm{mid}}, y^{\mathrm{mid}}\right)$ of the final centered pattern $A_C^{t=M}$ and goes in and opposite to the final movement direction of the activity mass center $(m_x, m_y)_{t=M}$. This line separates the grid in two equal areas. The asymmetry is computed by comparing the amount of activity in the grid right M_A^{right} and left M_A^{left} of the line. The normalized difference between both sides is the final asymmetry measure:

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

Activation centeredness C_A: Measures how strong the activation is distributed around the activity mass center:

$$C_A = \frac{1}{M_A} \sum_x \sum_y w_{xy} \cdot A_C^{t=M}(x,y) \quad \text{with} \quad w_{xy} = \left(1 - \frac{d(x,y)}{\max_{y,x} d(x,y)}\right)^2 ,$$

where $d(x,y) = \sqrt{(x - x^{\mathrm{mid}})^2 + (y - y^{\mathrm{mid}})^2}$ is the distance from the point (x,y) to the center point $\left(x^{\mathrm{mid}}, y^{\mathrm{mid}}\right)$. $A_C^{t=M}(x,y)$ is the centered activation that is updated every time step as for the asymmetry measure (Eq. 2). The weights w_{xy} decrease the farer a point is from the center. Thus, patterns that are concentrated around the center have a high value for C_A close to 1. Whereas, patterns whos activity is distributed throughout the whole grid have a smaller value. For patterns that are equally distributed ($\forall_{x,x'} : A(x) = A(x')$) is $C_A = 0$ defined as centeredness measure.

B Sampling of Parameters for Lenia

All exploration algorithms explore Lenia patterns by sampling the parameters θ that control Lenia. The parameters are comprised of the initial pattern $A^{t=1}$ and the parameters which control the dynamic behavior $(R, T, \mu, \sigma, \beta_1, \beta_2, \beta_3)$. There are two operations to sample parameters: 1) random initialization and 2) mutating an existing parameter θ. CPPNs are used for the random initialization and mutation of the initial pattern $A^{t=1}$. The details of this process are described in the next section. Afterwards, the initialization and mutation of Lenia's parameter that control its dynamics are described.

B.1 Sampling of Start Patterns for Lenia via CPPNs

Compositional Pattern Producing Networks (CPPNs) are recurrent neural networks that were developed for the generation and evolution of gray-scale 2D images (Stanley, 2006). We used CPPNs to generate and mutate the initial state of Lenia $A^{t=1}$ which resembles an image. CPPNs generate images pixel by pixel by taking as input a bias value, the x and y coordinate of the pixel in the image and its distance d to the image center (Fig. 8). Their output is the pixel value as a gray scale between 0 and 1 for the given (x, y) coordinate. For the generation of initial Lenia patterns is as input the x and y coordinate of the grid cells used. They were mapped to $x = [-2, 2]$ and $y = [-2, 2]$. The distance to the grid center is given by $d(x, y) = \sqrt{x^2 + y^2}$. The final activity of a cell is the remapped output p of the CPPN via $A(x, y) = 1 - |p|$.

CPPNs consist of several hidden neurons (typically between 4 to 6 in our experiments) that can have recurrent connections and self connections. Each CPPN has one output neuron. Two activation functions were used for the hidden neurons and the output neuron. The first is Gaussian and the second is sigmoidal:

$$\text{gauss}(x) = 2\exp\left(-(2.5x)^2\right) - 1\ , \tag{3}$$

$$\text{sigm}(x) = 2\left(\frac{1}{1+\exp(-5x)}\right) - 1\ . \tag{4}$$

To randomly initialize a Lenia initial pattern $A^{t=1}$ a CPPN is randomly sampled by sampling the number of hidden neurons, the connections between inputs and neurons and neurons to neurons, their connection weights and the activation functions for neurons. Afterwards the initial pattern is generated by it. In the history $\mathcal{H}$ of the IMGEPs is then the CPPN as part of the parameter θ added. If the parameter is mutated, then the weights, connections and activation functions of the CPPN are mutated and the new initial pattern $A^{t=1}$ generated by it. A CPPN is defined over its network structure (number of nodes, connections of nodes) and its connection weights. The number of parameters in θ is therefore variable and not fixed.

We used the *neat-python*[2] package for the random generation and mutation of CPPNs. It is based on the NeuroEvolution of Augmenting Topologies (NEAT) algorithm for the evolution of neural networks (Stanley & Miikkulainen, 2002). The meta-parameters for the initialization and mutation

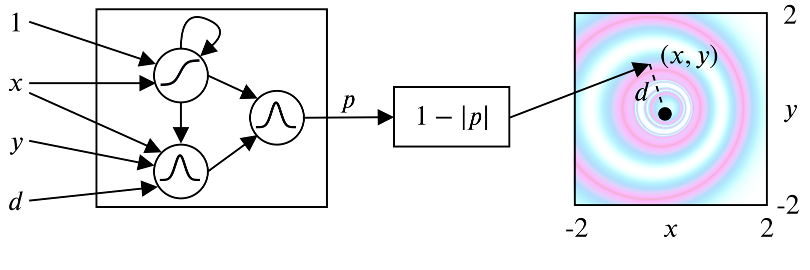


Figure 8: CPPNs are recurrent neural networks which take as input a bias of 1, the x and y coordinate of a point in the generated pattern and its distance r to the center of the pattern. Their output is the activity value of a grid cell.

[2]https://github.com/CodeReclaimers/neat-python

Parameter	Value
Initial number of hidden neurons	4
Initial activation functions	gauss, sigm
Initial connections	random connections with probability 0.6
Initial synapse weight	Gaussian distribution with $\mu = 0$, $\sigma = 0.4$
Synapse weight range	$[-3, 3]$
Mutation neuron add probability	0.02
Mutation neuron delete probability	0.02
Mutation connection add probability	0.05
Mutation connection delete probability	0.01
Mutation rate of activation functions	0.1
Mutation rate of synapse weights	0.05
Mutation replace rate of synapse weights	0.06
Mutation power of synapse weights σ_M	1
Mutation enable/disable rate of synapse weights	0.02

Table 1: Configuration for the initialization and mutation of CPPN networks that generate the initial state for the Lenia system.

of CPPNs are listed in Table 1. The random sampling and mutation of CPPNs allows to generate complex patterns as illustrated in Fig. 9.

The random sampling of a new CPPN is done by the following steps. All CPPNs are initialized with 4 hidden neurons and 1 output neuron. Their activation functions are randomly assigned. Each input-hidden, hidden-hidden and hidden-output neuron pair is connected with a probability of 0.6. The weights of each connection are sampled via a Gaussian distribution: $w_{ij} \sim \mathcal{N}(\mu = 0, \sigma = 0.4)$. The maximum and minimum weights for a connection are -3 and 3.

An existing CPPN is mutated by the following procedure. At first, structural mutations are performed. With probability 0.02 a new neuron z with a random activation function is added. The neuron is connected to the network by choosing randomly an existing connection. This connection is deleted. A connection from the source i of the deleted connection to the new neuron is added with weight $w_{iz} = 1.0$. Additionally, a new connection from the new neuron to the target j of the deleted connection is added with the old connection weight $w_{zj} = w_{ij}$, finishing the addition of a new neuron. With probability 0.02 one of the hidden neurons is deleted. With probability 0.05 a new connection is added between a random input-hidden, hidden-hidden or hidden-output neuron pair. The connection weight is sampled by the same method as for the sampling of new CPPNs. With probability 0.01 one random existing connection is removed. After the structural mutations the activation functions and weights are mutated. For each neuron the activation function is changed with probability 0.1 by randomly assigning a new activation function (either gauss or sigm). For each connection the weight is mutated by the following steps. With probability 0.05 the weight of the connection is changed according to:

$$w_{ij} \leftarrow [w_{ij} + \mathcal{N}(0, \sigma_M)]_{-3}^{3} \ ,$$

where $\sigma_M = 1$ is the mutation power and $[n]_b^a = \min(\max(n, a), b)$ is the clip function. With probability 0.06 the connection weight is completely replaced by sampling a new one as done for the sampling of weights of new CPPNs.

Please note, the neat-python package allows also the setting and mutation of response and bias weights for each neuron. Those settings were not used for the experiments. Moreover, we adjusted the sigmoid and Gaussian function in the neat-python package to the ones defined in Eq. 3 and Eq. 4 to be able to replicate similar images as in Stanley (2006).

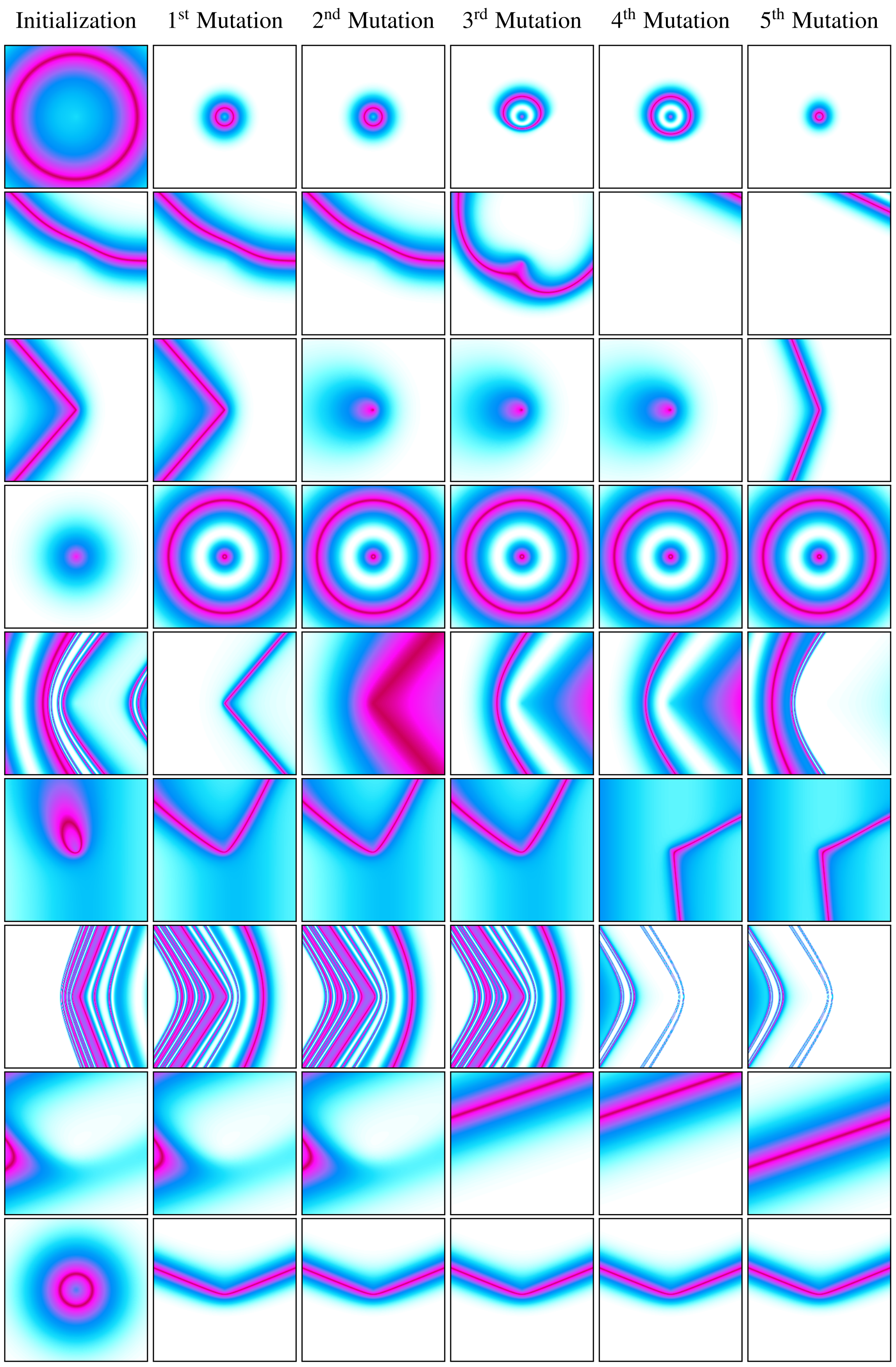


Figure 9: CPPNs can generate complex patterns via their random initialization and successive mutations. Each row shows generated patterns by one CPPN and its mutations.

B.2 Sampling of Lenia's Dynamic Parameters

The parameters that control the dynamics of Lenia ($R, T, \mu, \sigma, \beta_1, \beta_2, \beta_3$) are initialized and mutated via uniform and Gaussian distributions. Table 2 lists for each parameter the meta-parameters for their initialization and mutation. Each parameter is initialized by an uniform sampling $\theta_i \sim \mathcal{U}(a, b)$ with a and b as upper and lower border. An existing parameter θ_i is mutated by the following equation:

$$\theta_i \leftarrow [\theta_i + \mathcal{N}(0, \sigma_M)]_b^a \ ,$$

where σ_M is the mutation power and $[n]_b^a = \min(\max(n, a), b)$ is the clip function with a and b as upper and lower border. For natural numbers $\theta_i \in \mathbb{N}$ the resulting value is rounded towards the nearest natural number.

Parameter	Type	Value Range	Mutation σ_M
R	$\mathbb{N}$	$[2, 20]$	0.5
T	$\mathbb{N}$	$[1, 20]$	0.5
μ	$\mathbb{R}$	$[0, 1]$	0.05
σ	$\mathbb{R}$	$[0.001, 0.3]$	0.01
$\beta[1]$	$\mathbb{R}$	$[0, 1]$	0.05
$\beta[2]$	$\mathbb{R}$	$[0, 1]$	0.05
$\beta[3]$	$\mathbb{R}$	$[0, 1]$	0.05

Table 2: Settings for the initialization and mutation of Lenia system parameters θ.

