Hierarchical Neural Cellular Automata

Ritu Pande, Daniele Grattarola

Independent Researchers
ritz.pande@gmail.com

Abstract

As opposed to the traditional view wherein intelligence was believed to be a result of centralised complex monolithic rules, it is now believed that the phenomenon is multi-scale, modular and emergent (self-organising) in nature. At each scale, the constituents of an intelligent system are cognitive units driven towards a specific goal, in a specific problem space—physical, molecular, metabolic, morphological, etc. Recently, Neural Cellular Automata (NCA) have proven to be effective in simulating many evolutionary tasks, in morphological space, as self-organising dynamical systems. They are however limited in their capacity to emulate complex phenomena seen in nature such as *cell differentiation* (change in cell's phenotypical and functional characteristics), *metamorphosis* (transformation to a new morphology after evolving to another) and *apoptosis* (programmed cell death). Inspired by the idea of multi-scale emergence of intelligence, we present ***Hierarchical NCA***, a self-organising model that allows for phased, feedback-based, complex emergent behaviour. We show that by modelling emergent behaviour at two different scales in a modular hierarchy with dedicated goals, we can effectively simulate many complex evolutionary morphological tasks. Finally, we discuss the broader impact and application of this concept in areas outside biological process modelling.

Introduction

Intelligence can be thought of as a capacity to achieve a fixed goal by variable means (Fields and Levin, 2022). In other words, an intelligent system works towards a defined goal and, even on being disturbed from its path towards it, is able to use novel means, not explored hitherto, to reasonably achieve its objective. Intelligent agents typically exhibit collective behaviour whose mechanics are local in nature, *i.e.*, involve interactions between individual agents physically close to one another occur over short timescales. The emergent property arising out of interaction amongst multiple intelligent agents, collaborating and competing with each other, asynchronously, across large time scales, is termed *Collective intelligence*. Collective intelligence, by this definition, can be seen as a multi-scaled self-organising phenomenon, across multiple problem spaces, resulting in evolution from *physics to the mind*.[1] Laws of physics drive atomic and sub-atomic interactions in physical space. In molecular space, gene regulatory networks govern the formation of proteins. Proteins come together to form cells which interact in morphological space to form organs. Organs collectively form an organism which, sometimes, possesses the capacity for meta-cognition, *i.e.*, awareness of its cognitive capabilities. An intelligent system at each scale is made up of a collective of cognitive units, in a specific problem space with a distinct goal.

Cellular Automata have for long been used as computational models to represent the behaviour of dynamical systems that evolve by asynchronous local interactions between constituents of the system at discrete time intervals. There have been attempts to use them to model dynamical systems exhibiting intelligent behaviour at varied scales in nature. For instance, Vichniac (1984) used them to model physical systems, De Sales et al. (1997) to represent the gene regulatory networks and Li et al. (2013) to emulate the evolutionary dynamics of social networks. But how do these intelligent systems at different scales interact and how do their independent and potentially conflicting goals impact each others' dynamics? In this paper we take a hierarchical view of the evolutionary dynamics of intelligent self-organising systems and ask the following questions:

1. Can an artificially intelligent system operating at a higher scale in the hierarchy influence another system, at a lower scale, and guide it towards its own goals?

2. Can an artificially intelligent system lower in the hierarchical scale provide feedback to the system at a higher scale attempting to influence it?

3. How do the goals of a collective reconcile with the goals of its constituents?

To get insight into answers to these questions, we introduce *Hierarchical NCA*, a modelling architecture that allows modelling self-organising systems operating at different scales with the capability of bi-directional communication. In this paper, we focus our modelling efforts on two self-organising systems operating at different scales in morphological space. However, it should be noted that the architecture and the

[1] Concept popularised by Prof. Michael Levin.

concept itself are generic and can be applied across multiple scales and problem spaces.

Preliminaries and Related Work

Cellular Automata: Cellular automata are spatially and temporally discrete computational units used to model non-linear dynamical systems in various scientific disciplines. They are composed of distinct homogeneous computational units, called *cells*, arranged in a grid. The cells evolve in parallel at discrete time steps based on fixed transition rules. The transition rules are local, *i.e.*, they are a function of states of a cell's immediate neighbourhood (Von Neumann, 1963) and perform no action at a distance. The sequential execution of these simple local rules has been shown to result in the emergence of complex global patterns. Cellular automata were proposed as a possible model for biological systems (Wolfram et al., 2002). However, it is not easy to reverse-engineer local rules for desired complex global emergent patterns, especially biological ones.

Neural Cellular Automata: There have been many attempts to learn the local transition rules using neural networks. Most recently, transition rules of cellular automata have been shown to be learnable using convolutional neural networks (Gilpin, 2019). Also, recurrent convolutional neural networks have been used as lightweight networks to learn transition rules of cellular automata from a global objective function using backpropagation through time (BPTT). Such models are termed as *Neural Cellular Automata* (NCA) and have been shown to be effective in modelling biological phenomena such as morphogenesis using 2-D images (Mordvintsev et al., 2020). Each cell is represented by the RGB channels of the corresponding pixel in the image, representing the cell phenotype, and a set of latent channels representing its hidden states. The CA is trained on a specific target image using mean squared error (MSE) as the objective function. There have been many other interesting applications of NCA in 3-D morphogenesis and evolution, (Sudhakaran et al., 2021; Nichele et al., 2017; Pontes-Filho et al., 2022), regeneration (Horibe et al., 2021), classification (Randazzo et al., 2020) and control (Nadizar et al., 2022; Variengien et al., 2021).

