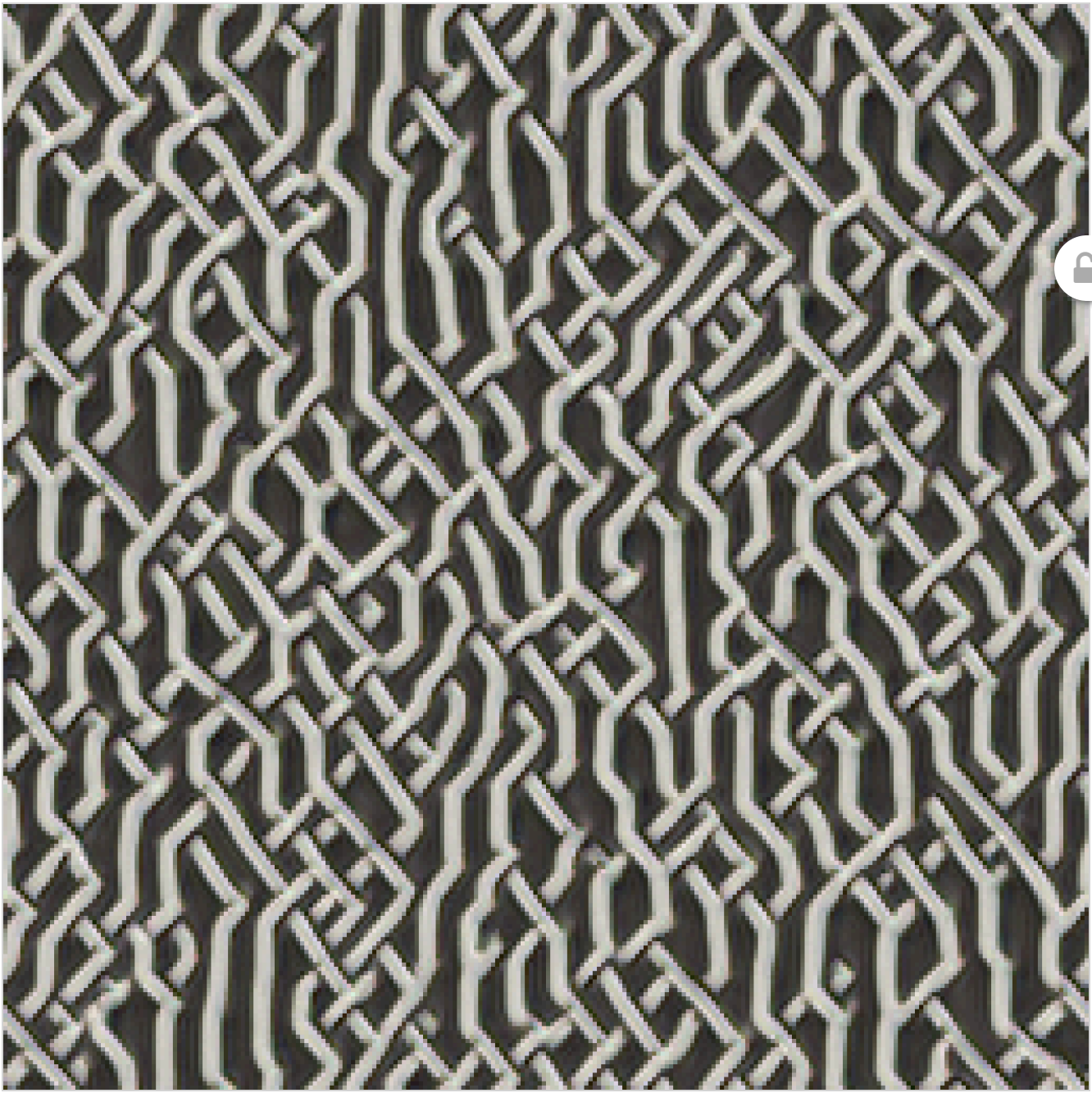
Self-Organising Textures

Neural Cellular Automata Model of Pattern Formation



Speed: 1x

Rotation: 0 deg

Cell alignment

Grid type

Textures

Inception

interlaced_0172 (DTD)

TRY IN A NOTEBOOK TRY IN A NOTEBOOK

AUTHORS
Eyvind Niklasson
Alexander Mordvintsev
Ettore Randazzo
Michael Levin

AFFILIATIONS
Google
Google
Google
Tufts

PUBLISHED
Feb. 11, 2021

DOI
10.23915/distill.00027.003

Contents

This article is part of the Differentiable Self-organizing Systems Thread, an experimental format collecting invited short articles delving into differentiable self-organizing systems, interspersed with critical commentary from several experts in adjacent fields.

