Progressive growing of self-organized hierarchical representations for exploration
Mayalen Etcheverry, Pierre-Yves Oudeyer, Chris Reinke

To cite this version:
Mayalen Etcheverry, Pierre-Yves Oudeyer, Chris Reinke. Progressive growing of self-organized hierarchical representations for exploration. ICLR 2020 workshop: Beyond tabula rasa in Reinforcement Learning, Apr 2021, Addis Ababa / Virtual, Ethiopia. hal-03122039

HAL Id: hal-03122039
https://inria.hal.science/hal-03122039
Submitted on 26 Jan 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
PROGRESSIVE GROWING OF SELF-ORGANIZED HIERARCHICAL REPRESENTATIONS FOR EXPLORATION

Mayalen Etcheverry & Pierre-Yves Oudeyer & Chris Reinke
Flowers Team, Inria, France
{mayalen.etcheverry,pierre-yves.oudeyer,chris.reinke}@inria.fr

ABSTRACT

Designing agents that can autonomously discover and learn a diversity of structures and skills in unknown changing environments is key for lifelong machine learning. A central challenge is how to learn incrementally representations in order to progressively build a map of the discovered structures and re-use it to further explore. To address this challenge, we identify and target several key functionalities. First, we aim to build lasting representations and avoid catastrophic forgetting throughout the exploration process. Secondly, we aim to learn a diversity of representations allowing to discover a “diversity of diversity” of structures (and associated skills) in complex high-dimensional environments. Thirdly, we target representations that can structure the agent discoveries in a coarse-to-fine manner. Finally, we target the reuse of such representations to drive exploration toward an “interesting” type of diversity, for instance leveraging human guidance. Current approaches in state representation learning rely generally on monolithic architectures which do not enable all these functionalities. Therefore, we present a novel technique to progressively construct a Hierarchy of Observation Latent Models for Exploration Stratification, called HOLMES. This technique couples the use of a dynamic modular model architecture for representation learning with intrinsically-motivated goal exploration processes (IMGEPs). The paper shows results in the domain of automated discovery of diverse self-organized patterns, considering as testbed the experimental framework from Reinke et al. (2019).

1 INTRODUCTION

Maintaining, fine-tuning and expanding the acquired knowledge of a learning agent in a continual way is a central challenge in reinforcement learning. Despite success of recent work in reinforcement learning to master complex tasks, current artificial agents still lack the necessary autonomy and versatility to properly interact with realistic environments (Santucci et al., 2019).

Exploration, or the ability of a learning agent to autonomously discover and reach a diversity of possible states in an unknown environment, is a key ingredient for lifelong machine learning. Inspired from developmental mechanisms observed in humans, “intrinsically-motivated” or “curiosity-driven” exploration (Oudeyer et al., 2007; Baldassarre & Mirolli, 2013) proposes to endow the learning agent with motivational signals to guide the search toward novel states, skills or goals. Such intrinsic rewards aim to generate a curriculum for “intelligently” exploring the environment and accumulate a repertoire of diverse (re-)usable skills (Forestier et al., 2017). Coupled with (goal-conditioned) reinforcement learning policies, intrinsically-motivated algorithms have enabled agents to acquire autonomously diverse skill repertoires that can be re-used to solve efficiently downstream tasks (Pathak et al., 2017; Mohamed & Rezende, 2015; Eysenbach et al., 2019), and to maintain diverse competences in non-stationary environments (Colas et al., 2018). While several works have studied these approaches with agents that perceive their environment at the pixel-level (Bellemare et al., 2016) and self-generate their own goals (Nair et al., 2018; Pong et al., 2019; Reinke et al., 2019), their efficiency relies on the ability to learn low-dimensional state/goal spaces that can adequately represent the different factors of variations of the environment. One key challenge is how to learn representations that will enable efficient exploration in environments where these underlying factors are initially unknown and may change as the agent discovers new areas, new objects, or new ways to interact with the environment.
Representation learning, and more specifically unsupervised feature learning, aims to automatically recover the underlying low-dimensional explanatory factors of complex observations data \cite{Bengio}. Replacing the need for human hand-designed features, they are particularly suited to encode high-dimensional observations into compact latent code and hence define a goal space $G$.