C Measurement of Diversity in the Analytic Parameter and Behavior Space

The algorithms are compared on their ability to explore a diverse set of patterns. The next section introduces the diversity measure, followed by sections that introduce the spaces in which the algorithms are compared.

C.1 Diversity Measure

Diversity is measured by the area that explored parameters cover in the parameter space of Lenia or that the identified patterns cover in the observation space. For the experiments the parameter space consisted of the initial start state of Lenia ($A^{t=1} \in [0,1]^{256 \times 256}$) and the settings for Lenia's dynamics ($R, T, \mu, \sigma, \beta_1, \beta_2, \beta_3$). The space consist therefore of 256^2 dimensions, each for a single grid cell of the initial pattern, plus 7 dimensions for the dynamic settings. The observation space consists of the final patterns $A^{t=200} \in [0,1]^{256 \times 256}$ resulting in 256^2 dimensions for the space. Each single exploration results in a new point in those spaces.

The diversity measures how much area the algorithms explored in those spaces (Fig. 10). The measurement is done by discretizing the space with a spatial grid and counting the number of discretized areas in which at least one point falls. For the discretization each dimension of the space is given a range, i.e. a minimum and maximum border. Each dimension is then split in a certain number of equally sized bins between those borders. The areas with values falling below the minimum or above the maximum border are counted as two additional bins.

The number of dimensions of the original parameter and observation space are too large to measure diversity in a meaningful manner. The initial pattern and the final pattern have 256^2 dimensions. We constructed therefore an analytic parameter and behavioral space where the latent representations of a β-VAE were used to reduce the high-dimensional patterns to 8 dimensions. The diversity in those spaces was compared between the algorithms. 5 bins (7 with the out of range values) per dimension were used for the discretization of those spaces for all experiments in the paper.

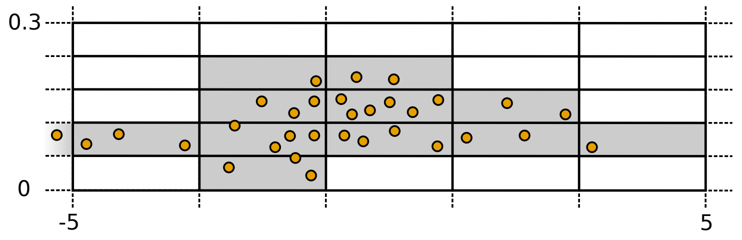


Figure 10: Illustration of the diversity measure in a two-dimensional space. The ranges for the dimensions were set to $[-5, 5]$ and $[0, 0.3]$. The number bins per dimension is 5. Including the outlier areas the number of discretized bins is $7^2 = 49$. The diversity is the number of bins in which points exist (grey areas) which are 12 in this example.

C.2 Analytic Parameter Space

The analytic parameter space was constructed by the 7 Lenia parameters that control its dynamics and 8 latent representation dimensions of a β-VAE (Table 3). The β-VAE was trained on initial patterns $A^{t=1}$ used during the experiments. The dataset was constructed by randomly selecting 42500 patterns (37500 as training set, 5000 as validation set) from the experiments of all algorithms and each of their 10 repetitions. The β-VAE uses the same structure, hyper-parameters, loss function and learning algorithm as described in Section E. It was trained for more than 1400 epochs with $\beta = 5$ (Fig. 11). The encoder which resulted in the minimal validation set error during the training was used. According to its reconstructed patterns it can represent the general form of patterns but often not individual details such as their texture (Fig. 12).

Analytic Parameter Space Definition

Parameter	min	max	Parameter	min	max
R	1	20	β-VAE latent 1	-5	5
T	2	10	β-VAE latent 2	-5	5
μ	0	1	β-VAE latent 3	-5	5
σ	0	0.3	β-VAE latent 4	-5	5
β_1	0	1	β-VAE latent 5	-5	5
β_2	0	1	β-VAE latent 6	-5	5
β_3	0	1	β-VAE latent 7	-5	5
			β-VAE latent 8	-5	5

Table 3: Features of the analytic parameter space and their min and max values

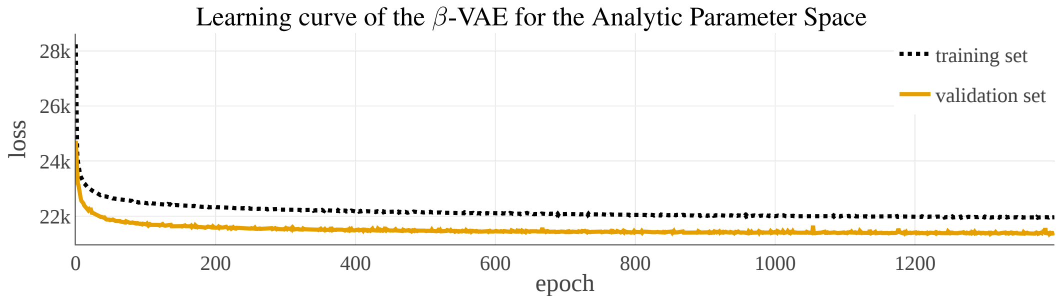


Figure 11: Learning curve of the β-VAE whose latent encoding was used for the analytic parameter space.

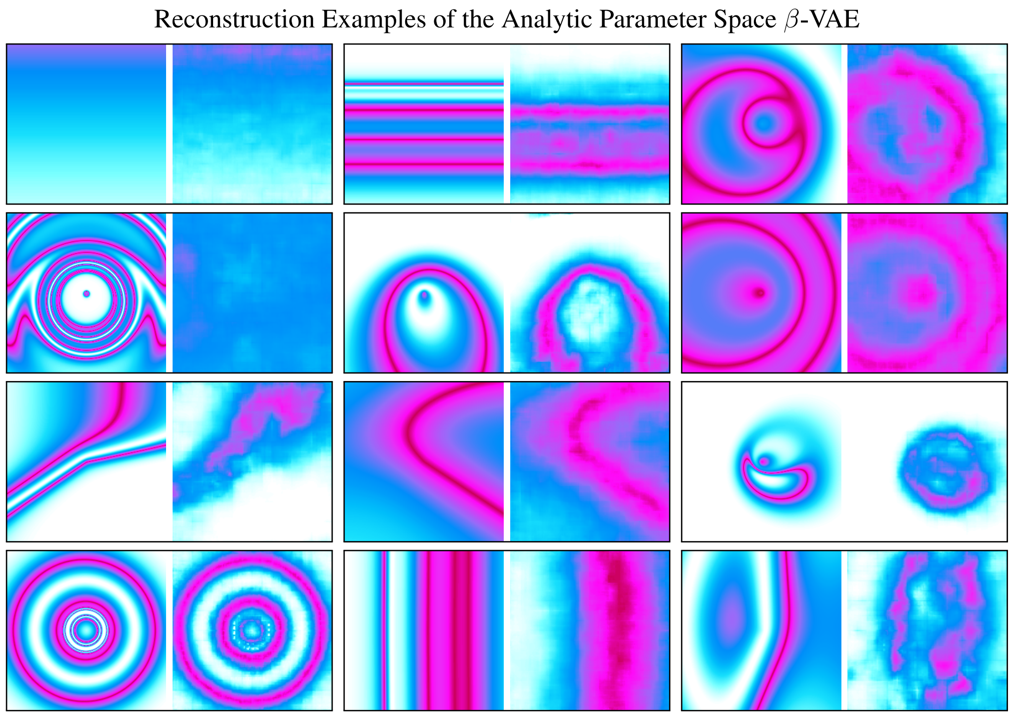

Reconstruction Examples of the Analytic Parameter Space β-VAE

Figure 12: Examples of patterns (left) and their reconstructed output (right) by the β-VAE used for the construction of the analytic parameter space. The patterns are sampled from the validation dataset.

Analytic Parameter Space Definition

Parameter	min	max	Parameter	min	max
mass M_A	0	1	β-VAE latent 1	-5	5
volume V_A	0	1	β-VAE latent 2	-5	5
density D_A	0	1	β-VAE latent 3	-5	5
asymmetry A_A	-1	1	β-VAE latent 4	-5	5
centeredness C_A	0	1	β-VAE latent 5	-5	5
			β-VAE latent 6	-5	5
			β-VAE latent 7	-5	5
			β-VAE latent 8	-5	5

Table 4: Features of the analytic behavior space and their min and max values

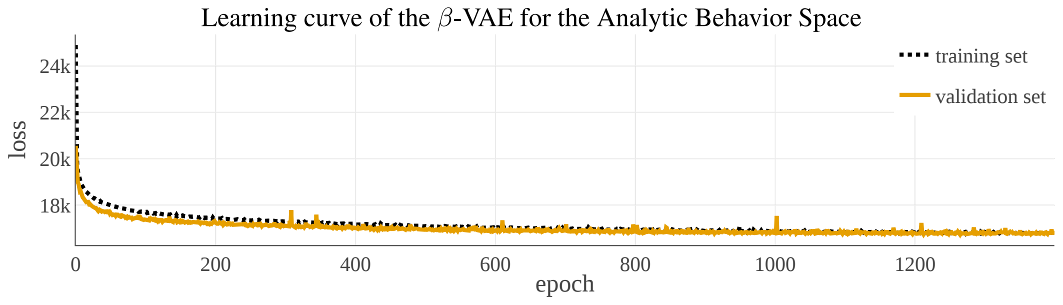


Figure 13: Learning curve of the β-VAE whose latent encoding was used for the analytic behavior space.

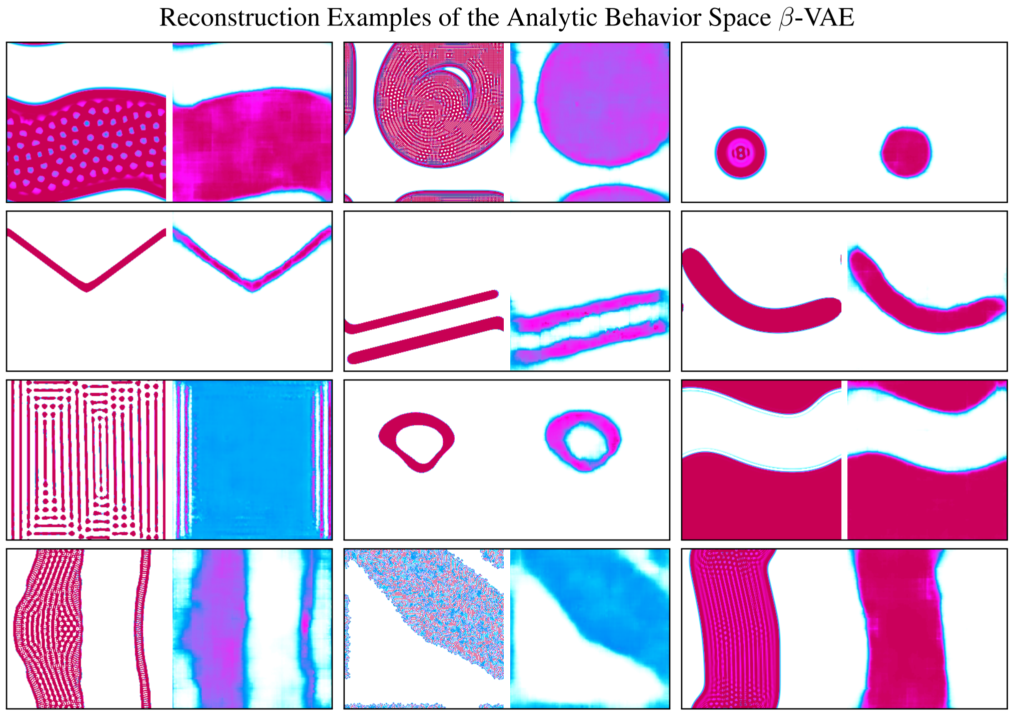

Reconstruction Examples of the Analytic Behavior Space β-VAE

Figure 14: Examples of patterns (left) and their reconstructed output (right) by the β-VAE used for the construction of the analytic behavior space. The patterns are sampled from the validation dataset.

C.3 Analytic Behavior Space

The analytic behavior space was constructed by combining the 5 statistical measures for final Lenia patterns (Section A.3) and 8 latent representation dimensions of a β-VAE (Table 4). The β-VAE was trained on final patterns $A^{t=200}$ observed during experiments. The dataset was constructed by randomly selecting 42500 patterns (37500 as training set, 5000 as validation set) from the experiments of all algorithms and each of their 10 repetitions. The dataset consists of 50% animal and 50% non-animal patterns. The β-VAE uses the same structure, hyper-parameters, loss function and learning algorithm as described in Section E. It was trained for more than 1400 epochs with $\beta = 5$ (Fig. 13). The encoder which resulted in the minimal validation set error during the training was used. Its reconstructed patterns show that it is able to represent the general form of patterns but often not individual details such as their texture (Fig. 14).

D Random Exploration and IMGEPs with Hand-Defined Goal Spaces

Two random explorations and several IMGEPs with different hand-defined goal spaces were evaluated and compared. The main paper and the additional results in Section F only report the results for the best random exploration and one IMGEP variant with a hand-defined goal space. This section introduces the implementation details and diversity results of all evaluated random explorations and IMGEPs with hand-defined goal spaces.

D.1 Random Explorations

We evaluated two random exploration strategies: Random Initialization and Random Mutation. The main paper and the additional results in Section F only discuss the Random Initialization approach.

Random Initialization: This approach sampled for each of the 5000 explorations a random parameter θ including a random CPPN to generate the initial state $A^{t=1}$. The approach can be replicated by using Algorithm 2 with $N_{init} = 5000$.

Random Mutation: This approach is closer to the principle of IMGEPs. It first performs $N_{init} = 1000$ random explorations and adds each explored parameter θ to a history $\mathcal{H}$. Afterwards, it randomly samples a parameter from the history and mutates it. The new parameter is also added to history $\mathcal{H}$. The approach can be replicated by using Algorithm 2 where line 6 is skipped and the parameter sampling distribution $\Pi(g, \mathcal{H})$ is selecting a random parameter from the history and mutating it.