← PREVIOUS ARTICLE
Self-classifying MNIST Digits

NEXT ARTICLE →
Adversarial Reprogramming of Neural Cellular Automata

Neural Cellular Automata (NCA [1]) are capable of learning a diverse set of behaviours: from generating stable, regenerating, static images [1], to segmenting images [2], to learning to "self-classify" shapes [3]. The inductive bias imposed by using cellular automata is powerful. A system of individual agents running the same learned local rule can solve surprisingly complex tasks. Moreover, individual agents, or cells, can learn to coordinate their behavior even when separated by large distances. By construction, they solve these tasks in a massively parallel and inherently degenerate [2] way. Each cell must be able to take on the role of any other cell - as a result they tend to generalize well to unseen situations.

In this work, we apply NCA to the task of texture synthesis. This task involves reproducing the general appearance of a texture template, as opposed to making pixel-perfect copies. We are going to focus on texture losses that allow for a degree of ambiguity. After training NCA models to reproduce textures, we subsequently investigate their learned behaviors and observe a few surprising effects. Starting from these investigations, we make the case that the cells learn distributed, local, algorithms.

To do this, we apply an old trick: we employ neural cellular automata as a differentiable image parameterization [4].

Patterns, textures and physical processes

A pair of Zebra. Zebra are said to have unique stripes.

Zebra stripes are an iconic texture. Ask almost anyone to identify zebra stripes in a set of images, and they will have no trouble doing so. Ask them to describe what zebra stripes look like, and they will gladly tell you that they are parallel stripes of slightly varying width, alternating in black and white. And yet, they may also tell you that no two zebra have the same set of stripes [3] . This is because evolution has programmed the cells responsible for creating the zebra pattern to generate a pattern of a certain quality, with certain characteristics, as opposed to programming them with the blueprints for an exact bitmap of the edges and locations of stripes to be moulded to the surface of the zebra's body.

Put another way, patterns and textures are ill-defined concepts. The Cambridge English Dictionary defines a pattern as "any regularly repeated arrangement, especially a design made from repeated lines, shapes, or colours on a surface". This definition falls apart rather quickly when looking at patterns and textures that impart a feeling or quality, rather than a specific repeating property. A coloured fuzzy rug, for instance, can be considered a pattern or a texture, but is composed of strands pointing in random directions with small random variations in size and color, and there is no discernable regularity to the pattern. Penrose tilings do not repeat (they are not translationally invariant), but show them to anyone and they'll describe them as a pattern or a texture. Most patterns in nature are outputs of locally interacting processes that may or may not be stochastic in nature, but are often based on fairly simple rules. There is a large body of work on models which give rise to such patterns in nature; most of it is inspired by Turing's seminal paper on morphogenesis. [5]

Such patterns are very common in developmental biology [6]. In addition to coat colors and skin pigmentation, invariant large-scale patterns, arising in spite of stochastic low-level dynamics, are a key feature of peripheral nerve networks, vascular networks, somites (blocks of tissue demarcated in embryogenesis that give rise to many organs), and segments of anatomical and genetic-level features, including whole body plans (e.g., snakes and centipedes) and appendages (such as demarcation of digit fields within the vertebrate limb [7] [8] [9]). These kinds of patterns are generated by reaction-diffusion processes, bioelectric signaling, planar polarity, and other cell-to-cell communication mechanisms [10] [11] [12] [13]. Patterns in biology are not only structural, but also physiological, as in the waves of electrical activity in the brain and the dynamics of gene regulatory networks. These gene regulatory networks, for example, can support computation sufficiently sophisticated as to be subject to Liar paradoxes [4]. Studying the emergence and control of such patterns can help us to understand not only their evolutionary origins, but also how they are recognized (either in the visual system of a second observer or in adjacent cells during regeneration) and how they can be modulated for the purposes of regenerative medicine.

As a result, when having any model learn to produce textures or patterns, we want it to learn a generative process for the pattern. We can think of such a process as a means of sampling from the distribution governing this pattern. The first hurdle is to choose an appropriate loss function, or qualitative measure of the pattern. To do so, we employ ideas from Gatys et. al [15]. NCA become the parametrization for an image which we "stylize" in the style of the target pattern. In this case, instead of restyling an existing image, we begin with a fully unconstrained setting: the output of an untrained, randomly initialized, NCA. The NCA serve as the "renderer" or "generator", and a pre-trained differentiable model serves as a distinguisher of the patterns, providing the gradient necessary for the renderer to learn to produce a pattern of a certain style.

