FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	Improving neural cellular automata by incorporating physical dynamics
Student:	Bc. František Koutenský
Supervisor:	Mgr. Petr Šimánek
Study program:	Informatics
Branch / specialization:	Computer Science
Department:	Department of Theoretical Computer Science
Validity:	until the end of summer semester 2023/2024

Instructions

Neural cellular automata (NCA) appeared recently and gained much attention. The behavior of these cellular automata is not prescribed by some simple rules (e.g. Game of Life), but by a small neural network that is learned in a traditional way. The resulting NCA has some interesting behavior that can be useful in studying natural patterns (Turing patterns), and artificial life, or can be helpful in synthetic biology. The important behavior is robustness, growth, self-replication, and the ability to repair damaged patterns. NCAs showed some practical results, for example, control of simple cart-pole or State-of-the-art results for image denoising. The goal of this thesis is to improve its application for some physics-based problems and study the behavior of the resulting patterns.

The task is as follows:

1. The student will study how NCAs work and conduct a literature review to gain a thorough understanding of NCAs.
2. The current NCAs are based on prescribed convolutional filters. The student will implement a different technique that helps the training process by including spatial derivatives as the convolutional filters. The combination of the spatial derivatives of different orders is then trained. This should allow us to explain what governing equations are "used" during the pattern generation process.
3. This new method will be applied to some natural Turing-like patterns. E.g. swordfish stripes and jaguar spots.

Electronically approved by doc. Ing. Jan Janoušek, Ph.D. on 7 February 2023 in Prague.

4. The student will further analyze the results to derive the governing equations.
5. The student will further experiment to understand how is the training process affected by different ways of symmetry breaking (e.g. by random noise, gradient, or point anchor).
6. The student will further analyze the hidden states of the learned NCA to understand if we can find some cellular clusters, thus showing the emergence of cell specialization, or will further perform experiments that could show the adaptability of the NCA patterns.

The student will analyze and interpret the results of the study.

Master’s thesis

IMPROVING NEURAL CELLULAR AUTOMATA BY INCORPORATING PHYSICAL DYNAMICS

Bc. František Koutenský

Faculty of Information Technology
Department of Theoretical Computer Science
Supervisor: Mgr. Petr Šimánek
June 29, 2023

Contents

Acknowledgments **iv**

Declaration **v**

Abstract **vi**

Introduction **1**

1 Literature Review **3**

1.1 Turing Patterns 3

1.2 Classical Cellular Automata 9

1.3 Neural Cellular Automata 12

1.4 Physics-Informed Machine Learning 19

2 Current Work **23**

2.1 Challenges and Structure 23

2.2 Model Development 24

2.3 Leopard Patterns with Unknown Configurations 27

2.4 Leopard Patterns with Known Configurations 33

2.5 Deriving the Governing Equations 37

2.6 Discussion 42

2.7 Pattern Analysis 44

Conclusion **47**

I wish to extend my deepest gratitude to my supervisor, Mgr. Petr Šimánek. Petr is not only an exceptional academic but also an incredibly cool person. His genuine enthusiasm for introducing me to the topic of this thesis was both inspiring and instrumental in fueling my passion for this subject. Thank you, Petr, for being an amazing guide and for your unwavering support throughout this journey.

Declaration

In Prague on June 29, 2023 . . . .

Abstract

This thesis presents the Physically Informed Neural Cellular Automaton (PINCA) model, a novel approach combining Neural Cellular Automata with the integration of physical dynamics. PINCA seeks to uncover governing equations from minimal datasets by learning differential operators through convolutional filters. A case study involving leopard coat patterns demonstrates the model's efficacy in deriving a unique reaction-diffusion system. Additionally, an exploration into the hidden states of Growing Neural Cellular Automata aims to shed light on cell specialization phenomena. The thesis comprises a literature review and the development and application of the PINCA model. This work bridges artificial intelligence and physics, offering new tools for modeling and understanding complex systems.

Keywords Neural Cellular Automata, Turing Patterns, Physically Informed Models

Abstrakt

Tato práce představuje nový Neurální celulární automat se zapojením známé fyzikální dynamiky (PINCA), který spojuje neurální celulární automaty s integrací fyzikální dynamiky. Cílem modelu PINCA je naučit se diferenciální operátory přes učící konvoluční filtry a odhalit řídící rovnice daného procesu i z velmi malých datasetů. Funkce modelu je demonstrována na procesu vývoje vzorů na srsti leoparda, kde byla odhalena unikátní řídící reakčně-difuzní rovnice. Práce se dále zabývá buněčnou specializací skrze zkoumání chování skrytých kanálů modelu Growing neural cellular automata. Hlavním obsahem této práce je podrobná rešerše literatury a vývoj modelu PINCA. Práce spojující umělou inteligenci a fyziku nabízí nové nástroje pro modelování a chápání komplexních systémů.

Klíčová slova Neurální celulární automaty, Turingovy vzory, Modely zahrnující fyziku

Introduction

The field of Artificial Intelligence (AI) has experienced numerous transformations over the past decades, with each evolution enhancing our understanding of the world. From simple rule-based systems to more complex structures like deep neural networks, AI has increasingly approached the level of sophistication observed in natural phenomena. An emerging field in AI that holds immense potential in this regard is Neural Cellular Automata (NCA), which is a fusion of traditional cellular automata and neural networks.

Traditional cellular automata (CA) are discrete models that have been instrumental in simulating physical and biological processes such as fluid dynamics and the growth patterns of colonies of bacteria [1]. In contrast to traditional CA, where behavior is determined by simple predefined rules, NCA leverage small neural networks trained in the traditional way to dictate the behavior of the automata. This innovation has led NCA to exhibit behavior such as robustness, growth, self-replication, and the ability to repair damaged patterns - qualities that are crucial for studying natural patterns, artificial life, synthetic, and developmental biology. A groundbreaking paper by Mordvintsev et al. [2], exemplifies the innovative capabilities of NCA.

One of the longstanding challenges in scientific research is to unravel the governing equations of physical processes. These equations embody the fundamental principles and dynamics that guide the behavior of systems under study. Extracting these equations from observed data can lead to the development of robust models capable of making predictions, running simulations, and providing deeper insights into the underlying mechanisms.

In recent years, various methodologies have been developed to recover governing equations from data. Among these, Sparse Identification of Nonlinear Dynamics (SINDy) [3] is notable for its ability to identify sparse, interpretable models of dynamical systems. Symbolic Regression [4] seeks mathematical expressions that best describe given data, while recently introduced PDE-Net [5, 6] uses deep neural networks for learning partial differential equations (PDEs) directly from observational data. One of the key components of PDE-Net is that the differential operators were actually learned by convolutional filters.

In this thesis, a novel approach which combines the principles of Neural Cellular Automata with the integration of physics through the learning of differential operators via convolutional filters is introduced. This innovative method is termed as the Physically Informed Neural Cellular Automaton (PINCA) model. The PINCA model aims effectively discover governing equations from minimal datasets. As an illustrative case, this thesis tests the efficacy of the PINCA model in the challenging task of recovering governing equations from an exceptionally minimal dataset, comprised solely of three images depicting leopard coat patterns. Through this exercise, we were successful in deriving a reaction-diffusion system involving two substances. Interestingly, in contrast to the classical Turing systems, the equations derived from PINCA exhibited anisotropic behavior.

Additionally, this work explores an auxiliary experiment investigating the hidden states of

Growing NCA, as introduced in [2]. This investigation is aimed at discerning patterns indicative of cellular clusters, which could provide insights into the emergence of cell specialization.

In conclusion, this thesis serves as a pathway to unraveling the hidden complexities within Neural Cellular Automata and exploring their capabilities in tackling intricate physics-oriented challenges. Through the introduction of the PINCA model, this research bridges the gap between Artificial Intelligence and Physics, introducing a novel approach that could pave the way for the development of potent tools and methodologies for comprehending and simulating natural phenomena.

Thesis Structure

The thesis is structured into two primary chapters. The opening chapter offers an extensive review of the literature encompassing various fields that have relevance to the subject matter. It initiates with Turing Patterns, where the Reaction-Diffusion (RD) equations are introduced and demonstrated. Subsequently, the chapter delves into the domain of classical Cellular Automata. This is followed by an introduction to Neural Cellular Automata. Lastly, the Physics-Informed Machine Learning is discussed, to cover the required mathematical background for the PINCA model.

The subsequent chapter of the thesis presents the original contribution, which is the introduction of the PINCA model. The application of this model is explored through the simulation of the growth patterns observed in leopard coats under various conditions. The chapter further expounds on the methodology adopted for extracting the governing equations from the model. Furthermore, in this chapter, an analysis of the hidden states of the PINCA model is conducted, with a focus on examining the emergence of cellular clusters and the specialization of individual cells.

Chapter 1
Literature Review

1.1 Turing Patterns

Turing patterns owe their name to the British mathematician Alan Turing, who introduced the concept in his seminal paper The Chemical Basis of Morphogenesis published in 1952 [7]. In his work, Turing proposed that biological patterns such as the spots and stripes on animal coats could arise naturally from a homogeneous, isotropic state through a process of symmetry breaking, driven by what came to be known as Turing instability.

Morphogenesis, the biological process that causes an organism to develop its shape, had traditionally been viewed as a process guided by a pre-existing pattern or template. However, as the observations and studies from developmental biology imply, embryos often begin in a state of approximate or perfect spherical symmetry. Based on that, Turing argued that a system that has spherical symmetry, and whose state is changing because of chemical reactions and diffusion, will remain spherically symmetrical forever. However, this would not lead to the development of complex organisms like a horse, which exhibit non-spherical symmetry.

Instead, he hypothesized that pattern formation is an emergent property of a system of reacting and diffusing chemicals – a system that can start from an entirely uniform state. According to Turing, the symmetry within such systems is disrupted by necessary small deviations. These deviations arise from inherent irregularities, stochastic fluctuations, or statistical fluctuations in the biochemical reactions and molecular concentrations within the system.

As a mathematical model of this system, Turing introduced the reaction-diffusion equations. These equations describe how the concentration of certain chemicals changes over time and space, as they react with each other and diffuse through the organism's tissue. Turing revealed that under certain conditions, a stable, uniform solution to these equations can undergo instability, giving rise to pattern formation, which is known as Turing instability.

A general reaction-diffusion equation for two variables u and v, representing the concentrations of two chemical species, can be represented mathematically as a system of partial differential equations (PDEs):

$$\begin{aligned} \frac{\partial u}{\partial t} &= D_u \nabla^2 u + R(u, v) \\ \frac{\partial v}{\partial t} &= D_v \nabla^2 v + S(u, v) \end{aligned} \tag{1.1}$$

where $\frac{\partial u}{\partial t}$ and $\frac{\partial v}{\partial t}$ represent the rate of change of the concentrations of u and v with respect to time, depending on diffusion and chemical reactions. The diffusion terms $D_u \nabla^2 u$ and $D_v \nabla^2 v$ describe spatial diffusion, while the reaction terms $R(u, v)$ and $S(u, v)$ capture the interactions between u and v. Specifically, the diffusion coefficients D_u and D_v control the spreading of concentrations, and the Laplacian operators $\nabla^2 u$ and $\nabla^2 v$ represent spatial diffusion. The reaction terms $R(u, v)$ and $S(u, v)$ account for the effects of chemical reactions or interactions. They encompass various aspects, such as reaction rates, reaction mechanisms, and the influence of one chemical species on the other.

Sadly, the minimum number of chemicals required to generate Turing patterns is a topic of ongoing research and can depend on the specific system and the desired pattern characteristics. The work by Alan Turing suggested that at least two chemicals are necessary for the formation of Turing patterns. In some cases, systems with only two chemicals can indeed produce a variety of Turing patterns.

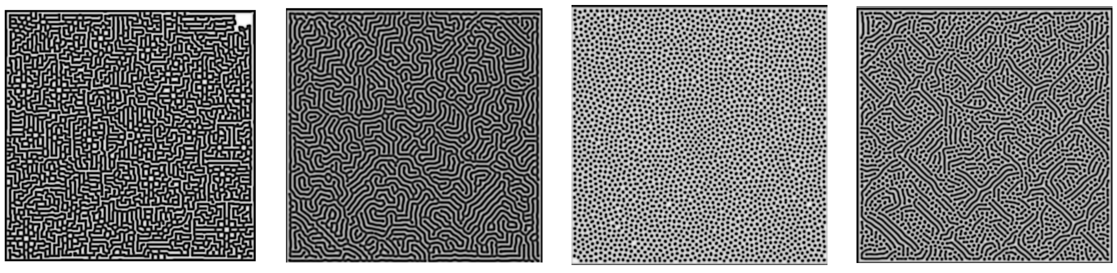
Gray-Scott Model

The Gray-Scott model is a classic example of a reaction-diffusion system that involves two interacting chemicals u and v and can generate a wide range of Turing-like patterns [8]. The Gray-Scott system models the chemical reaction $U + 2V \rightarrow 3V$, where the chemical u is consumed and v is synthesized. The regulation of the quantities of both chemicals and thus the maintenance of the reaction is accomplished by introducing u at a specific feeding rate F, while simultaneously extracting v at a rate denoted by the parameter k, which is referred to as the killing rate. The removal of v is described as a separate chemical reaction, $V \rightarrow P$, where P symbolizes an inert by-product that does not participate in any further reactions and thus does not contribute to the observations. The parameter k regulates the rate at which this secondary reaction takes place.

Mathematically, the Gray-Scott model can be written as follows:

$$\begin{aligned} \frac{\partial u}{\partial t} &= D_u \nabla^2 u - u \cdot v^2 + F \cdot (1 - u) \\ \frac{\partial v}{\partial t} &= D_v \nabla^2 v + u \cdot v^2 - (F + k) \cdot v \end{aligned} \tag{1.2}$$

Figure 1.1 illustrates various patterns generated through the Gray-Scott model, where D_u is set to 0.16 and D_v to 0.8, which corresponds to the Pearson's parametrization. The initial configurations of the system consisted of the entire grid filled with the first chemical u (represented by white color), with some cells occupied by the second chemical v (represented by black color) randomly placed. The randomness in the positioning of the inhibitor cells was introduced to break the symmetry of the system. The images show the quantities of the chemical u only, as the occurrence of v is inversely related to the occurrence of u.


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

■ **Figure 1.1** Patterns generated by the Gray-Scott model with Pearson's parametrization after they filled the whole simulation-frame, taken from [9]. (a) The Flower-pattern: F=0.055, k=0.062. (b) The Mazes-pattern: F=0.029, k=0.057. (c) The Mitosis-pattern: F=0.028, k=0.062. (d) The Solitons-pattern: F=0.03, k=0.06.

Gierer-Meinhardt Model

In the realm of Turing Patterns, the exploration of pattern formation in biological systems has been of utmost interest. A central question in biology is the development of complex multicellular organisms from a single fertilized egg. The intriguing aspect is the emergence of cellular diversity despite each cell possessing identical genetic information. This phenomenon is not solely restricted to biological systems; it is also observed in inorganic systems, such as the formation of large sand dunes and branching river systems, despite seemingly homogeneous initial conditions.

A common thing in these pattern-forming systems is the positive feedback mechanism that increases deviations from homogeneity. For instance, an initial depression on a surface gathers more water, which further deepens the depression due to erosion. Similarly, a slight elevation in sand creates a wind shelter, leading to sand deposition behind it. However, for pattern formation to occur, this self-enhancement must be accompanied by a longer-ranged inhibitory effect, preventing unlimited growth. This can stem from inhibitory signaling that spreads from the region or due to the consumption of essential materials from the surrounding areas for self-enhancement [10].

In the early 1970s, Alfred Gierer and Hans Meinhardt, inspired by Alan Turing's pioneering work, formalized these observations into a molecular model for pattern formation. This model, known as the Gierer-Meinhardt model [11], is also composed of two reaction-diffusion PDEs. It characterizes the interaction between an activator substance, which has a short-range autocatalytic effect, and its antagonist, the inhibitor, which operates over a longer range.

The mathematical description of the Gierer-Meinhardt model can be represented by the following equations:

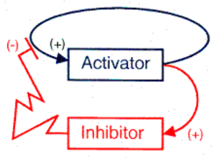
$$\begin{aligned}\frac{\partial a}{\partial t} &= \rho \cdot (\frac{a^2}{h} - \mu \cdot a) + D_a \nabla^2 a \\ \frac{\partial h}{\partial t} &= \rho \cdot (a^2 - v \cdot h) + D_h \nabla^2 h\end{aligned} \tag{1.3}$$

Here, the chemicals a and h stand for the activator and the inhibitor respectively. The parameters ρ, μ, and v have specific roles in the reaction kinetics of the model.

In the first equation, the term $\frac{\partial a}{\partial t}$ represents the rate of change of the activator's concentration over time. The reaction term $\rho \cdot (\frac{a^2}{h} - \mu \cdot a)$ represents the auto-catalytic production of the activator and its decay. Specifically, the activator promotes its own production in a nonlinear manner proportional to a^2, and this production is inhibited by the inhibitor concentration h. The decay rate of the activator is proportional to μ.

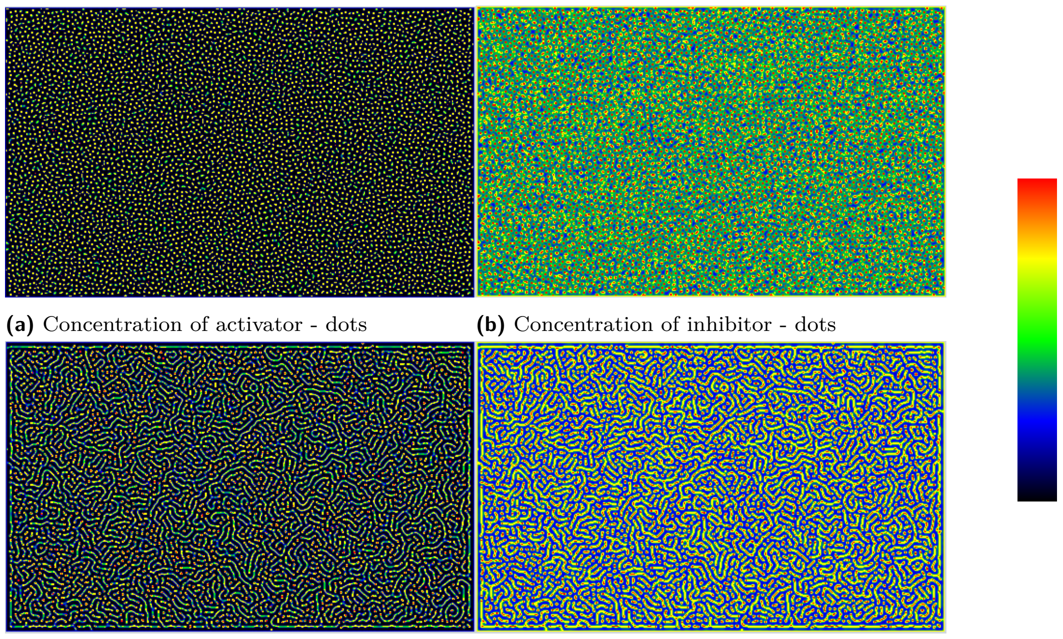
In the second equation, $\frac{\partial h}{\partial t}$ represents the rate of change of the inhibitor's concentration over time. The reaction term $\rho \cdot (a^2 - v \cdot h)$ captures the production of the inhibitor, which is promoted by the activator concentration squared a^2 and decays proportional to v.

It is important to notice the role of the activator and inhibitor in this model. The activator acts in an auto-catalytic manner - it reinforces its own production - but is regulated by the inhibitor. Contrarily, the production of the inhibitor is promoted by the activator. This leads to a balance between activation and inhibition, a key feature in the formation of patterns.



■ **Figure 1.2** Activator and inhibitor in Gierer-Meinhardt model [10].