Self-Organising Textures: NCAs have also been shown to be effective in the task of texture synthesis using 2-D images (Niklasson et al., 2021). Again, each cell is described by the RGB channels of the corresponding pixel representing its phenotype and latent channels representing its hidden states. The NCA is trained by calculating the distance between the Gram matrix of the predicted and target image representations derived from a pre-trained VGG-16 model.

Adversarial Reprogramming of NCA: Many complex biological processes rely on the ability of external agents to modify their behaviour, *e.g.*, attack by a bacteria disturbing the stability of a living system and then action by antibodies to restore homeostasis. Randazzo et al. (2021) evaluated multiple methods to perturb the behaviour of an NCA, either by perturbing selective cells via a parallel NCA or by global perturbation of all the cells via a vector derived from a trainable symmetric perturbation matrix.

Goal-Guided NCA: Goal-Guided NCA (Sudhakaran et al., 2022) focused on external perturbation of NCA using goal-encoding generated by a neural network. The goal encoding is used as a global vector perturbing all the cells of NCA at every step of its evolution to continually change/control its behaviour as desired.

However, none of the above approaches are capable of modelling a system that changes its behaviour in programmed phases, wherein the trigger for change comes from within the system itself. Modelling many biological processes, such as, cell differentiation and metamorphosis require the model to exhibit these properties. Our work seeks to fill this gap in the field of modelling self-organising systems by taking a hierarchical view of their evolution, and in the process, answer important questions about the nature of emergence of collective intelligence.

While our work also focuses on modification of NCA behaviour, like Randazzo et al. (2021) and Sudhakaran et al. (2022), it differs from the existing approaches in that:

1. It takes a hierarchical view of self-organising systems, in which cells at a higher level in the hierarchy control the cells at a lower level.

2. Each NCA in the hierarchy is trained towards a distinctive goal.

3. The trigger to change NCA behaviour is feedback-based and governed from within the system itself. Only once the NCA evolves to a defined stable state it sends feedback to the NCA next up in the hierarchy, which in turn signals its modification.

Methodology: Hierarchical NCA

In order to study the behaviour and capabilities of self-organising systems operating at different scales we propose *Hierarchical NCA* (H-NCA), an architectural framework that arranges self-organising systems hierarchically, each operating at a different scale evolving towards their own goal. The basic computational units of the H-NCA model are NCA whose design is the same as that proposed by Niklasson et al. (2021), with minor modifications (refer to Figure 2). Each cell of the NCA higher in the hierarchy controls a subset of cells, called *cell clusters*, of the NCA immediately below it in the hierarchy. It is also capable of receiving feedback on the current state of the cell cluster it controls.

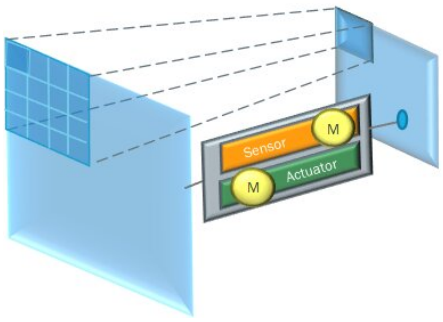


Figure 1: H-NCA architecture. Each cell of **parent-NCA** (right) controls a cluster of **child-NCA** (left) cells. **Sensor** is used to detect the state of the child-NCA clusters by the parent-NCA. **Actuator** is used by the parent-NCA to influence the child-NCA via its signal channels. Both the actuator and the sensor contain a **multiplexer** responsible for mixing signals from the source NCA to the destination NCA.

In this paper, we work with only two scales in the hierarchy, but the architecture is extensible, given enough computational resources. The NCA lowest in the hierarchy is referred to as the *child-NCA* and the one above it, the *parent-NCA*. The child-NCA and the parent-NCA communicate with each other via a *communication layer*, which contains an *actuator* and a *sensor*. Each cell in the parent-NCA uses the sensor to obtain the current state of the child-NCA cell cluster it controls and then uses the actuator to signal modification of the cell cluster in service of its own objective. Both the sensor and the actuator use a *multiplexer* for mixing the information in the signal with its destination cell state.

Model Composition

This section provides the details of each constituent component of the H-NCA model.

Child-NCA: The child-NCA, represented as f_{θ_c}, models a self-organising system operating at the lowest scale in the hierarchy of emergent systems being modelled by the H-NCA. It contains learnable parameters θ_c, that can be trained to learn transition rules such that:

$$x_c^{t+1} = f_{\theta_c}(x_c^t) \tag{1}$$

where $x_c^t \in \mathbb{R}^{128\times128\times12}$ is the state of the child-NCA, at time step t.