From Turing, to Cellular Automata, to Neural Networks

NCA are well suited for generating textures. To understand why, we'll demonstrate parallels between texture generation in nature and NCA. Given these parallels, we argue that NCA are a good model class for texture generation.

PDEs

In "The Chemical Basis of Morphogenesis" [16], Alan Turing suggested that simple physical processes of reaction and diffusion, modelled by partial differential equations, lie behind pattern formation in nature, such as the aforementioned zebra stripes. Extensive work has since been done to identify PDEs modeling reaction-diffusion and evaluating their behaviour. One of the more celebrated examples is the Gray-Scott model of reaction diffusion ([17], [18]). This process has a veritable zoo of interesting behaviour, explorable by simply tuning the two parameters. We strongly encourage readers to visit this interactive atlas of the different regions of the Gray-Scott reaction diffusion model to get a sense for the extreme variety of behaviour hidden behind two simple knobs. The more adventurous can even play with a simulation locally or in the browser.

To tackle the problem of reproducing our textures, we propose a more general version of the above systems, described by a simple Partial Differential Equation (PDE) over the state space of an image.

$$\frac{\partial \mathbf{s}}{\partial t} = f(\mathbf{s}, \nabla_{\mathbf{x}}\mathbf{s}, \nabla^2_{\mathbf{x}}\mathbf{s})$$

Here, f is a function that depends on the gradient ($\nabla_{\mathbf{x}}\mathbf{s}$) and Laplacian ($\nabla^2_{\mathbf{x}}\mathbf{s}$) of the state space and determines the time evolution of this state space. s represents a k dimensional vector, whose first three components correspond to the visible RGB color channels.

Intuitively, we have defined a system where every point of the image changes with time, in a way that depends on how the image currently changes across space, with respect to its immediate neighbourhood. Readers may start to recognize the resemblance between this and another system based on immediately local interactions.

To CAs

Differential equations governing natural phenomena are usually evaluated using numerical differential equation solvers. Indeed, this is sometimes the **only** way to solve them, as many PDEs and ODEs of interest do not have closed form solutions. This is even the case for some deceptively simple ones, such as the three-body problem. Numerically solving PDEs and ODEs is a vast and well-established field. One of the biggest hammers in the metaphorical toolkit for numerically evaluating differential equations is discretization: the process of converting the variables of the system from continuous space to a discrete space, where numerical integration is tractable. When using some ODEs to model a change in a phenomena over time, for example, it makes sense to advance through time in discrete steps, possibly of variable size.

We now show that numerically integrating the aforementioned PDE is equivalent to reframing the problem as a Neural Cellular Automata, with f assuming the role of the NCA rule.

The logical approach to discretizing the space the PDE operates on is to discretize the continuous 2D image space into a 2D raster grid. Boundary conditions are of concern but we can address them by moving to a toroidal world where each dimension wraps around on itself.

Similarly to space, we choose to treat time in a discretized fashion and evaluate our NCA at fixed-sized time steps. This is equivalent to explicit Euler integration. However, here we make an important deviation from traditional PDE numerical integration methods for two reasons. First, if all cells are updated synchronously, initial conditions s_0 must vary from cell-to-cell in order to break the symmetry. Second, the physical implementation of the synchronous model would require the existence of a global clock, shared by all cells. One way to work around the former is by initializing the grid with random noise, but in the spirit of self organisation we instead choose to decouple the cell updates by asynchronously evaluating the CA. We sample a subset of all cells at each time-step to update. This introduces both asynchronicity in time (cells will sometimes operate on information from their neighbours that is several timesteps old), and asymmetry in space, solving both aforementioned issues.

Our next step towards representing a PDE with cellular automata is to discretize the gradient and Laplacian operators. For this we use the sobel operator and the 9-point variant of the discrete Laplace operator, as below.

$$\underset{Sobel_x}{\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}} \quad \underset{Sobel_y}{\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}} \quad \underset{Laplacian}{\begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix}}$$