While the Gray-Scott model also involves the interaction between an activator and an in-



(a) Concentration of activator - dots **(b)** Concentration of inhibitor - dots

(c) Concentration of activator - snakes **(d)** Concentration of inhibitor - snakes

■ **Figure 1.3** Snapshots of the time evolution of the activator and inhibitor concentrations in the Gierer-Meinhardt model are shown for two parameters presets, namely ”dots” and ”snakes”. In these visual representations, the color red is indicative of the highest concentration, as per the reference color map located on the right-hand side. The initial state for these simulations features a random distribution of the activator. The specific parameters utilized for generating the ”dots” and ”snakes” patterns in the Gierer-Meinhardt model can be explored through an interactive web-based simulation interface, from which this figure was sourced [16].

hibitor, the key difference between these two models is that the Gray-Scott model does not feature the auto-catalytic reinforcement of the activator as seen in the Gierer-Meinhardt model. Instead, the Gray-Scott model has a reaction term where the activator is consumed to produce the inhibitor.

Applications of the Gierer-Meinhardt model include explanations for natural patterns such as pigmentation in animal coats, distribution of hairs or bristles, and the arrangement of leaves or floral organs in plants. It has been an essential tool in developmental biology to understand how simple interactions at the cellular level can lead to complex structures at the organism level [12, 13, 14, 15].

Equations Behind Nature

While models like Gray-Scott and Gierer-Meinhardt provide a theoretical framework for pattern formation, adapting these models to accurately represent real-life biological processes is a challenging task. Moreover, these models are simplifications and represent idealized versions of reaction-diffusion systems. In real biological processes, the mechanisms are often more complex, involving lots of interacting species, feedback loops, and spatial structures; different organisms or tissues within the same organism may exhibit vastly different patterns. Additionally, even slight variations in genetic structure or environmental conditions can result in substantially different outcomes. This diversity makes it difficult to find a one-size-fits-all governing equation for biological pattern formation.

Another problem is data acquisition. While collecting precise data is essential for constructing and validating mathematical models, obtaining detailed quantitative data at the cellular or molecular level is often challenging due to limitations in experimental techniques. Moreover, ethical considerations can limit the types of experiments that are feasible, particularly in higher organisms.

Subsequently, when the suitable mathematical model for the underlying process is formulated, we still need to estimate the parameters of the governing equations, such as reaction rates and diffusion coefficients. This is also a challenging job, as biological systems often exhibit high sensitivity to these parameters and a slight change in any of them can lead to a dramatically different behavior.

Despite these challenges in finding the governing equations in real-life biological processes, there has recently been a remarkable success in understanding complex biological systems. For instance, reaction-diffusion models were developed to understand the processes such as the formation of pigmentation patterns in zebrafish [17], the development of limbs in vertebrates [18], the generation of fur patterns in mammals [12], the creation of coat patterns in marine angelfish [19], or to understand how cells achieve polarity and how this polarity influences morphogenesis, particularly in the context of the development of complex structures in plants and animals [20]. These works are just the tip of the iceberg. For a more comprehensive exploration of examples and insights into the subject, the classical textbook ”Mathematical Biology” by Murray [13] serves as an invaluable resource.

Modeling Leopard Patterns

In [21], Liu et al. successfully constructed a reaction-diffusion model for the leopard and jaguar patterns formation. As this paper has influenced the research conducted in this thesis, we outline its contents.

The authors used a simplified reaction-diffusion model inspired by Barrio et al. [22], which they fundamentally altered such that it reaches the steady state, a state where the concentrations of the substances involved no longer change with time when the concentrations of both substances are at zero. This implies that the variables should be interpreted as deviations from a steady state rather than as actual concentrations. The model incorporates basic nonlinear interaction terms, specifically quadratic and cubic interactions between two morphogens, u and v. The governing equations are:

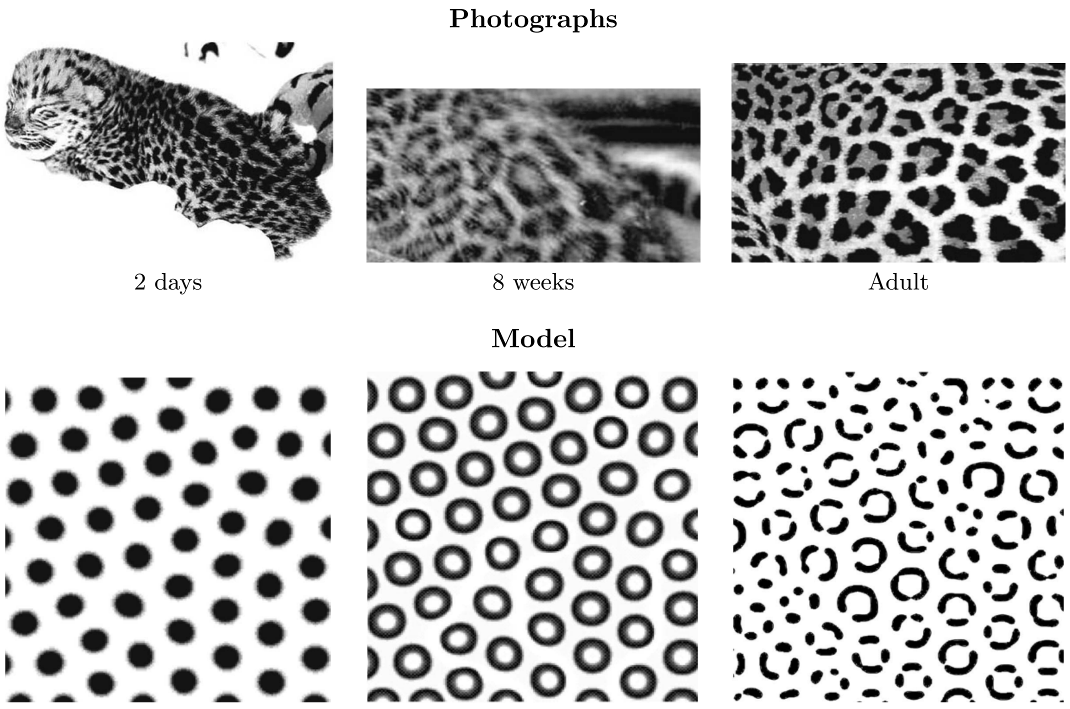
$$\begin{aligned} \frac{\partial u}{\partial t} &= D\delta\nabla^2 u + \alpha u + v - r_2 uv - \alpha r_3 uv^2 \\ \frac{\partial v}{\partial t} &= \delta\nabla^2 v + \gamma u + \beta v + r_2 uv + \alpha r_3 uv^2 \end{aligned} \tag{1.4}$$

where D denotes the ratio of the diffusion coefficient of the activator u to that of the inhibitor v, and δ represents a scaling variable that can be interpreted either as the relative strength of the diffusion compared to the interaction terms or as a quantification of the length scale in the problem. Next, there are two interaction parameters, r_2 and r_3, corresponding to a cubic and a quadratic term, respectively, and parameters α, γ and β. The zero flux boundary conditions, which in the case of a square domain means that the concentrations of both substances are zero at each boundary, are imposed on the system.

The authors focused on three distinct stages of leopard coat patterns, which were classified according to the different growth phases. These stages, illustrated in Figure 1.4, include the spot pattern observed at 2 days old, the ring pattern at 8 weeks, and the rosette pattern in adulthood.

At first, they determined the values for the parameters of the model in order to generate the first stage of the leopard pattern. The initial values of u and v at each point were assigned randomly between 0 and 1. The corresponding parameters are as follows:

$$D = 0.45,\ \delta = 6,\ \alpha = 0.899,\ \gamma = -\alpha,\ \beta = -0.91,\ r_2 = 2,\ r_3 = 3.5.$$



■ **Figure 1.4** The comparison between the patterns on real leopards and those generated by the reaction-diffusion models proposed in [21].

Next, through a trial-and-error approach, the parameters were iteratively fine-tuned to enable the model to evolve from the distribution initially yielded by the model to produce the second and third stages. Namely, they increased r_2 from 2 to 7 to obtain the second stage patterns, and then decreased δ from 6 to 3.8 to obtain the third stage, which was then stabilized by decreasing D from 0.45 to 0.15. The produced results are depicted in Figure 1.4.

The authors conclude that the ability of a Turing reaction-diffusion model to exhibit a temporal sequence of spatial patterning was exhibited, while being consistent with some examples in nature. This adds additional credence to the involvement of morphogens in developmental biology, though it does not serve as definitive proof.

1.2 Classical Cellular Automata

Cellular Automata (CA) is a model that is extensively used in various disciplines and has strong ties with Turing patterns and reaction-diffusion systems. A cellular automaton is a dynamic system that evolves over time. It consists of a regular grid of cells of an arbitrary dimension, where each cell can be in one of a finite number of states. The evolution of the system is determined by the synchronous application of some local predefined rules that update the state of each cell based on the cell's current state and the states of its immediate neighbors.

Mathematically, the state of a cell at position x at the next time step $t+1$, denoted as $s_{t+1}(x)$, is determined by a transition function F, which takes as its inputs the states of a set of neighboring cells around x, denoted as $s_t(x_0), s_t(x_1), \ldots, s_t(x_{k-1})$, at the current time step t. The evolution of a cell's state is described by:

$$s_{t+1}(x) = F(s_t(x_0), s_t(x_1), ..., s_t(x_{k-1})). \tag{1.5}$$

Here, the transition function F represents the local predefined set of rules. The idea that the same state transition function and the same neighborhood apply uniformly to all spatial locations is the most characteristic assumption of CA [23].

In the context of the reaction-diffusion equations, CA can be utilized to approximate these continuous systems. Reaction-diffusion equations usually describe the change in concentrations of one or more chemical substances over time as they react with each other and diffuse through space. These equations are continuous in both time and space. However, by discretizing space into a grid of cells and considering the time in discrete steps, one can adapt CA to model reaction-diffusion systems. In this scenario, the state of each cell represents the concentration of the substances, and the transition function F mimics both the reaction and diffusion processes. The neighborhood is defined in a way to account for the diffusion process, where substances move from regions of high concentration to regions of low concentration.

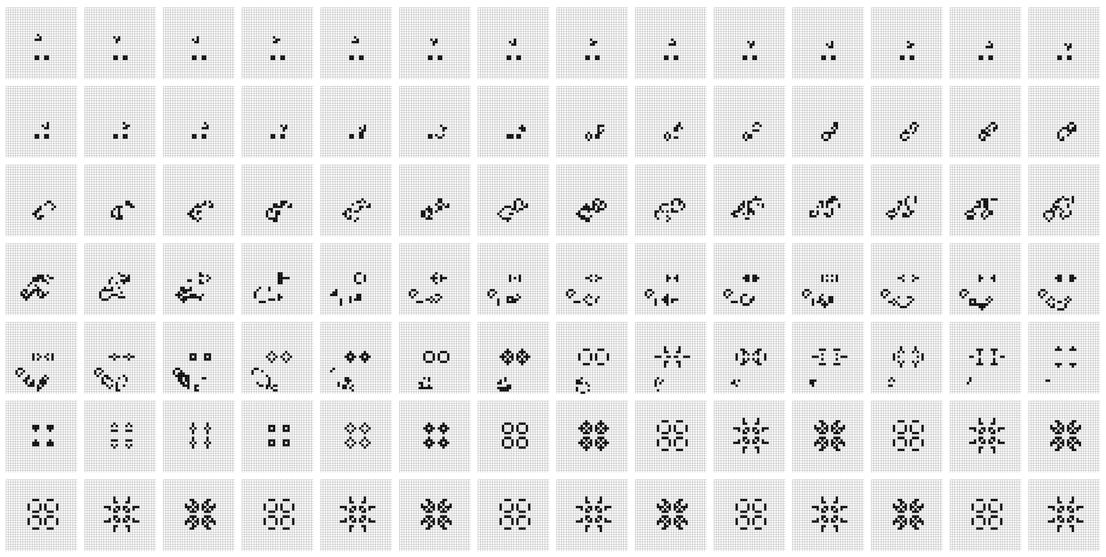
Development and Milestones

The invention of cellular automata can be credited to more prominent researchers of that time, but mainly to John von Neumann and Stanislaw Ulam. In [24], Neumann solved the question of whether a machine can autonomously self-reproduce itself by proposing a cellular automaton with 2×10^5 initial configuration that would, given an initial pattern, make infinitely many copies of that pattern. Another important contribution to the use of CA was made by Konrad Zuse in 1969, who, in his book [25], introduced and laid the foundation of the *Digital physics* hypothesis. According to the theory, the universe could be the result of computation by a discrete computing system, such as cellular automata. The hypothesis challenges the traditional belief that certain laws of physics are inherently continuous.

Perhaps the most famous finding in the study of cellular automata is John Conway's *Game of Life* [26] automaton, which was created in 1970. The Game of Life consists of an infinite two-dimensional grid with the following rules:

- Any live cell with fewer than two live neighbors dies, as if by underpopulation.
- Any live cell with two or three live neighbors lives on to the next generation.
- Any live cell with more than three live neighbors dies, as if by overpopulation.
- Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

These simple rules lead to surprisingly fascinating patterns and behaviors. Some patterns are stable and remain unchanged, while others oscillate or move across the grid. Certain configurations even generate self-replicating structures or "glider" patterns that traverse the grid indefinitely. An example of the Game of Life cellular automaton that, given the correct initial configuration, creates an interesting pulsating pattern can be seen in Figure 1.5.



■ **Figure 1.5** The run of the Game of Life cellular automaton starts in an initial configuration which creates an interesting pulsating pattern also known as *Pulsar* found by John Conway [27]. The initial configuration is positioned in the upper left corner, while the following iterations are displayed sequentially from left to right. Notably, on the 82nd iteration, the system returns to a state identical to that observed three iterations earlier, indicating an ongoing repetition of the subsequent three iterations that together form the Pulsar pattern. The figure was created by the author of this thesis, who discovered the corresponding initial configuration in [27].

In the 1980s, Wolfram proposed a conjecture stating that a one-dimensional cellular automaton called *Rule 110* is Turing-complete. This conjecture was later proven by Cook in 2004, confirming that Rule 110 indeed possesses the computational power of a Turing machine [28]. Similarly, the Game of Life was also shown to be Turing-complete in 2002, thanks to the efforts of Rendell et al [29]. Eventually, these proofs showed the potential of CA and paved the way for their further investigation.

Applications

Cellular automata have had a considerable impact on a variety of areas, including physics, biology, chemistry, social sciences, and urban planning, among others.

In the field of biology, as previously mentioned, CA played a significant role in the development of Turing Patterns. Alan Turing's paper [7], and the Meinhardt and Gierer model [11] were instrumental in laying the foundation for how reaction-diffusion systems, often modeled using CA, can explain pattern formation in biological systems.

Furthermore, [30] investigated various biologically motivated CA arising in excitable and oscillatory media, developmental biology, neurobiology, and population biology. Cellular automata have also been used to model mammalian hair growth [31], simulate an immune system [32], and model the spread of infectious diseases [33].

In the realm of chemistry, reaction-diffusion systems modeled by CA have been used to study chemical reactions and the diffusion processes [34]. For instance, in [35], reaction-diffusion computing is explored as a substrate for unconventional computing, using chemical waves and their interactions.

In [36], the possibility of application CA to simulate physics was investigated, with a conclusion that cellular automata are capable of various levels of physics simulation and that CA, even though having no structural resemblance with any physical system, exhibits features that are unquestionably physical. The potential of CA in modeling complex systems like reaction-diffusion systems, pattern formation phenomena, fluid flows, fracture processes, and road traffic models was examined thoroughly in the paper of [37]. Another extensive work discussed in [38] tries to establish an alternative to classical realistic fully continuous molecular dynamics by employing cellular automata. Additionally, the Lattice Boltzmann method, which is a popular CA-based approach, has been extensively used for simulating fluid dynamics [39].

Urban planning is another area where CA have found application. In particular, CA models have been used to simulate urban growth and to predict land-use changes [40].

In the social sciences, CA can model and simulate complex social systems. Schelling's segregation model is an example of how simple local interactions in a CA can lead to emergent behavior resembling societal segregation [41].

In computer science, the concept of CA has proven to be of fundamental importance. For instance, Stephen Wolfram's work [42] and the discovery of Rule 110 [28], which is Turing complete, show that CA can perform computations of arbitrary complexity. Moreover, CA have been used in various engineering disciplines including architecture [42] for pattern generation and structural design, and logistics [43] for optimizing transportation networks. Lastly, cellular automata were also used to model mammalian hair growth [31] and simulate an immune system [32].

1.3 Neural Cellular Automata

Despite the effectiveness of classical cellular automata in modeling certain systems, they suffer when it comes to capturing dynamic and adaptive behaviors. Classical CA typically rely on fixed rules that do not change over time or adapt to the environment. This makes them less suitable for modeling systems with evolving dynamics or systems that require learning and adaptation. Additionally, classical cellular automata may struggle to capture the complex interactions and feedback loops present in many real-world systems.

A prominent issue of classical CA that limits their use is that they often require manual tuning of parameters in order to achieve desired behaviors or patterns. These parameters include the initial configuration, the rules governing the state transitions, and other factors such as the grid size or neighborhood structure. Mathematicians and researchers often need to invest significant effort in fine-tuning these parameters to obtain the desired emergent behavior from the automaton. This process can be time-consuming and challenging, particularly when dealing with complex systems or trying to capture specific dynamic or adaptive behaviors.

The answer to these drawbacks is the *Neural Cellular Automata* (NCA). Like CA, NCA also operate on a grid of cells by applying some local rules. However, in NCA, the cells' states are influenced not only by their immediate neighbors but also by the activation patterns of their neural networks. These neural networks receive inputs from neighboring cells and use them to update the cell's state. The neural networks can learn and adapt based on the patterns they observe in the local environment. The complexity as well as the learning method of the underlying neural network can, of course, vary, based on the specific requirements, but perhaps the most feasible way to train NCA has recently been through backpropagation [44]. The usual procedure involves iterating the system for a fixed number of time steps, computing the loss function between the evolved final state and the target output, and using the computed loss to update the weights of the neural network. This process is repeated iteratively until convergence, refining the neural networks' weights to gradually align the NCA's final state with the desired final state. Additionally, a cell state is typically a vector of real numbers in NCA.

Interestingly, the concept of having the transition function of CA learned by neural networks is not new. The first realization of this idea was done in 1992 by Wulff and Hertz [45]. However, recent advancements in hardware capabilities, particularly highly-parallelizable differentiable operations implemented on GPUs, have enabled the utilization of deep learning techniques in NCA. This has demonstrated that NCA can be effectively tuned to learn remarkably complex desired behaviors. Furthermore, one can easily see another immediate benefit of NCA in contrast to traditional CA - an experimenter is only required to define the initial configuration, the number of time steps, and the desired final state, as opposed to providing a full set of rules as in traditional CA. While it is true that the hyperparameters of neural networks often require manual configuration by the experimenter, optimizing these hyperparameters is generally much less challenging compared to finding the parameters for rules in cellular automata.

Mathematical Definition

According to the notation used in the classical CA, the state of a cell at position x at the next time step $t+1$, denoted as $s_{t+1}(x)$, is determined by a transition function F_{NCA}, which takes as its inputs the states of a set of neighboring cells around x, denoted as $s_t(x_0), s_t(x_1), \ldots, s_t(x_{k-1})$, at the current time step t, as well as the weights W of the neural network. The evolution of a cell's state in NCA is described in the following way:

$$s_{t+1}(x) = F_{\text{NCA}}(s_t(x_0), s_t(x_1), \ldots, s_t(x_{k-1}), W). \tag{1.6}$$

For a simple feedforward neural network with one hidden layer within the NCA, this can be further expanded as:

$$h = \sigma(W_1 \cdot [s_t(x_0), s_t(x_1), \ldots, s_t(x_{k-1})] + b_1)$$
$$s_{t+1}(x) = \sigma(W_2 \cdot h + b_2) \tag{1.7}$$

where σ is the activation function (e.g., ReLU, sigmoid), W_1, b_1 and W_2, b_2 are the weight matrices and the bias vectors, respectively, of the neural network, and h is the hidden state.