D.2 IMGEPs with Hand-Defined Goal Spaces

We evaluated several IMGEP variants with goal spaces that were hand-defined (IMGEP-HGS). Each space was constructed by a different combination of statistical measures of the final Lenia patterns (Tables 5 and 6) which are described in Section A.3. The main paper and the additional results in Section F only discuss the IMGEP-HGS 9 approach. Algorithm 2 lists the steps of the IMGEP-HGS variants. They begin with $N_{init} = 1000$ random explorations, followed by 4000 explorations based on randomly generated goals. Each goal was sampled from a uniform distribution within the ranges defined in Table 5. Then the parameter from a previous exploration that resulted in the closest outcome to the current goal was mutated and explored.

Algorithm 2: IMGEP-HGS

```
1 Initialize goal space representation $\mathcal{R}$ by hand-defined features
2 for $i \leftarrow 1$ to $N$ do
3     if $i < N_{init}$ then              // Initial random iterations to populate $\mathcal{H}$
4         Sample $\theta \sim \mathcal{U}(\Theta)$
5     else                                // Intrinsically motivated iterations
6         Sample a goal $g \sim \mathcal{G}(\mathcal{H})$ based on the space represented by $\mathcal{R}$
7         Choose $\theta \sim \Pi(g, \mathcal{H})$
8     Perform an experiment with $\theta$ and observe $o$
9     Append $(o, \theta, \mathcal{R}(o))$ to the history $\mathcal{H}$
```

D.3 Results

The random explorations and IMGEP-HGS variants are compared by their resulting diversity in the analytic parameter and behavior space (Fig. 15). The diversity is measured by the number of reached bins in each space using a binning of 7 bins per dimension.

The Random Initialization approach reached for all diversity measures a higher diversity than the Random Mutation approach. Therefore, the Random Initialization approach is used for the comparison to IMGEP approaches in the main paper and the additional results in Section F.

Most IMGEP-HGS variants had a higher diversity in the analytic behavior space compared to random explorations, although their diversity in the analytic parameter space is lower. This shows the advantage of IMGEPs over random searches in discovering a wider range of patterns in the target system. The best overall diversity had IMGEP-HGS 3, 4 and 9. We chose IMGEP-HGS 9 to compare it with learned goal spaces in the main paper and for the additional results in Section F. It identified the highest diversity of non-animals of the three variants (3, 4, 9) reaching a higher diversity for non-animals than any IMGEP with a learned goal space. It was therefore selected to show that the choice of the goal space has an influence on the patterns that IMGEPs identify.

Depending on the statistical measures used to define the goal space the diversity between the IMGEP-HGS variants varied. IMGEPs that use the volume measure (HGS 1 - 4) reach in general a higher overall diversity which can be attributed to their higher diversity of animal patterns than goal spaces with the density measure (HGS 5 - 8) (Fig. 15, b, c). In terms of diversity of identified animals showed the inclusion of several measures the best performance (HGS 4 and HGS 8 in Fig. 15, c). In terms of diversity of identified non-animals showed the inclusion of several measures besides the centeredness C_A measure the best performance (HGS 3 and HGS 7 in Fig. 15, d). The results show that the choice of the goal space has an important influence on the diversity of identified patterns and their type (animal or non-animal).

Feature	min	max
mass M_A	0	1
volume V_A	0	1
density D_A	0	1
asymmetry A_A	-1	1
centeredness C_A	0	1

Table 5: HGS Goal Space Ranges

	HGS-Variants								
Feature	1	2	3	4	5	6	7	8	9
mass M_A	×	×	×	×	×	×	×	×	×
volume V_A	×	×	×	×					×
density D_A					×	×	×	×	×
centeredness C_A		×		×		×		×	×
asymmetry A_A			×	×			×	×	×

Table 6: IMGEP-HGS Variants

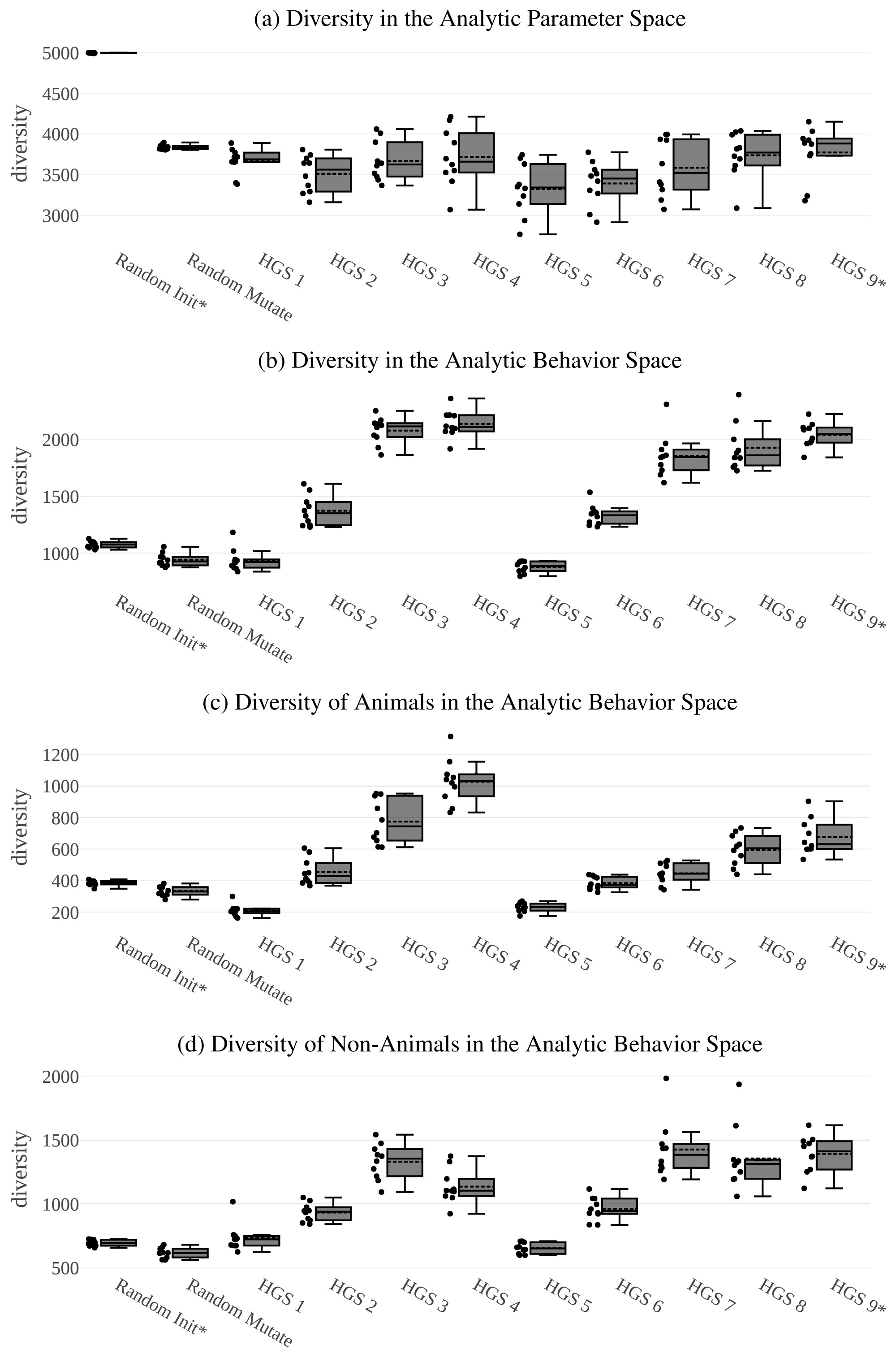


Figure 15: Although all IMGEP-HGS variants have lower diversity in the analytic parameter space compared to the Random Initialization approach, most of them have a higher diversity in the analytic behavior space. Each dot besides the boxplot shows the diversity of found patterns for each repetition ($n = 10$). The box ranges from the upper to the lower quartile. The whiskers represent the upper and lower fence. The mean is indicated by the dashed line and the median by the solid line.

E IMGEPs with Random and Learned Goal Spaces using Deep Variational Autoencoders

We considered three random initializations for the VAE representation used in the IMGEP-RGS as well as three different training objectives for learning the VAE goal space used in IMGEP-PGL and IMGEP-OGL. Variational Autoencoders (VAEs) (Kingma & Welling, 2013; Rezende et al., 2014) are commonly used deep generative models that can unsupervisedly learn a latent representation of the data. The latent representation has a reduced number of dimensions and should capture the important features of the input data. We use VAEs to learn the important features that describe Lenia patterns. The features are then used to define goal spaces for IMGEPs. This section details the different variants that were implemented for IMGEP with random and learned VAE goal spaces, the implementation details and compares the diversity results as well as the VAEs reconstruction accuracy.

E.1 IMGEP with Random VAE Goal Spaces

To study the impact of learning representations we implemented IMGEP-RGS as an ablated version of IMGEPs where the goal space is based on the encoder of a VAE with random weights. We evaluated three variants to randomly set the weights of the VAE encoder: Pytorch (Paszke et al., 2017), Xavier (Glorot & Bengio, 2010) and Kaiming (He et al., 2015). The VAE is composed of four 2D convolutional layers (with ReLU activations) followed by three fully-connected layers. Table 7 shows the different sampling distributions from which the encoder parameters are initialized. We used uniform distributions for both Xavier and Kaiming variants and set all the layers bias parameters to zero.

		Convolutional Layers		Linear Layers	
RGS Variants	bound b	weight	bias	weight	bias
Pytorch	$\frac{1}{\sqrt{\text{fan}_{\text{in}}}}$	$\mathcal{U}(-b,b)$	$\mathcal{U}(-b,b)$	$\mathcal{U}(-b,b)$	$\mathcal{U}(-b,b)$
Xavier	$\sqrt{\frac{6}{\text{fan}_{\text{in}}+\text{fan}_{\text{out}}}}$	$\mathcal{U}(-b,b)$	0	$\mathcal{U}(-b,b)$	0
Kaiming	$\sqrt{\frac{6}{\text{fan}_{\text{in}}}}$	$\mathcal{U}(-b,b)$	0	$\mathcal{U}(-b,b)$	0

Table 7: IMGEP-RGS variants initialization schemes. fan_{in} is the number of input units in the weight tensor and fan_{out} is the number of output units in the weight tensor.

E.2 IMGEP with Learned VAE Goal Spaces

E.2.1 VAE framework

VAEs have two components: a neural *encoder* and *decoder*. The encoder $q(\mathbf{z}|\mathbf{x},\chi)$ represents a given data point x in a latent representation $\mathbf{z}$. In variational approaches the encoder describes a data point by a representative distribution in the latent space of reduced dimension d. A standard Gaussian prior $p(\mathbf{z}) = \mathcal{N}(0, I)$ and a diagonal Gaussian posterior $q(\mathbf{z}|\mathbf{x},\chi) = \mathcal{N}(\mu,\sigma)$ are used for this purpose. Given a data point x, the encoder outputs the mean μ and variance σ of the representative distribution in the latent space. The decoder $p(\mathbf{x}|\mathbf{z},\psi)$ tries to reconstruct the original data x from a sampled latent representation $\mathbf{z}$ for the distribution given by the encoder.

Under these assumptions, training is done by maximizing the computationally tractable evidence lower bound (with $\beta = 1$):

$$\mathcal{L}(\chi,\psi) = \underbrace{\mathbb{E}_{\mathbf{z}\sim q_\chi(\mathbf{z}|\mathbf{x})}[\log p_\psi(\mathbf{x}|\mathbf{z})]}_{a} - \beta \times \underbrace{\mathbb{D}_{KL}[q_\chi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})]}_{b} \ . \tag{5}$$

The first term (a) represents the expected reconstruction accuracy while the second (b) is the KL divergence of the approximate posterior from the prior.

$$b = \mathbb{D}_{KL}[\mathcal{N}(\mu(\mathbf{x}), \Sigma(\mathbf{x})) \| \mathcal{N}(0, I)] = \sum_{i=1}^{d} \underbrace{\mathbb{D}_{KL}[\mathcal{N}(\mu(\mathbf{x})_i, \sigma(\mathbf{x})_i) \| \mathcal{N}(0, 1)]}_{b_i} . \tag{6}$$

E.2.2 VAE VARIANTS

The recent growing interest in unsupervised representation learning, and therefore in VAEs, resulted in a plethora of proposed losses, network designs and choices of family for the encoder, decoder and prior distributions (Tschannen et al., 2018). In order to enhance desired properties such as interpretability and disentanglement of the latent variables, many current state-of-the-art approaches build on the VAE framework and augment the VAE objective (Higgins et al., 2017; Burgess et al., 2018; Kim & Mnih, 2018; Chen et al., 2018; Kumar et al., 2017).

In this paper, we couple the VAE architecture with three different objectives: the classical VAE objective (Kingma & Welling, 2013) (equation 5 with $\beta = 1$), the β-VAE objective (Higgins et al., 2017) equation 5 with $\beta > 1$) and an augmented β-VAE objective (equation 7).

The β-VAE objective re-weights the b term by a factor $\beta > 1$, aiming to enhance the disentangling properties of the learned latent factors. We are interested in such properties as it has been shown that it can benefit exploration (Laversanne-Finot et al., 2018). However, heavily penalizing b can result in the network learning to "sacrifice" one or more of the learned latent variables in order to nullify their contribution b_i (equation 6). Those dimensions become completely uninformative and useless for further exploration in the learned latent space. This phenomenon is known as *posterior collapse* and is a common problem when training VAEs (Bowman et al., 2015; Chen et al., 2016; He et al., 2019; Kingma et al., 2016).