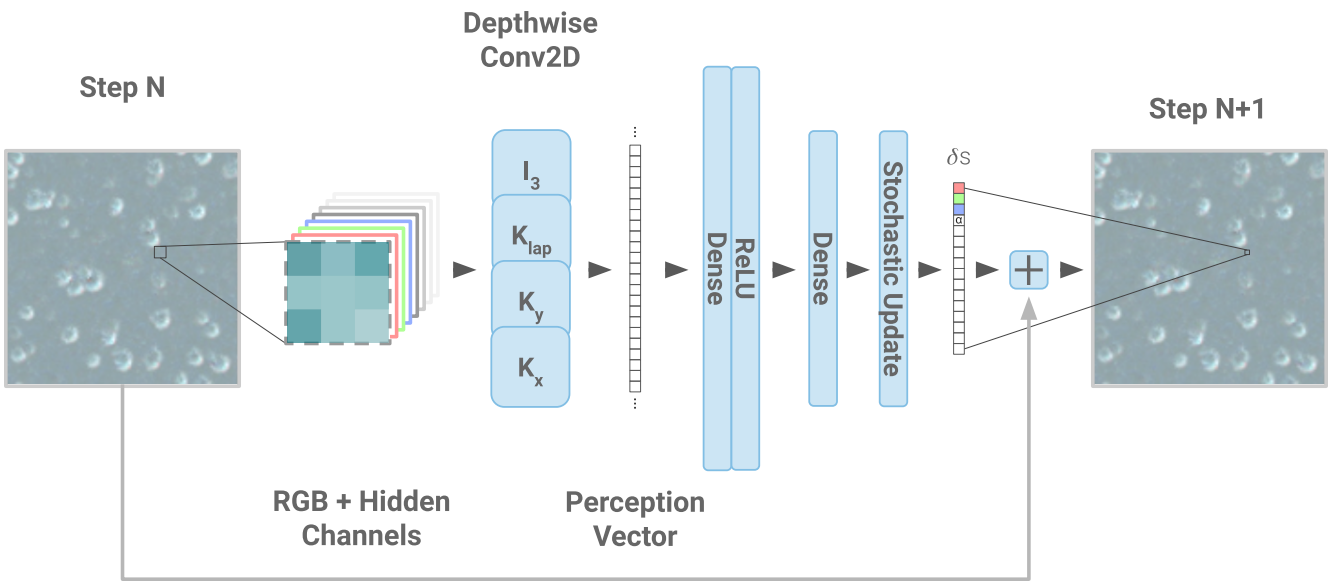
With all the pieces in place, we now have a space-discretized version of our PDE that looks very much like a Cellular Automata: the time evolution of each discrete point in the raster grid depends only on its immediate neighbours. These discrete operators allow us to formalize our PDE as a CA. To double check that this is true, simply observe that as our grid becomes very fine, and the asynchronous updates approach uniformity, the dynamics of these discrete operators will reproduce the continuous dynamics of the original PDE as we defined it.

To Neural Networks

The final step in implementing the above general PDE for texture generation is to translate it to the language of deep learning. Fortunately, all the operations involved in iteratively evaluating the generalized PDE exist as common operations in most deep learning frameworks. We provide both a Tensorflow and a minimal PyTorch implementation for reference, and refer readers to these for details on our implementation.

NCA as pattern generators

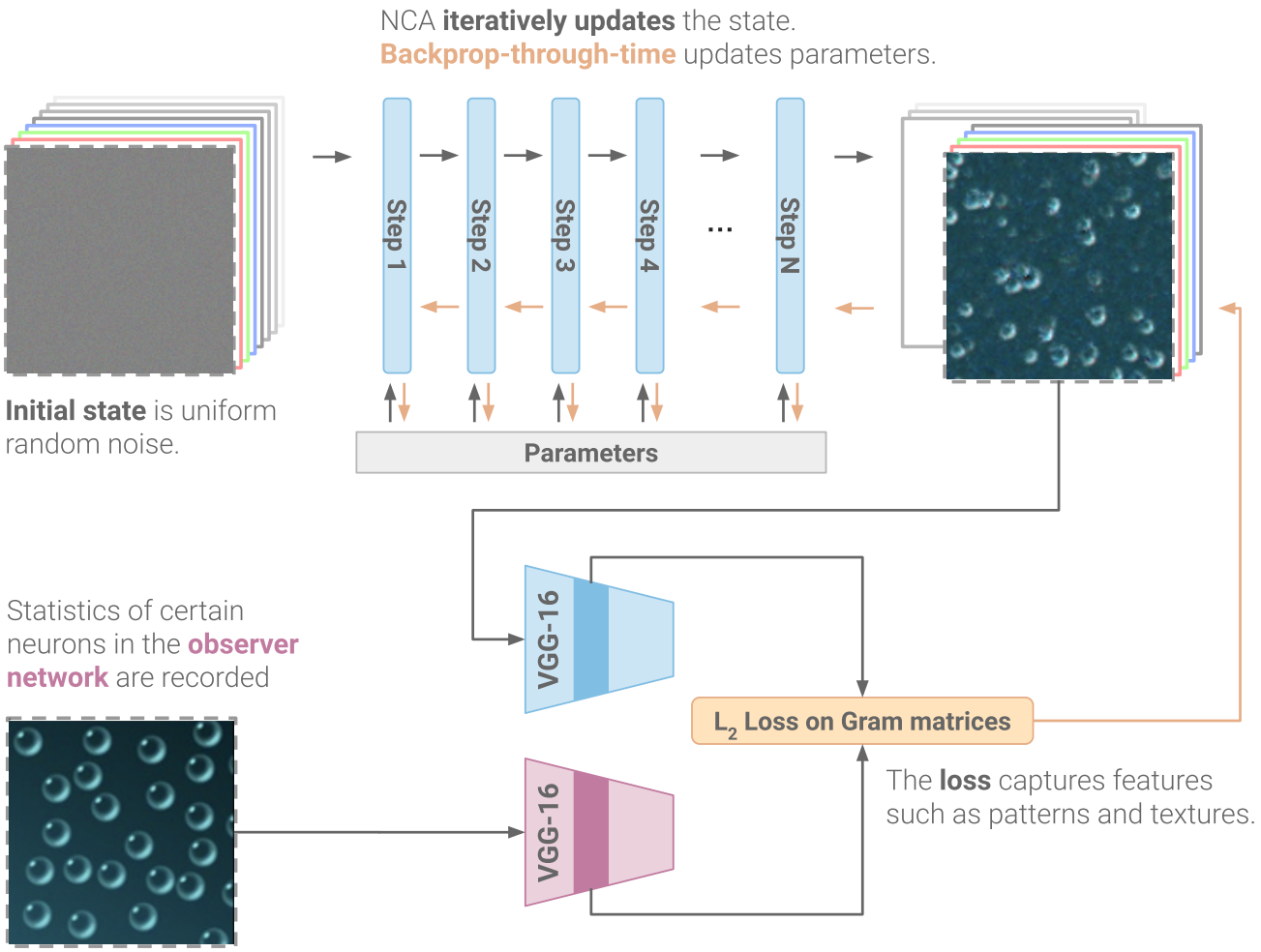
Model:



Texture NCA model.

We build on the Growing CA NCA model [1], complete with built-in quantization of weights, stochastic updates, and the batch pool mechanism to approximate long-term training. For further details on the model and motivation, we refer readers to this work.

Loss function:



Texture NCA model.

We use a well known deep convolutional network for image recognition, VGG (Visual Geometry Group Net [19]) as our differentiable discriminator of textures, for the same reasons outlined in Differentiable Parametrizations [4]. We start with a template image, $\vec{x}$, which we feed into VGG. Then we collect statistics from certain layers (block$[1...5]$_conv1) in the form of the raw activation values of the neurons in these layers. Finally, we run our NCA forward for between 32 and 64 iterations, feeding the resulting RGB image into VGG. Our loss is the L_2 distance between the gram matrix [5] of activations of these neurons with the NCA as input and their activations with the template image as input. We keep the weights of VGG frozen and use ADAM [20] to update the weights of the NCA.

Dataset:

The template images for this dataset are from the Oxford Describable Textures Dataset [21]. The aim of this dataset is to provide a benchmark for measuring the ability of vision models to recognize and categorize textures and describe textures using words. The textures were collected to match 47 “attributes” such as “bumpy” or “polka-dotted”. These 47 attributes were in turn distilled from a set of common words used to describe textures identified by Bhusan, Rao and Lohse [22].

Results:

After a few iterations of training, we see the NCA converge to a solution that at first glance looks similar to the input template, but not pixel-wise identical. The very first thing to notice is that the solution learned by the NCA is **not** time-invariant if we continue to iterate the CA. In other words it is constantly changing!

This is not completely unexpected. In *Differentiable Parametrizations*, the authors noted that the images produced when backpropagating into image space would end up different each time the algorithm was run due to the stochastic nature of the parametrizations. To work around this, they introduced some tricks to maintain **alignment** between different visualizations. In our model, we find that we attain such alignment along the temporal dimension without optimizing for it; a welcome surprise. We believe the reason is threefold. First, reaching and maintaining a static state in an NCA appears to be non-trivial in comparison to a dynamic one, so much so that in Growing CA a pool of NCA states at various iteration times had to be maintained and sampled as starting states to simulate loss being applied after a time period longer than the NCAs iteration period, to achieve a static stability. We employ the same sampling mechanism here to prevent the pattern from decaying, but in this case the loss doesn’t enforce a static fixed target; rather it guides the NCA towards any one of a number of states that minimizes the style loss. Second, we apply our loss after a random number of iterations of the NCA. This means that, at any given time step, the pattern must be in a state that minimizes the loss. Third, the stochastic updates, local communication, and quantization all limit and regularize the magnitude of updates at each iteration. This encourages changes to be small between one iteration and the next. We hypothesize that these properties combined encourage the NCA to find a solution where each iteration is **aligned** with the previous iteration. We perceive this alignment through time as motion, and as we iterate the NCA we observe it traversing a manifold of locally aligned solutions.