During training, a loss function L is defined to measure the discrepancy between the evolved final state and the target output.

$$L = \mathrm{Loss}(s_T(x), s_{\mathrm{target}}(x)) \tag{1.8}$$

where T is the final time step and $s_{\mathrm{target}}(x)$ is the target state for cell at position x. The weights of the neural network are updated to minimize this loss, typically by using the back-propagation algorithm.

$$W \leftarrow W - \alpha \nabla_W L \tag{1.9}$$

where α is the learning rate and $\nabla_W L$ is the gradient of the loss with respect to the weights. This process is repeated until the NCA's behavior converges to the desired outcome.

From the perspective of classical CA, NCA can be seen as an extension of cellular automata that leverages the capabilities of neural networks, which are known to be universal function approximators [46], to construct the transition function F.

Growing NCA

The crucial paper that has recently awakened the interest in NCA was done by Mordvintsev et al. [2]. In their highly comprehensible interactive paper, they trained several NCA models that simulate morphogenesis, which is the biological process by which an organism develops its shape, structure, and form. Specifically, multiple images of different emojis like lizards, smiley faces, fish, etc. were grown from a single pixel cell seed to their final form by NCA. The image, displayed in Figure 1.6, provides an explanation of how the model works, while Figure 1.7 shows the step-by-step iteration of the model. There were also presented experiments that aimed not only to grow the desired pattern from a single cell, but also to persist in the final state after the number of training steps is exceeded by the system, or even regenerate the desired final state when parts of it are spoiled. The promising results brought by this paper opened the door for an investigation of NCA in the field of developmental biology and artificial life.

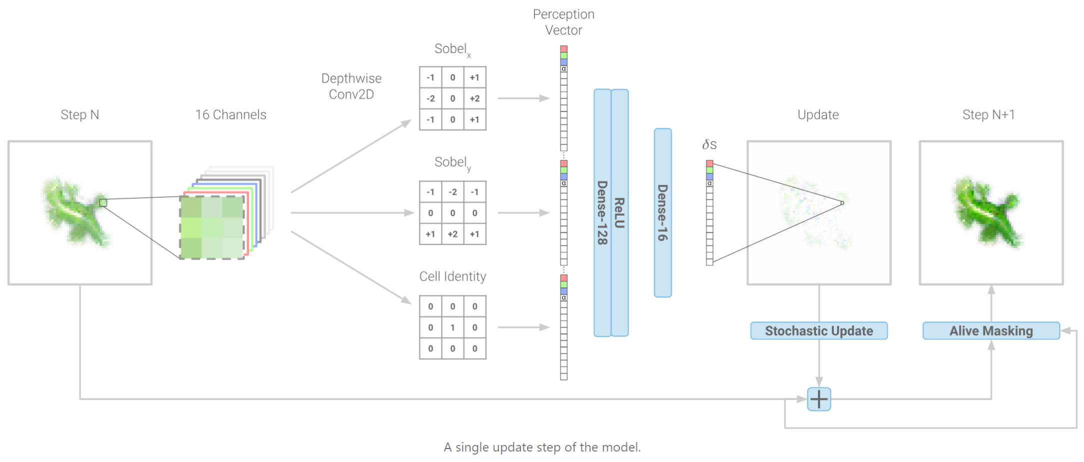
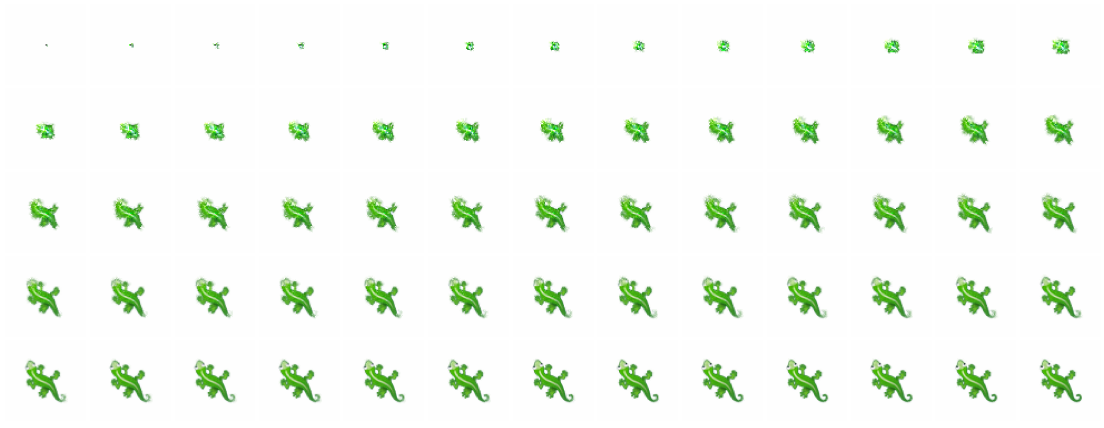


Figure 1.6 The description of the model proposed in the pioneering work on NCA [2].

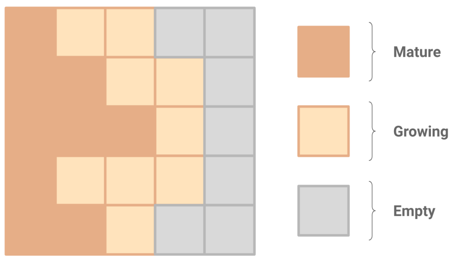


■ **Figure 1.7** The results produced by the growing neural cellular automaton constructed in [2], which was trained to develop a lizard from a single cell seed.

In the following discussion, it will be delved into the details of the growing NCA architecture introduced in [2], as it serves as the foundation for the model proposed in this thesis.

Cell State

The growing NCA model employs a 2D grid with each cell represented as a 16-dimensional vector. The first three dimensions represent the RGB color, and the fourth dimension is the alpha (α) channel, which has a special meaning. It specifies which cells are considered to be a part of the growing pattern (”alive”) and which ones are not (”dead”). A cell is considered alive if it has at least one neighbor in its immediate 3x3 neighborhood with $\alpha > 0.1$. The other cells are considered empty (or ”dead”) and their state vector is set to 0 at every time step. The authors label the cells with $\alpha > 0.1$ as ”mature”, and their neighbors with $\alpha \leq 0.1$ as ”growing”. The remaining dimensions are hidden channels, which can represent various signals, such as chemical concentrations. Figure 1.8 illustrates the cell states.



■ **Figure 1.8** The cell states of the growing NCA.

Cellular Automaton rule

To update the cells, a differentiable update rule is used. This allows the use of gradient-based numerical optimization for modifying cell states. The differentiable update rule is essentially a combination of convolution operations and non-linear functions. The update rule is applied in four phases:

- **Perception**: Each cell perceives its environment through a 3x3 convolution with a set of three fixed (non-learnable) kernels. The Sobel filters for both x and y directions are used together with the identity filter. This step produces gradients for each cell state channel and concatenates them into a 48-dimensional perception vector for each cell.

- **Update Rule**: The update rule is executed by first passing the perception vector through a dense layer with 128 units. Following this, a ReLU activation function is applied, and then another dense layer reduces the dimensions back to 16. This update rule outputs an incremental update to the cell's state.
- **Stochastic Cell Update**: Classical CA models update all cells simultaneously, which requires global synchronization. This model adopts a more organic approach by updating each cell independently at random intervals. Specifically, it is achieved through the *fire rate* parameter, which represents the probability of a cell's state being updated. With a fire rate set to 0.5, there is a 50% chance that the incremental update will be applied to a cell's current state.
- **Living Cell Masking**: Cells with no living neighbors (based on the alpha channel) are set to empty by zeroing out all their channels.

Learning to Grow Experiment

In the "Learning to Grow" experiment, the authors train the NCA model to achieve a target image through a series of updates. Initially, the grid is filled with zeros, except for a single seed cell in the center.

The update rule is iteratively applied for a randomly chosen number of steps within the range of 64 to 96. This randomization aims to ensure that the pattern remains stable across multiple iterations. After the final step, a pixel-wise L2 loss is computed between the RGBA channels in the grid and the target pattern. This loss is then optimized with respect to the update rule parameters through backpropagation, similar to training recurrent neural networks.

One of the resulting simulations of this experiment can be seen in Figure 1.7.

Recent Publications

In [47], NCA are used in the self-classifying task on the MNIST dataset. The goal of the task, which is described in Figure 1.9, is to develop a communication protocol among the cells so that, after a series of iterations of communication, all cells can correctly identify the digit they are collectively forming. The task also explores whether the agents can adapt to changes in the digit formation, such as adding or removing agents to create a new digit. They also adapted their model for writing and erasing, which showed surprising robustness to certain ranges of digit stretching and brush widths. Moreover, a hypothesis that self-organizing models with constrained capacity may be inherently robust and have good generalization properties was proposed. The research serves as a proof-of-concept for how simple self-organizing systems such as CA can be used for classification when trained end-to-end through backpropagation.

The authors of the growing NCA paper also delved into the exploration of the adversarial attack on NCA [48]. Their goal was to have some adversarial cells that would change the global configuration and behavior of all the cells. In the case of growing NCA, they trained two adversarial NCAs: one that aims to turn the lizard into a tailless lizard, and the other one trying to make the lizard red. Besides that, they also used the self-classifying MNIST NCA, mentioned above. In the case of the MNIST NCA, the goal of the trained adversarial NCA was to highjack the collective's classification consensus to always classify an eight. The results suggest that by introducing adversarial cellular automata into a pretrained self-classifying MNIST CA, a cell system that heavily relies on information exchange is highly susceptible to manipulation by deceptive signals. The authors also state that this scenario mirrors the challenges encountered by biological systems, where parasites and other competitive agents in the biosphere can hijack physiological or behavioral mechanisms. On the other hand, the adversarial injection attack was much less effective against the growing CA and resulted in overall unstable CA.

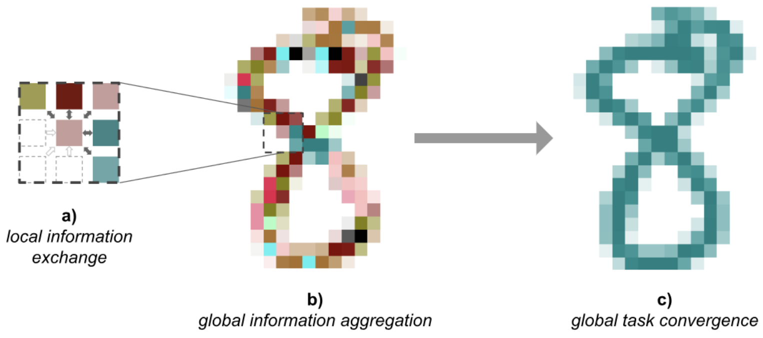


Figure 1.9 The description of the self-clasifying MNIST NCA [47].

Another interesting paper of Sinapayen [49] examined self-replication, spontaneous mutations, and exponential genetic drift in NCA. The study revealed that NCA patterns can exhibit self-replication accompanied by spontaneous, inheritable mutations and exponential genetic drift. Surprisingly, even deterministic NCAs displayed a divergence between descendant patterns and ancestral patterns over time, suggesting the emergence of evolutionary dynamics. The paper highlights the unique capabilities of NCA, expanding the potential for open-ended evolution and providing insights into the interplay between replication, mutation, and genetic drift in complex systems. The detailed description of the self-replication experiment can be seen in Figure 1.10.

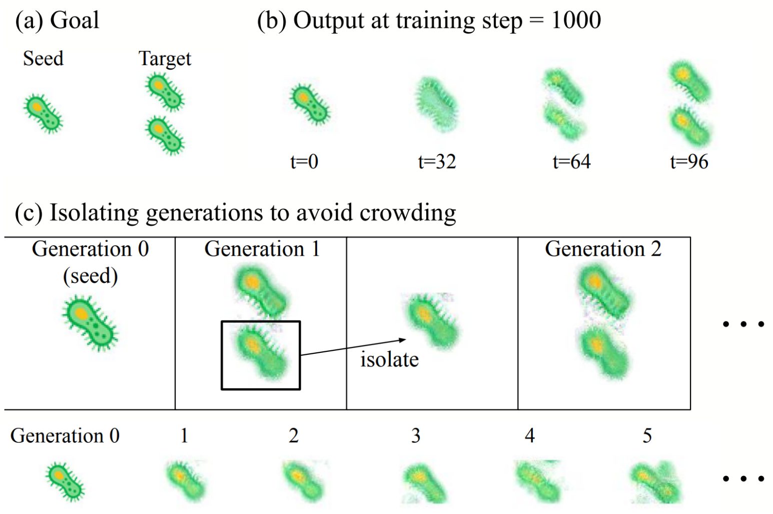


Figure 1.10 The simple self-replication experiment done in [49] on the bacteria emoji. After 96 steps, when the new bacteria is fully evolved, one is taken randomly to the next generation and the same process is repeated. The author highlights the visual difference between successive generations G5 that seem to have 2 nuclei (yellow central patch).

In the study on self-organizing textures by Niklasson et al. [50], where the task is to generate new images that express perceptual similarity to a provided texture sample, an intriguing approach was introduced. The research demonstrated how NCAs can be trained to generate and sustain various synthesized textures. Rather than computing the loss as a per-pixel mean squared error (MSE) between the NCA's final state and the reference output, the researchers employed the VGG-16 network, a well-known deep convolutional network for image recognition, to assess the similarity between the generated image and the reference image and to provide the required gradients. The findings propose that this approach led to the creation of NCAs capable of producing intricate and captivating textures and that by enforcing local interactions, the NCAs were encouraged to discover algorithms that could generate such patterns, prioritizing high-level consistency and robustness.

The potential of NCA in reinforcement learning (RL) tasks was shown in the work of Variengien et al. [51], where the simple RL cart-pole problem was successfully solved by NCA. The authors used Deep-Q learning [52], where the states of the output cells were used as the q-value estimates to be optimized.

Najarro et al. [53] introduce HyperNCA, which is NCA acting as hypernetwork, capable of producing policy networks with a larger number of parameters. They demonstrate the effectiveness of HyperNCA in solving modern reinforcement learning tasks, such as discrete action environments and quadrupedal robot locomotion. Although the results did not reach SotA in any RL benchmark, it brings a novel and promising approach to growing neural networks through local self-organizing processes.

In [54], a Vision Transformer Cellular Automata (ViTCA) is introduced, which combines the idea of Vision Transformer (ViT) [55] with the NCA framework, resulting in attention-based NCA for the denoising autoencoding. The localized self-attention in ViTCA exhibits global self-organization, enabling per-pixel dense processing while avoiding the quadratic complexity of explicit global self-attention. ViTCA architectures outperform other baselines, including U-Net [56], the U-Net-based CA (UNetCA) model, which the authors implemented, and ViT, across multiple denoising autoencoding benchmarks.

The utilization of NCA in the field of generative models was done in the work of [57] Palm et al., who came up with a generative model inspired by the biological processes of cellular growth and differentiation called Variational Neural Cellular Automata (VNCA). The VNCA combines NCA with the principles of the probabilistic generative model Variational Auto-Encoder [58]. Although not reaching state-of-the-art performance, the VNCA demonstrates the capability to learn a self-organizing generative model of data. Additionally, the model exhibits the ability to learn a distribution of stable attractors of data, which allows for recovery from damage or perturbations.

In the paper of Sandler et al. [59], a novel approach for solving real-world segmentation problems using cellular automata is proposed. They design and train a cellular automaton that can effectively segment high-resolution images. The automaton consists of a colony of cells densely inhabiting the pixel grid, with each cell governed by a randomized update rule based on the current state, color, and the state of its 3×3 neighborhood. The space of possible update rules is defined by a small neural network. The update rule is applied in parallel to a large random subset of cells, and after convergence, segmentation masks are generated and back-propagated to learn the optimal update rules using gradient descent. The experiments demonstrate that these models can be efficiently learned with short trajectory lengths and exhibit an impressive ability to produce globally consistent segmentations using only local information exchange. The authors highlight the advantages of cellular automata, such as incremental computation, suitability for continuously changing data like videos, and compact model size. They also discuss the potential for further advancements in cellular automata as a framework for solving complex tasks with simple independent agents.

The power of NCA

As NCA directly comes from the classical CA that were proven, in certain cases, to be Turing-complete, it is natural to ask how they can be classified within the realm of computational models. As mentioned in [54], NCA is fundamentally not a “very large” convolutional neural network, as might be suggested from [60], where the author claims that *arbitrary* CA may be analytically represented by convolutional perceptrons with finite layers and units - in fact, the convolutional neural network (CNN) was used to predict the next state at $t+1$, given the current state at time t, which introduces recurrence. Thus, NCA should rather be viewed as a type of recurrent neural network (RNN), also because both NCA and RNN induce a directed *cyclic* computation graph, while CNN produce a directed *acyclic* computation graph. Moreover, RNN were, like CA, shown to be Turing-complete [61].

In terms of computational efficiency, NCA offers advantages over traditional neural networks. Since the update of each cell in NCA is performed in parallel without requiring global synchronization, it can be implemented efficiently on parallel architectures, such as GPUs or distributed computing systems. This parallelism allows NCA to process large-scale data and perform real-time simulations, making it suitable for applications requiring high computational capacity.

As mentioned in [53], the main motivation behind using NCA to grow a neural network in the HyperNCA architecture is the presumably shared characteristic of both the developmental process of biological systems and NCA, which is the *computational irreducibility.* Computational irreducibility refers to the idea that certain complex systems cannot be efficiently simulated or predicted without actually running them. The definition of computational irreducibility used in [62] states that a system is computationally irreducible if in order to determine a system's state after n steps, one must evolve the system for n steps according to its dynamical equation. In other words, there is no analytical formula available that can directly provide the system's state at any given step without actually evolving the system through those steps.

As an example of the computational irreducible biological system, one might consider the embryogenesis of a fertilized egg, which is during its development transformed into a fully developed organism. During this process, the egg is influenced by numerous factors such as genetic information, biochemical signals, and environmental interactions. Thus it is impossible to predict the exact structure and morphology of an organism at a particular stage of its development. It is conjectured that the developmental process of biological systems is computationally irreducible [42]. Moreover, as implied by many papers mentioned above, the NCA is able to exhibit computational irreducibility as well.

1.4 Physics-Informed Machine Learning

Physics-Informed Machine Learning (PIML) represents a fusion of empirical data and prior physical knowledge to improve performance in tasks that are governed by physical or biological mechanisms. Integrating physical knowledge into machine learning models can uplift their learning process, reduce the required amount of training data, improve generalization, and enhance interpretability [63]. As an example, consider training a neural network to predict the motion of a pendulum. Without physical priors, the network may fit the training data well but fail to generalize to different initial conditions. By incorporating the conservation of energy as a physical constraint, the neural network is encouraged to learn the underlying physical dynamics, improving generalization. For a more comprehensive understanding of this topic, it is recommended that interested readers refer to the extensive survey conducted by Hao et al. [63].

Prior physical knowledge encompasses the fundamental laws and principles of physics that dictate the behavior of the physical world. Such knowledge can vary in strength, from potent inductive biases like PDEs, to weaker constraints based on symmetries and intuitive physical concepts. The survey on Physics-Informed Machine Learning conducted by Hao et al. [63] presents the following strategies to incorporate prior physical knowledge into ML models:

- **Data**: The dataset can be augmented using physical priors, such as symmetries in our data. One example of incorporating symmetry in a data strategy is by augmenting a dataset of images with rotational symmetry. This can be done by adding rotated versions of the images to the dataset, which helps improve the model's training by including variations that align with the underlying symmetries present in the objects.