To prevent this phenomenon to happen, we then considered an augmented β-VAE objective with a new term that encourages the network to decrease *together* the individual contributions b_i of the different latent variables. This augmented loss term not only minimizes the averaged contribution (sum) but also the variance of the individual contributions:

$$b_{aug} = \sum_{i=1}^{d} b_i + \mathrm{Var}\left([b_1, \dots, b_d]\right) . \tag{7}$$

Similarly other modifications of the training objective can be found in the literature to avoid posterior collapse (Tolstikhin et al., 2017; Zhao et al., 2017a).

By writing the VAE training objective as stated in equation 8, the three different variants outlined above correspond to the following set of hyper-parameters $\{\beta = 1, \gamma = \mathbf{0}\}$, $\{\beta = 5, \gamma = \mathbf{0}\}$ and $\{\beta = 5, \gamma = \mathbf{1}\}$.

$$L = -a + \beta\left(b + \gamma \times c\right) \tag{8}$$

E.3 IMPLEMENTATION DETAILS

This section describes the IMGEP approaches (RGS, PGL and OGL) and the network architecture, training procedure, hyper-parameters and datasets for the training of their VAEs.

All VAEs use the same architecture (Table 8). The encoder network has as input the Lenia pattern and as outputs for each latent variable $\mathbf{z}_i$ the mean μ_i and log-variance $\log(\sigma_i^2)$. The decoder takes as input during the training for each latent variable a sampled value $z_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$. For validation runs and the generation of all reconstructed patterns shown in figures the decoder takes the mean $z_i = \mu_i$ as input. Its output is the reconstructed pattern.

The training objectives of all three variants are given in section E.2.1. The resulting loss function (Eq. 8) of all VAE variants for a batch is:

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

Algorithm 3: IMGEP-RGS

```
1 Initialize goal space encoder $\mathcal{R}$ with a VAE with random weights
2 for $i \leftarrow 1$ to $N$ do
3     if $i < N_{init}$ then          // Initial random iterations to populate $\mathcal{H}$
4         Sample $\theta \sim \mathcal{U}(\Theta)$
5     else                            // Intrinsically motivated iterations
6         Sample a goal $g \sim \mathcal{G}(\mathcal{H})$ based on space represented by $\mathcal{R}$
7         Choose $\theta \sim \Pi(g, \mathcal{H})$
8     Perform an experiment with $\theta$ and observe $o$
9     Append $(o, \theta, \mathcal{R}(o))$ to the history $\mathcal{H}$
```

where x are the input patterns, $\hat{x}$ are the reconstructed patterns, μ, σ are the outputs of the decoder network and d is the number of latent dimensions. The reconstruction accurray part a of the loss is given by a binary cross entropy with logits:

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

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

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

All VAEs were trained for 2000 epochs and initialized with pytorch default initialization. We used the Adam optimizer (Kingma & Ba, 2014) ($lr = 1\mathrm{e}{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1\mathrm{e}{-8}$, weight decay=$1\mathrm{e}{-5}$) with a batch size of 64.

The patterns from the datasets were augmented by random x and y translations (up to half the pattern size and with probability 0.3), rotation (up to 40 degrees and with probability 0.3), horizontal and vertical flipping (with probability 0.2). The translations and rotations were preceded by spherical padding to preserve Lenia spherical continuity.

Three types of IMGEPs were evaluted:

IMGEP-RGS (random goal space): IMGEP with a goal space defined by an encoder network with random weights (Algorihm 3). The network architecture of the encoder is the same that the one of the VAEs used for IMGEP with learned goal spaces. In the other IMGEP algorithms (HGS/PGL/OGL), the goals are sampled uniformly within fixed-range boundaries that are chosen in advance. However, in the case of random goal spaces, we do not know in advance in which region of the space goals will be encoded. Therefore, we set the range to $[-min, max]$ for each of the latent variables, to also bias exploration towards the boundaries of the discovered goal space.

IMGEP-PGL (prelearned goal space): IMGEP (Algorihm 4) with a goal space defined by a VAE that was trained before the exploration starts. The VAE is trained on a dataset with precollected Lenia

Encoder	**Decoder**
Input pattern A: $256 \times 256 \times 1$	Input latent vector z: 8×1
Conv layer: 32 kernels 4×4, stride 2, 1-padding + ReLU	FC layers : 256 + ReLU, $16 \times 16 \times 32$ + ReLU
Conv layer: 32 kernels 4×4, stride 2, 1-padding + ReLU	TransposeConv layer: 32 kernels 4×4, stride 2, 1-padding + ReLU
Conv layer: 32 kernels 4×4, stride 2, 1-padding + ReLU	TransposeConv layer: 32 kernels 4×4, stride 2, 1-padding + ReLU
Conv layer: 32 kernels 4×4, stride 2, 1-padding + ReLU	TransposeConv layer: 32 kernels 4×4, stride 2, 1-padding + ReLU
FC layers : 256 + ReLU, 256 + ReLU, FC: 2×8	TransposeConv layer: 32 kernels 4×4, stride 2, 1-padding

Table 8: VAE architecture for the pretrained and online experiments.

patterns. The best VAE model obtained during the training phase, i.e. the one with with the highest accuracy on the validation data, is used for the exploration.

The dataset used to train the VAE has 558 patterns which are distributed into a training (75%), validation (10%) and testing (15%) datasets. Half of the patterns (279) were manually identified animal patterns by Chan (2018) (Fig. 5). The other half (279) are randomly initialized CPPN patterns as described in Section B.1 (Fig. 9).

During the intrinsically motivated iterations, goals are uniformly sampled in the hypercube $[-3, 3]^8$. This values were chosen because the encoder of the VAE is trained to match a prior standard normal distribution (through the KL divergence term), therefore we can assume that most area of the covered goal space will fall into that hypercube.

Algorithm 4: IMGEP-PGL

```
1 Initialize goal space encoder $\mathcal{R}$ with the pretrained VAE
2 for $i \leftarrow 1$ to $N$ do
3     if $i < N_{init}$ then           // Initial random iterations to populate $\mathcal{H}$
4         Sample $\theta \sim \mathcal{U}(\Theta)$
5     else                             // Intrinsically motivated iterations
6         Sample a goal $g \sim \mathcal{G}(\mathcal{H})$ based on space represented by $\mathcal{R}$
7         Choose $\theta \sim \Pi(g, \mathcal{H})$
8     Perform an experiment with $\theta$ and observe $o$
9     Append $(o, \theta, \mathcal{R}(o))$ to the history $\mathcal{H}$
```

IMGEP-OGL (online learned goal space): IMGEP (Algorihm 1 in the main paper) that trains the VAE which defines the goal space during the exploration. The VAE is trained on Lenia patterns discovered by the algorithm. Every $K = 100$ explorations the VAE model is trained for 40 epochs resulting in 2000 epochs in total (less if there is not enough data after the first T runs to start the training).

Importance sampling is used to give the patterns in the training dataset a different weight during the training. A weighted random sampler is used that samples newly discovered patterns from the training dataset half of the time. Each pattern that has been added to the training dataset during the last period of 100 explorations has a probability of $\frac{0.5}{N}$ to be sampled (N is the total number of new patterns in the dataset). Older patterns are also sampled half of the time each one with probability $\frac{0.5}{|D_{\mathcal{T}}|-N}$. As a result, newer discovered patterns have a higher weight and a stronger influence on the training of the VAE model.

The datasets were constructed incrementally during the exploration by gathering non-dead patterns. One pattern every ten is added to the validation set (10%) and the rest is used in the training set. At the initial period of training, the training dataset amounts approximately 50 patterns and at the last period of training the dataset amounts approximately 3425 patterns (Fig. 16). The validation dataset only serves for checking purposes and has no influence on the learned goal space.

During the intrinsically motivated iterations, goals are uniformly sampled in the hypercube $[-3, 3]^8$.

E.4 Results

We compared the different IMGEP-RGS variants as well as the different objective variants for IMGEP with learned goal spaces (PGL and OGL) with each other on the basis of the diversity of their identified patterns. Furthermore, the pattern reconstruction ability of the VAEs is analyzed.

E.4.1 Diversity

The algorithms are compared by their diversity in the analytic parameter and behavior space (Section C). Diversity is measured by the number of discretized bins that were explored by the algorithms in each space if each dimension of the space is seperated in 7 bins.

All the IMGEPs with learned goal spaces reached a higher diversity in the analytic behavior space compared to random explorations (Fig. 17, b), although random explorations have a higher diversity in the analytic parameter space (Fig. 17, a). This result confirms further the advantage of IMGEPs over random explorations in terms of identifying diverse patterns.

Furthermore, all the IMGEPs with learned goal spaces outperformed the IMGEP with random goal space. This result shows the importance of learning relevant pattern features that, combined with an effective exploration process, is key to discover a high diversity of patterns.

There is no significant differences between Xavier and Kaiming IMGEP-RGS variants. They both seem to present a higher variance than the Pytorch variant and reach therefore a higher average performance, but it is unclear why. Because the Xavier initialization performed slightly the best for IMGEP-RGS, it was used for the results in the main paper and in Section F.

The difference between the PGL and OGL variants were small for all diversity measures. The OGL showed a slight advantage over the PGL versions in all diversity measures. Thus, an online version of the IMGEP can learn an appropriate goal space during the exploration. A precollected dataset as for the PGL is not necessary to successfully use IMGEPs.

The difference between the VAE objective variants (VAE, β-VAE and augmented β-VAE) was very small. The β-VAE was slightly better than the other two variants for the diversity in the analytic parameter space and for both IMGEP variants. All VAEs seemed to learn similar features for our datasets. It might be possible that the different VAE variants show different behaviors if their parameters are fine-tuned, such as the β parameter, but this was out of the scope of this paper. Because the β-VAE objective performed slighly the best for IMGEP-PGL and IMGEP-OGL, it was used for the results in the main paper and in Section F.

E.4.2 VAE Pattern Reconstruction

All learned VAE variants showed similar learning curves on the precollected dataset and the online collected dataset (Fig. 18 and 20). Their ability to reconstruct patterns based on the encoded latent representation is also qualitatively similar. For both datasets the VAEs are able to learn the general form of the activity pattern (Fig. 19 and 21). Nonetheless, the compression of the images to a 8-dimensional vector results in a general blurriness in the reconstructed patterns. As a result, the VAEs are not able to encode finer details and textures of patterns (Fig. 22). We believe this is the reason for their ability to identify more animals compared to the random exploration or the IMGEP-RGS and IMGEP-HGS. Different animals have often a different form, whereas non-animals span often over the whole area of Lenia's grid and differentiate mainly in their textures and small details. Because the VAE seem to encode more the general form a goal space based on them is more appropriate to find patterns with different forms such as the animals and not different textures which are important for non-animals.

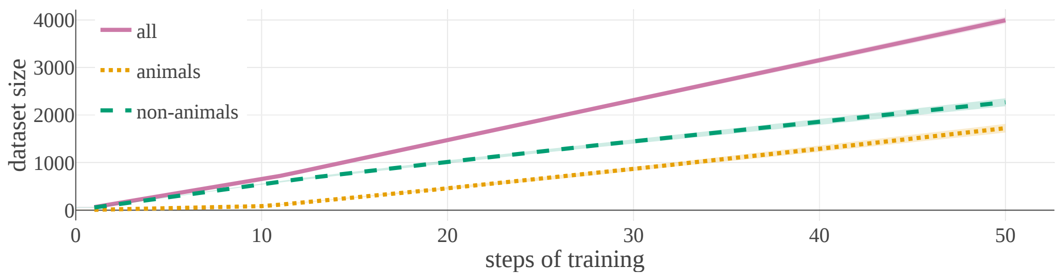
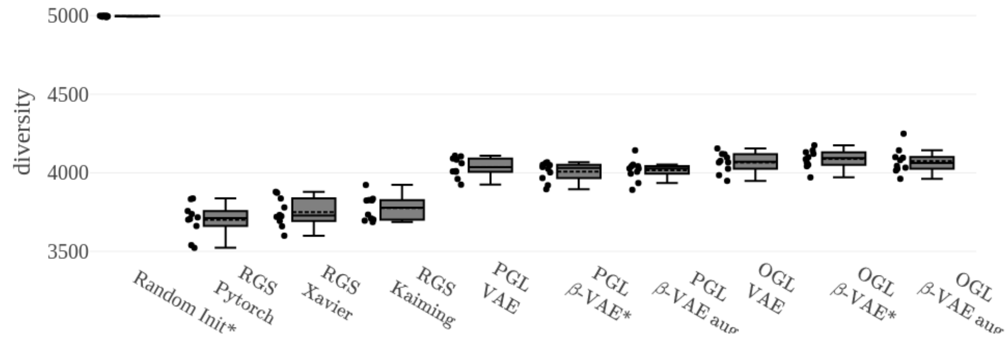
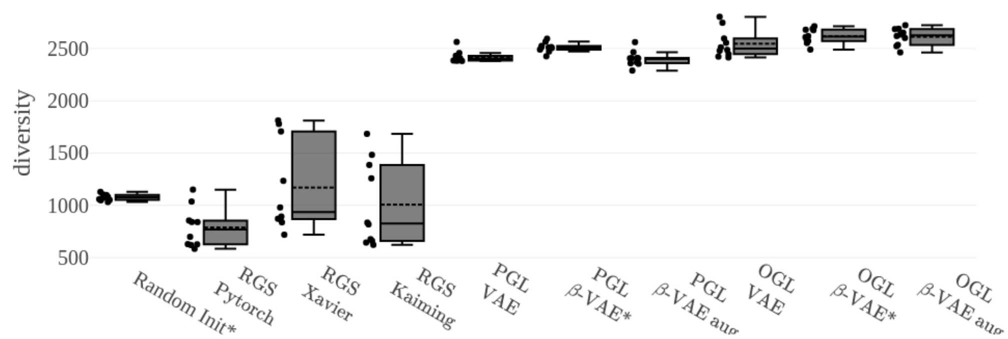


Figure 16: The IMGEP-OGL collects during the exploration animal and non-animal patterns to add them to its dataset for the training of the VAE. The figure shows the development of the averaged dataset size over all repetitions ($n = 10$) of the IMGEP-OGL algorithm with a β-VAE. Standard deviation is depicted as a shaded area but for some not visible because it is too small.