We now **posit** *that finding temporally aligned solutions is equivalent to finding an algorithm, or process, that generates the template pattern*, based on the aforementioned findings and qualitative observation of the NCA. We proceed to demonstrate some exciting behaviours of NCA trained on different template images.

An NCA trained to create a pattern in the style of **chequered_0121.jpg**.

Here, we see that the NCA is trained using a template image of a simple black and white grid.

We notice that:

- Initially, a non-aligned grid of black and white quadrilaterals is formed.
- As time progresses, the quadrilaterals seemingly grow or shrink in both $\vec{x}$ and $\vec{y}$ to more closely approximate squares. Quadrilaterals of both colours either emerge or disappear. Both of these behaviours seem to be an attempt to find local consistency.
- After a longer time, the grid tends to achieve perfect consistency.

Such behaviour is not entirely unlike what one would expect in a hand-engineered algorithm to produce a consistent grid with local communication. For instance, one potential hand-engineered approach would be to have cells first try and achieve local consistency, by choosing the most common colour from the cells surrounding them, then attempting to form a diamond of correct size by measuring distance to the four edges of this patch of consistent colour, and moving this boundary if it were incorrect. Distance could be measured by using a hidden channel to encode a gradient in each direction of interest, with each cell decreasing the magnitude of this channel as compared to its neighbour in that direction. A cell could then localize itself within a diamond by measuring the value of two such gradient channels. The appearance of such an algorithm would bear resemblance to the above - with patches of cells becoming either black, or white, diamonds then resizing themselves to achieve consistency.

An NCA trained to create a pattern in the style of **bubbly_0101.jpg**.

In this video, the NCA has learned to reproduce a texture based on a template of clear bubbles on a blue background. One of the most interesting behaviours we observe is that the density of the bubbles remains fairly constant. If we re-initialize the grid states, or interactively destroy states, we see a multitude of bubbles re-forming. However, as soon as two bubbles get too close to each other, one of them spontaneously collapses and disappears, ensuring a constant density of bubbles throughout the entire image. We regard these bubbles as "solitons" in the solution space of our NCA. This is a concept we will discuss and investigate at length below.

If we speed the animation up, we see that different bubbles move at different speeds, yet they never collide or touch each other. Bubbles also maintain their structure by self-correcting; a damaged bubble can re-grow.

This behaviour is remarkable because it arises spontaneously, without any external or auxiliary losses. All of these properties are learned from a combination of the template image, the information stored in the layers of VGG, and the inductive bias of the NCA. The NCA learned a rule that effectively approximates many of the properties of the bubbles in the original image. Moreover, it has learned a process that generates this pattern in a way that is robust to damage and looks realistic to humans.

0:00 / 0:14

An NCA trained to create a pattern in the style of **interlaced_0172.jpg**.

Here we see one of our favourite patterns: a simple geometric “weave”. Again, we notice the NCA seems to have learned an algorithm for producing this pattern. Each “thread” alternately joins or detaches from other threads in order to produce the final pattern. This is strikingly similar to what one would attempt to implement, were one asked to programmatically generate the above pattern. One would try to design some sort of stochastic algorithm for weaving individual threads together with other nearby threads.

0:00 / 0:14

An NCA trained to create a pattern in the style of **banded_0037.jpg**.

Here, misaligned stripe fragments travel up or down the stripe until either they merge to form a single straight stripe or a stripe shrinks and disappears. Were this to be implemented algorithmically with local communication, it is not infeasible that a similar algorithm for finding consistency among the stripes would be used.

Related work

This foray into pattern generation is by no means the first. There has been extensive work predating deep-learning, in particular suggesting deep connections between spatial patterning of anatomical structure and temporal patterning of cognitive and computational processes (e.g., reviewed in [23]). Hans Spemann, one of the heroes of classical developmental biology, said "Again and again terms have been used which point not to physical but to psychical analogies. It was meant to be more than a poetical metaphor. It was meant to my conviction that the suitable reaction of a germ fragment, endowed with diverse potencies, in an embryonic 'field'... is not a common chemical reaction, like all vital processes, are comparable, to nothing we know in such degree as to vital processes of which we have the most intimate knowledge." [24]. More recently, Grossberg quantitatively laid out important similarities between developmental patterning and computational neuroscience [25]. As briefly touched upon, the inspiration for much of the work came from Turing's work on pattern generation through local interaction, and later papers based on this principle. However, we also wish to acknowledge some works that we feel have a particular kinship with ours.