- **Model**: Physical priors can be embedded within the model design, such as network architecture. This is achieved by introducing inductive biases guided by physical priors to narrow down the range of possible models or solutions. For example, in the case of a fluid dynamics problem, it might be beneficial to inherently respect the conservation of mass and momentum, which can be done by designing a network architecture that processes inputs and outputs in a way that the sum of mass or momentum remains constant.

- **Objective**: The loss functions or regularization terms can be modified using physical priors like PDEs. For example, convolutional kernels can be used as differential operators with known priors, and a loss can be imposed on these kernels to enforce the desired physical behavior.

- **Optimizer**: Optimization methods can be modified to be more stable or to converge faster by incorporating physical priors.

- **Inference**: Physical constraints can be enforced by modifying inference algorithms. For example, in the case of a ML model that predicts the temperature distribution in a system based on input data, we know that the temperature should always be non-negative, as negative temperatures are physically impossible. Thus, one can modify the post-processing function to enforce this physical constraint during inference.

Physics-informed neural networks (PINNs) [64] are a type of a ML model that utilizes physical knowledge through the objective. The training of PINNs is enhanced by incorporating the governing equations directly into the loss function, which allows the learning of the underlying physics from data and enables to solve forward and inverse problems of PDEs. Here, the forward PDE problem refers to finding a data-driven solution of a PDE, while the objective of the inverse PDE problem is to discover the unknown parameters of a PDE based on the observed data. PINNs utilize the neural networks to satisfy the following physical constraint:

$$\mathcal{L}(u(x,t),\theta) = g$$

where $u(x, t)$ represents the output of the neural network, where x and t are the spatial and time variables, θ denotes the parameters of the neural network, $\mathcal{L}$ is an operator that represents the physics of the problem, and g represents the known information or constraints from the physical laws, which could include boundary conditions, initial conditions, conservation laws, or other physical properties.

The range of applications for PINNs is extensive. For instance, in solving forward PDE problems, PINNs have been effectively utilized in tackling the Navier-Stokes equations that characterize the motion of viscous fluids [65]. Additionally, they have been applied to solve elasticity equations, which govern the behavior of deformable solid materials [66]. In the context of inverse PDE problems, PINNs have been used for determining unknown parameters in PDEs using observed data [67]. Moreover, a more generalized model has been developed which is capable of learning the PDE directly from data [6].

Inverse PDE Problem

Identifying the governing equations of physical systems is crucial for understanding and predicting their behavior. Sometimes these equations are not known or are too complex to be derived from first principles. In such cases, extracting equations from experimental or simulation data becomes a valuable approach.

Besides PINNs, a method to extract governing equations directly from data represents the Sparse Identification of Nonlinear Dynamics (SINDy) algorithm [3], which works by formulating the problem as a sparse regression task. The main idea behind SINDy is to identify a sparse set of nonlinear terms that best explain the dynamics of the system. The limitation of this approach is its fundamental reliance on an effective coordinate system in which the dynamics have a simple representation. To address this drawback, a novel method employing a custom deep autoencoder network was recently developed [68]. This method identifies a coordinate transformation leading to a sparse representation of dynamics while simultaneously learning the governing equations and their associated coordinate system.

Another method that for solving the inverse PDE problem from a given set of data is symbolic regression [4]. The process of symbolic regression involves searching through a space of mathematical expressions to find the one that optimally describes the data. Unlike traditional linear and nonlinear regression methods that fit parameters to an equation of a given form, symbolic regression searches both the parameters and the form of equations simultaneously. This process automatically forms mathematical equations that are amenable to human interpretation and help explicate observed phenomena [69]. Symbolic regression algorithms typically use a population-based search strategy, inspired by evolutionary computation techniques such as genetic programming.

Both SINDy and symbolic regression are powerful tools for discovering mathematical models from data. SINDy is especially effective for identifying sparse dynamical systems, where the underlying dynamics can be represented by a small number of governing equations. On the other hand, symbolic regression offers a more general approach but can be computationally expensive. In comparison, PINNs combine the strengths of both SINDy and symbolic regression while addressing some of their limitations. They excel at capturing complex and nonlinear systems while providing accurate predictions. However, PINNs can be computationally demanding for large-scale problems and require a lot of training data for good generalization.

Recently, a paper presenting PDE-Net [5], a unique approach combining deep neural networks with the learning of differential operators as convolutional kernels (filters) and the approximation of unknown nonlinear relationships, was introduced. In contrast, the advantage of this novel method is that it does not require any prior knowledge about the differential operators in the underlying PDE, except for their maximum possible order. The way how differential operators are learned in PDE-Net will be discussed in the following section, as it is closely related to the main work conducted in this thesis.

Learning Differential Operators

In many cases of searching for the governing equation of a physical process, the specific form of its differential operators is often unknown. Thus, the approach of learning the differential operators directly from data can be highly promising. In [5] and [6], the learnable filters are properly constrained so that their correspondence to differential operators can be easily identified. This approximation can be seen as a discrete approximation of the derivatives, similar to the finite difference method [70]. The theory behind constraining the filters to produce differential operators is based on two papers [71, 72], where the authors discussed the connection between the order of sum rules of filters and the orders of differential operators. In the following text, the definitions and concepts presented in [5] and [6] will be reproduced to provide the necessary theoretical foundation that the work conducted in this thesis is based on.

The definition of convolution that is used follows the convention of deep learning and is defined followingly:

$$(f \circledast q)[l_1, l_2] := \sum_{k_1, k_2} q[k_1, k_2] f[l_1 + k_1, l_2 + k_2]$$

Further, the concept of the order of sum rules, which is closely related to the order of vanishing moments in wavelet theory [73, 74], is defined:

▶ **Definition 1.1** (Order of Sum Rules)**.** *For a filter q, we say q to have sum rules of order $\alpha = (\alpha_1, \alpha_2)$, where $\alpha \in \mathbb{Z}^2_+$, provided that*

$$\sum_{k \in \mathbb{Z}^2} k^\beta q[k] = 0 \tag{1.10}$$

for all $\beta = (\beta_1, \beta_2) \in \mathbb{Z}^2_+$ with $|\beta| := \beta_1 + \beta_2 < |\alpha|$ and for all $\beta \in \mathbb{Z}^2_+$ with $|\beta| = |\alpha|$ but $\beta \neq \alpha$. If 1.10 holds for all $\beta \in \mathbb{Z}^2_+$ with $|\beta| < K$ except for $\beta \neq \bar{\beta}$ with certain $\bar{\beta} \in \mathbb{Z}^2_+$ and $|\bar{\beta}| = J < K$, then we say q to have total sum rules of order $K \setminus \{J + 1\}$.

The proposition presented in [71] establishes a connection between the orders of sum rules and the orders of differential operators.

▶ **Proposition 1.2.** *Let q be a filter with sum rules of order $\alpha \in \mathbb{Z}^2_+$. Then for a smooth function $F(x)$ on $\mathbb{R}^2$, we have*

$$\frac{1}{\varepsilon^{|\alpha|}} \sum_{k \in \mathbb{Z}^2} q[k] F(x + \varepsilon k) = C_\alpha \frac{\partial^\alpha}{\partial x^\alpha} F(x) + O(\varepsilon), \ \text{as } \varepsilon \to 0. \tag{1.11}$$

If, in addition, q has total sum rules of order $K \setminus \{|\alpha| + 1\}$ for some $K > |\alpha|$, then

$$\frac{1}{\varepsilon^{|\alpha|}} \sum_{k \in \mathbb{Z}^2} q[k] F(x + \varepsilon k) = C_\alpha \frac{\partial^\alpha}{\partial x^\alpha} F(x) + O(\varepsilon^{K - |\alpha|}), \ \text{as } \varepsilon \to 0. \tag{1.12}$$

Proposition 1.2 suggests that a differential operator of order α can be approximated through the convolution of a filter that possesses sum rules of order α. Moreover, as indicated by equation 1.12, a more accurate approximation of a particular differential operator can be achieved if the filter being used has an order of total sum rules where $K > |\alpha| + k$, with $k \geq 1$.

▶ **Example 1.3.** Considering filter

$$q = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix},$$

it has a sum rules of order $(1, 0)$, and total sum rules of order $3 \setminus \{2\}$. Thus, up to a constant and a proper scaling, q corresponds to a discretization of $\frac{\partial}{\partial x}$ with second-order accuracy.

The *moment matrix* is used to approximate differential operators with prescribed orders of accuracy.

▶ **Definition 1.4** (Moment matrix). *For a filter q of size $N \times N$, the moment matrix of q, denoted by $M(q)$, is defined as an $N \times N$ matrix with entries $m_{i,j}$, where each entry is given by*

$$m_{i,j} = \frac{1}{i!j!} \sum_{k_1,k_2=-\frac{N-1}{2}}^{\frac{N-1}{2}} k_1^i k_2^j q[k_1,k_2], \tag{1.13}$$

for $i, j = 0, 1, \ldots, N-1$.

In what follows, $m_{i,j}$ refers to the elements of the moment matrix of a filter q as previously defined. For any smooth function $f : \mathbb{R}^2 \to \mathbb{R}$, convolution is applied on its sampled version with respect to the filter q. The following formula can be derived from the Taylor's expansion:

$$\begin{aligned} &\sum_{k_1,k_2=-\frac{N-1}{2}}^{\frac{N-1}{2}} q[k_1,k_2] f(x+k_1\delta x, y+k_2\delta y) \\ =& \sum_{k_1,k_2=-\frac{N-1}{2}}^{\frac{N-1}{2}} q[k_1,k_2] \sum_{i,j=0}^{N-1} \left.\frac{\partial^{i+j} f}{\partial x^i \partial y^j}\right|_{(x,y)} \frac{k_1^i k_2^j}{i!j!} \delta x^i \delta y^j + o\left(|\delta x|^{N-1} + |\delta y|^{N-1}\right) \\ =& \sum_{i,j=0}^{N-1} m_{i,j} \delta x^i \delta y^j \cdot \left.\frac{\partial^{i+j} f}{\partial x^i \partial y^j}\right|_{(x,y)} + o\left(|\delta x|^{N-1} + |\delta y|^{N-1}\right). \end{aligned} \tag{1.14}$$

The implications from Equation 1.14 is that the filter q could be designed to approximate any differential operator with the prescribed order of accuracy by imposing constraints on $M(q)$.

▶ **Example 1.5.** For approximation of $\frac{\partial u}{\partial x}$ by convolution $q \circledast u$ where q is a 3×3 filter, the following constraints on $M(q)$ can be considered:

$$\begin{pmatrix} 0 & 0 & \star \\ 1 & \star & \star \\ \star & \star & \star \end{pmatrix} \text{ or } \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & \star \\ 0 & \star & \star \end{pmatrix} \tag{1.15}$$

Here, $\star$ means no constraint on the corresponding entry. The constraints described by the moment matrix on the left of 1.15 guarantee the approximation accuracy is at least first order, and the one on the right guarantees an approximation of at least second order. In particular, when all entries of $M(q)$ are constrained, e.g.

$$M(q) = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

the corresponding filter can be uniquely determined. The work presented in this thesis follows the approach of PDE-Net and PDE-Net 2.0, where all filters are learned with partial constraints on their associated moment matrices to ensure at least second-order accuracy.

Chapter 2
Current Work

2.1 Challenges and Structure

The development of the PINCA model was an iterative process, characterized by a multitude of trials and refinements. The foundation for our model was laid using the publicly available growing NCA model, which can be accessed here [75]. In the initial phase, we retained the model's configurations from the original growing NCA paper [2], particularly employing 16 channels. The model adapted well and exhibited proficiency in pattern learning. However, the use of 16 channels led to exceedingly complex governing equations, which lacked practicality and violated the principle of Occam's razor.

Upon observation, several challenges emerged with the 16-channel model. Among these challenges, the main issue was the intricacy involved in identifying the appropriate initial configuration. Moreover, it was uncertain if it is even possible to find the correct initial configuration for a small number of channels. Another concern was the partial stochastic nature of the process, which often culminated in slightly varied final states despite identical settings. Additionally, the impact of altering a key parameter – the number of channels – on the model was unknown.

Furthermore, there was a particular interest in investigating how the integration of physical principles could enhance the model's performance and capabilities. Consequently, we aimed to compare an NCA that integrated physical knowledge with one lacking such incorporation.

In light of these insights and objectives, a series of experiments was orchestrated. The initial experiment focused on training the model to simulate all three stages of leopard coat patterns from an unknown initial state. The subsequent experiment narrowed the scope to training the model to transition from the first (known) stage to the second and third stages. An auxiliary experiment was conducted to facilitate the extraction of the governing equations, which involved subtle modifications to the model's architecture.

Additionally, a supplementary experiment employing the original growing NCA model was conducted to analyze the formation of clusters within hidden states. This component deviates from the central theme, which is PINCA. Thus, this topic is presented in the concluding section of this chapter.

2.2 Model Development

Implementation and Code

For implementing our model, we opted to utilize the PyTorch library [76], which is widely recognized for its flexibility and efficiency, and also the initial code was built using this platform. The code accompanies this submission and includes detailed implementations of the model used in the first and second experiments, as well as the modified version employed in the third experiment for extracting governing equations from data. Additionally, we have provided several pre-trained versions of the models to facilitate further exploration.

To make the exploration process more intuitive and interactive, Jupyter notebooks [77] are included. There are two notebooks: one is dedicated to the model used in the first and second experiments, and the other corresponds to the model used in the third experiment.

Furthermore, the source code for the cellular clustering experiment is provided.

Data

The data utilized in our first two experiments comprise images of the leopard coat pattern at its first, second, and third developmental stages. These images, generated via a reaction-diffusion model as presented in the study by [21], are shown in Figure 2.1.

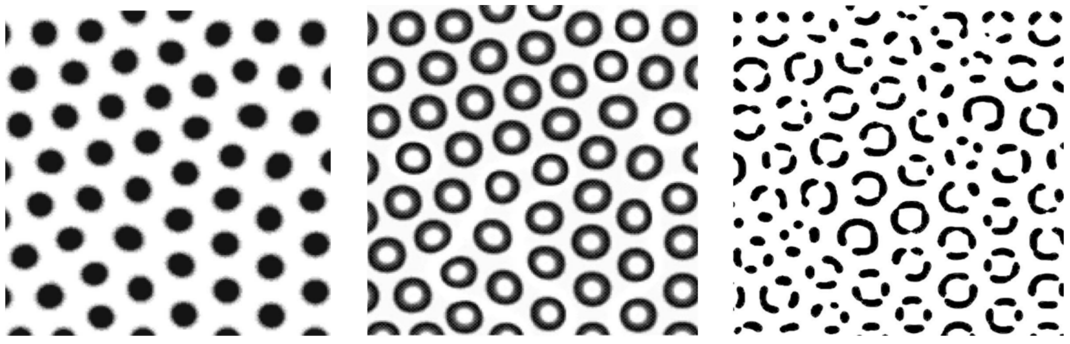


Figure 2.1 The figure displays three distinct stages of leopard coat patterns during growth, as simulated through a reaction-diffusion system, according to [21]. These three images were employed as the dataset for our experiments.

To accelerate the training process, all images utilized in our experiments were resized to a resolution of 40x40 pixels.

Architecture

Our architecture builds on the architecture of growing NCA [2]. The primary difference lies in the way the perception vector is assembled. Moreover, unlike the original architecture where the alive masking process was involved, in our version, all cells are considered to be active and are subject to evolution.

Similar to traditional CA, the model iterates for a predefined number of steps. In each iteration, the process of cells evolution is as follows: At every position, all C channels are taken. Subsequently, each channel is convolved individually with nine learnable filters $K_1, \ldots, K_9$. The convolution outputs are then assembled together, forming a vector whose size is the product of the number of channels and the number of kernels (9 in this case). Within this vector, each group of 9 consecutive values corresponds to a single channel. For each channel, this set of

9 values is subsequently fed into a dense layer consisting of a single hidden unit, which then yields a single output value per channel, producing C values in total. In parallel, an identity filter is applied to each channel, resulting in C additional values. By combining both outputs, a perception vector of size $2 \times C$ is obtained. This vector is then fed into a dense layer of size 128, using ReLU activation. The values are subsequently transformed back to C dimensions through another dense layer of size C. The resulting output vector contains the cell value change for each channel. Finally, stochasticity is applied, and the output vector is summed with the grid at step N, thereby generating the new grid at step $N + 1$.

The detailed description of a single iteration of our model is depicted in Figure 2.2.

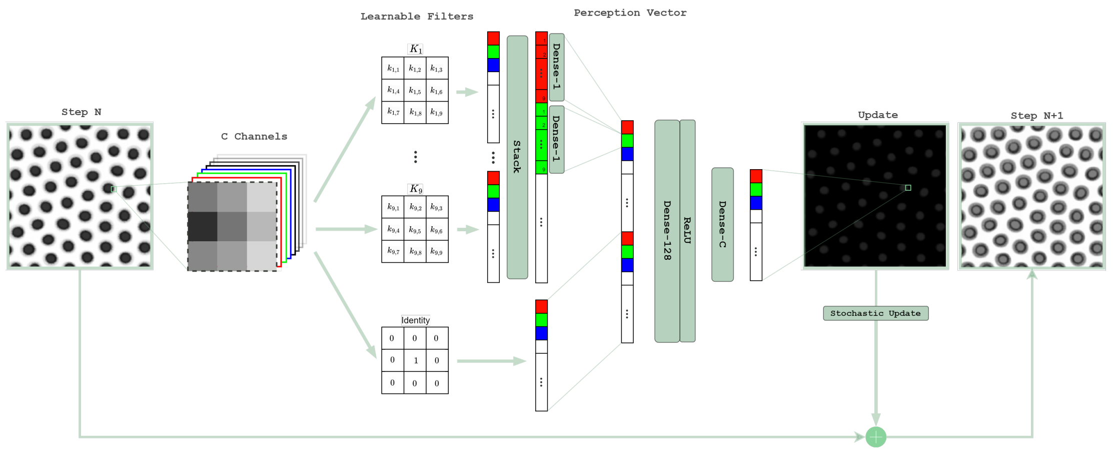


Figure 2.2 The complete description of the model developed within this work, capturing a single iteration step from the image at step N to the image at step $N + 1$.

In the context of the convolutional operation we employed, each of the nine filters was represented as a 3x3 matrix. During each weight update of the model, a moment matrix was computed for each filter, yielding nine moment matrices in total. The computation of these moment matrices adhered to the definition of the moment matrix as introduced in the preceding chapter, under the section titled “Physics-Informed Machine Learning”, in Definition 1.4. Upon obtaining the moment matrices, each one was compared against its predefined counterpart, leading to the calculation of the MSE loss between the paired matrices.

According to the theory, the nine predefined matrices were selected such that the sum of the elements within each matrix equaled one, and no two predefined matrices were identical. The significance of the filters’ moments converging toward these predefined matrices lies in the transformation of the filters into approximations of specific differential operators. This is a pivotal achievement since the filters now carry physical meaning and are not mere abstract constructs.

Subsequently, the MSE losses associated with all filters are aggregated to compute the total loss attributable to the filters. This synthesized measure serves as an index of how closely the filters are aligning with their physical counterparts, aiding in providing the model with a more grounded and interpretable structure.

Parameters and Conditions

All the parameters related to backpropagation were adopted from the growing NCA model, as these parameters were already fine-tuned and demonstrated to be effective. Additionally, the cell fire rate, which governs the stochasticity (i.e., whether the cell will be updated during a given iteration), was maintained at 0.5.

Given that the data provided are not cyclic in nature (meaning they do not wrap around) and instead represent a segment of the coat pattern, it is crucial to handle edge cases carefully to avoid introducing biases into our model. To this end, considerations include the size and value of padding, as well as the treatment of edges. With respect to edge treatment during convolution in the model, there are two primary strategies that can be employed: zero padding and circular padding. Zero padding entails adding zeros along the periphery of the image, while circular padding wraps the image to form a continuous loop. Zero padding has the advantage of implicitly breaking system symmetry, usually facilitating the learning process for the model. However, a limitation of this setting is that the model may become overly dependent on the edges of the input. In contrast, circular padding makes the training process more difficult but forces the model to concentrate exclusively on the core content of the initial configuration. The selection between these two approaches depends on the specific experiment being conducted, and hence, the choice for each parameter is described within the respective sections of each experiment.