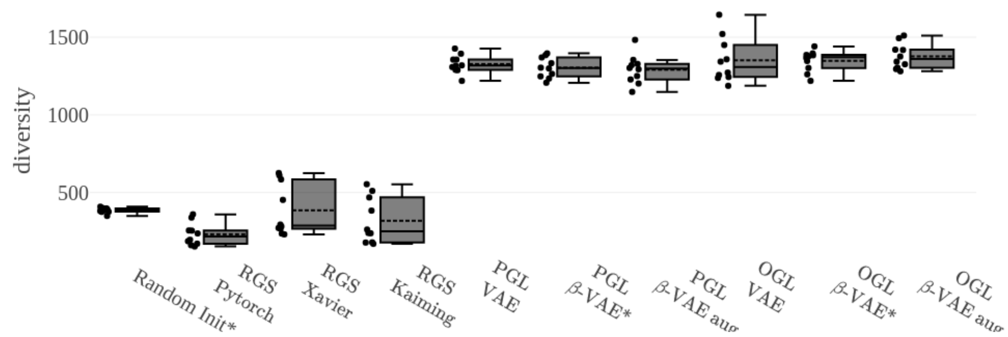
(a) Diversity in the Analytic Parameter Space



(b) Diversity in the Analytic Behavior Space



(c) Diversity of Animals in the Analytic Behavior Space



(d) Diversity of Non-Animals in the Analytic Behavior Space

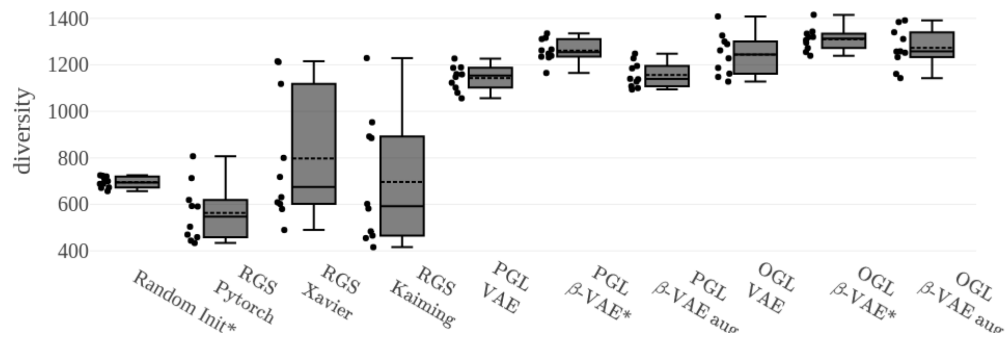


Figure 17: The different VAE algorithms showed only small differences in terms of diversity. The β-VAE had a slighlty better diversity for the analytic behavior space for both IMGEP variants (PGL and OGL). Each dot besides the boxplot shows the diversity of found patterns for each repetition ($n = 10$). The box ranges from the upper to the lower quartile. The whiskers represent the upper and lower fence. The mean is indicated by the dashed line and the median by the solid line.

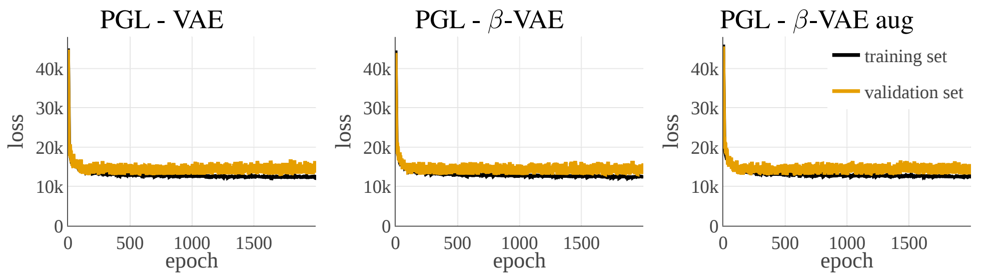


Figure 18: Averaged learning curves ($n = 10$) of the VAEs for the IMGEP-PGL experiments.

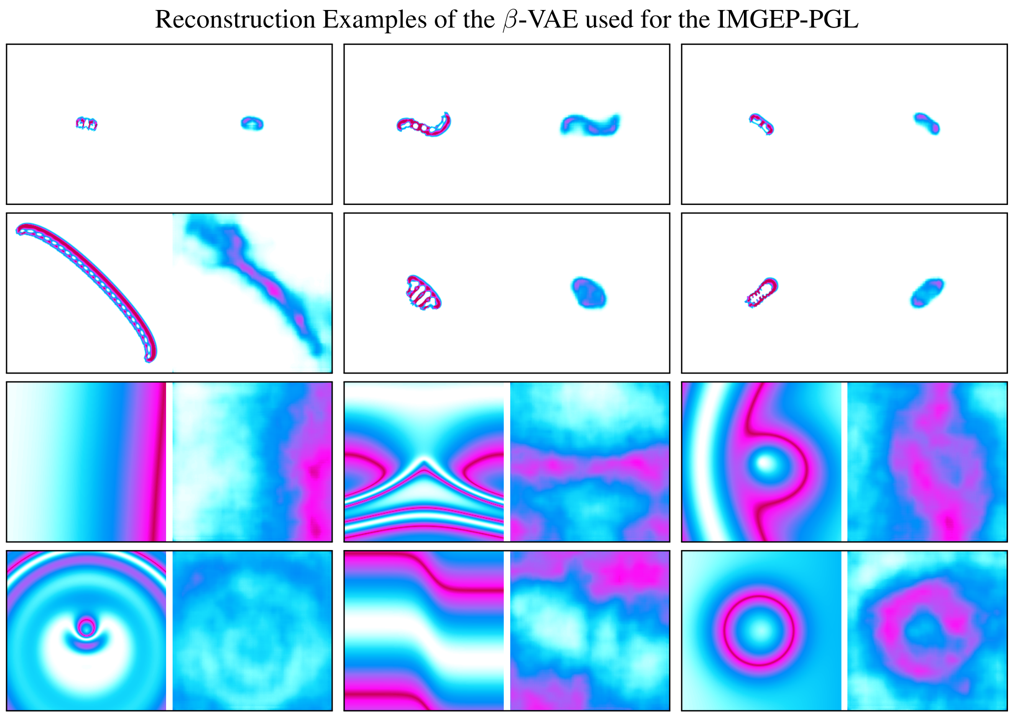

Reconstruction Examples of the β-VAE used for the IMGEP-PGL

Figure 19: Examples of patterns (left) and their reconstructed output (right) by a VAE network used for the IMGEP-PGL. The patterns are sampled from its validation dataset. The dataset is composed of half animal patterns (rows 1 and 2) and half randomly generated CPPN patterns (rows 3 and 4).

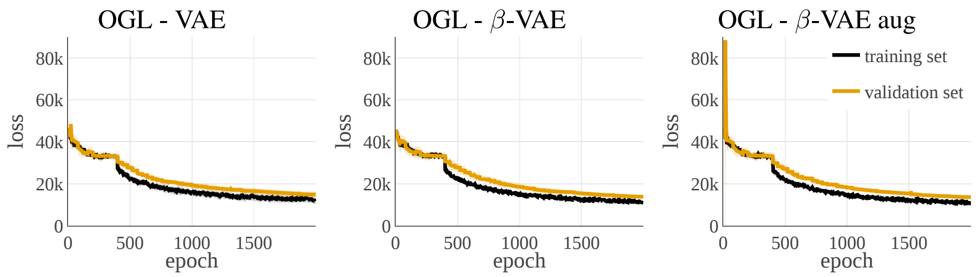


Figure 20: Averaged learning curves ($n = 10$) of the VAEs for the IMGEP-OGL experiments.

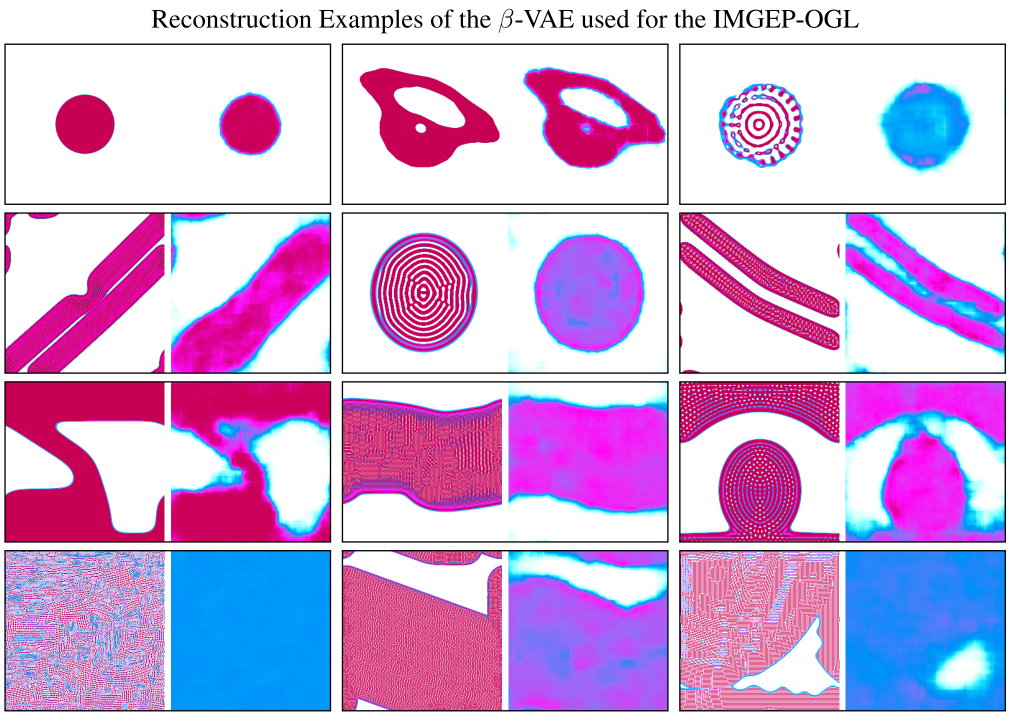

Reconstruction Examples of the β-VAE used for the IMGEP-OGL

Figure 21: Examples of patterns (left) and their reconstructed output (right) by a VAE network used for the IMGEP-OGL. The patterns are sampled from its validation dataset. Animal patterns (rows 1 and 2) and non-animal patterns (rows 3 and 4) are shown.

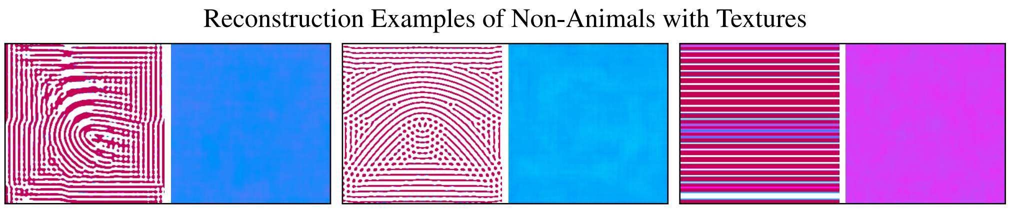


Figure 22: Examples of “textured” patterns that the VAE networks is unable to reconstruct. While a human eye can differentiate the input patterns (spatial frequency, orientation, etc.) the VAE reconstructs all images identically.

F Additional Results

This section lists additional results. The results are only for a subset of all algorithm variants that have been evaluated. The results correspond to the following algorithms: Random to Random Initialization (Section D), IMGEP-RGS to IMGEP-RGS Xavier (Section E), IMGEP-HGS to IMGEP-HGS 9 (Section D), IMGEP-PGL to IMGEP-PGL with a β-VAE (Section E) and IMGEP-OGL to IMGEP-OGL with a β-VAE (Section E).

F.1 Number of Identified Patterns

The main paper used the measure of diversity of the found patterns per algorithm to compare their performance. Another measure to compare the algorithms is the number of the patterns they identified for each of the three pattern classes: dead, animals, non-animals (Fig. 23).

The results deviate slightly from the diversity measures. In terms of identified non-dead patterns, all IMGEP approaches outperform a random exploration by finding between 10 to 20% more patterns. Although the IMGEP-RGS and IMGEP-HGS find more non-dead patterns than the IMGEPs with learned goal spaces (OGL, PGL) its overall diversity in the analytic behavior space is smaller (Fig. 3, b of the main paper).

In the case of animal patterns, all IMGEP approaches outperform the random exploration (8%). Within the IMGEP approaches the online learned goal space approach (IMGEP-OGL, 34%) and the pretrained goal space approach (IMGEP-PGL: 35%) find a similar amount. The hand-defined goal space approach identified less animal patterns (IMGEP-HGS: 19%). The random goal space approach is the one that finds the least (IMGEP-RGS: 10%). For non-animal patterns, the random goal space approach identifies most patterns (IMGEP-RGS: 79%) , followed by the hand-defined approach (IMGEP-HGS: 67%), the random exploration (56%) and both learned goal space approaches (IMGEP-OGL: 45%, IMGEP-PGL: 43%). Although the number of identified non-animal patterns for the learned goal space approaches is low, their diversity is higher than for a random exploration and the random goal space approach, and only slightly lower than for the hand-defined goal space approach (Fig. 3, d of the main paper).