There are a total of 128×128 cells in the child-NCA, with each cell containing 12 channels, of which the first 3 channels are responsible for the cell phenotype and the remaining 9 are responsible for processing signals intended for the cell. The first layer of the model concatenates the input *feature channels*, x_f^t and the *signal channels* x_s^t:

$$x_c^t = x_f^t \parallel x_s^t \tag{2}$$

where $x_f^t \in \mathbb{R}^{128\times128\times3}$, $x_s^t \in \mathbb{R}^{128\times128\times9}$ and $x_c^t \in \mathbb{R}^{128\times128\times12}$.

Next, circular padding is applied to the concatenated input to maintain the size of the automaton once it goes through the *Perception* layer.

The Perception layer is responsible for obtaining a feature representation of the cell's current state and that of its immediate neighbourhood. It does so by performing depthwise-convolution on the input with 4, 3×3 (non-trainable) filters, namely $sobel_x$, $sobel_y$, *Laplacian* and *Cell Identity* filters. Please refer Niklasson et al. (2021) for further details. This layer outputs a 48-channel perception vector which has information about the cell and its immediate neighbourhood. The is followed by a 1×1 convolution hidden layer with 64 trainable filters and the final 1×1 convolution output layer with 12 trainable filters.

The update to the cell state by the child-NCA is stochastic in nature, *i.e.*, at every time step t, the state of any cell is updated with a probability $\delta_c \in [0, 1]$. This accounts for the fact that there is no global clock in nature that synchronises cell updates in self-organising systems (Mordvintsev et al., 2020).

Parent-NCA: The parent-NCA, represented as f_{θ_p}, with learnable parameters θ_p, models the self-organising system operating above the child-NCA in the H-NCA emergent hierarchy. The design of parent-NCA is the same as that of child-NCA with the following exceptions:

1. There are a total of 32×32 cells in the parent-NCA. This is because we use a *signalling factor*, $s_f = 4$, *i.e.*, each cell of parent-NCA controls 4×4 cells of child-NCA.
2. There is no demarcation between cell feature and signal channels, since the mechanism of sensing child-NCA cell cluster states and constructing signals to modify their behaviour by parent-NCA is the same for both phenotype and signal channels of the child-NCA.
3. The Perception layer is trainable and only initialised with $sobel_x$, $sobel_y$, *Laplacian* and *cell identity* filters. This design choice is empirically driven based on the results obtained while training parent-NCA.
4. The hidden layer has 256, 1×1 convolution filters. Again, this design choice is empirically driven based on the results obtained while training parent-NCA.

Communication Layer : Communication layer acts as a conduit between the child-NCA and the parent-NCA. It is composed of a sensor, for detecting the current state of the child-NCA cell clusters by the parent-NCA, and an actuator, for sending signals from parent-NCA to child-NCA to modify its behaviour. Both the sensor and the actuator contain an independent instance of a multiplexer, responsible for mixing the signals to their destination cell states. We describe details of the constituent elements of the communication layer in this section.

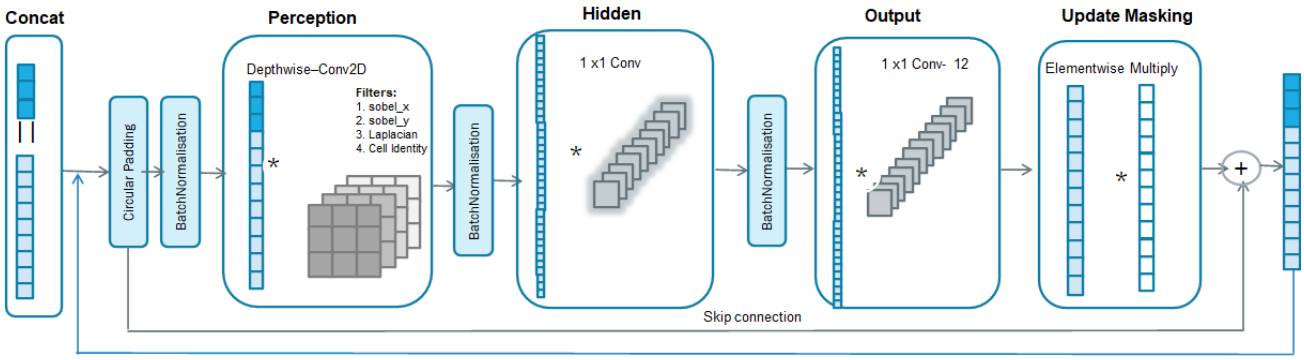


Figure 2: The NCA framework used in the H-NCA model is the same as proposed by Niklasson et al. (2021) with a minor modification to perform batch normalisation after every layer to speed up training. The same framework is used by both child-NCA and parent-NCA with the following differences: 1. The child-NCA perception layer is non-trainable, while the parent-NCA perception layer is trainable; 2. The number of 1×1 filters in the child-NCA hidden layer are 64 while in the parent-NCA they are 256.

1. *Sensor*: Sensor, represented as f_s, is used by parent-NCA to detect the current state of child-NCA cell clusters. It takes as input the signalling factor s_f, which determines the number of cells in the child-NCA cluster that are controlled by a single parent-NCA cell, *i.e.*, the cells in child-NCA are divided into clusters of size $s_f \times s_f$. The sensor averages the state of the cells in each of the child-NCA cell clusters to construct a feedback signal for parent-NCA, *i.e.*, if x_{c_i} is defined as the state of any cell i of the child-NCA, and N_j the set of $s_f \times s_f$ child-NCA cells that are controlled by the parent-NCA cell j, then:

$$b_j = \frac{1}{|N_j|} \sum_{i}^{|N_j|} x_{c_i} \tag{3}$$

where $b_j \in \mathbb{R}^{1\times1\times12}$

In this paper, we have used the value $s_f = 4$.