A crucial parameter is the number of steps that the NCA takes to produce the first, second, and third stages of the coat pattern. It is important to note that the number of steps required for each stage of growth varies among the three experiments, and as such, these are detailed within the respective sections dedicated to each experiment.

Loss Function

The selection of the loss function was relatively straightforward. Depending on the number of channels we experimented with, two cases emerged: When there were more than four channels, we calculated the pixel-wise Mean Squared Error (MSE) on the first four channels (RGBA) of the model's output after a specific number of steps, and compared it against the label image corresponding to a certain stage. Conversely, if there were three channels or fewer, we assessed the loss in a similar fashion but solely on the first channel, which was also employed for visualization purposes. Ultimately, the losses from each stage were aggregated. In addition to this, the loss from the constrained convolutional filters was also aggregated.

Furthermore, L1 regularization was implemented in certain instances. The particulars of when and how it was utilized are contingent upon the specific experiment and will be discussed in the subsequent sections.

2.3 Leopard Patterns with Unknown Configurations

In this initial experiment, the primary objectives are threefold: (a) to find the optimal initial configuration that enables the model to generate all three stages of coat patterns, (b) to minimize the number of channels as much as possible in order to yield a parsimonious model, which can represent governing equations that are as simple as possible, and (c) to evaluate the extent to which incorporating physics enhances the model.

For this experiment, padding was configured to extend 8 pixels from each side, and these spaces were populated with a constant value of 1 (white pixels). Naturally, the loss was exclusively measured within the original, non-padded parts of the image. Additionally, the edges were subjected to circular padding (wrapping) during the convolution process.

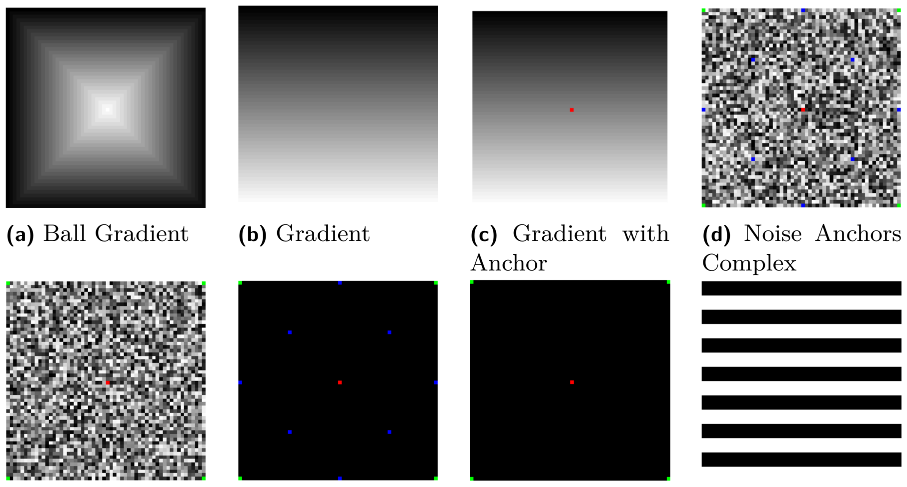
Additionally, an L1 regularization term was applied to the dense layers corresponding to the outputs of the convolution. This was employed with the intent to encourage the model to utilize only the necessary differential operators for each channel, thus promoting sparsity. The overall loss can be expressed as a combination of three terms:

$$loss_{total} = 1 \cdot loss_{data} + 100 \cdot loss_{filters} + 0.0001 \cdot loss_{L1_f}$$

In this equation, $loss_{total}$ represents the aggregate loss that is used during the backpropagation step. The term $loss_{data}$ accounts for the pixel-wise mean squared error between the model's output and the corresponding target pattern for each stage involved in training. The loss computed on the constrained filters is denoted by $loss_{filters}$. Lastly, $loss_{L1_f}$ represents the L1 regularization term applied to the dense layers corresponding with the outputs of the convolution.

Initial Configuration and Channel Count

Guided by insights we accumulated through hands-on engagement with the model, we identified eight promising configurations. These configurations are visually represented in Figure 2.3. The hidden channels were all set to zero for each configuration.



(a) Ball Gradient **(b)** Gradient **(c)** Gradient with Anchor **(d)** Noise Anchors Complex

(e) Noise Anchors Simple **(f)** Plain Anchors Complex **(g)** Plain Anchors Simple **(h)** Stripes

■ **Figure 2.3** The chosen initial configuration candidates.

To make the training process smoother and enhance efficiency, we divided it into two phases. During the initial phase, the model was trained to exclusively develop the first stage of the leopard pattern (spots). This training, where the initial configuration was one of the candidate initial configurations, was allowed to progress to the subsequent stages - forming the rings and rosettes - only after reaching a certain loss threshold. This threshold was selected based on the observation that, when the models' outputs reached a loss in proximity to this threshold, they visually resembled the desired spots pattern.

Specifically, the NCA ran for 70 steps to evolve from the initial configuration to the first stage, followed by an additional 40 steps to reach the second stage, and a final set of 40 steps to complete the pattern formation.

We conducted a set of experiments, where the initial configuration and the number of channels were explored. Furthermore, we conducted experiments without constraints on the convolutional kernels, allowing the learnable filters complete freedom to adapt. Moreover, we explored different edge treatments and, in one instance, utilized zero padding, which, however, did not yield any significant improvements and was thus discontinued.

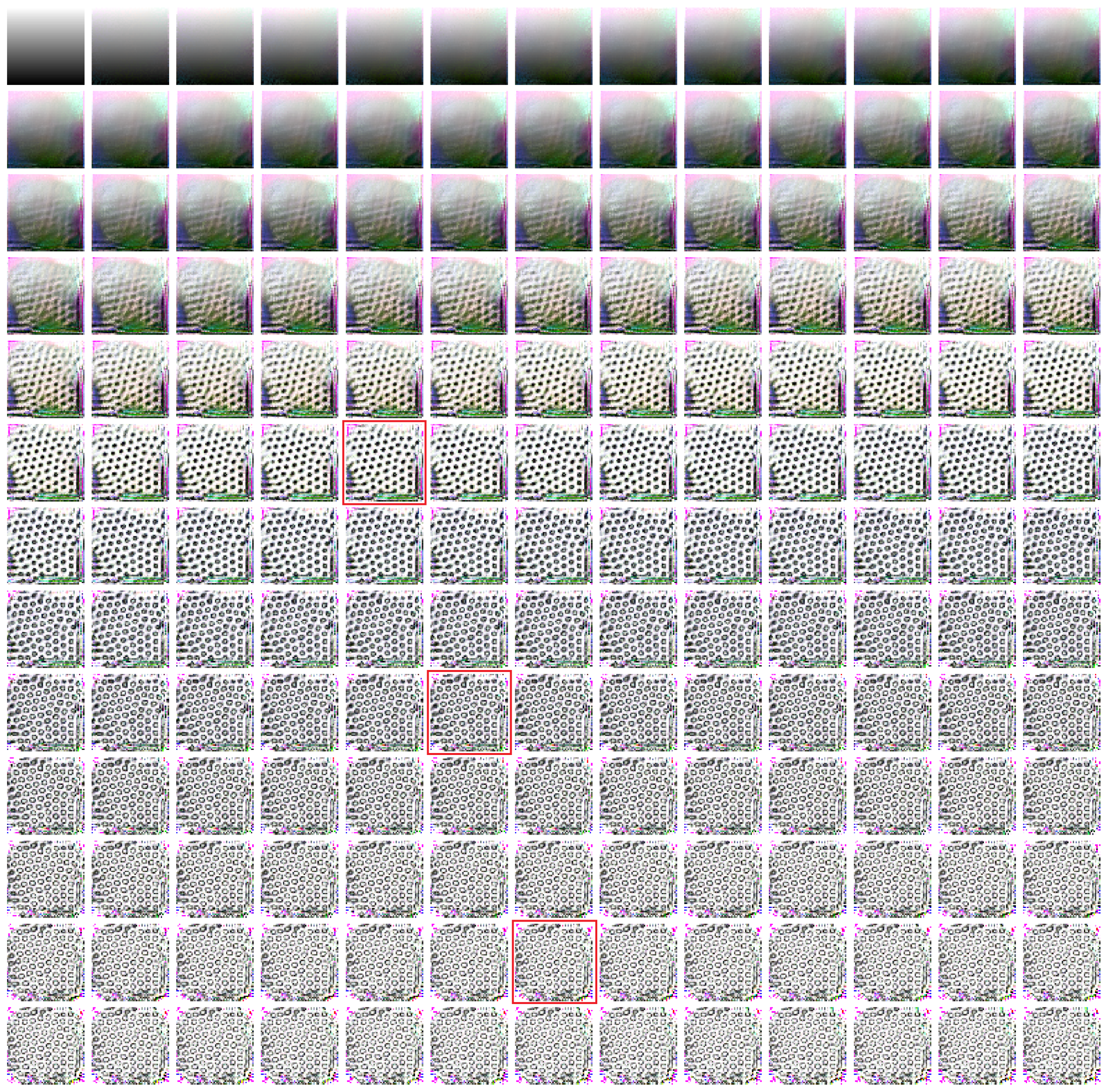
The losses associated with various model configurations are depicted in Figure 2.5 for successful cases and Figure 2.6 for unsuccessful ones. Figure 2.5 illustrates the configurations where the model successfully reduced the first-stage loss below the threshold (marked by a red dotted line). The green line indicates the lowest loss achieved across all models when growing all three stages.

To vividly showcase the intricate and captivating progression of how the model evolves the leopard pattern, we have incorporated a detailed grid plot that traces the step-by-step evolution under one particular set of settings - employing 7 channels and initiating with the 'gradient' configuration. The grid plot can be found in Figure 2.4.

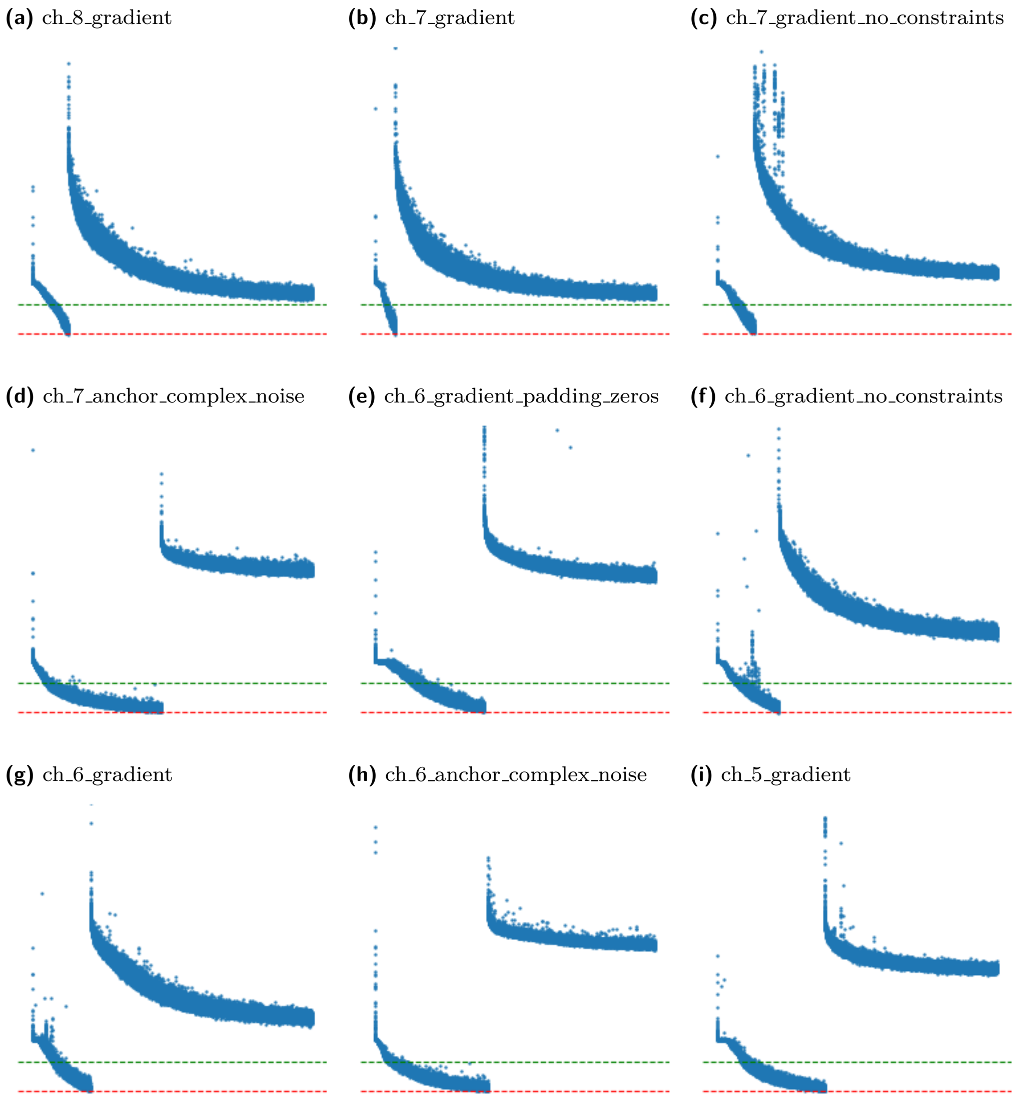
Constrained vs. Unconstrained Models

Having gained insights into how the model responds to various parameters and initial configurations, the focus shifted to identifying parameter settings that produce minimalistic and accurate solutions, capable of generating all the patterns well. Our initial experiments discovered the 'gradient' initial configuration as consistently yielding lower losses. Additionally, it was observed that using 7 channels provided favorable results, whereas employing 6 channels only occasionally succeeded.

Armed with this knowledge, a subsequent experiment was run in which the number of channels was fixed at 7, and the initial configuration was set to 'gradient'. In this experiment, we conducted five separate runs of the model with the filters constrained and another five without constraints. The losses from each set of runs were then averaged to show the overall performance. Figure 2.7 depicts the results. It is noteworthy that the plot starts at the 10,000th iteration; this is because it was only after this point that all five runs from both sets surpassed the threshold, and hence began training on all three patterns.



■ **Figure 2.4** The step-by-step evolution of our model employing 7 channels and beginning with the 'gradient' configuration. Each image in the grid plot represents a distinct state of the leopard pattern formation. Red frames encircle the images where loss measurements were taken.



(a) ch_8_gradient **(b)** ch_7_gradient **(c)** ch_7_gradient_no_constraints

(d) ch_7_anchor_complex_noise **(e)** ch_6_gradient_padding_zeros **(f)** ch_6_gradient_no_constraints

(g) ch_6_gradient **(h)** ch_6_anchor_complex_noise **(i)** ch_5_gradient

■ **Figure 2.5** The losses for the successful models with different parameters are displayed in the plots. The y-axis represents the loss, while the x-axis corresponds to the number of iterations, capped at 40,000. The y-axis range remains consistent across all images. A red dotted line demarcates the threshold for transitioning to the second stage of training. The green line signifies the lowest loss attained by any model during the development of all three stages. The title of the images indicates which parameters were used in that particular run. For instance, ch_7_gradient_no_constraints implies that the model utilized 7 channels, was initialized with the gradient configuration, and did not impose any constraints on the convolutional filters. In cases where no_constraints is not part of the title, it indicates that constraints were enforced during the model's training.

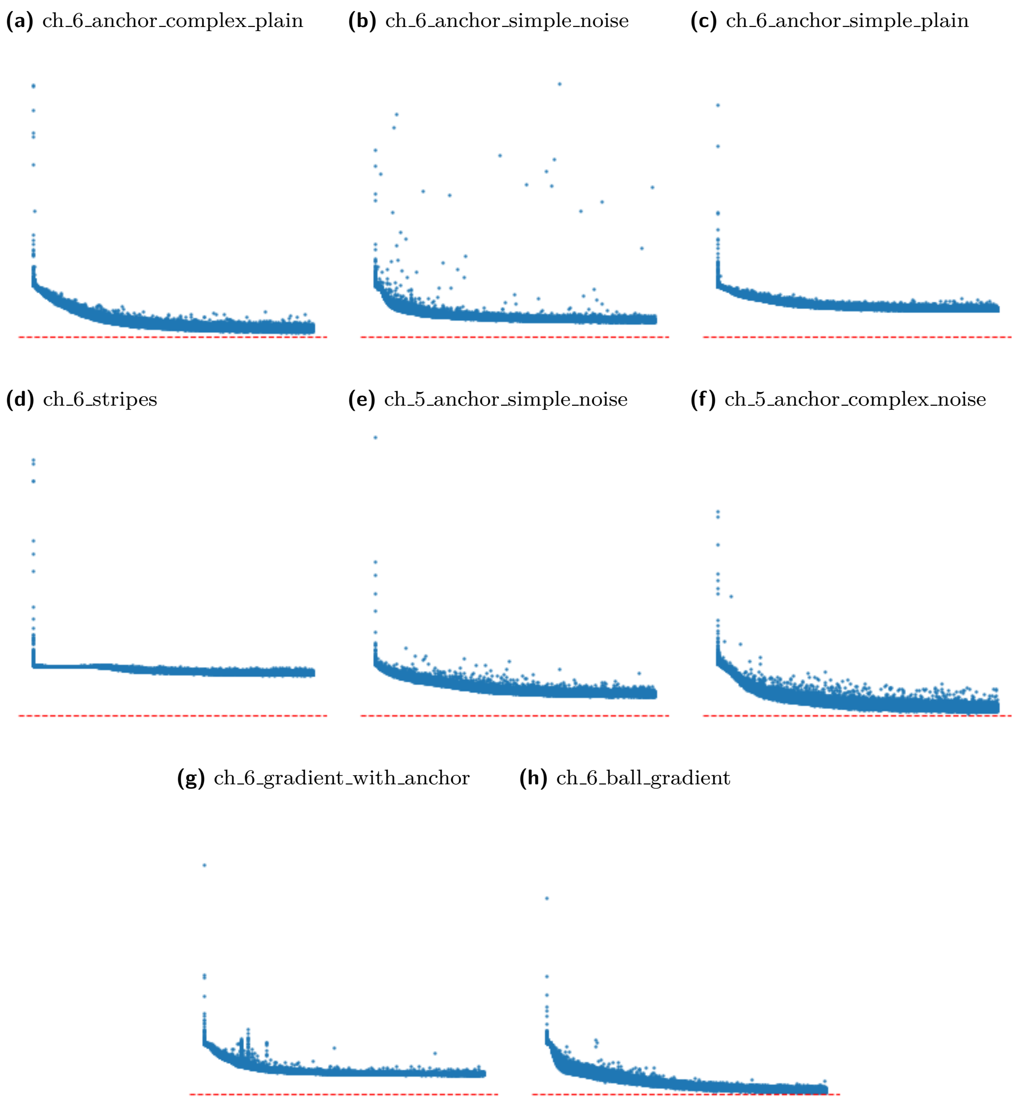
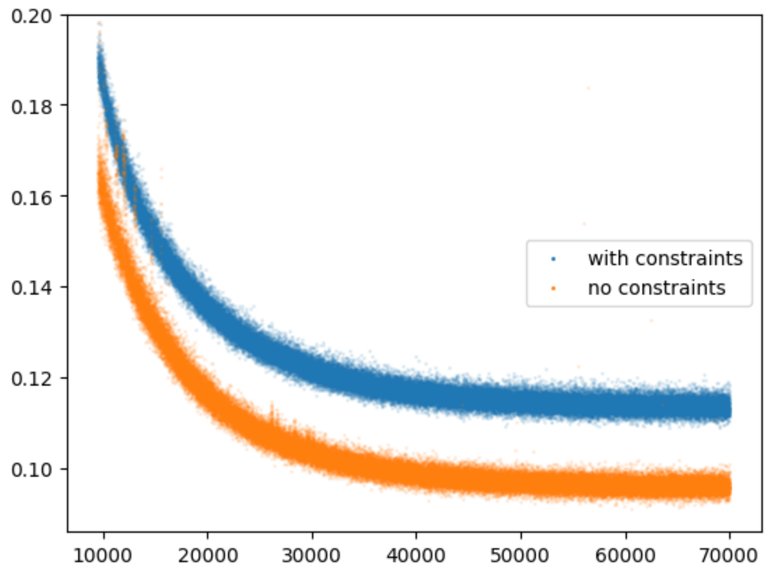


Figure 2.6 The losses for the unsuccessful models with different parameters are displayed in the plots. The y-axis represents the loss, while the x-axis corresponds to the number of iterations, capped at 40,000. The y-axis range remains consistent across all images. A red dotted line demarcates the threshold for transitioning to the second stage of training, which was never reached by any of these models. The title of the images indicates which parameters were used in that particular run.