Deep generative models have the additional advantage to generate a distribution of “plausible” latent points from which new unseen goals can easily be sampled. Recent work in goal-directed exploration extensively reuses different variants of such models as variational auto-encoders (VAEs) \cite{VAEs}, generative adversarial networks (GANs) \cite{GANs}, noise-contrastive estimation of mutual information \cite{NCE} and autoregressive methods \cite{Ostrovski}. The representation $R$ is either pretrained before exploration \cite{VAEs}, or learned incrementally \cite{Nair} or from a generative replay model \cite{Caselles-Dupré}. However, they all rely on a monolithic representation model $R$ to recover all the factors of variations, preventing the agent to actively organize the discoveries in different modules and at different levels of granularity. Even though the use of a replay can mitigate the phenomenon of catastrophic forgetting, such architecture generally lacks flexibility to encode new types of information, i.e. to learn diverse representations associated to diverse kinds of structures, and to adapt to the environment increasing complexity.

In this paper, we propose a novel method to give the agent more versatility to augment and structure its world model representation and reuse it for the goal sampling strategy. Following the intuition of Elman \cite{Elman} on the importance of “starting small” both on the task data distribution and on the network memory capacity, we propose to actively grow a hierarchy of embedding networks (deep generative models such as VAEs) as the agent is discovering novel structures in its environment. The agent starts with a small network capacity and can incrementally augment it by freezing an existing module and splitting it into two child modules with their own capacity, preventing by construction the phenomenon of catastrophic forgetting. The tree-structured representation unsupervisedly partitions the observations into distinct branches leading to a hierarchy of specialized goal space representations. Moreover, by encoding observations (and hence goals) at different levels of granularity, the proposed architecture automatically produces an exploration stratification that can target discovery of a “diversity of diversity”. As a proof of concept, we use as test-bed environment a continuous game of life where diverse visual structures can self-organize. We compare the discoveries of IMGEPs equipped with different goal space representations: a fixed-architecture VAE and the proposed adaptive architecture HOLMES. We also implemented as use-case of our architecture an algorithm that leverages the learned structure to guide exploration toward a desired type of diversity. Our contributions are twofold. First, we introduce a dynamic modular model architecture for representing the “diversity of diversity” present in complex environments. This is, to our knowledge, the first work that proposes to progressively grow the capacity of the agent visual world model into an organized hierarchical representations. Secondly, we propose to leverage the structure of the hierarchy to guide the exploration toward a certain type of diversity, opening interesting perspectives for the integration of a human evaluator in the loop.

![Figure 1: Hierarchy of Observation Latent Models for Exploration Stratification (HOLMES).](image.png)
2 Approach

We first explain the architectural approach for learning representations provided that the agent receives an input flow of observations. Then we explain how it can be coupled with intrinsically-motivated goal exploration processes (IMGEP) where the data is collected by the exploring agent.

Hierarchical Observation Latent Models The model architecture takes inspiration from Progressive Neural Networks (PNN) (Rusu et al., 2016), a dynamic model architecture that was proposed for continual learning and applied to a given sequence of reinforcement learning tasks. PNN explicitly prevent catastrophic forgetting by instantiating a new neural network (column) for each new task, and support transfer between tasks by connecting the new column to all the previously trained columns via learned lateral connections. In the following, we explain the different modifications made to adapt PNN to deep generative models in the context of continual state representation learning, which remains an unexplored area (Lesort et al., 2019).