F.2 Dependence of the Diversity Measure on the Number of Bins per Dimension

The diversity of identified patterns measures the spread of the area which the identified patterns cover in the analytic behavior space (Fig. 3 in the main paper). The measure is defined by dividing the space in a number of discrete areas or bins (Section C.1). The diversity is then measured by how many bins are covered during an exploration. The bins are created by dividing each dimension of the space into a number of equally-sized bins. We analyzed how the number of bins per dimension influences the diversity measure (Fig. 24).

Although the diversity difference between the algorithms depends on the number of bins per dimension for each space, the order of the algorithms, i.e. which algorithm has a higher diversity, is generally constant. Only if the number of bins per dimension grows large ($>$10) the order of the algorithms changes for some spaces and subpatterns. The order starts to follow the order seen for the number of identified patterns (compare the diversity with 25 bins per dimension in Fig. 24 with the number of identified patterns in Fig. 23). In this case the discretization of the space becomes too fine and each pattern falls into its own discretized area. We chose therefore a smaller number of bins per dimension of 7 (including the out of border bins) for all other diversity plots in the main paper and the Supplementary Material to compare the algorithms in a meaningful way.

F.3 Dimension Reduction of the Analytic Parameter and Behavior Space

A two-dimensional reduction of the identified patterns in the analytic parameter and behavior space (Section C) visualizes the diversity of the parameters and identified patterns. The dimension reduction of the parameter space is based on all explored parameters encoded in the analytic parameter space from the first repetition experiment of all 4 algorithms. All encoded points were normalized so that the overall minimum value became 0 and the maximum value 1 for each dimension. Afterwards a principle component analysis (PCA) was performed to detect the 2 principle components (Jolliffe, 1986). The found patterns for each algorithm are plotted according to those components.

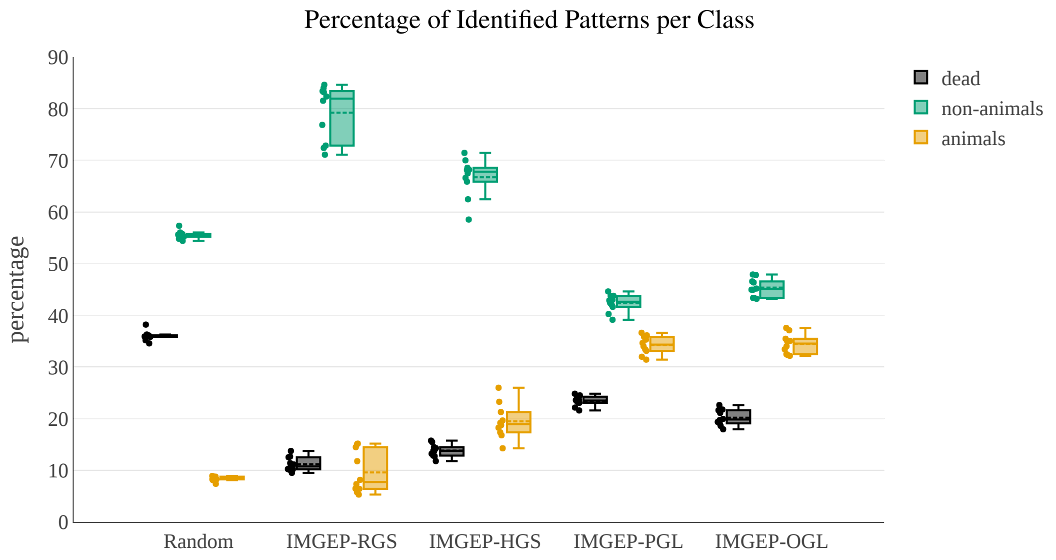


Figure 23: IMGEPs found less dead patterns compared to the random exploration. In terms of animals, the learned goal space approaches (IMGEP-PGL and OGL) found most animal patterns. For non-animals, the random goal space (IMGEP-RGS) found most patterns, followed by the hand-defined goal space (IMGEP-HGS). The plot illustrates the percentage of found patterns for each class. Each dot besides the boxplot shows the percentage of found patterns for each repetition ($n = 10$). The box ranges from the upper to the lower quartile. The whiskers represent the upper and lower fence. The mean is indicated by the dashed line and the median by the solid line.

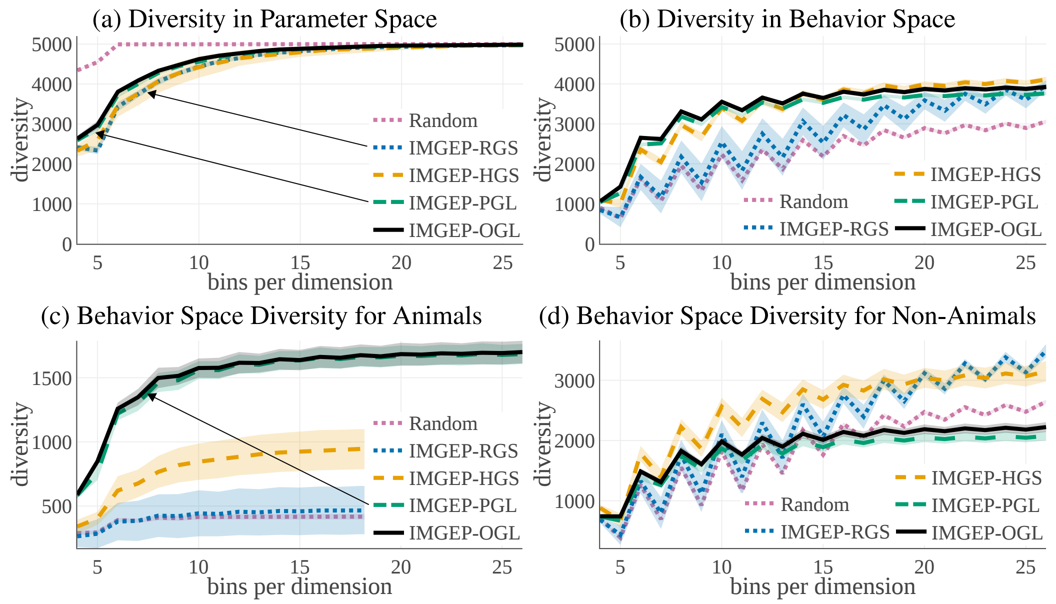


Figure 24: Dependencies of the diversity measure on the number of bins per dimensions for (a) the analytic parameter and (b-d) the behavior space. Depicted is the average diversity ($n = 10$) with the standard deviation as shaded area (for some not visible because it is too small).

The results show that the random exploration has a stronger uniform distribution than any of the IMGEP algorithms in the analytic parameter space (Fig. 25). The IMGEP algorithms show concentrations of explorations in specific regions of the parameter space. The visualization shows also that

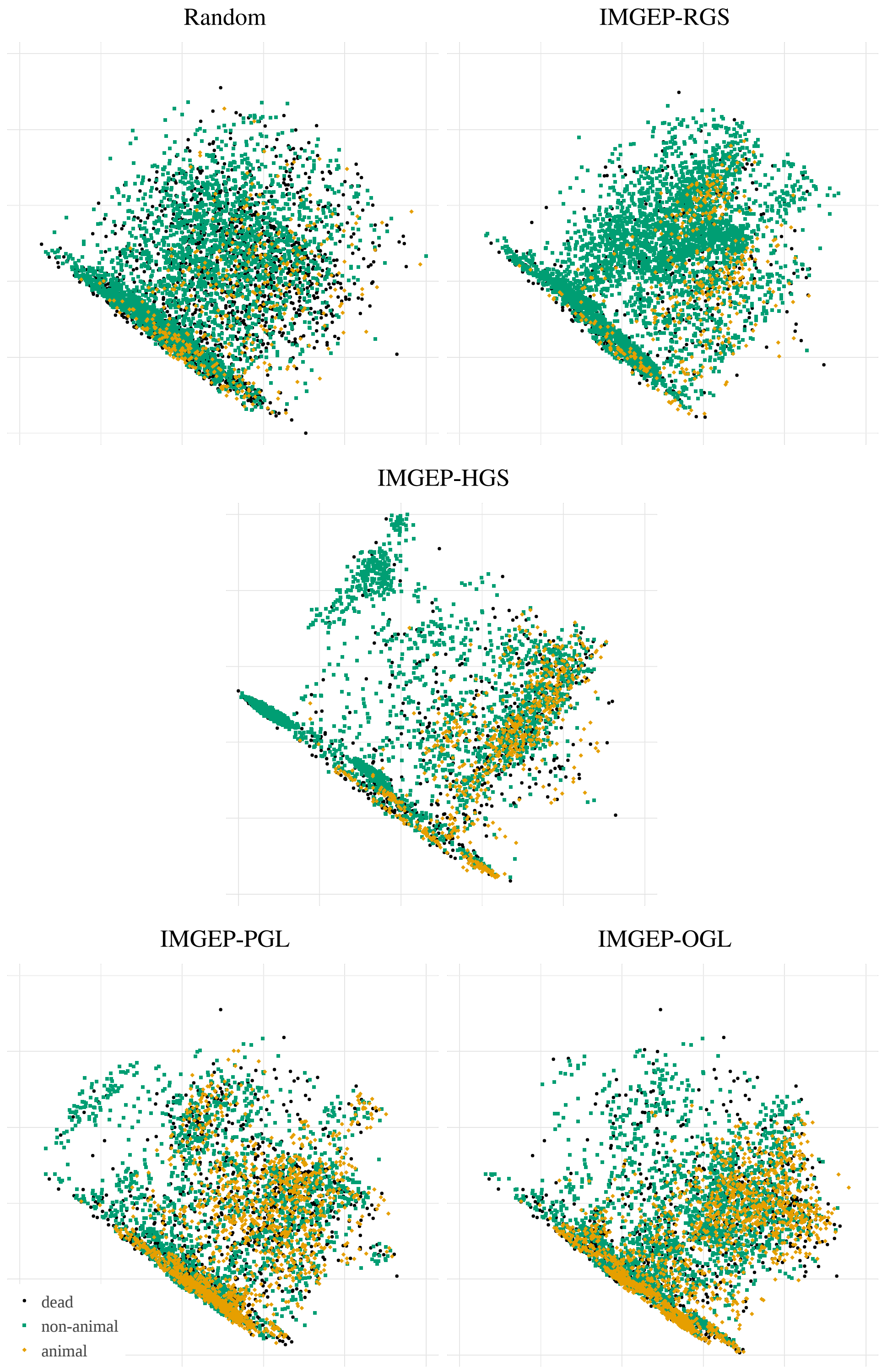


Figure 25: A random exploration covers the analytic parameter space more uniformly than IMGEP algorithms which form clusters at certain areas. PCA dimension reduction of the analytic parameter space which illustrates all explored parameters by the first repetition experiment per algorithm.

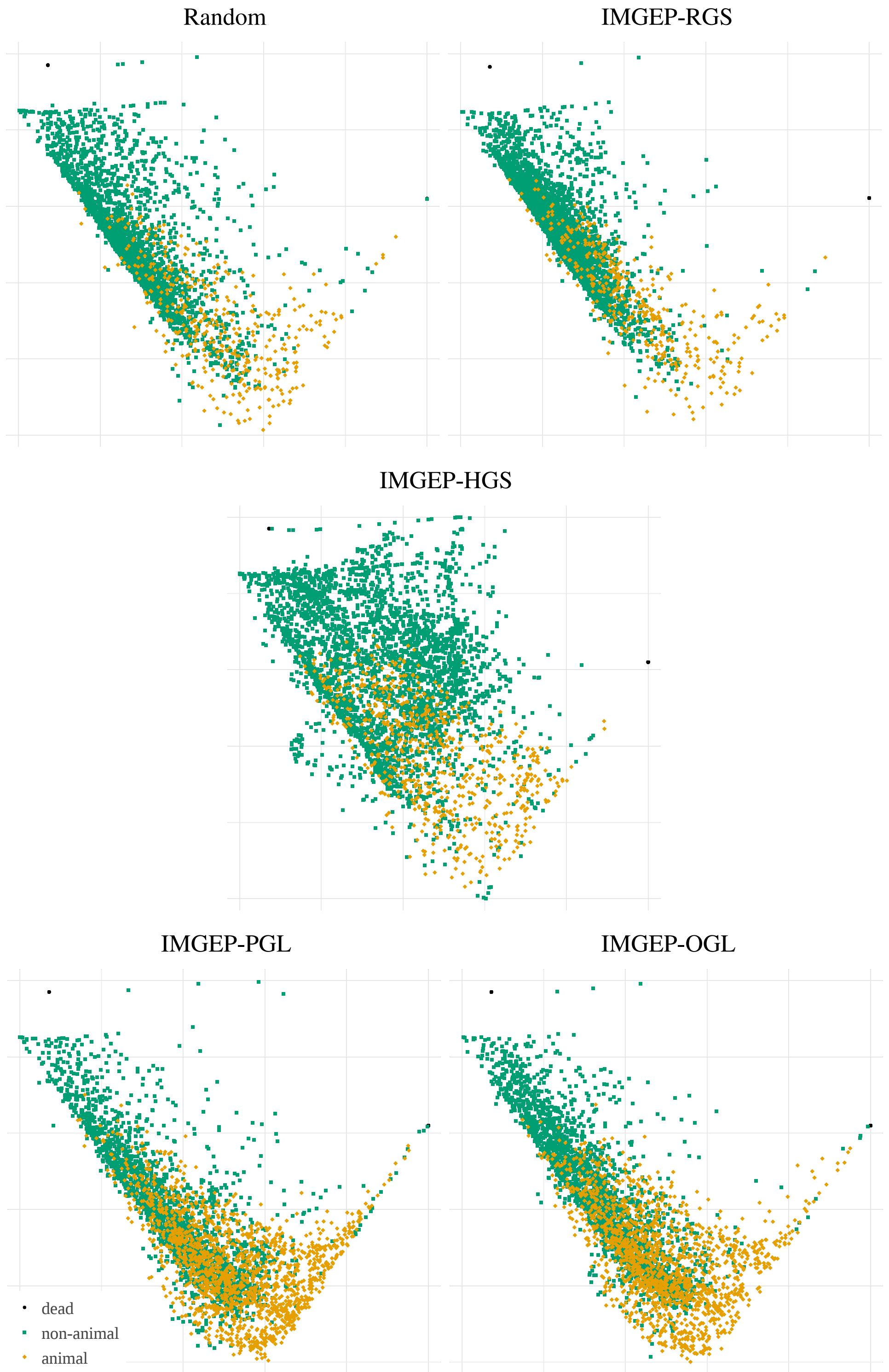


Figure 26: In the analytic behavior space IMGEPs reach a higher diversity compared to a random exploration, except IMGEP-RGS which covers similar areas. The HGS approach explores more non-animal areas and the PGL and OGL more animal areas. PCA dimension reduction of the analytic behavior space which illustrates all identified patterns by the first repetition experiment per algorithm.

it is not possible to define distinct regions in the parameter space that allow to differentiate between dead, animal and non-animal patterns.