Next, the feedback signal b_j is multiplexed to the current parent-NCA cell states via the multiplexer.

2. *Actuator*: The actuator, represented as f_{θ_a}, with learnable parameters θ_a, is responsible for signalling child-NCA cell clusters to modify their behaviour based on the parent-NCA current state. It takes as input the signalling factor, s_f and up-scales each parent-NCA cell by $s_f \times s_f$, with all its channels, using *nearest neighbour* interpolation.

   Next, the up-scaled signal goes through a single 1×1 convolution layer, called the *signal-creator*, with the number of filters equalling the number of signalling channels in child-NCA. In our case, this value is 9. The signal thus created is sent to the multiplexer to be delivered to the child-NCA. This ensures that the parent-NCA can only influence the child-NCA via its signalling channels.

3. *Multiplexer*: Both the actuator and the sensor contain an instance of the multiplexer to integrate the signal from its source to the destination. The multiplexer takes as the input, the source signal, the destination NCA cell values and whether the destination NCA distinguishes between the feature and signal channels. If the destination NCA distinguishes between feature and signal channels, the multiplexer ensures that the signal is only integrated into the signal channels. This enables the multiplexer in the actuator to only send signals to the child-NCA signal channels while in the sensor to provide the feedback on the state of child-NCA cell clusters to all the parent-NCA channels. The integration, in this paper, is performed by adding the values of source channels to the corresponding destination channel values. However, the model framework allows for designs wherein the multiplexer can be configured to use more complex integration mechanisms such as cross-attention, for other use-cases and NCA goals.

Hierarchical NCA Model Evolution

This section explains the interaction between various constituents of the H-NCA model and how they come together to model hierarchical emergent behaviour.

1. The first step in H-NCA evolution performs the following operations:

   (a) The child-NCA evolves for a pre-defined number of steps, τ_c before sensor can detect its cell clusters. In this paper, we have chosen $\tau_c = 100$:

   $$x_c^{t+1} = f_{\theta_c}(x_c^t) \tag{4}$$

   where $t = 1, 2, ..., \tau_c$.

   (b) In the first step of H-NCA evolution, the parent-NCA has not yet evolved. Hence, the feedback signal provided by the sensor is used as the initial state of the parent-NCA to perform one step in its evolution:

   $$x_p^{\tau_c} = f_s(x_c^{\tau_c}); \quad x_p^{\tau_c+1} = f_{\theta_p}(x_p^{\tau_c}) \tag{5}$$

2. Each subsequent H-NCA step performs the following operations in this order:

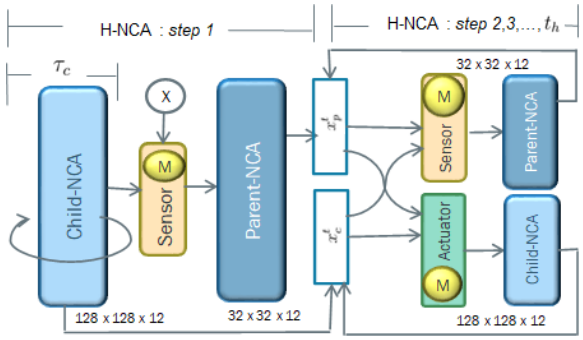


Figure 3: H-NCA model evolution. The child-NCA evolves for τ_c time steps after which the sensor averages the values of the cell states of child-NCA cell clusters. They become the initial state of the parent-NCA. Subsequently, at each time step, both the child-NCA and the parent-NCA consume signals via the actuator and the sensor and perform one step in their evolution, respectively.

(a) The parent-NCA and the child-NCA simultaneously consume the signals from the communication layer. The parent-NCA detects the state of child-NCA cell clusters via the sensor while the child-NCA gets the signals from the actuator:

$$x_c^{'t} = f_{\theta_a}(x_p^t, x_c^t) \tag{6}$$

$$x_p^{'t} = f_s(x_c^t, x_p^t). \tag{7}$$

(b) The the child-NCA and the parent-NCA both perform one step in their evolution:

$$x_c^{t+1} = f_{\theta_c}(x_c^{'t}); \;\; x_p^{t+1} = f_{\theta_p}(x_p^{'t}). \tag{8}$$

This evolution mechanism ensures that parent-NCA takes control over the child-NCA only after it has evolved to a state wherein it is ready to process signals. This is akin to many biological processes such as cell differentiation and metamorphosis wherein the change in the target morphology is triggered in a phased manner based on the existing state of the organism. Refer to Figure 3 summarising the H-NCA evolution scheme.

Model Training Procedure

Training the H-NCA model is a two-step process. As a first step, the child-NCA in the H-NCA model is pre-trained. Subsequently, only the parent-NCA and the communication layer in the H-NCA model are trained to modify the child-NCA output to the parent-NCA target, with only the signalling channels of the child-NCA available to the parent-NCA for influence.