■ **Figure 2.7** The graph depicts the average loss over time for two variants of the model: one with physics-based constraints (blue line) and another without constraints (orange line). Each variant was executed five times, and the losses were averaged. The x-axis represents the iteration count, while the y-axis denotes the average loss.

2.4 Leopard Patterns with Known Configurations

As previously mentioned, the objective in deriving the governing PDEs was to ensure that our model remained as subtle as possible. However, the initial experiments revealed that this task is exceedingly challenging when the correct initial configuration is unknown. Consequently, the following experiment was designed to bypass the need to evolve from an unidentified configuration. Instead, the model was programmed to initiate directly from the first stage (spots). We hypothesized that starting from a known initial configuration, which potentially aligns with what occurs in nature, would facilitate the process, possibly requiring fewer channels since the complexity associated with evolving from an unknown configuration is eliminated.

In this experiment, the threshold element was removed. The model was trained with the objective of transitioning from a spots pattern to a rings pattern, ultimately culminating in the rosettes pattern. We allocated 40 steps for the model to progress from one stage to the next. Drawing parallels with the previous experiment, we also conducted a variation where convolutional kernels were unconstrained, permitting the learnable filters free adaptability. The approach to edge treatment was less critical here, as padding (the added white pixels to the sides) effectively signaled the model to recognize edges, similar to zero padding. However, we retained the 'circular' edge treatment to allow the model an additional degree of flexibility.

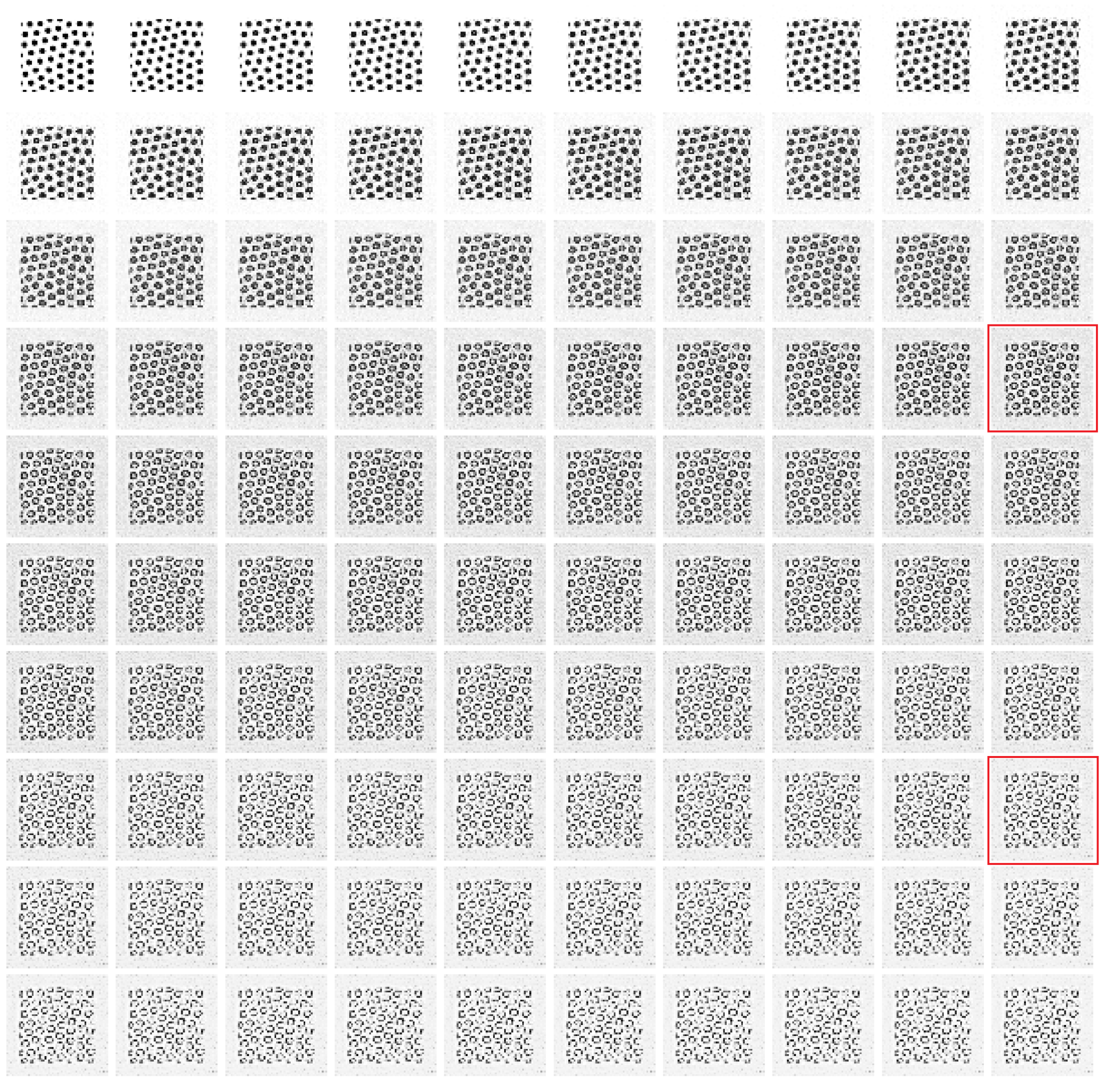
Furthermore, in this experiment, the loss was solely measured on the first channel since there were not enough channels to represent RGBA channels. The remaining hidden channels were all set to the values of the first visible channel.

We examined three different channel configurations – 1, 2, and 3 channels, with two variations for each – one incorporating physical constraints, and another without such limitations. This resulted in a total of six experimental runs. The outcomes of this investigation are illustrated in Figure 2.9.

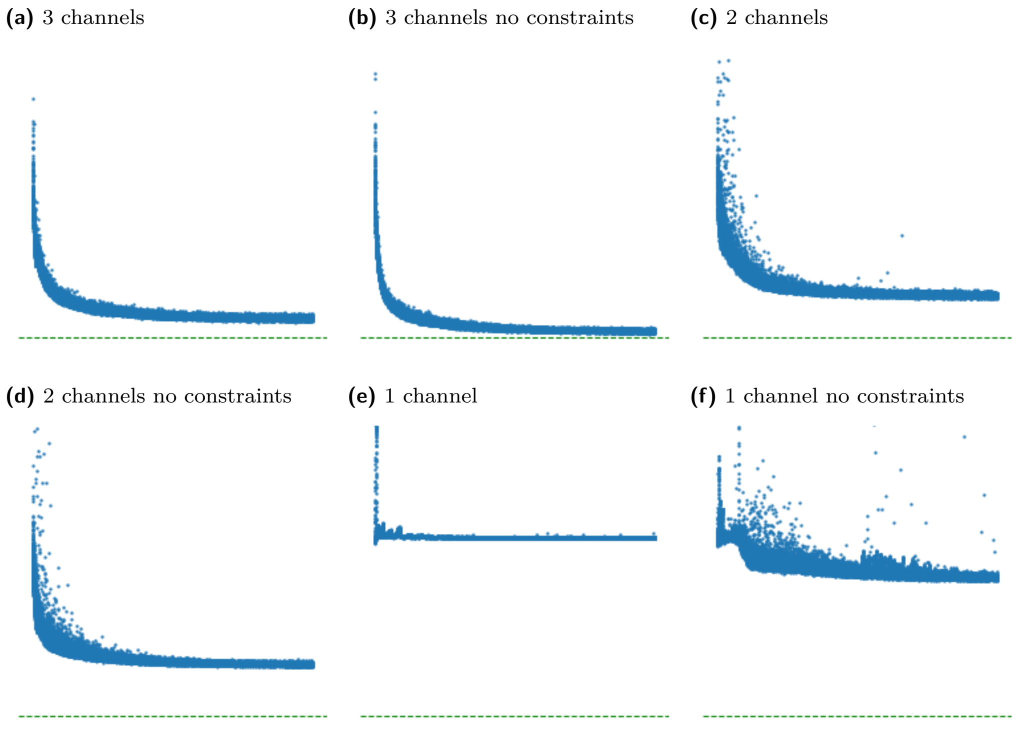
To provide an insightful and visually engaging representation of the model's intricate evolution in creating the leopard pattern in this second experiment, a detailed grid plot that traces each step of the model's evolution has also been included. It can be viewed in Figure 2.8.

Constrained vs. Unconstrained Models

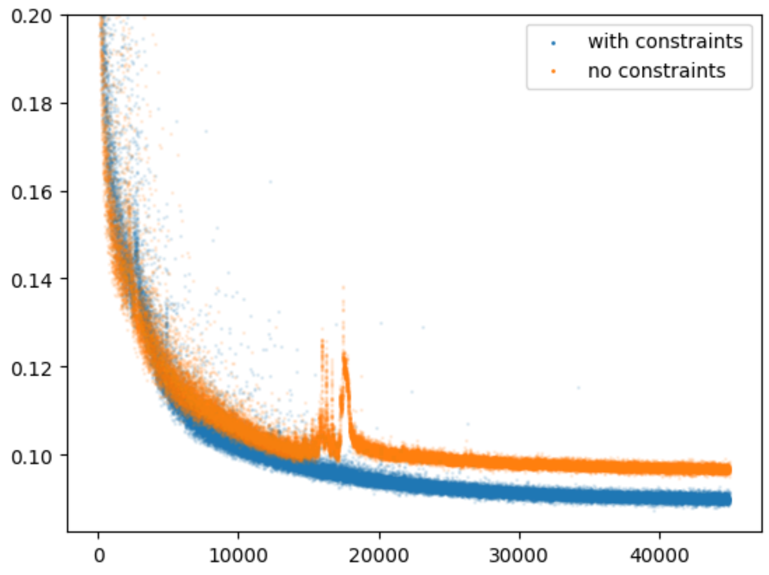
As with the initial experiment, it was imperative to draw a comparison between the physics-informed models and their unconstrained counterparts. Like in the previous experiment, five runs for the model with physical constraints were conducted and another five for the unconstrained version. For this set of experiments, the number of channels was fixed at 2. Subsequently, the losses for both variants were aggregated and averaged across the five runs. Figure 2.10 visually presents the comparative results.



■ **Figure 2.8** The step-by-step evolution of our model in the second experiment, employing 2 channels. Each image in the grid plot represents a distinct state of the leopard pattern formation. Red frames encircle the images where loss measurements were taken.



■ **Figure 2.9** The losses for the models with different parameters are displayed in the plots. Here, the model evolved from the spots pattern to the rings pattern, ending in the rosettes pattern. The y-axis represents the loss, while the x-axis corresponds to the number of iterations, capped at 40,000. The y-axis range remains consistent across all images. The green line signifies the lowest loss attained by any model during the development.



■ **Figure 2.10** The average loss over time for two variants of the model is depicted: one with physics-based constraints (blue line) and another without constraints (orange line). Each variant was executed five times, and the losses were averaged. The x-axis represents the iteration count, while the y-axis denotes the average loss.

2.5 Deriving the Governing Equations

The final conducted experiment involved modifying the model that evolves the leopard patterns, with the aim of deciphering the governing equations underlying the process. The interrelations between the differential operators and combinations of the channels (or more aptly termed 'substances') in the initial model were highly intricate, and not conducive to extracting the underlying PDEs. Nonetheless, the second experiment presented a positive aspect by utilizing only two channels, which is aligned with the two substances typically found in the original governing equations. Therefore, the objective of this experiment was to simplify the architecture to facilitate the extraction of the governing equations.

Adjusted Architecture

To accomplish this goal, alterations were made to the architecture. The learnable filters component was retained, but the dense layer consisting of 128 units was eliminated alongside the ReLU activations to enhance the transparency of the process. Together with the convolutions, the values of the channels at specific positions were collected. These values were then subjected to polynomial combinations up to the third degree. Mathematically, the set of terms generated encompassed:

$$\{c_1, \ldots, c_C\} \cup \{c_i^k \cdot c_j^\ell \mid i \neq j \wedge i, j \in \{1, \ldots, C\} \wedge k, \ell \in \{1, 2, 3\}\}$$

This essentially accounts for polynomial combinations between the channels up to the third degree plus their identities. Upon formation of these terms, they were passed through a dense layer that condensed the input values into a vector of size C. This gave each channel the autonomy to select which combinations it would employ in its equation.

Consequently, two vectors of size C are derived: one from the convolution component, and another from the polynomial combinations. These vectors are merged, yielding a stacked vector wherein each channel is associated with two adjacent values. Finally, for each channel, a dense layer of size 1 combines the two values - one representing the differential operators and the other the nonlinear combinations among the substances. This yields the update vector, encapsulating the change in values at a particular position.

A comprehensive illustration of the modified model's architecture is provided in Figure 2.11.

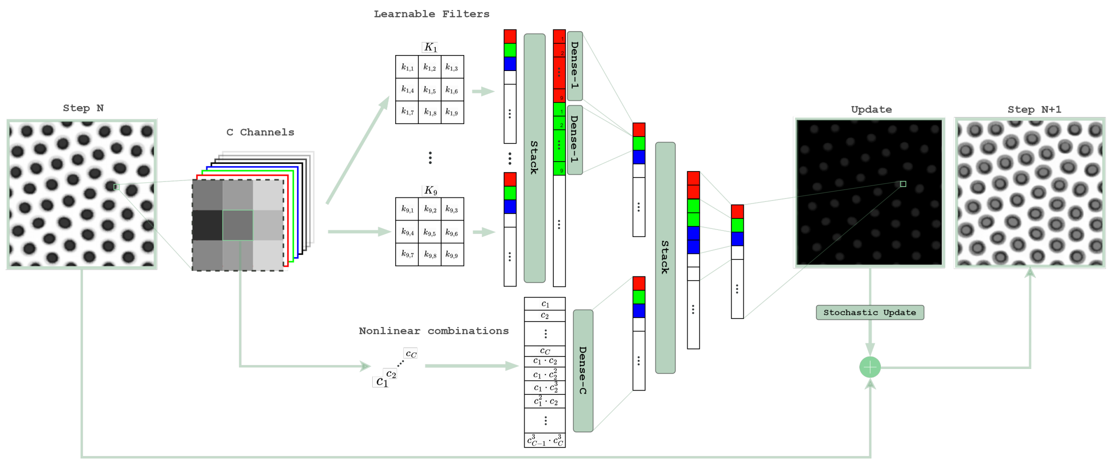


Figure 2.11 The complete description of the adjusted model for deriving the governing equations developed within this work, capturing a single iteration step from the image at step N to the image at step $N+1$.

Another distinction between this version and the previous one is the imposition of an L1 loss on the dense layer associated with the 'nonlinear combinations' part, while the L1 loss on the output of the convolutional layers is retained. This time, the total loss was computed with a much more pronounced emphasis on the L1 losses to strongly encourage each channel to selectively adopt the differential operators or combinations essential to its role. The loss was computed using the following equation:

$$loss_{total} = 1 \cdot loss_{data} + 100 \cdot loss_{filters} + 1 \cdot loss_{L1_f} + 1 \cdot loss_{L1_c}$$

In this equation, $loss_{total}$ represents the composite loss utilized during the backpropagation step. The term $loss_{data}$ reflects the pixel-wise mean squared error between the model's output and the corresponding target patterns for each stage involved in training. The term $loss_{filters}$ denotes the loss computed on the constrained filters. Additionally, $loss_{L1_f}$ signifies the L1 regularization term imposed on the dense layer that corresponds to the convolutional outputs, whereas $loss_{L1_c}$ represents the L1 loss applied to the combinations.

This formulation of the loss function, with its emphasized L1 terms, is instrumental in ensuring that the channels make the correct and minimalistic choices in terms of differential operators and combinations, thereby approximating the governing dynamics more effectively.

During the development phase of this modified model, we initially encountered a challenge wherein the model occasionally produced 'NaN' (Not-a-Number) values as output. The root cause of this issue was traced back to the freedom accorded to the cells, which enabled them to venture outside the prescribed range of 0-1 during the step-by-step development. On certain occasions, this behavior led to overflow, which in turn resulted in NaN values contaminating the gradients, causing the entire model to malfunction.

To overcome this problem, we integrated a safeguard into the model by incorporating a code that clips the values to fall within the range of 0-1 after each update. This seemingly simple but effective modification ensures that the values remain within the desired range, and, as a result, the model operates seamlessly without experiencing any NaN-related disruptions.

Training and Simulation

In this final experiment, the padding was eliminated and the edge pixels were assigned a value of 1 (white) during the convolution operations. Additionally, as the focus transitions from a machine learning perspective to a biological one, it is more appropriate to refer to the two channels we are working with as 'substances'.

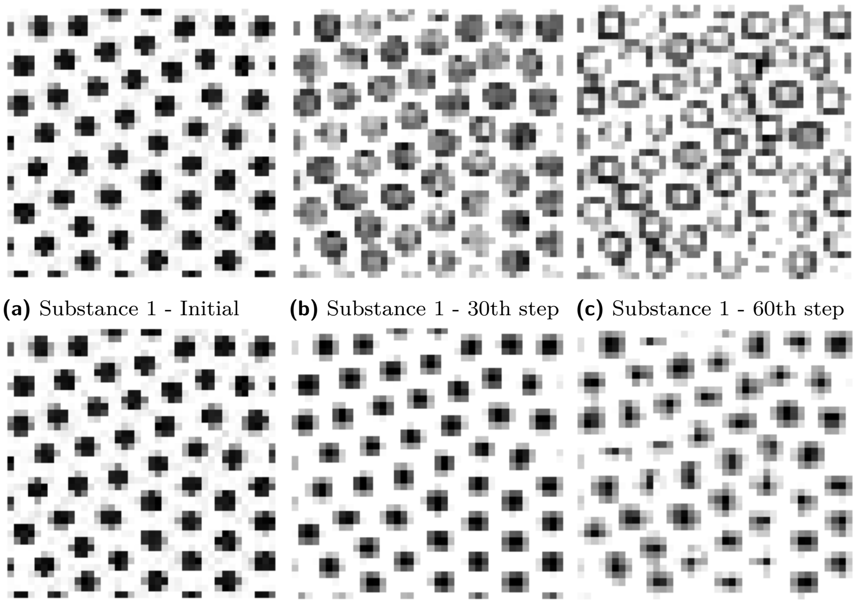
Our reconfigured architecture was subsequently trained to evolve the leopard pattern starting from the spots pattern. To encourage the model to sustain the final pattern for an extended period, the number of steps allotted for transitioning to each subsequent stage was randomized between 29 and 31 for each iteration.

The outcomes are depicted in Figure 2.12. This time, the development of both substances is presented.