The same analysis was performed for the identified patterns of each algorithm encoded in the analytic behavior space (Fig. 26). It is visible that the random exploration and the random goal space (RGS) approaches are more concentrated compared to the hand-defined (HGS) or learned goal spaces approaches (PGL and OGL), especially in a region with many non-animal patterns (north-west). The IMGEP-RGS has a similar spread than random exploration, with an even higher concentration of non-animals (north-west). The IMGEP-HGS has a wider spread in the non-animal area. The IMGEPs with a learned goal space (PGL and OGL) show a stronger distribution in an area that encodes mostly animals (south).

F.4 Identified Patterns

Fig. 27, 28, 29, 30 and 31 illustrate examples of identified pattern per class (animal, non-animal, dead) and their ratio for the random exploration, IMGEP-RGS, IMGEP-HGS, IMGEP-PGL and IMGEP-OGL. The patterns have been randomly sampled from the results of the first exploration repetition experiment of each algorithm.

F.5 Visualization of Goal Spaces

The goal space of IMGEPs is their most important element because it defines which type of patterns are set as goals for the exploration. This section provides complementary material for the analysis made in Section 5.2 of the main paper and shows further visualizations of goal spaces. The goal spaces of all IMGEP algorithms are visualized via a two-dimensional reduction of each goal space. Two techniques for dimensionality reduction were applied: PCA (Jolliffe, 1986) and t-Distributed Stochastic Neighbor Embedding (t-SNE) (Maaten & Hinton, 2008).

The visualization was constructed by using for each exploration algorithm its goal space representations of all patterns it explored from a single repetition experiment. All goal representations were normalized so that the overall minimum value became 0 and the maximum value 1 for each goal space dimension. Afterwards the PCA was performed to detect the 2 principle components. T-SNE was executed by using the default standard Euclidean distance metric and default hyper-parameters (perplexity set to 50).

The resulting two-dimensional visualizations of the goal spaces make the differences between the algorithms visible (Fig. 32). For both aproaches (PCA, t-SNE), the random goal space (RGS) and hand-defined goal space (HGS) have only a small area and a few clusters for animal patterns. In contrast, the learned goal spaces based on β-VAEs (PGL and OGL) have larger areas and more clusters for animal patterns. As a result, the learned goal spaces explore more animal patterns and find a higher diversity of them (Fig. 3, c) compared to the hand-defined goal space and the random goal space. The reason for this effect seems to be that the β-VAE which defines the goal space for the PGL and OGL is learning to represent the shape of patterns. The shape is an important feature of animals. Whereas, non-animals often cover the whole Lenia grid and differ mainly in their textures which the β-VAE does not represent well (Section E).

The visualization serves as a support in qualitatively evaluating and comparing the efficiency of each algorithm in extracting a diversity of patterns from the data. Integrated into an interactive interface, these graphs are also useful for a potential human end-user to easily explore and visualize the different type of found patterns during the exploration phase. Videos and demonstrations of the interface can be found on the website `https://automated-discovery.github.io/`.

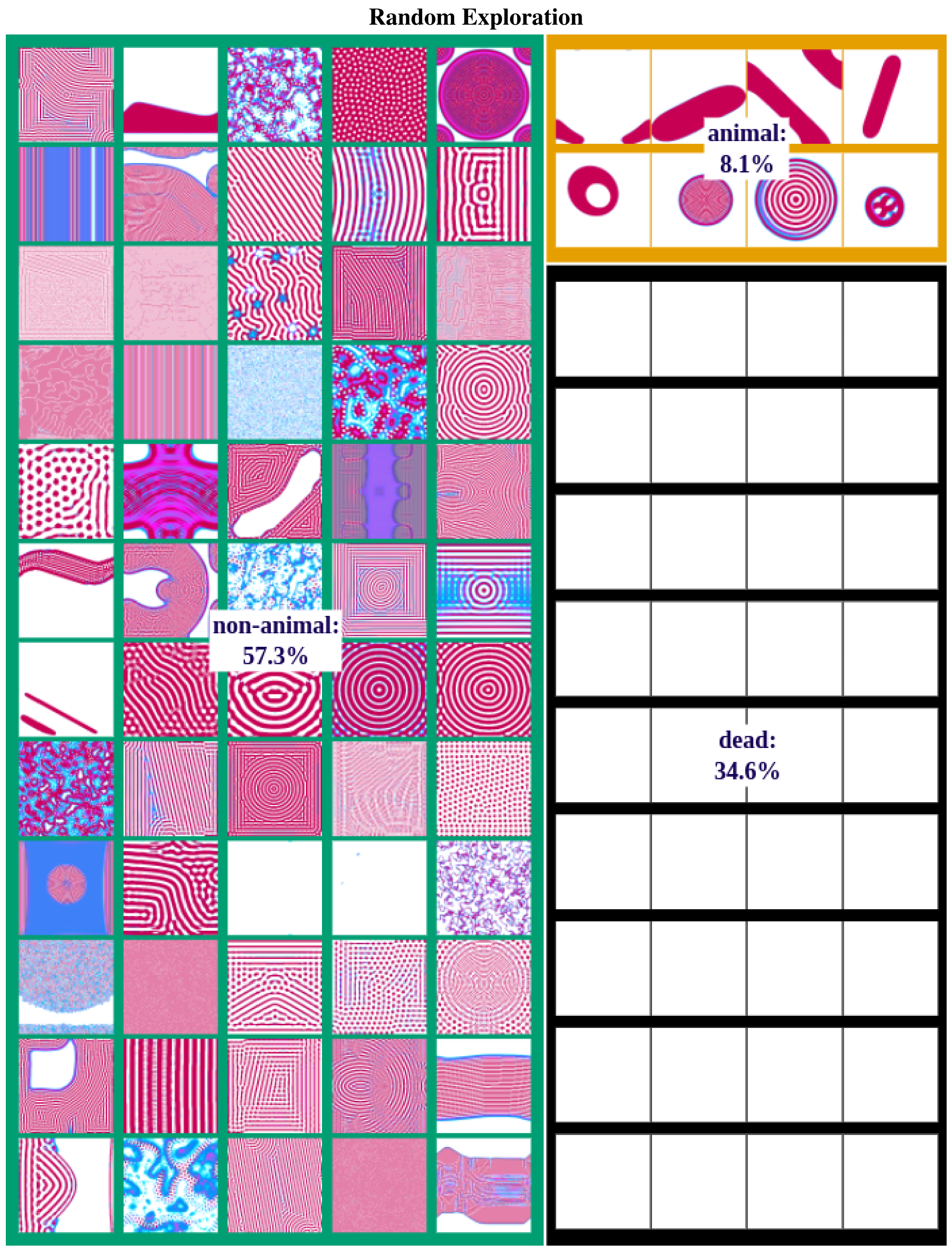


Figure 27: Examples of identified patterns for the random exploration algorithm from the first repetition of experiments.

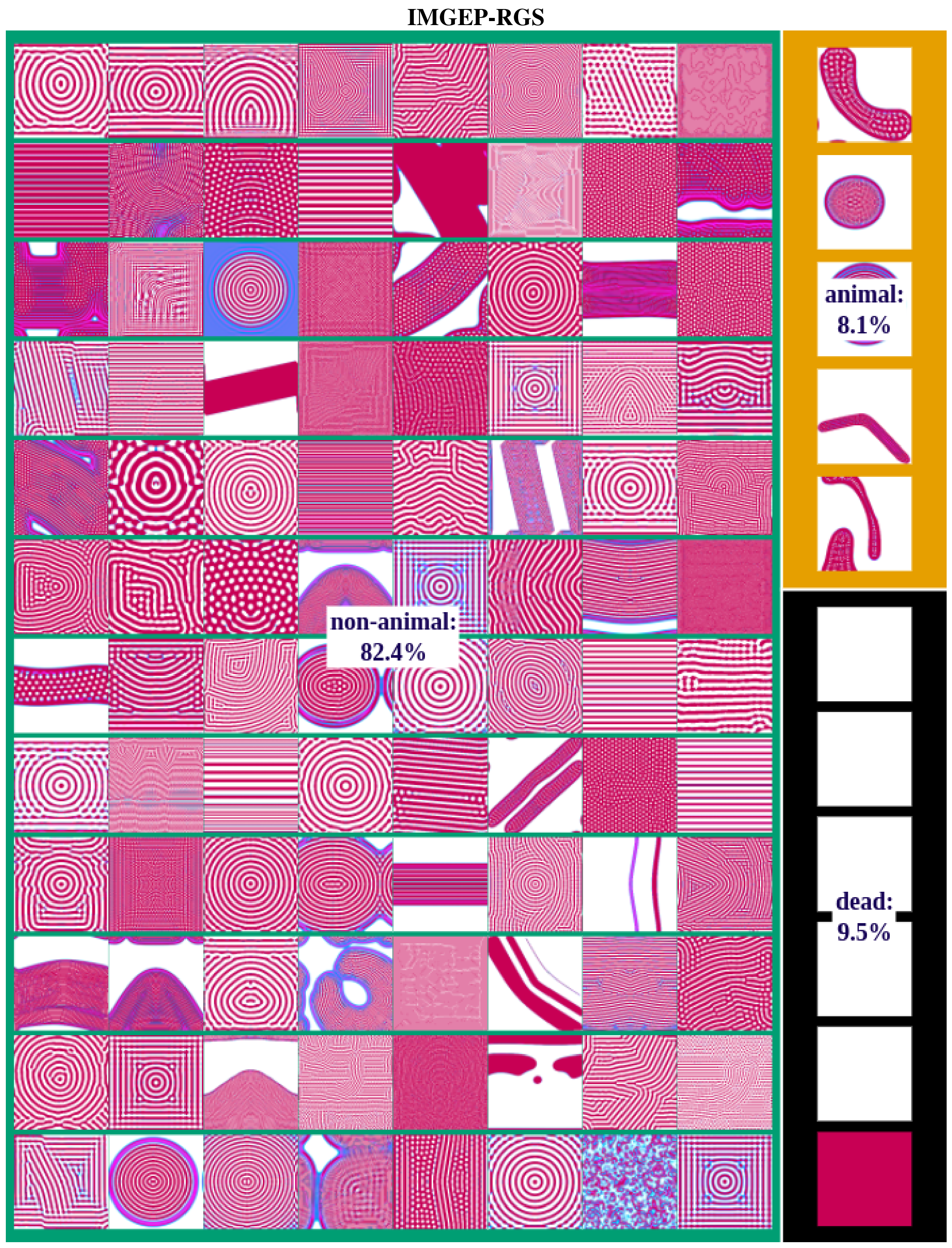


Figure 28: Examples of identified patterns for the IMGEP-RGS algorithm from the first repetition of experiments.

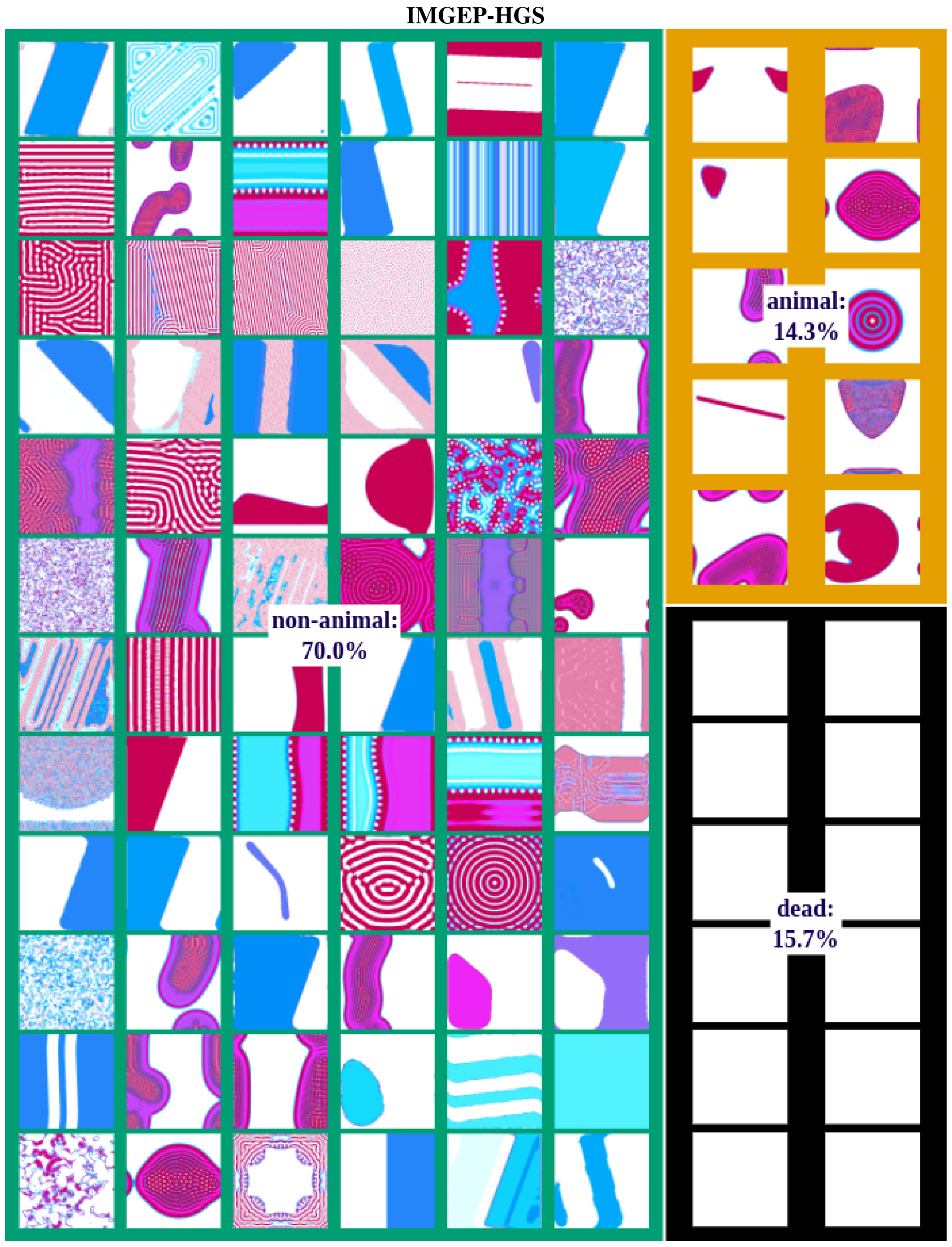


Figure 29: Examples of identified patterns for the IMGEP-HGS algorithm from the first repetition of experiments.