Pre-Training child-NCA: During pre-training, child-NCA does not distinguish between feature channels and signal channels. The target for the child-NCA is an RGB image of size 128×128 pixels. The training procedure starts by initialising a pool $\mathcal{P}$ of size $P_n \times 128 \times 128 \times 12$, where P_n is the number of elements in the pool, with a uniform random seed, $s_{ur} \in \mathbb{R}^{128\times\ 128\times 12}$, representing the initial state of the child-NCA.

1. The training starts by randomly sampling a batch s_b from $\mathcal{P}$ and replacing the first entry of the batch with s_{ur}.

2. The child-NCA is evolved for $t_c \in [t_c^{min}, t_c^{max}]$ steps

$$x_c^{t+1} = f_{\theta_c}(x_c^t) \tag{9}$$

where $t = 1, 2, 3, ..., t_c$ and $x_c^0 = s_b$.

3. The output of the child-NCA at step t_c is then added to the pool $\mathcal{P}$. Hence, $\mathcal{P}$ acts as a replay buffer stabilising the child-NCA evolution beyond the number of steps it has been trained for (Mordvintsev et al., 2020).

4. The feature channels (RGB) of $x_c^{t_c}$ are extracted to get the prediction by child-NCA, $\hat{x}_c$ to compare against the child-NCA target img_c

$$\hat{x}_c = \phi_{rgb}(x_c^{t_c}) \tag{10}$$

where ϕ_{rgb} is a function that extracts the RGB channels from the image.

5. Next, image representations of img_c and $\hat{x_c}$ are extracted from layers 1,4,7,11 and 15 of pre-trained VGG-16 model (Simonyan and Zisserman, 2014), represented as img_r and $\hat{x}_r$, respectively. These representations are then projected on 32 random vectors $V \in \mathbb{R}^{3\times 32}$ resulting in $\hat{x}_w$ and img_w respectively.

$$\hat{x}_w = \text{Proj}_V(\hat{x}_r); \;\; x_w = \text{Proj}_V(img_r) \tag{11}$$

where Proj_V denotes projection on vectors V, $x_r \in \mathbb{R}^{5\times n_i\times 3}$ and $\hat{x_r} \in \mathbb{R}^{5\times n_i\times 3}, i = 1, ..., 5$, n_i is the number of features in the respective VGG16 layer.

6. Sliced Wasserstein loss (Heitz et al., 2021) between the child-NCA target and child-NCA prediction is then calculated as:

$$L_{SW}(img_w, \hat{x}_w) = \frac{1}{N}||\text{sort}(img_w) - \text{sort}(\hat{x}_w)||^2 \tag{12}$$

where $N = |x_w|$ and *sort* arranges its inputs in ascending order.

8. The child-NCA learnable parameters, θ_c, are finally updated via BPTT using L_{SW}.

7. The above steps are repeated for each training epoch

Parent-NCA & Communication Layer Training: While training parent-NCA and the communication layer of the H-NCA model, the child-NCA parameters θ_c^* learnt during pre-training are not changed. The child-NCA distinguishes between the feature and the signal channels allowing the parent-NCA to influence it via its signalling channels to

modify its output to service its own objective. The target for the parent-NCA is an RGB image of size 128×128 pixels. The training procedure starts by initialising a pool $\mathcal{P}$ of size $P_n \times 128 \times 128 \times 12$, where P_n is the number of elements in the pool, with the same uniform random seed, s_{ur} used to pre-train the child-NCA. The remaining procedure is the same as that for training child-NCA with the following exceptions:

1. H-NCA is evolved for $t_h \in [t_h^{min}, t_h^{max}]$. Refer to *Hierarchical NCA Model Evolution* section for details.
2. The loss function between the parent-NCA target img_h and the prediction of child-NCA $\hat{x}_h$ is either Sliced Wasserstein loss, L_{WS}, or mean squared error, L_{MSE}, depending on the nature of the experiment.
3. The learnable parameters in the actuator, θ_a, and the parent-NCA parameters, θ_p, are updated via BPTT using L_{WS} or L_{MSE}, as the case may be. There are no learnable parameters in the sensor.

Experiments

As discussed, the main objective of the H-NCA model is to:

- Model phased morphological changes in one self-organising system triggered by another operating at a higher scale.
- Prompt the change in the morphology from within the system based on its current state.
- Study the impact of the objective of the collective on its constituents.

To prove that the H-NCA models can satisfy these objectives we designed two experiments, namely:

1. *Cell Differentiation*: This experiment is designed to study whether morphological characteristics of structures created by a self-organising system can be altered by another self-organising system operating at a larger scale. The child-NCA is designed to create basic structural units/patterns emulating the formation of biological cells from different cell organelles in living organisms. This is achieved by using a 128×128 sized RGB image with a random distribution of structural patterns of the desired shape as the child-NCA target. The parent-NCA target is, again, a random distribution of patterns, but different in shape from those of the child-NCA. The child-NCA, the parent-NCA, and the actuator are trained using sliced Wasserstein loss. If our hypothesis is correct, once the structural patterns generated by the child-NCA are evolved to a stage where they can be detected by the H-NCA sensor, their shape should be altered to that of the parent-NCA target. In this way, we can prove that the trigger for modification of structural characteristics of the output of a self-organising system can be controlled from within itself.
2. *Metamorphosis*: This experiment is designed to study whether the structures created by a self-organising system can be organised in a different target morphology by another self-organising system operating at a larger scale. The child-NCA is trained using sliced Wasserstein loss, while the parent-NCA and the actuator are trained using mean squared error (MSE). If our hypothesis is correct, once the structural patterns generated by the child-NCA are evolved to a stage where they can be detected by the H-NCA sensor, they should be arranged as per the target morphology of the parent-NCA.