The global sequential architecture is modified into a hierarchical representational architecture $R$. The hierarchy starts with a single root neural network $R_0$. When a saturation signal is triggered, the parameters of $R_0$ are frozen and two child networks $R_{00}$ and $R_{01}$ are instantiated. Input observations are forward first through $R_0$ and are then send to one child network based on a boundary criterion defined in the feature space of $R_0$. Each time a node gets saturated, the split procedure is repeated in that node, resulting in a progressively deeper hierarchy of specialized goal spaces.

We replace the “column” network with a VAE composed of an encoder and a decoder network. To mitigate the growing number of parameters, “lateral connections” are only created between a node and its ancestors and between a reduced number of layers (original, local, global, and embedding levels). The connection scheme is summarized in figure 1. Transfer is beneficial in the decoder network so a child module can reconstruct “as well as” its parent, however connections are removed between encoders as new complementary type of features should be learned. We preserve connections only at the local feature level, as CNN first layers tend to learn similar features (Yosinski et al., 2014). Connections between convolutional layers are defined as convolutions with $1 \times 1$ kernel.

Finally, at the difference of Rusu et al. (2016), the extension into deeper levels of refinement is automatically handled during exploration, removing the need for a predefined sequence of tasks.

Exploration Stratification IMGEPs are goal-oriented exploration processes, operating in a given goal space $G$ which is computed by an an encoding function $R$. We combine HOLMES with IMGEPs by replacing $R$ with the proposed modular hierarchy $\{R_k\}$. Henceforth, IMGEP operates in a hierarchy of goal spaces $\{G_k\}$ and the agent has an additional degree of control in the goal sampling strategy by selecting first a goal space to explore and then a goal in that space. In this paper we considered two setups for the goal space sampling strategy: 1) the target goal space $G_k$ is sampled uniformly over the tree leaves 2) After each split in the hierarchy, we “pause” exploration and assign a fixed probability to each leaf goal space. This second variant is intended to simulate the integration of a human evaluator in the loop that could, by visually browsing the current results made by the agent (see appendix A for a possible visualisation) assign a score to each goal space. During exploration, the agent selects one of the leaf goal spaces with softmax sampling on the assigned probabilities. Then, we follow Reinke et al. (2019) for sampling a goal $g$ in the selected space.

The other way round, the IMGEP influences the training of HOLMES by generating the data distribution and splits in the hierarchy. For evolving HOLMES, we trigger a saturation signal when the population of a goal space go past a threshold of $N_{max}$ points and use the reconstruction performance to create a boundary $B_k$ in the goal space (see appendix B.2). For details on IMGEP and the integration of HOLMES we refer to appendix B.

3 Experimental Results

We use the same experimental testbed as Reinke et al. (2019). The environment is a continuous Game of Life, Lenia (Chan, 2018), where a variety of visual structures can self-organize but still are difficult to discover by manual parameter tuning, making it an interesting testbed for pattern exploration algorithms. We compare IMGEP-VAE equipped with a monolithic high-capacity VAE and IMGEP-HOLMES equipped with the proposed hierarchy of smaller-capacity VAEs and where the goal space selection is done uniformly over the leaf nodes. Additionally, using the classifiers from Reinke et al. (2019) to categorize the patterns of Lenia as “animals” or “non-animals”, we imple-
mented two guided variants where we assume that an external evaluator is interested in discovering a diversity of animals (or non-animals) patterns. Each time a split is triggered, the leaf nodes of the hierarchy get scored with the number of animals (or non-animals) patterns they currently contain, serving as basis for the softmax goal space sampling strategy. The variants are denoted **IMGEP-HOLMES(A)** (guided toward animals) and **IMGEP-HOLMES(NA)** (guided toward non-animals).