IMGEP-PGL

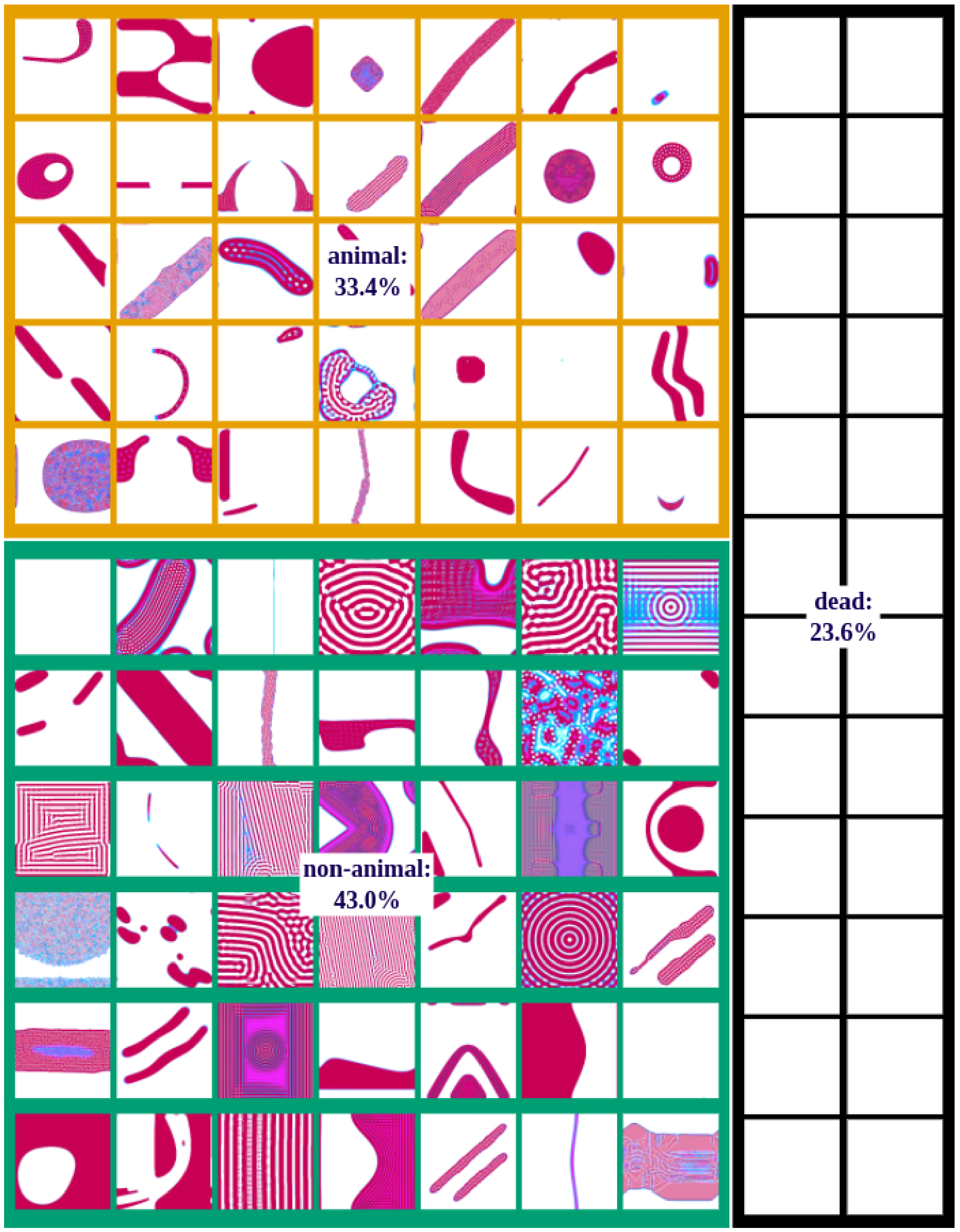


Figure 30: Examples of identified patterns for the IMGEP-PGL algorithm from the first repetition of experiments.

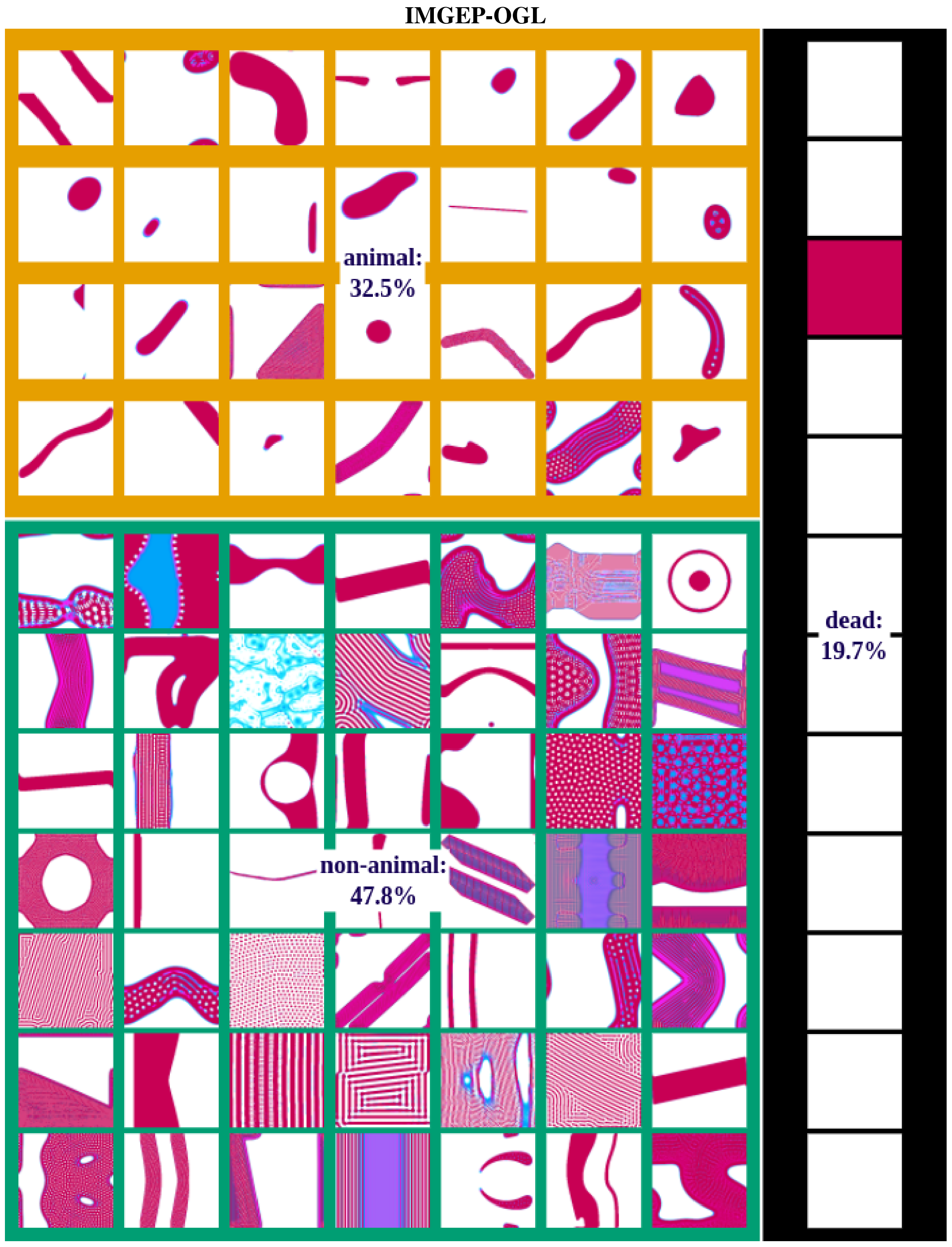


Figure 31: Examples of identified patterns for the IMGEP-OGL algorithm from the first repetition of experiments.

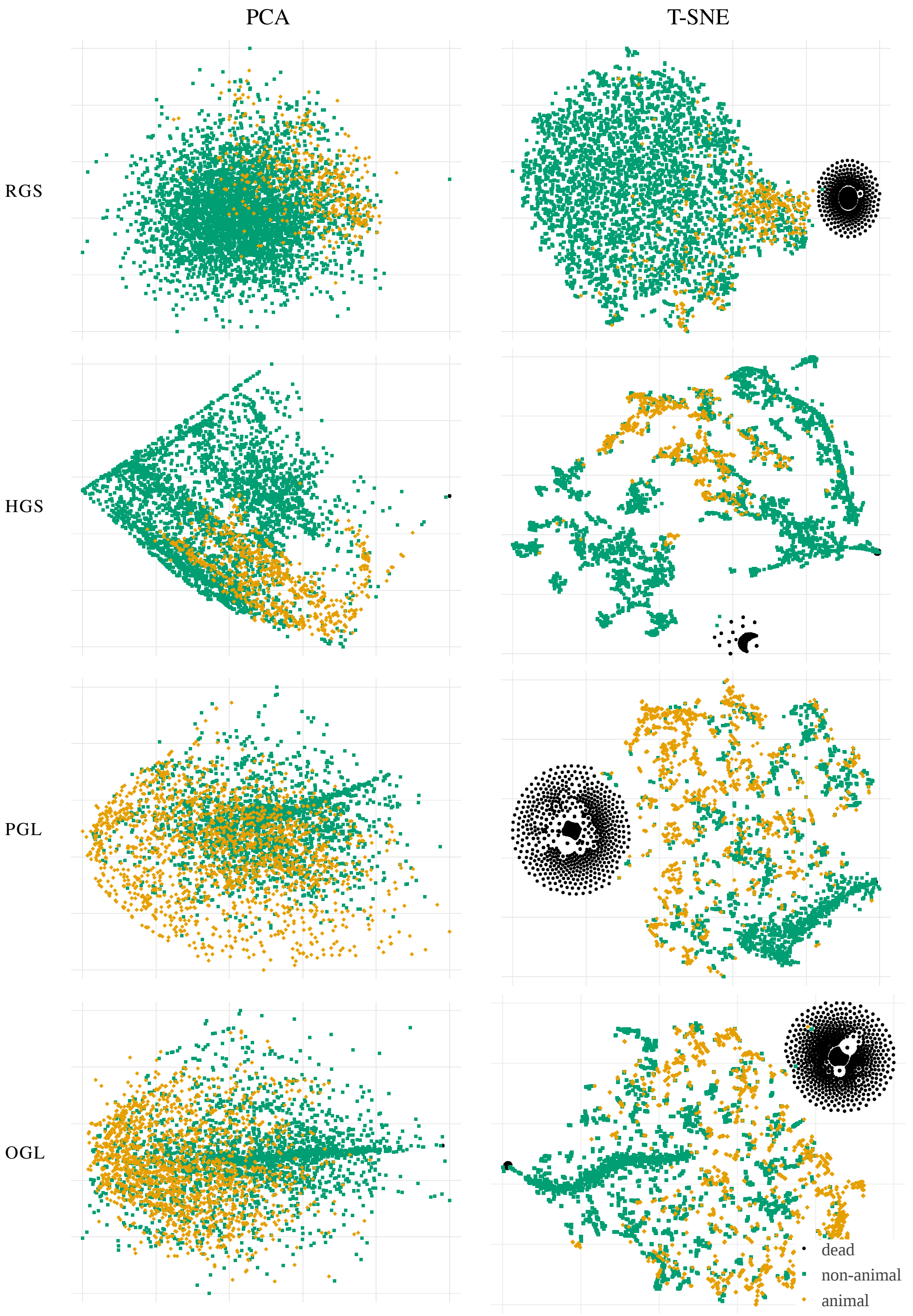


Figure 32: PCA and t-SNE visualization of the goal spaces for the IMGEP variants show that HGS has more area (PCA) and clusters (t-SNE) for non-animals compared to learned goal spaces (PGL and OGL) and vice versa for animals. t-SNE shows that the hand-defined goal space (HGS) and learned goal spaces (PGL and OGL) structure and cluster more the discovered patterns compared to random goal space (RGS).