Patch sampling

Early work in pattern generation focused on texture sampling. Patches were often sampled from the original image and reconstructed or rejoined in different ways to obtain an approximation of the texture. This method has also seen recent success with the work of Gumin [26].

Deep learning

Gatys et. al's work [15], referenced throughout, has been seminal with regards to the idea that statistics of certain layers in a pre-trained network can capture textures or styles in an image. There has been extensive work building on this idea, including playing with other parametrisations for image generation [4] and optimizing the generation process [27].

Other work has focused on using a convolutional generator combined with path sampling and trained using an adversarial loss to produce textures of similar quality [28].

Interactive Evolution of Camouflage

Perhaps the most unconventional approach, with which we find kinship, is laid out in *Interactive Evolution of Camouflage* [29]. Craig Reynolds uses a texture description language, consisting of generators and operators, to parametrize a texture patch, which is presented to human viewers who have to decide which patches are the worst at "camouflaging" themselves against a chosen background texture. The population is updated in an evolutionary fashion to maximize "camouflage", resulting in a texture exhibiting the most camouflage (to human eyes) after a number of iterations. We see strong parallels with our work - instead of a texture generation language, we have an NCA parametrize the texture, and instead of human reviewers we use VGG as an evaluator of the quality of a generated pattern. We believe a fundamental difference lies in the solution space of an NCA. A texture generation language comes with a number of inductive biases and learns a deterministic mapping from coordinates to colours. Our method appears to learn more general algorithms and behaviours giving rise to the target pattern.

Two other noteworthy examples of similar work are Portilla et. al's work with the wavelet transform [30], and work by Chen et al with reaction diffusion [31].

Feature visualization

A butterfly with an "eye-spot" on the wings.

We have now explored some of the fascinating behaviours learned by the NCA when presented with a template image. What if we want to see them learn even more "unconstrained" behaviour?

Some butterflies have remarkably lifelike eyes on their wings. It's unlikely the butterflies are even aware of this incredible artwork on their own bodies. Evolution placed these there to trigger a response of fear in potential predators or to deflect attacks from them [32]. It is likely that neither the predator nor the butterfly has a concept for what an eye is or what an eye does, or even less so any theory of mind regarding the consciousness of the other, but evolution has identified a region of morphospace for this organism that exploits pattern-identifying features of predators to trick them into fearing a harmless bug instead of consuming it.

Even more remarkable is the fact that the individual cells composing the butterfly's wings can self assemble into coherent, beautiful, shapes far larger than an individual cell - indeed a cell is on the order of $1^{-5}m$ [33] while the features on the wings will grow to as large as $1^{-3}m$ [34]. The coordination required to produce these features implies self-organization over hundreds or thousands of cells to generate a coherent image of an eye that evolved simply to act as a visual stimuli for an entirely different species, because of the local nature of cell-to-cell communication. Of course, this pales in comparison to the morphogenesis that occurs in animal and plant bodies, where structures consisting of millions of cells will specialize and coordinate to generate the target morphology.

A common approach to investigating neural networks is to look at what inhibits or excites individual neurons in a network [35]. Just as neuroscientists and biologists have often treated cells and cell structures and neurons as black-box models to be investigated, measured and reverse-engineered, there is a large contemporary body of work on doing the same with neural networks. For instance the work by Boettiger [36] [37].

We can explore this idea with minimal effort by taking our pattern-generating NCA and exploring what happens if we task it to enter a state that excites a given neuron in Inception. One of the common resulting NCAs we notice is eye and eye-related shapes - such as the video below - likely as a result of having to detect various animals in ImageNet. In the same way that cells form eye patterns on the wings of butterflies to excite neurons in the brains of predators, our NCA's population of cells has learned to collaborate to produce a pattern that excites certain neurons in an external neural network.

An NCA trained to excite **mixed4a_472** in Inception.

NCA with Inception

Model:

We use a model identical to the one used for exploring pattern generation, but with a different discriminator network: Imagenet-trained Inception v1 network [38].

Loss function:

Our loss maximizes the activations of chosen neurons, when evaluated on the output of the NCA. We add an auxiliary loss to encourage the outputs of the NCA to be $\in [0, 1]$, as this is not inherently built into the model. We keep the weights of the Inception frozen and use ADAM [20] to update the weights of the NCA.

Dataset:

There is no explicit dataset for this task. Inception is trained on ImageNet. The layers and neurons we chose to excite are chosen qualitatively using OpenAI Microscope.

Results:

Similar to the pattern generation experiment, we see quick convergence and a tendency to find temporally dynamic solutions. In other words, resulting NCAs do not stay still. We also observe that the majority of the NCAs learn to produce solitons of various kinds. We discuss a few below, but encourage readers to explore them in the demo.