**Can HOLMES represent a “diversity of diversity”?** We use Representational Similarity Analysis (RSA), technique coming from systems neuroscience (Kriegeskorte et al., 2008), to compare the different goal spaces representations. Given the set of encoder representations learned with the IMGEP-VAE and IMGEP-HOLMES variants, and an independent set of lenia patterns (750 images), we compute the RSA matrix between all pairs of encoders. We refer to appendix 10 for computation details and the full matrix result. Figure 1 shows the dissimilarity between the goal space learned by IMGEP-VAE versus the modular goal spaces learned by IMGEP-HOLMES. The results indicate a high similarity between the representations learned by the VAE and the root node HOLMES 0 (which can be seen as an early frozen version of the VAE). This suggests that, although the VAE is additionally trained on new unseen patterns, the monolithic representation does not significantly update the type of encoded information/diversity. However, the RSA matrix shows strong dissimilarities between HOLMES different node representations (see figure 10 in appendix), confirming our intuition that HOLMES can better encode a “diversity of diversity” by learning different sets of features per node.

![Figure 2: RSA heatmap showing disagreement (colorscale) among the different goal space representations. Numbers represent the lenia patterns (over 750) shared between each pair of goal spaces.](image)

**Can HOLMES drive exploration toward an “interesting type” of diversity?** Table 1 reports the percentage of identified patterns by the different IMGEP-HOLMES variants. The results show that the IMGEP-HOLMES(A) variant (resp IMGEP-HOLMES(NA)) is finding more animals (resp non-animals) patterns, confirming that HOLMES modular architecture can be exploited to drive exploration toward a desired type of diversity. We refer to appendix A.1 for a qualitative illustration of these results and appendix A.2 for additional statistical analysis on the diversity.

## 4 Conclusion

We presented a hierarchical model architecture for incremental learning of goal space representations, with core functionalities for dealing with open-ended environments. Specifically, it prevents the phenomenon of catastrophic forgetting, can be adaptively augmented to encode new type of information, and self-organize the agent discoveries in hierarchically organized modules. Moreover, by combining the representational architecture with intrinsically-motivated goal exploration, we showed that our approach can target discovery of a “diversity of diversity” and that the exploring agent can exploit the learned structure to efficiently drive exploration. This work opens interesting perspectives to leverage human guidance for exploration in complex systems. Future direction of research should analyze further the capabilities and limits of this architecture and consider experiments that directly integrate a human end-user.

Table 1: Comparison of percentage, across three categories, of discovered patterns for each IMGEP-HOLMES variant. For each algorithm 5 repetitions of the exploration experiment were conducted.

<table>
<thead>
<tr>
<th></th>
<th>animal patterns</th>
<th>non-animal patterns</th>
<th>dead patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IMGEP-HOLMES</strong></td>
<td>15.4 ± 2.4</td>
<td>62.2 ± 2.3</td>
<td>22.4 ± 0.8</td>
</tr>
<tr>
<td><strong>IMGEP-HOLMES(A)</strong></td>
<td><strong>26.5 ± 3.8</strong></td>
<td>45.9 ± 3.7</td>
<td>27.7 ± 1.1</td>
</tr>
<tr>
<td><strong>IMGEP-HOLMES(NA)</strong></td>
<td>4.9 ± 0.4</td>
<td><strong>79.6 ± 3.4</strong></td>
<td>15.5 ± 3.1</td>
</tr>
</tbody>
</table>
REFERENCES


Figure 3, 5 and 7 illustrate the final hierarchical tree that has been incrementally created by the different IMGEP-HOLMES variants. We can see how the discovered patterns are partitioned along the hierarchy. Figure 4, 6 and 8 show additional illustration of patterns (randomly selected) in the final leaves of the tree. These figures illustrate how exploration guidance can drive the type of found diversity. For instance, we can see that IMGEP-HOLMES(A) allocates more goal space nodes for “animal” patterns whereas IMGEP-HOLMES(NA) discovers majority “non-animal” patterns. Additionally, section A.1.4 provides examples of patterns reconstructed by HOLMES representation showing the coarse-to-fine specialisation along the tree.
A.1.1 IMGEP-HOLMES DISCOVERIES

Figure 3: Tree constructed by the IMGEP-HOLMES algorithm during a single exploration with 5000 iterations. We display (randomly selected) discovered pattern that are send to the different nodes of the hierarchy.