Refer to Figure 4 for details of the targets used in each experiment and the source code to reproduce the results.[2]

Training Data

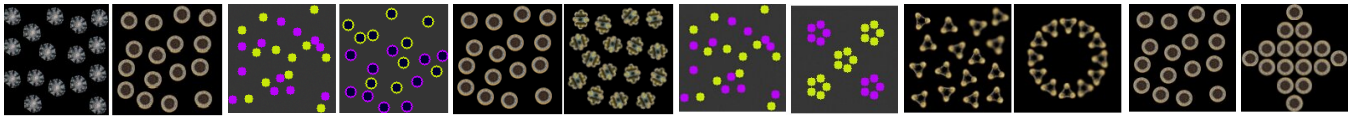
Target images for the child-NCA and the parent-NCA for both experiments are generated via two mechanisms, namely via a script to programmatically generate RGB images of size 128×128 with custom patterns and by manually arranging images of photosynthesising micro-algae called *diatoms* to create more life-like patterns.

Training

The training of the H-NCA model in the experiments follows the procedure detailed in the section *Model Training Procedure*. In this section, we provide the details of the regularisation techniques, parameters and the hyper-parameters used during the training and the rationale behind using them. The training parameters and hyper-parameters are summarised in Table 1 and the loss curves captured during training are available in Figure 5

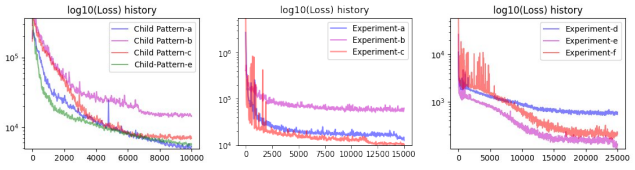
- *Replay buffer*: The size of the replay buffer to store the child-NCA output is set to $P_n = 4096$. This value served well to stabilise the H-NCA evolution beyond the training steps.
- *Batch size*: Batch size represents the number of child-NCA states sampled from the replay buffer during each training epoch. The value used for both the child-NCA pre-training and the H-NCA training is 4, and is driven by the GPU resource limit in Google Colab Pro.
- $t_c^{min}, t_c^{max}, t_h^{min}, t_h^{max}$: The values for t_c^{min}, t_c^{max} during child-NCA pre-training are chosen as 32 and 96, as proposed by Niklasson et al. (2021). The choice of values for t_h^{min}, t_h^{max} as 20 and 60, during H-NCA training, is driven by the GPU resource limit in Google Colab Pro. The values used allow for the training to converge without exceeding the available GPU memory.
- *Learning rate:* The training starts with the learning rate of 10^{-3} as proposed by Niklasson et al. (2021). We attempted

[2] https://github.com/RituPande/hnca



(a) (b) (c) (d) (e) (f)

Figure 4: Experiments. (a), (b) and (c) are pair of child-NCA (left) and parent-NCA (right) targets used in cell differentiation experiments. (d), (e) and (f) are child-NCA (left) and parent-NCA (right) targets used in metamorphosis experiments.



(a) Child-NCA Pre-training (L_{SW}) **(b)** Cell Differentiation Training (L_{SW}) **(c)** Metamorphosis Training (L_{MSE})

Figure 5: Loss curves for experiments training H-NCA models. The losses are pre-processed before plotting to smoothen them. Legend labels refer to Figure 4.

to increase the rate to 10^{-2}, but it destabilised the training after a few epochs.

- *Cell update rate:* The probability that a cell in the child-NCA or the parent-NCA is updated at any time step is governed by δ_c and δ_p, respectively. We have used the value of both to be 0.5 during both the child-NCA pre-training and H-NCA training. Increasing their values beyond 0.5 destabilised the training and decreasing their values below it made the convergence very slow.

- *Optimiser:* The training used *Adam:* optimiser as proposed by (Niklasson et al., 2021)

- *Epochs:* In our experiments we found that pre-training child-NCA for 10000 epochs achieved the best results. The training required 15000 epochs for the cell differentiation experiments and 25000 epochs for the metamorphosis experiments, in order to achieve stable results.

- *Learning rate patience and early stopping patience:* Learning rate patience is the number of epochs of training with no improvement in loss after which the learning rate was decreased by a factor of 10^{-1}. Early stopping patience is the total number of epochs of training without improvement in loss after which the training was terminated. For child-NCA pre-training, learning rate patience and early stopping patience were 1000 and 1500 respectively, and for H-NCA training they were 3000 and 4000, respectively.

- *Gradient clipping:* We perform gradient clipping during training to regularise the weight updates during backpropagation.