Extracting Governing Equations

With our trained model in hand, we are now fully equipped to retrieve the governing equations. In particular, within the convolution section, each channel is associated with 9 weights, making for a total of 9 times the number of channels. These weights serve as indicators of the filters that are engaged and critically, due to the integration of physical dynamics, they also signify the degree to which specific differential operators are implicated in the system's evolution.

Analogously, in the combinations component, we have an array of weights that is a product of the number of channels and the number of combinations. Each weight in this array provides insights into the involvement of a particular combination of substances in the governing equations.



(a) Substance 1 - Initial **(b)** Substance 1 - 30th step **(c)** Substance 1 - 60th step

(d) Substance 2 - Initial **(e)** Substance 2 - 30th step **(f)** Substance 2 - 60th step

■ **Figure 2.12** The evolution of leopard coat pattern presented by our physics-informed NCA model that allows to extract the governing equations of the process.

In order to synthesize these elements for each channel (or substance), two final weights are utilized. These drive the contribution of the differential component and the combination component, consolidating them into a cohesive framework.

Mathematically, our aim is to derive the equations for the rate of change of both channels (substances) in time. Let u be the first channel and v the other one. According to our architecture, the emerged reaction-diffusion system can be described as follows:

$$\begin{aligned} \frac{\partial u}{\partial t} &= \alpha_1 \cdot \mathcal{D}_u + \beta_1 F_u \\ \frac{\partial v}{\partial t} &= \alpha_2 \cdot \mathcal{D}_v + \beta_2 F_v \end{aligned} \tag{2.1}$$

where $\mathcal{D}_u$ and $\mathcal{D}_v$ represent the weighted differential operators of the first and second equation respectively, F_u and F_v are the weighted combinations of the first and second equation respectively, and the parameters α_1, α_2, β_1 and β_2 are the weights of these terms learned by our model. In what follows, x denotes the horizontal direction while y denotes the vertical direction.

The raw terms for the first substance u are as follows:

$$\begin{aligned} \mathcal{D}_u \approx 0.047 \cdot u - 0.001 \cdot \frac{\partial u}{\partial x} + 0.021 \cdot \frac{\partial^2 u}{\partial x^2} \\ + 0.002 \cdot \frac{\partial u}{\partial y} + 0.003 \cdot \frac{\partial^2 u}{\partial x \partial y} - 0.001 \cdot \frac{\partial^2 u}{\partial y^2} \\ + 0.023 \cdot \frac{\partial^2 u}{\partial y^2} + 0 \cdot \frac{\partial^2 u}{\partial x^2} + 0.001 \cdot \frac{\partial^2 u}{\partial x \partial y} \end{aligned}$$

$$\begin{aligned} F_u \approx & -0.039 \cdot u + 0.123 \cdot v + 0.009 \cdot u \cdot v + 0 \cdot u \cdot v^2 \\ & -0.006 \cdot u \cdot v^3 + 0.005 \cdot u^2 \cdot v + 0 \cdot u^2 \cdot v^2 + 0 \cdot u^2 \cdot v^3 \\ & + 0 \cdot u^3 \cdot v + 0 \cdot u^3 \cdot v^2 - 0.043 \cdot u^3 \cdot v^3 \end{aligned}$$

The raw terms for the second substance v are as follows:

$$\begin{aligned}\mathcal{D}_v \approx\ & -0.007 \cdot v + 0 \cdot \frac{\partial v}{\partial x} - 0.004 \cdot \frac{\partial^2 v}{\partial x^2} \\ & + 0.001 \cdot \frac{\partial v}{\partial y} + 0 \cdot \frac{\partial^2 v}{\partial x \partial y} + 0 \cdot \frac{\partial^2 v}{\partial y^2} \\ & - 0.002 \cdot \frac{\partial^2 v}{\partial y^2} + 0 \cdot \frac{\partial^2 v}{\partial x^2} - 0.015 \cdot \frac{\partial^2 v}{\partial x \partial y}\end{aligned}$$

$$\begin{aligned}F_v \approx\ & 0 \cdot u - 0.019 \cdot v + 0.037 \cdot u \cdot v + 0 \cdot u \cdot v^2 \\ & - 0.03 \cdot u \cdot v^3 + 0.051 \cdot u^2 \cdot v + 0 \cdot u^2 \cdot v^2 + 0 \cdot u^2 \cdot v^3 \\ & + 0.011 \cdot u^3 \cdot v + 0 \cdot u^3 \cdot v^2 + 0 \cdot u^3 \cdot v^3\end{aligned}$$

The parameters α_1, α_2, β_1 and β_2 were set followingly:

$$\alpha_1 = 1.274, \quad \alpha_2 = 2.735, \quad \beta_1 = -0.894, \quad \beta_2 = 2.785$$

As can be seen, we have incorporated the terms in their unaltered state, meaning that they include certain terms that carry negligible weights and are effectively redundant. By employing a filtering process where terms with weights below a specified threshold are omitted, we consolidate the equation to a more concise form, as illustrated in the following system:

$$\begin{aligned}\mathcal{D}_u &\approx 0.047 \cdot u + 0.021 \frac{\partial^2 u}{\partial x^2} + 0.023 \frac{\partial^2 u}{\partial y^2} \\ F_u &\approx -0.039 \cdot u + 0.123 \cdot v - 0.043 \cdot u^3 \cdot v^3 \\ \mathcal{D}_v &\approx -0.007 \cdot v - 0.015 \frac{\partial^2 v}{\partial x \partial y} \\ F_v &\approx -0.019 \cdot v + 0.037 \cdot u \cdot v - 0.03 \cdot u \cdot v^3 + 0.051 \cdot u^2 \cdot v + 0.011 \cdot u^3 \cdot v\end{aligned}$$

Ultimately, the resulting governing equations have the following form:

$$\begin{aligned}\frac{\partial u}{\partial t} &\approx 1.274 \cdot \mathcal{D}_u - 0.894 \cdot F_u \\ \frac{\partial v}{\partial t} &\approx 2.735 \cdot \mathcal{D}_v + 2.785 \cdot F_v\end{aligned} \tag{2.2}$$

An interesting observation earns attention regarding the filters that were employed to approximate the differential operators. The convolutional filters converged towards the matrices depicted in Equation 2.3. Each matrix is accompanied by a title that specifies the order and direction of the derivative approximated by the filter. In Equation 2.3, s denotes the substance being approximated.

$$\begin{array}{ccc}
\frac{\partial^0 s}{\partial x^0 \partial y^0} & \frac{\partial s}{\partial x} & \frac{\partial^2 s}{\partial x^2} \\
\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ -0.5 & 0 & 0.5 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{pmatrix} \\
\frac{\partial s}{\partial y} & \frac{\partial^2 s}{\partial x \partial y} & \frac{\partial^2 s}{\partial y^2} \\
\begin{pmatrix} 0 & -0.5 & 0 \\ 0 & 0 & 0 \\ 0 & 0.5 & 0 \end{pmatrix} & \begin{pmatrix} 0.25 & 0 & -0.25 \\ 0 & 0 & 0 \\ -0.25 & 0 & 0.25 \end{pmatrix} & \begin{pmatrix} -0.5 & 1 & -0.5 \\ 0 & 0 & 0 \\ 0.5 & -1 & 0.5 \end{pmatrix} \\
\frac{\partial^2 s}{\partial y} & \frac{\partial^2 s}{\partial x^2} & \frac{\partial^2 s}{\partial x \partial y} \\
\begin{pmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{pmatrix} & \begin{pmatrix} -0.5 & 0 & 0.5 \\ 1 & 0 & -1 \\ -0.5 & 0 & 0.5 \end{pmatrix} & \begin{pmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{pmatrix}
\end{array} \tag{2.3}$$

2.6 Discussion

Our study embarked on the development of a Physics-Informed Neural Cellular Automata (PINCA) model that exhibits the capability to both simulate and extract the governing equations of an underlying physical process while requiring minimal data. The growth process of leopard patterns was employed as a case study, utilizing an exceptionally minimal dataset containing a mere three images. The investigation encompassed various alterations to the model's architecture and an assortment of experiments leading to the successful extraction of the system's governing equations.

Interpretation of Key Findings

Our research has yielded several intriguing findings. We present these findings in relation to each of the three experiments conducted:

- **Leopard Patterns with Unknown Configurations.** The initial experiment revealed the challenges associated with identifying an optimal initial configuration. Utilizing fewer than seven channels consistently failed to produce satisfactory results. Nevertheless, the 'gradient' configuration consistently emerged as the most effective. This underscores the significance of symmetry breaking in influencing the eventual outcomes, as different kinds of symmetry breaking can lead to various results.

  The second part of this experiment involved comparing a physics-informed constrained model with an unconstrained version. Our findings (see Figure 2.7) revealed that the unconstrained model converged more rapidly and achieved greater loss reduction. This can be attributed to two factors. Firstly, the initial configuration was unknown, and the one employed ('gradient') likely did not align with the configurations found in nature. Secondly, there may have been external factors that could not be captured by the reaction-diffusion system, but were adeptly approximated by the unconstrained NCA.

- **Leopard Patterns with Known Configurations.** The importance of the second experiment lied in showing how the initial configurations are vital for accurate modeling. When the model starts in a known configuration (the configuration that likely occur in nature), in this case, in the spots pattern, the process can be modeled to a high degree of accuracy using just two substances (channels). This finding is in harmony with the traditional Turing patterns theory. Additionally, the experiment revealed that the inclusion of a third channel rendered only a modest enhancement in results. Moreover, Figure 2.8 serves as visual evidence, proving the model's adeptness at evolving competently using a mere two channels.

  Comparing the physics-informed model with its non-physics counterpart (see Figure 2.10) when using two channels and starting from known configuration demonstrated the former's superiority in consistently reducing loss. This affirms that the physics-informed model, when initialized with the correct configuration, can more effectively capture the process while retaining the physical interpretations through the constrained convolutional filters.

- **Deriving the Governing Equations.** In pursuit of concise governing equations, modifications to the model architecture were necessitated. The 128-unit dense layer and ReLU were removed from the model in favor of a more parsimonious structure. While Figure 2.12 indicates that the modified model does not simulate growth patterns as convincingly as the original model, it nonetheless exhibits remarkable accuracy in replicating both stages of leopard pattern formation.

  Employing this altered parsimonious model not only facilitated the simulation of pattern growth but also enabled the extraction of the governing equations. This represents a monumental advancement, considering that the derivation of governing equations was based on a trifling set of three images. The PINCA model's efficacy is unequivocally established through this achievement.

Analysis of the Derived Equations

The extracted governing equations are presented in Equation 2.2. Interestingly, the second equation (representing the rate of change of the second substance v in time) contains a mixed second-order derivative term, indicating the presence of anisotropy. This characteristic is rarely observed in traditional Turing reaction-diffusion models.

Anisotropy in Turing patterns refers to diffusion that exhibits directional dependency. Classical Turing systems are characterized by isotropic diffusion, where substances spread uniformly in all directions. Anisotropic diffusion, on the other hand, manifests directionality in diffusion, which could be influenced by factors such as cell polarity or external constraints.

The appearance of the mixed second-order derivative term ($\frac{\partial^2 s}{\partial x \partial y}$) in our model is indicative of anisotropic behavior, as it represents the coupling of diffusion in the x and y directions. This coupling suggests that diffusion in these directions is not independent and can be viewed as an anisotropic characteristic.

Remarkably, our findings align with those of [78], who explored the role of anisotropy in pattern formation. Their model demonstrated that incorporating anisotropic diffusion could yield more complex and biologically realistic patterns.

Limitations and Future Directions

As this thesis pioneers the application of this approach to simulate and distill governing PDEs from an underlying process described through a minimalistic dataset, it paves the way for an array of future research and exploratory endeavors.

A primary avenue for enhancement lies in fine-tuning the architecture of the PINCA model. Specifically, refining the model such that terms with negligible weights are systematically pruned during training could be highly beneficial. This modification would facilitate a more precise representation of the underlying dynamics through the governing equations.

Furthermore, devising a more sophisticated strategy for identifying the optimal initial configuration warrants investigation. Our research indicates that the selection of an initial configuration is a critical factor; therefore, developing an intelligent and robust method for this selection could substantially bolster the model's efficacy.

As a natural extension, the versatility of the PINCA model can be tested by applying it to a diverse range of problems beyond leopard pattern formation. The model's capacity to generalize and adapt to different datasets and physical systems is an exciting prospect that earns thorough exploration.

2.7 Pattern Analysis

This section presents an additional experiment done in this thesis. Here, we focus on the formation of cellular clusters in the growing NCA model that was presented in [2]. A depiction of the model's architecture is available in Figure 1.6. Aligning with the original paper, we employed the emoji dataset, from which three distinct shapes for examination were selected: a lizard, a smiling face, and a fish, all of which are displayed in Figure 2.13.

Figure 2.13 The emoji from the growing NCA paper [2].

Methodology

The training phase of the growing NCA model emulated the protocol designed in the original work, employing 16 channels. We conducted training sessions for three separate models; one designed for the growth of the lizard shape, another one for the growth of the smiley face, and the final one for the growth of the fish shape. After the training, the models were let to evolve from a single-pixel seed configuration to the fully grown pattern. Subsequently, alive masking was applied, effectively filtering out the dead cells.

With the data from these living cells, we created a dataset wherein the columns ranged from channel numbers 4 to 16, giving rise to 12 columns in total, and each row represented an individual cell. In essence, each row in our dataset encapsulated the values of hidden channels for a specific living cell.

Having these datasets formed for each respective pattern, we engaged two clustering algorithms in an endeavor to unravel any hidden cellular specializations. Specifically, we employed the traditional KMeans algorithm [79] along with the DBSCAN algorithm [80], each offering unique attributes in the art of cluster analysis.

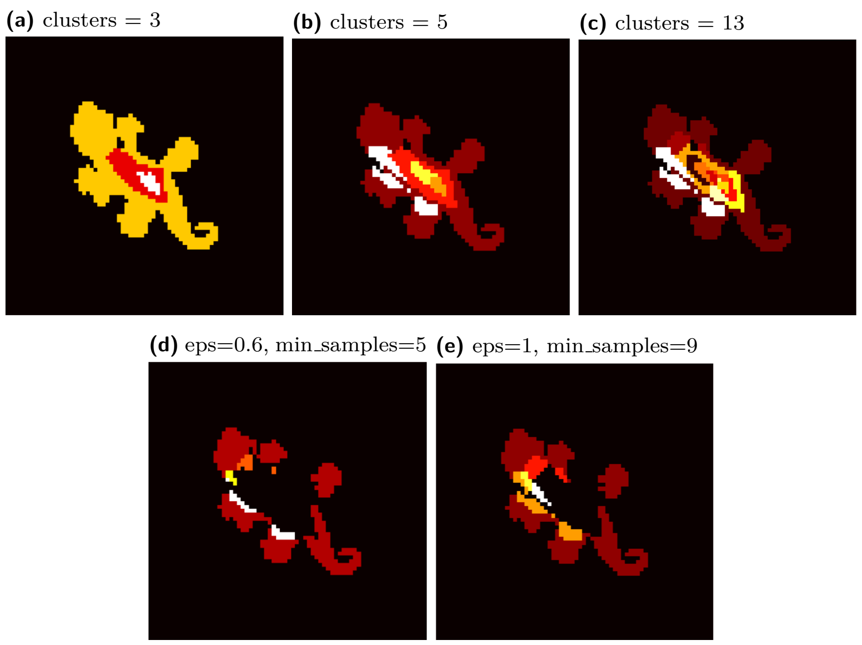
Results

We collected a series of clusterings by changing the parameters of the clustering algorithms (KMeans and DBScan). Subsequently, the ones that provided the greatest insight about the hidden clusters were selected. In Figures 2.14, 2.15 and 2.16, the results for lizard, smiley face, and fish pattern respectively can be found. The parameters employed by each clustering are included in the title of each image.

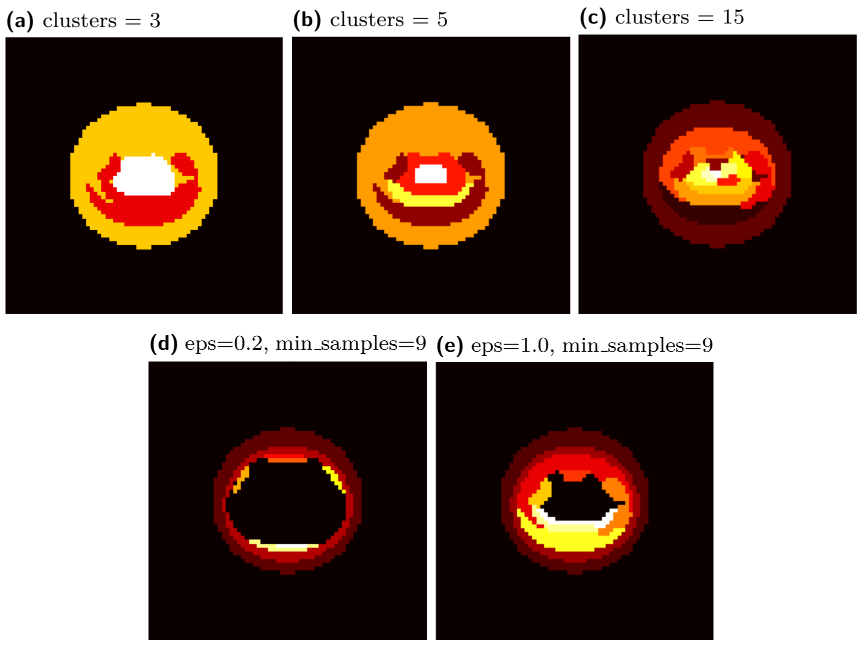
Discussion

In multicellular organisms, cellular specialization is a foundation of functionality and development. Cellular specialization refers to the phenomenon where different cells in an organism adopt distinct roles and functions, enabling complex behaviors and responses. This specialization is often manifested through differences in gene expression, morphology, and interaction with surrounding cells. In essence, cellular specialization allows the cells to work in cooperation, but with diversified responsibilities, to maintain the intricacy of a living organism.

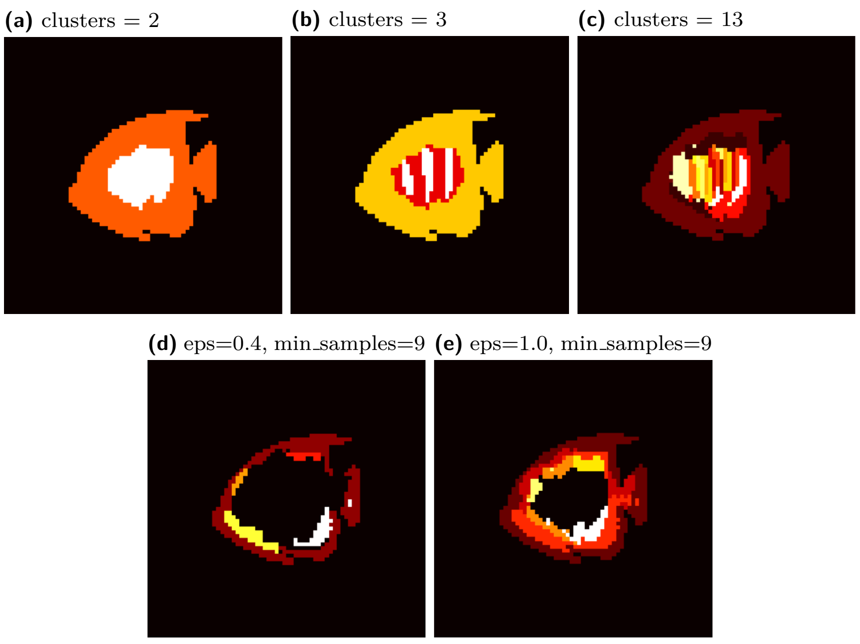
The motivation behind this auxiliary experiment was to explore if analogous behavior could be observed within NCA models - whether cells within our simulated environment adopt a form



■ **Figure 2.14** The figure showcases the clustering results for the lizard pattern. In the first row, the clusters obtained using the KMeans algorithm are presented. On the second row, the clusters formed by the DBScan algorithm are displayed.