Figure 4: More example of discovered patterns in all leaf nodes (except GS 000 which gathers “dead” patterns).
A.1.2 IMGEP-HOLMES(A) DISCOVERIES

Figure 5: Tree constructed by the IMGEP-HOLMES(A) algorithm during a single exploration with 5000 iterations. We display (randomly selected) discovered pattern that are send to the different nodes of the hierarchy.

Figure 6: More example of discovered patterns in several leaf nodes.
A.1.3 IMGEP-HOLMES(NA) DISCOVERIES

Figure 7: Tree constructed by the IMGEP-HOLMES(NA) algorithm during a single exploration with 5000 iterations. We display (randomly selected) discovered patterns that are sent to the different nodes of the hierarchy.

Figure 8: More example of discovered patterns in several leaf nodes.
A.1.4 Coarse-to-fine specialisation

This section provides the reconstruction performances of HOLMES representation, learned during the IMGEP-HOLMES experiment, on an external test dataset of 750 images. The results are summarized in Table 2. Figure 9 provides additional examples of patterns and their reconstructions. We can see that HOLMES progressively learns to reconstruct more and more fine-grained details, which is a good proxy evaluation of HOLMES ability to learn coarse-to-fine representations.

Table 2: Reconstruction error, measured by pixel-wise binary cross entropy loss (BCE), on the test dataset. We provide mean and standard deviation over the different repetitions (n=5).

<table>
<thead>
<tr>
<th>IMGEP-HOLMES root node representation</th>
<th>IMGEP-HOLMES leaf node representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCE 19710 ± 722</td>
<td>BCE 17383 ± 301 (↓ 2327)</td>
</tr>
</tbody>
</table>

![Figure 9: Examples of patterns and their reconstructions along HOLMES tree. Please note that all patterns are originally gray-scale and that, for visualisation purpose, we follow the color scheme of Reinke et al. (2019).](image-url)
A.2 Statistical Results

A.2.1 Representational Similarity Analysis

Figure 10: RSA heatmap showing Spearman’s $\rho$ correlation (color scale) between disagreement among the different goal space representations. The displayed numbers represent the count of Lennia patterns (over the 750 patterns from an external precollected dataset) that are shared between the respective pair of goal spaces and based on which the dissimilarity was computed.

To evaluate the diversity of the representations achieved by the VAE and HOLMES architecture, the representational similarity matrix has been calculated (Figure 10). Both VAE and HOLMES encode a set of pre-selected images (unbiased by exploration) and the achieved representations are compared by using the Spearman’s $\rho$ correlation measure. Since VAE has only one goal space and HOLMES has one per node of the hierarchy, each goal space is mutually compared. Additionally, since HOLMES redirects images through the hierarchy only the images which are in common to both compared goal spaces have been used. The goal spaces which have no images in common are marked with the value 0 in the table.

The dissimilarity index of two compared representations $R_n \in G_n$ and $R_m \in G_m$ is calculated in two stages. First, correlation distance of all the image representation pairs $x = [z_i, z_j]$ is calculated as follows as a dissimilarity measure.

$$corr(x) = 1 - \frac{(z_i - \overline{z}_i) \cdot (z_j - \overline{z}_j)}{||z_i - \overline{z}_i||_2 ||z_j - \overline{z}_j||_2}$$

The $\overline{z}_i$ is a mean value of $z_i$ elements, $\cdot$ is a dot product and $||.||_2$ is the $l_2$ norm. The result of this step is a $N \times N$ sized matrix for each representation $R_n$ and $R_m$, where $N$ is the number of images in common, showing the correlation distance of each image pairs in its goal space. Second phase is calculation of the Spearman’s $\rho$ rank correlation coefficient. The Spearman’s coefficient is a standard statistical method for determining the significance of correlation between two data sets where $\rho \in [0, 1]$. The closer the value of $\rho$ to 1 the higher the correlation in representations. In the figure 10 the $\rho$ coefficient is depicted by using a heatmap colors and the displayed numbers indicate the number of images in common for each representation pair.