Table 1: H-NCA training parameters.

Parameters	Child-NCA Pre-training	Parent-NCA Training
replay buffer size	4096	4096
batch size	4	4
learning rate	1e-3	1e-3
lr_patience	1000	3000
es_patience	1500	4000
t^{min}	32	20
t^{max}	96	60
cell update rate	0.5	0.5
optimiser	Adam	Adam

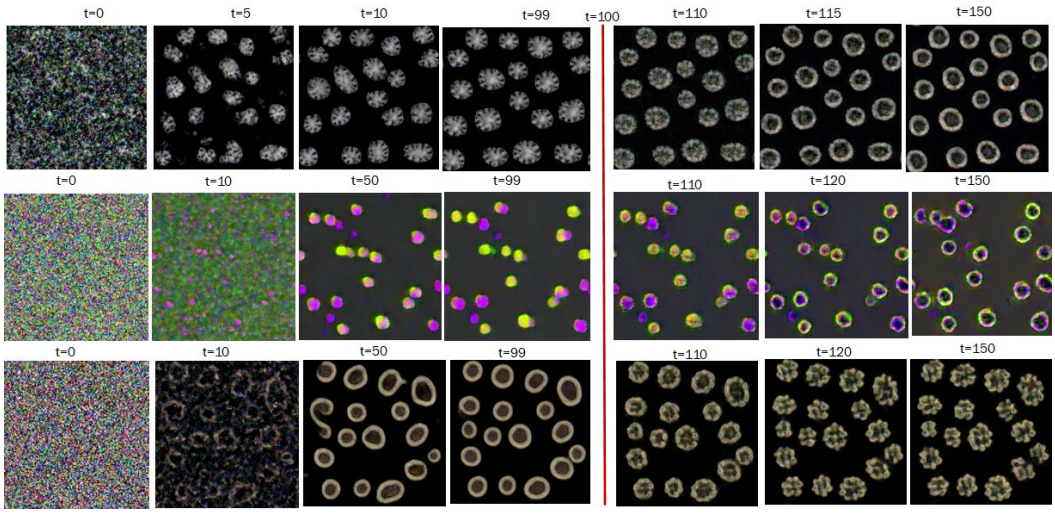
Results

We observe that H-NCA evolves to patterns in the child-NCA target image. After τ_c time steps, in the cell differentiation experiments, the same structural units whose shape is determined by the child-NCA target morph into the new shape desired by the parent-NCA target. This is similar to the process of cell differentiation seen in organisms wherein the phenotypical and functional characteristics of a cell change over time. In the metamorphosis experiments, after τ_c time steps, the structural units generated by the child-NCA change to morphology determined by the parent-NCA target image. In the process, the patterns generated by the child-NCA are destroyed. This is akin to the process of metamorphosis seen in amphibian development wherein at the cellular level, apoptosis occurs in a spatiotemporally regulated fashion in different amphibian organs to ensure timely removal of larval organs or tissues and the development of adult ones (Ishizuya-Oka et al., 2010).

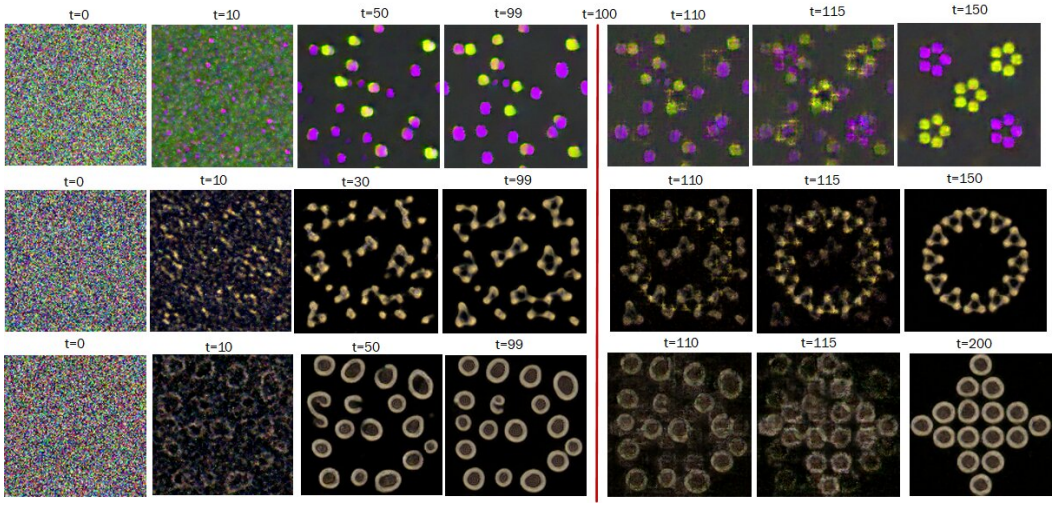
Generally, we observed that the model is stable w.r.t. the choice of τ_c during evolution in the cell differentiation experiments. However, we noticed a growing instability when the value of τ_c during evolution was significantly smaller than the average number of training steps in the metamorphosis experiments. Refer to Figure 6a and Figure 6b for full trajectories of the model evolution captured during the experiments.