■ **Figure 2.15** The figure showcases the clustering results for the smiley face pattern. In the first row, the clusters obtained using the KMeans algorithm are presented. On the second row, the clusters formed by the DBScan algorithm are displayed.



■ **Figure 2.16** The figure showcases the clustering results for the fish pattern. In the first row, the clusters obtained using the KMeans algorithm are presented. On the second row, the clusters formed by the DBScan algorithm are displayed.

of specialization analogous to that seen in real-world multicellular organisms. This investigation was grounded in an attempt to comprehend the internal dynamics of the model and examine if cellular behavior is dictated by relative positions, emergent properties, or any other factors.

Our findings reveal an intriguing aspect of cellular behavior within the model. Rather than adapting or altering behavior based on relative positioning or environmental factors, cells in the model seem to adopt roles or functions during the initial stages of emergence. This allocation appears to remain constant throughout the life-cycle, with cells persistently acting in accordance to their predetermined function.

This observation bears resemblance to the concept of cell specialization in biological systems, though through a distinct mechanism. Rather than complex gene regulation and environmental interactions, NCA models showcase that simple rules and emergent properties can lead to cells assuming specialized roles.

Conclusion

In this thesis, an extensive exploration of literature was carried out, encompassing Turing patterns, Cellular Automata, Neural Cellular Automata, and Physics-Informed Machine Learning. This thorough foundation was instrumental for the subsequent development of the innovative Physics-Informed Neural Cellular Automata (PINCA) model.

The PINCA architecture emerges as a groundbreaking concept, combining Neural Cellular Automata with Physics-Informed Machine Learning. It showcases a remarkable ability to simulate and extract the governing equations of underlying physical processes using minimal data. Through a series of experiments, including the simulation of leopard coat patterns, PINCA demonstrated its efficiency and precision in capturing complex dynamics.

In addition, the thesis presented evidence that Neural Cellular Automata exhibit cell specialization, similar to multicellular organisms in real nature.

In summary, PINCA represents a significant advancement in data-driven modeling by incorporating physical principles. It holds the potential to substantially impact scientific computing and suggests a direction where data and physics can be more effectively combined. This thesis provides an initial framework that can serve as a basis for future research, opening doors for further exploration and incremental advancements in the field.

Bibliography

1. WIMPENNY, Julian WT; COLASANTI, Ric. A unifying hypothesis for the structure of microbial biofilms based on cellular automaton models. *FEMS microbiology ecology*. 1997, vol. 22, no. 1, pp. 1–16.
2. MORDVINTSEV, Alexander; RANDAZZO, Ettore; NIKLASSON, Eyvind; LEVIN, Michael. Growing Neural Cellular Automata. *Distill*. 2020. Available from DOI: `10.23915/distill.00023`. https://distill.pub/2020/growing-ca.
3. BRUNTON, Steven L; PROCTOR, Joshua L; KUTZ, J Nathan. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*. 2016, vol. 113, no. 15, pp. 3932–3937.
4. KOZA, John R. Genetic Programming, On the Programming of Computers by Means of Natural Selection. A Bradford Book. *MIT Press*. 1992.
5. LONG, Zichao; LU, Yiping; MA, Xianzhong; DONG, Bin. Pde-net: Learning pdes from data. In: *International conference on machine learning*. PMLR, 2018, pp. 3208–3216.
6. LONG, Zichao; LU, Yiping; DONG, Bin. PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*. 2019, vol. 399, p. 108925.
7. TURING, Alan Mathison. The chemical basis of morphogenesis. *Bulletin of mathematical biology*. 1990, vol. 52, no. 1-2, pp. 153–197.
8. GRAY, Peter; SCOTT, Stephen K. Autocatalytic reactions in the isothermal, continuous stirred tank reactor: Oscillations and instabilities in the system A+ 2B$\rightarrow$ 3B; B$\rightarrow$ C. *Chemical Engineering Science*. 1984, vol. 39, no. 6, pp. 1087–1097.
9. KÄFER, Katharina; SCHULZ, Mirjam. *Gray-Scott Model of a Reaction-Diffusion System Pattern Formation* [`https://itp.uni-frankfurt.de/~gros/StudentProjects/Projects_2020/projekt_schulz_kaefer/`]. [N.d.]. Accessed June 22, 2023.
10. HANS MEINHARDT. *Gierer-Meinhardt Model* [`http://www.scholarpedia.org/article/Gierer-Meinhardt_model`]. [N.d.]. Accessed on 22 June 2023.
11. GIERER, Alfred; MEINHARDT, Hans. A theory of biological pattern formation. *Kybernetik*. 1972, vol. 12, pp. 30–39.
12. KONDO, Shigeru; MIURA, Takashi. Reaction-diffusion model as a framework for understanding biological pattern formation. *science*. 2010, vol. 329, no. 5999, pp. 1616–1620.
13. MURRAY, James D. Mathematical biology: I. An introduction. Interdisciplinary applied mathematics. *Mathematical Biology, Springer*. 2002, vol. 17.
14. MAINI, Philip K; BAKER, Ruth E; CHUONG, Cheng-Ming. The Turing model comes of molecular age. *Science*. 2006, vol. 314, no. 5804, pp. 1397–1398.

15. JÖNSSON, Henrik; HEISLER, Marcus G; SHAPIRO, Bruce E; MEYEROWITZ, Elliot M; MJOLSNESS, Eric. An auxin-driven polarized transport model for phyllotaxis. *Proceedings of the national academy of sciences*. 2006, vol. 103, no. 5, pp. 1633–1638.

16. ELIAS FISCHER AND STEFANIE MROZINSKI. *Display of Pattern Formation using the Gierer-Meinhardt Model* [`https://itp.uni-frankfurt.de/~gros/StudentProjects/Projects_2020/projekt_fischer_mrozinski/`]. [N.d.]. Accessed on 22 June 2023.

17. NAKAMASU, Akiko; TAKAHASHI, Go; KANBE, Akio; KONDO, Shigeru. Interactions between zebrafish pigment cells responsible for the generation of Turing patterns. *Proceedings of the National Academy of Sciences*. 2009, vol. 106, no. 21, pp. 8429–8434.

18. RASPOPOVIC, Jelena; MARCON, Luciano; RUSSO, Laura; SHARPE, James. Digit patterning is controlled by a Bmp-Sox9-Wnt Turing network modulated by morphogen gradients. *Science*. 2014, vol. 345, no. 6196, pp. 566–570.

19. KONDO, Shigeru; ASAI, Rihito. A reaction–diffusion wave on the skin of the marine angelfish Pomacanthus. *Nature*. 1995, vol. 376, pp. 765–768.

20. ALTSCHULER, Steven J; ANGENENT, Sigurd B; WANG, Yanqin; WU, Lani F. On the spontaneous emergence of cell polarity. *Nature*. 2008, vol. 454, no. 7206, pp. 886–889.

21. LIU, RT; LIAW, SS; MAINI, Philip K. Two-stage Turing model for generating pigment patterns on the leopard and the jaguar. *Physical review E*. 2006, vol. 74, no. 1, p. 011914.

22. BARRIO, RA; VAREA, C; ARAGÓN, JL; MAINI, PK. A two-dimensional numerical study of spatial pattern formation in interacting Turing systems. *Bulletin of mathematical biology*. 1999, vol. 61, no. 3, pp. 483–505.

23. SAYAMA, Hiroki. *Introduction to the modeling and analysis of complex systems*. Open SUNY Textbooks, 2015.

24. NEUMANN, J von. Theory of self-reproducing automata. *Mathematics of Computation*. 1966, vol. 21, p. 745.

25. ZUSE, Konrad. *Calculating space*. Massachusetts Institute of Technology, Project MAC Cambridge, 1970.

26. GARDNER, Martin. The Fantastic Combinations of Jhon Conway's New Solitaire Game'Life. *Sc. Am.* 1970, vol. 223, pp. 20–123.

27. *Pulsar* [ConwayLife.com Wiki]. [N.d.]. Available also from: `https://conwaylife.com/wiki/Pulsar`. Retrieved June 7, 2023.

28. COOK, Matthew et al. Universality in elementary cellular automata. *Complex systems*. 2004, vol. 15, no. 1, pp. 1–40.

29. RENDELL, Paul. Turing universality of the game of life. *Collision-based computing*. 2002, pp. 513–539.

30. ERMENTROUT, G Bard; EDELSTEIN-KESHET, Leah. Cellular automata approaches to biological modeling. *Journal of theoretical Biology*. 1993, vol. 160, no. 1, pp. 97–133.

31. WANG, Yanshu; BADEA, Tudor; NATHANS, Jeremy. Order from disorder: Self-organization in mammalian hair patterning. *Proceedings of the National Academy of Sciences*. 2006, vol. 103, no. 52, pp. 19800–19805.

32. CELADA, Franco; SEIDEN, Philip E. A computer model of cellular interactions in the immune system. *Immunology today*. 1992, vol. 13, no. 2, pp. 56–62.

33. SIETTOS, Constantinos I; RUSSO, Lucia. Mathematical modeling of infectious disease dynamics. *Virulence*. 2013, vol. 4, no. 4, pp. 295–306.

34. ADAMATZKY, Andrew; COSTELLO, Benjamin De Lacy; ASAI, Tetsuya. *Reaction-diffusion computers*. Elsevier, 2005.

35. ASAI, Tetsuya; ADAMATZKY, Andrew; AMEMIYA, Yoshihito. Towards reaction–diffusion computing devices based on minority-carrier transport in semiconductors. *Chaos, Solitons & Fractals*. 2004, vol. 20, no. 4, pp. 863–876.

36. VICHNIAC, Gérard Y. Simulating physics with cellular automata. *Physica D: Nonlinear Phenomena*. 1984, vol. 10, no. 1-2, pp. 96–116.

37. CHOPARD, Bastien; DUPUIS, Alexandre; MASSELOT, Alexandre; LUTHI, Pascal. Cellular automata and lattice Boltzmann techniques: An approach to model and simulate complex systems. *Advances in complex systems*. 2002, vol. 5, no. 02n03, pp. 103–246.

38. MANNEVILLE, Paul; BOCCARA, Nino; VICHNIAC, Gerard Y; BIDAUX, Roger. *Cellular Automata and Modeling of Complex Physical Systems: Proceedings of the Winter School, Les Houches, France, February 21–28, 1989*. Vol. 46. Springer Science & Business Media, 2012.

39. SUCCI, Sauro. *The lattice Boltzmann equation: for fluid dynamics and beyond*. Oxford university press, 2001.

40. BATTY, Michael. *Cities and complexity: understanding cities with cellular automata, agent-based models, and fractals*. The MIT press, 2007.

41. SCHELLING, Thomas C. Dynamic models of segregation. *Journal of mathematical sociology*. 1971, vol. 1, no. 2, pp. 143–186.

42. WOLFRAM, Stephen et al. *A new kind of science*. Vol. 5. Wolfram media Champaign, 2002.

43. CHAI, Chen; WONG, Yiik Diew. Fuzzy cellular automata model for signalized intersections. *Computer-Aided Civil and Infrastructure Engineering*. 2015, vol. 30, no. 12, pp. 951–964.

44. HECHT-NIELSEN, Robert. Theory of the backpropagation neural network. In: *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

45. WULFF, N; HERTZ, J A. Learning cellular automaton dynamics with neural networks. *Advances in Neural Information Processing Systems*. 1992, vol. 5.

46. HORNIK, Kurt; STINCHCOMBE, Maxwell; WHITE, Halbert. Multilayer feedforward networks are universal approximators. *Neural networks*. 1989, vol. 2, no. 5, pp. 359–366.

47. RANDAZZO, Ettore; MORDVINTSEV, Alexander; NIKLASSON, Eyvind; LEVIN, Michael; GREYDANUS, Sam. Self-classifying MNIST Digits. *Distill*. 2020. Available from DOI: `10.23915/distill.00027.002`. https://distill.pub/2020/selforg/mnist.

48. RANDAZZO, Ettore; MORDVINTSEV, Alexander; NIKLASSON, Eyvind; LEVIN, Michael. Adversarial Reprogramming of Neural Cellular Automata. *Distill*. 2021. Available from DOI: `10.23915/distill.00027.004`. https://distill.pub/selforg/2021/adversarial.

49. SINAPAYEN, Lana. Self-Replication, Spontaneous Mutations, and Exponential Genetic Drift in Neural Cellular Automata. *arXiv preprint arXiv:2305.13043*. 2023.

50. NIKLASSON, Eyvind; MORDVINTSEV, Alexander; RANDAZZO, Ettore; LEVIN, Michael. Self-Organising Textures. *Distill*. 2021. Available from DOI: `10.23915/distill.00027.003`. https://distill.pub/selforg/2021/textures.

51. VARIENGIEN, Alexandre; NICHELE, Stefano; GLOVER, Tom; PONTES-FILHO, Sidney. Towards self-organized control: Using neural cellular automata to robustly control a cart-pole agent. *arXiv preprint arXiv:2106.15240*. 2021.

52. MNIH, Volodymyr; KAVUKCUOGLU, Koray; SILVER, David; RUSU, Andrei A; VENESS, Joel; BELLEMARE, Marc G; GRAVES, Alex; RIEDMILLER, Martin; FIDJELAND, Andreas K; OSTROVSKI, Georg, et al. Human-level control through deep reinforcement learning. *nature*. 2015, vol. 518, no. 7540, pp. 529–533.

53. NAJARRO, Elias; SUDHAKARAN, Shyam; GLANOIS, Claire; RISI, Sebastian. HyperNCA: Growing developmental networks with neural cellular automata. *arXiv preprint arXiv:2204.11674*. 2022.

54. TESFALDET, Mattie; NOWROUZEZAHRAI, Derek; PAL, Chris. Attention-based Neural Cellular Automata. *Advances in Neural Information Processing Systems*. 2022, vol. 35, pp. 8174–8186.

55. DOSOVITSKIY, Alexey; BEYER, Lucas; KOLESNIKOV, Alexander; WEISSENBORN, Dirk; ZHAI, Xiaohua; UNTERTHINER, Thomas; DEHGHANI, Mostafa; MINDERER, Matthias; HEIGOLD, Georg; GELLY, Sylvain, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. 2020.

56. RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. U-net: Convolutional networks for biomedical image segmentation. In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015, pp. 234–241.

57. PALM, Rasmus Berg; GONZÁLEZ-DUQUE, Miguel; SUDHAKARAN, Shyam; RISI, Sebastian. Variational neural cellular automata. *arXiv preprint arXiv:2201.12360*. 2022.

58. KINGMA, Diederik P; WELLING, Max. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*. 2013.

59. SANDLER, Mark; ZHMOGINOV, Andrey; LUO, Liangcheng; MORDVINTSEV, Alexander; RANDAZZO, Ettore, et al. Image segmentation via cellular automata. *arXiv preprint arXiv:2008.04965*. 2020.

60. GILPIN, William. Cellular automata as convolutional neural networks. *Physical Review E*. 2019, vol. 100, no. 3, p. 032402.

61. SIEGELMANN, Hava T; SONTAG, Eduardo D. On the computational power of neural nets. In: *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, pp. 440–449.

62. ZWIRN, Hervé; DELAHAYE, Jean-Paul. Unpredictability and computational irreducibility. *Irreducibility and Computational Equivalence: 10 Years After Wolfram's A New Kind of Science*. 2013, pp. 273–295.

63. HAO, Zhongkai; LIU, Songming; ZHANG, Yichi; YING, Chengyang; FENG, Yao; SU, Hang; ZHU, Jun. Physics-Informed Machine Learning: A Survey on Problems, Methods and Applications. *arXiv preprint arXiv:2211.08064*. 2022.

64. RAISSI, M; PERDIKARIS, P; KARNIADAKIS, GEM. A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* 2019, vol. 378, pp. 686–707.

65. RAISSI, Maziar; YAZDANI, Alireza; KARNIADAKIS, George Em. Hidden fluid mechanics: A Navier-Stokes informed deep learning framework for assimilating flow visualization data. *arXiv preprint arXiv:1808.04327*. 2018.

66. ZHU, Yinhao; ZABARAS, Nicholas; KOUTSOURELAKIS, Phaedon-Stelios; PERDIKARIS, Paris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*. 2019, vol. 394, pp. 56–81.

67. LUSCH, Bethany; KUTZ, J Nathan; BRUNTON, Steven L. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*. 2018, vol. 9, no. 1, p. 4950.

68. CHAMPION, Kathleen; LUSCH, Bethany; KUTZ, J Nathan; BRUNTON, Steven L. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*. 2019, vol. 116, no. 45, pp. 22445–22451.

69. SCHMIDT, Michael; LIPSON, Hod. Symbolic regression of implicit equations. In: *Genetic programming theory and practice VII*. Springer, 2009, pp. 73–85.

70. GROSSMANN, Christian; ROOS, Hans-Görg; STYNES, Martin. *Numerical treatment of partial differential equations*. Vol. 154. Springer, 2007.

71. DONG, Bin; JIANG, Qingtang; SHEN, Zuowei. Image restoration: Wavelet frame shrinkage, nonlinear evolution pdes, and beyond. *Multiscale Modeling & Simulation*. 2017, vol. 15, no. 1, pp. 606–660.

72. CAI, Jian-Feng; DONG, Bin; OSHER, Stanley; SHEN, Zuowei. Image restoration: total variation, wavelet frames, and beyond. *Journal of the American Mathematical Society*. 2012, vol. 25, no. 4, pp. 1033–1089.

73. CALDERBANK, A Robert; DAUBECHIES, Ingrid; SWELDENS, Wim; YEO, Boon-Lock. Lossless image compression using integer to integer wavelet transforms. In: *Proceedings of International Conference on Image Processing*. IEEE, 1997, vol. 1, pp. 596–599.

74. STEPHANE, Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

75. *Growing Neural Cellular Automata Repository* [`https://github.com/chenmingxiang110/Growing-Neural-Cellular-Automata`]. [N.d.]. GitHub repository.

76. PASZKE, Adam; GROSS, Sam; MASSA, Francisco; LERER, Adam; BRADBURY, James; CHANAN, Gregory; KILLEEN, Trevor; LIN, Zeming; GIMELSHEIN, Natalia; ANTIGA, Luca; DESMAISON, Alban; KOPF, Andreas; YANG, Edward; DEVITO, Zachary; RAISON, Martin; TEJANI, Alykhan; CHILAMKURTHY, Sasank; STEINER, Benoit; FANG, Lu; BAI, Junjie; CHINTALA, Soumith. *PyTorch: An Imperative Style, High-Performance Deep Learning Library* [`https://pytorch.org/`]. 2021.

77. KLUYVER, Thomas; RAGAN-KELLEY, Benjamin; PÉREZ, Fernando; GRANGER, Brian; BUSSONNIER, Matthias; FREDERIC, Jonathan; KELLEY, Kyle; HAMRICK, Jessica; GROUT, Jason; CORLAY, Sylvain; IVANOV, Paul; AVILA, Damián; ABDALLA, Safia; WILLING, Carol; TEAM, Jupyter Development. *Jupyter Notebook* [`https://jupyter.org/`]. 2016.

78. GOMENSORO MALHEIROS, Marcelo de; WALTER, Marcelo. Pattern formation through minimalist biologically inspired cellular simulation. In: *Graphics Interface*. 2017, pp. 148–155.

79. MACQUEEN, J. Classification and analysis of multivariate observations. In: *5th Berkeley Symp. Math. Statist. Probability*. University of California Los Angeles LA USA, 1967, pp. 281–297.

80. ESTER, Martin; KRIEGEL, Hans-Peter; SANDER, Jörg; XU, Xiaowei, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *kdd*. 1996, vol. 96, pp. 226–231. No. 34.