A.2.2 Diversity explored by the IMGEP variant in the different goal space representations

Additionally to the percentage of identified patterns presented in table 1 of the main paper, we provide in this section an analysis of the diversity discovered by the different IMGEP variants.
Figure 11: Depicted is the average diversity ($n = 5$) with the standard deviation as shaded area for the different IMGEP variants. The diversity is computed in the goal space learned by the IMGEP-VAE algorithm and in the different goal spaces learned by the IMGEP-HOLMES variant.
During the exploration phase, the different IMGEP variants sample in their goal spaces respectively. Single VAE and HOLMES embedding networks have their own goal spaces and their own representations of the explored images. To evaluate the discovered diversity by each variant, we project each set of exploration images to each goal space and calculate the diversity measure defined by Reinke et al. (2019) in section B.7.1. The diversity measure is defined as the area covered by the representations of a certain image set in the respective goal space. Additionally, the goal spaces are divided into bins to simplify the area calculation and the final diversity measure is calculated as the count of goal space bins which have at least one representation point inside. Each axis of goal space is divided into 4 bins (including the out of range areas).

Figure 11 shows the diversity measure in each goal space for each exploration strategy (IMGEP-VAE, IMGEP-HOLMES(A), IMGEP-HOLMES(NA)) over time (runs of exploration). Each experiment starts with 1000 steps of random exploration after which goal oriented strategy starts.

As shown in Figure 10, VAE goal space and HOLMES 0 root node goal space encode the same type of information, and therefore have similar diversity profiles. For this “type” of diversity, IMGEP-VAE reaches a higher diversity than HOLMES as the goal-sampling strategy operates in that space during the whole course of exploration and therefore manage to cover it better. However, HOLMES nodes encode different type of information and hence diversity, resulting in different diversity profiles. For a better understanding of the different “type” of diversity encoded in the different goal spaces, we refer to figure 3 that illustrate the kind of patterns onto which each goal space representation was trained. In HOLMES 000 goal space, every algorithm reaches the same (null) diversity because this space gathers only dead (all-white) patterns which are all encoded to the same feature point. We can see the impact of guidance in exploration as IMGEP-HOLMES (A) reaches a higher diversity in the goal spaces 00, 001, 0010 and 0011 of IMGEP-HOLMES and these goal spaces were trained mainly on “animals” as depicted in figure 3 and figure 4. Similarly IMGEP-HOLMES (NA) reaches a higher diversity in the goal spaces 011, 0111, 01110 and 01111 of IMGEP-HOLMES and these goal spaces were trained mainly on “non-animals”.

B IMPLEMENTATION DETAILS

This appendix complements section 2 of the main paper. First we detail the framework of intrinsically-motivated goal exploration processes (IMGEPs). Then we detail the integration of HOLMES into the IMGEP process.

B.1 INTRINSICALLY MOTIVATED GOAL EXPLORATION PROCESSES (IMGEP)

This section retakes the IMGEP formalization of Reinke et al. (2019), we refer to this paper for additional details. An IMGEP is an algorithmic process which automatically generates a sequence of goals in order to explore the parameters of an unknown complex system. It aims to maximize the diversity of observations from that system within a budget of \( N \) experiments. IMGEPs are equipped with a memory of past experimental parameters and observations, denoted as the history \( H \), which is used to guide the exploration process.

The explored system is characterized with three components. A parameter space \( \Theta \) corresponding to the system parameters \( \theta \) that are under the agent control. An observation space \( O \) where an observation \( o \) is a vector representing all the signals captured from the system, in our case raw sensory images of the discovered patterns. The (unknown) environment dynamics \( D: \Theta \rightarrow O \) mapping parameters to observations.

To explore a system, an IMGEP uses a goal space \( G \) computed by an encoding function \( \hat{g} = R(o) \), where the goal sampling strategy is implemented.