Discussion

In this paper, we have modelled how self-organising systems operating at multiple scales with distinct goals can interact to emulate emergent phenomena in morphological space. The NCA higher in the hierarchy influences the cells of the NCA



(a) Results of cell differentiation experiments. The child-NCA in each experiment evolves towards its own target for 100 time steps, after which it gradually morphs to the parent-NCA target.



(b) Results of metamorphosis experiments. The child-NCA in each experiment evolves towards its own target for 100 time steps, after which its structure is destroyed and it evolves to the parent-NCA target.

lower in the hierarchy, modifying their behaviour in order to achieve the higher-level objective. In the process, we have been able to show that:

1. It is possible for self-organising systems higher in the hierarchy to influence the cognitive units lower in the hierarchy via signalling channels dedicated to this purpose.
2. It is possible for self-organising systems higher in the hierarchy to receive feedback on the state of the cognitive units lower in the hierarchy.
3. The goal of the collective can often result in actions that might not be of benefit to its constituents' own goals.

This opens up the possibility of breaking down emergent systems into modular tasks at different scales instead of training a huge monolithic model, like it is often done in many areas of research involving self-organising systems; ranging from evolutionary game theory to modelling complexity in social, economic and political systems. It can also be useful in the field of sociology to study the impacts on individuals as members of collectives at multiple scales.

Future Work

In the future, we would like to perform a stability analysis of H-NCA to identify how perturbation of child-NCA, parent-NCA and the communication layer impact the overall stability of the system. We would also want to investigate whether instability in NCA at one scale can be eliminated by some modification of NCA at another scale. We believe that this can provide significant insight into mechanisms that can be used to cure diseases which arise as a result of failed system homeostasis, such as cancer and autoimmune disorders.

References

De Sales, J., Martins, M., and Stariolo, D. (1997). Cellular automata model for gene networks. *Physical Review E*, 55(3):3262.

Fields, C. and Levin, M. (2022). Competency in navigating arbitrary spaces as an invariant for analyzing cognition in diverse embodiments. *Entropy*, 24(6):819.

Gilpin, W. (2019). Cellular automata as convolutional neural networks. *Physical Review E*, 100(3):032402.

Heitz, E., Vanhoey, K., Chambon, T., and Belcour, L. (2021). A sliced wasserstein loss for neural texture synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9412–9420.

Horibe, K., Walker, K., and Risi, S. (2021). Regenerating soft robots through neural cellular automata. In *Genetic Programming: 24th European Conference, EuroGP 2021, Held as Part of EvoStar 2021, Virtual Event, April 7–9, 2021, Proceedings 24*, pages 36–50. Springer.

Ishizuya-Oka, A., Hasebe, T., and Shi, Y.-B. (2010). Apoptosis in amphibian organs during metamorphosis. *Apoptosis*, 15:350–364.

Li, J., Chen, Z., and Qin, T. (2013). Using cellular automata to model evolutionary dynamics of social network. In *11th International Symposium on Operations Research and its Applications in Engineering, Technology and Management 2013 (ISORA 2013)*, pages 1–6. IET.

Mordvintsev, A., Randazzo, E., Niklasson, E., and Levin, M. (2020). Growing neural cellular automata. *Distill*, 5(2):e23.

Nadizar, G., Medvet, E., Nichele, S., and Pontes-Filho, S. (2022). Collective control of modular soft robots via embodied spiking neural cellular automata. *arXiv preprint arXiv:2204.02099*.

Nichele, S., Ose, M. B., Risi, S., and Tufte, G. (2017). Ca-neat: evolved compositional pattern producing networks for cellular automata morphogenesis and replication. *IEEE Transactions on Cognitive and Developmental Systems*, 10(3):687–700.

Niklasson, E., Mordvintsev, A., Randazzo, E., and Levin, M. (2021). Self-organising textures. *Distill*, 6(2):e00027–003.

Pontes-Filho, S., Walker, K., Najarro, E., Nichele, S., and Risi, S. (2022). A single neural cellular automaton for body-brain co-evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 148–151.

Randazzo, E., Mordvintsev, A., Niklasson, E., and Levin, M. (2021). Adversarial reprogramming of neural cellular automata. *Distill*, 6(5):e00027–004.

Randazzo, E., Mordvintsev, A., Niklasson, E., Levin, M., and Greydanus, S. (2020). Self-classifying mnist digits. *Distill*, 5(8):e00027–002.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sudhakaran, S., Grbic, D., Li, S., Katona, A., Najarro, E., Glanois, C., and Risi, S. (2021). Growing 3d artefacts and functional machines with neural cellular automata. *arXiv preprint arXiv:2103.08737*.

Sudhakaran, S., Najarro, E., and Risi, S. (2022). Goal-guided neural cellular automata: Learning to control self-organising systems. *arXiv preprint arXiv:2205.06806*.

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

Vichniac, G. Y. (1984). Simulating physics with cellular automata. *Physica D: Nonlinear Phenomena*, 10(1-2):96–116.

Von Neumann, J. (1963). *Collected works. 5. Design of computers, theory of automata and numerical analysis*. Pergamon Press.

Wolfram, S. et al. (2002). *A new kind of science*, volume 5. Wolfram media Champaign.