The exploration process iterates through \( N \) exploration runs with the following strategy. First, sample a goal \( g \) from a goal sampling distribution \( G \) defined in \( G \) and based on the history of reached points \( \mathcal{H} \). Then, infer corresponding parameter \( \theta \) using a parameter sampling policy \( \Pi = \Pr(\theta; g, \mathcal{H}) \) and roll-out an experiment with \( \theta \). Observe the outcome \( o \) and compute the corresponding encoding \( R(o) \). Store the experimental parameters, observation and reached goal \( (\theta, o, R(o)) \) in history \( \mathcal{H} \).
Because the parameter sampling policy $\Pi$ and the goal sampling distribution $G$ generally take into account previous explorations runs, the history is first populated through exploring $N_{\text{init}}$ randomly sampled parameters after which the intrinsically motivated goal exploration process starts.

### B.2 IMGEP-HOLMES

IMGEP-HOLMES replaces the representation $R$ with the proposed hierarchy of deep generative models $\{R_k\}$ to encode the observations and goals at different levels along the hierarchy.

Because this new representation creates a hierarchy of modular goal spaces, the goal-sampling strategy is divided in two steps: 1) sample a target goal space $G_k$ according to a goal space sampling distribution $G_{\text{space}}(H)$, 2) sample a target goal $g$ in this space according to a goal sampling distribution $G_k(H)$.

During exploration, when a certain goal space $G_k$ gets saturated, it is split into two new goal spaces $G_{k,\text{left}}$ and $G_{k,\text{right}}$. Each goal space inherits from a part of the population of $G_k$. Two child module representations $R_{k,\text{left}}$ and $R_{k,\text{right}}$ are instantiated with new neural architecture and randomly initialized.

A pseudo-code for IMGEP-HOLMES implementation is given in algorithm 1.

---

### Algorithm 1: IMGEP-HOLMES

Initialize the goal space representation $R = R_0$ with random weights

```plaintext
for $i \leftarrow 1$ to $N$ do
    if $i < N_{\text{init}}$ then // Initial random iterations to populate $H$
        Sample $\theta \sim U(\Theta)$
    else // Intrinsically motivated iterations
        Sample a target goal space $G_k \sim G_{\text{space}}(H)$ in the hierarchy
        Sample a goal $g \sim G_k(H)$ in $G_k$
        Choose $\theta \sim \Pi(G_k, g, H)$
        Perform an experiment with $\theta$ and observe $o$
        $R_k \leftarrow R_0$
        while $R_k$ not a leaf module do // Encode reached goals in the hierarchy
            $\hat{g} = R_k(o, \{R_{\text{ancestors}}(k)(o)\})$
            Append $(\theta, o, \hat{g})$ to the history $H[k]$
            $R_k \leftarrow \text{child}(R_k|\hat{g})$
        if a goal space $G_k$ is saturated then // Augment representational capacity
            Freeze $R_k$ weights
            Define a boundary $B_k$ splitting $G_k$ in two subspaces $G_{k,\text{left}}$ and $G_{k,\text{right}}$
            Instantiate two child modules $R_{k,\text{left}}$ and $R_{k,\text{right}}$, for $(\theta, o, \hat{g}) \in H[k]$ do
                if $\hat{g}$ is on the left side of $B_k$ then
                    Append $(\theta, o, R_{k,\text{left}}(o))$ to the history $H[k,\text{left}]$
                else
                    Append $(\theta, o, R_{k,\text{right}}(o))$ to the history $H[k,\text{right}]$
            if $i \mod T == 0$ then // Periodically train the network
                for $E$ epochs do
                    Train the hierarchy $R$ on observations in $H$ with importance sampling
            for $k \in H$ do // Update the database of reached goals
                for $(\theta, o, \hat{g}) \in H[k]$ do
                    $H[k][\hat{g}] \leftarrow R_k(o)$
```

---

In this paper, the following design choices were made to decide when and how to split a node in the hierarchy. When the population of a goal space go past a threshold $N_{\text{max}}$, we trigger a split in that space. Other approaches could be considered as trigger signal such as a drop in the reconstruction.
loss (Caselles-Dupré et al. 2019), a low increase of diversity progress, etc. We use the reconstruction
performance to separate the population in the selected goal space in two: the median reconstruction
error serves as threshold to classify the population as “badly” versus “well” reconstructed and a
Support Vector Machine (SVM) classifier is then fitted generating boundary $B_k$ in the goal space.
From that boundary, the frozen node redirects incoming data flow to a certain child module.

C EXPERIMENTAL SETTINGS

In this section we detail the experimental settings and hyperparameters.

We refer to the appendix of Reinke et al. (2019) for Lenia settings (section B.1) and sampling
mechanisms for Lenias initial state via CPPN and dynamic parameters (section B.4). The same
hyperparameters were used in this paper.

Table 3 reports the VAE neural network architecture for the IMGEP-V AE representation and table
4 reports the neural architecture of the core module for the IMGEP-HOLMES variant. We give
hyperparameters were used in this paper.

The networks are trained 400 epochs every 400 runs of exploration, and initialized with kaiming
uniform initialization. For HOLMES child modules, the first convolutional layers are initialized
with the values of the trained parent module. We used the Adam optimizer ($lr = 1e-3$, $\beta_1 = 0.9$,
$\beta_2 = 0.999$, $\epsilon = 1e-8$, weight decay=1e−5) with a batch size of 128.

Table 3: VAE architecture used in the IMGEP-V AE variant.

<table>
<thead>
<tr>
<th>Encoder</th>
<th>Decoder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input pattern A: 256 × 256 × 1</td>
<td>Input latent vector $z$: 16 × 1</td>
</tr>
<tr>
<td>Conv layer: 32 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
<td>FC layers : 256 + ReLU, 16 × 16 × 32 + ReLU</td>
</tr>
<tr>
<td>Conv layer: 32 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
<td>TransposeConv layer: 32 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
</tr>
<tr>
<td>Conv layer: 32 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
<td>TransposeConv layer: 32 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
</tr>
<tr>
<td>Conv layer: 32 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
<td>TransposeConv layer: 32 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
</tr>
<tr>
<td>Conv layer: 32 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
<td>TransposeConv layer: 32 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
</tr>
<tr>
<td>FC layers : 256 + ReLU, 256 + ReLU, FC: 2 × 16</td>
<td>TransposeConv layer: 32 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
</tr>
</tbody>
</table>

Table 4: Basis VAE architecture used in the IMGEP-HOLMES variant.

<table>
<thead>
<tr>
<th>Encoder</th>
<th>Decoder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input pattern A: 256 × 256 × 1</td>
<td>Input latent vector $z$: 16 × 1</td>
</tr>
<tr>
<td>Conv layer: 8 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
<td>FC layers : 64 + ReLU, 16 × 16 × 8 + ReLU</td>
</tr>
<tr>
<td>Conv layer: 8 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
<td>TransposeConv layer: 8 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
</tr>
<tr>
<td>Conv layer: 8 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
<td>TransposeConv layer: 8 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
</tr>
<tr>
<td>Conv layer: 8 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
<td>TransposeConv layer: 8 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
</tr>
<tr>
<td>Conv layer: 8 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
<td>TransposeConv layer: 8 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
</tr>
<tr>
<td>Conv layer: 8 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
<td>TransposeConv layer: 8 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
</tr>
<tr>
<td>Conv layer: 8 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
<td>TransposeConv layer: 32 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
</tr>
<tr>
<td>FC layers : 64 + ReLU, 64 + ReLU, FC: 2 × 16</td>
<td>TransposeConv layer: 32 kernels 4 × 4, stride 2, 1-paddin + ReLU</td>
</tr>
</tbody>
</table>