An NCA trained to excite **mixed4c_439** in Inception.

Solitons in the form of regular circle-like shapes with internal structure are quite commonly observed in the inception renderings. Two solitons approaching each other too closely may cause one or both of them to decay. We also observe that solitons can divide into two new solitons.

An NCA trained to excite **mixed3b_454** in Inception.

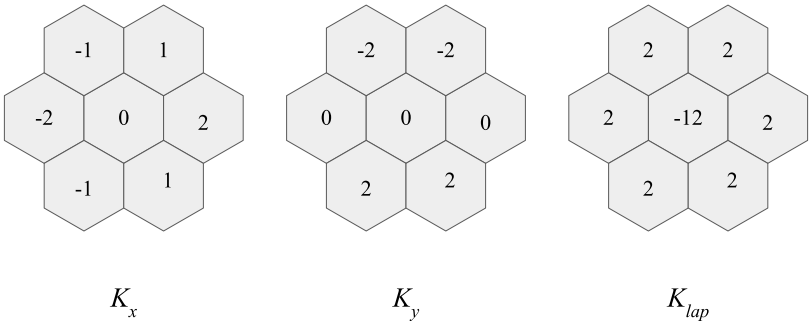
In textures that are composed of threads or lines, or in certain excitations of Inception neurons where the resulting NCA has a “thread-like” quality, the threads grow in their respective directions and will join other threads, or grow around them, as required. This behaviour is similar to the regular lines observed in the striped patterns during pattern generation.

Other interesting findings

Robustness

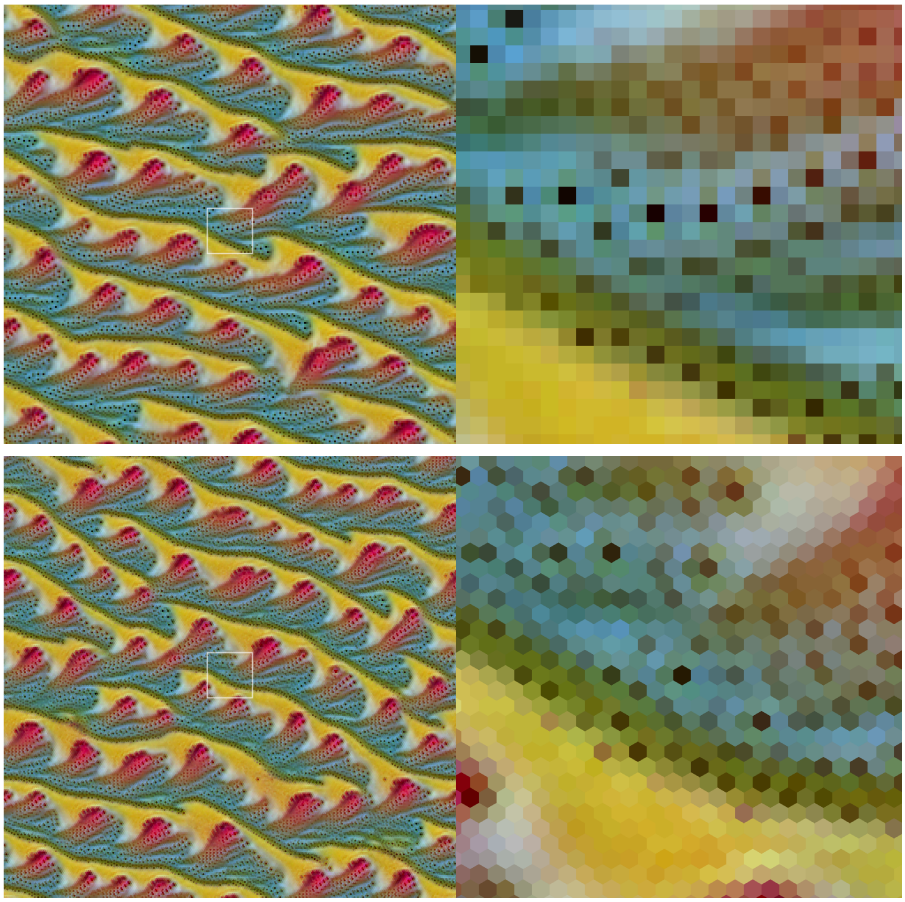
Switching manifolds

We encode local information flow within the NCA using the same fixed Laplacian and gradient filters. As luck would have it, these can be defined for most underlying manifolds, giving us a way of placing our cells on various surfaces and in various configurations without having to modify the learned model. Suppose we want our cells to live in a hexagonal world. We can redefine our kernels as follows:



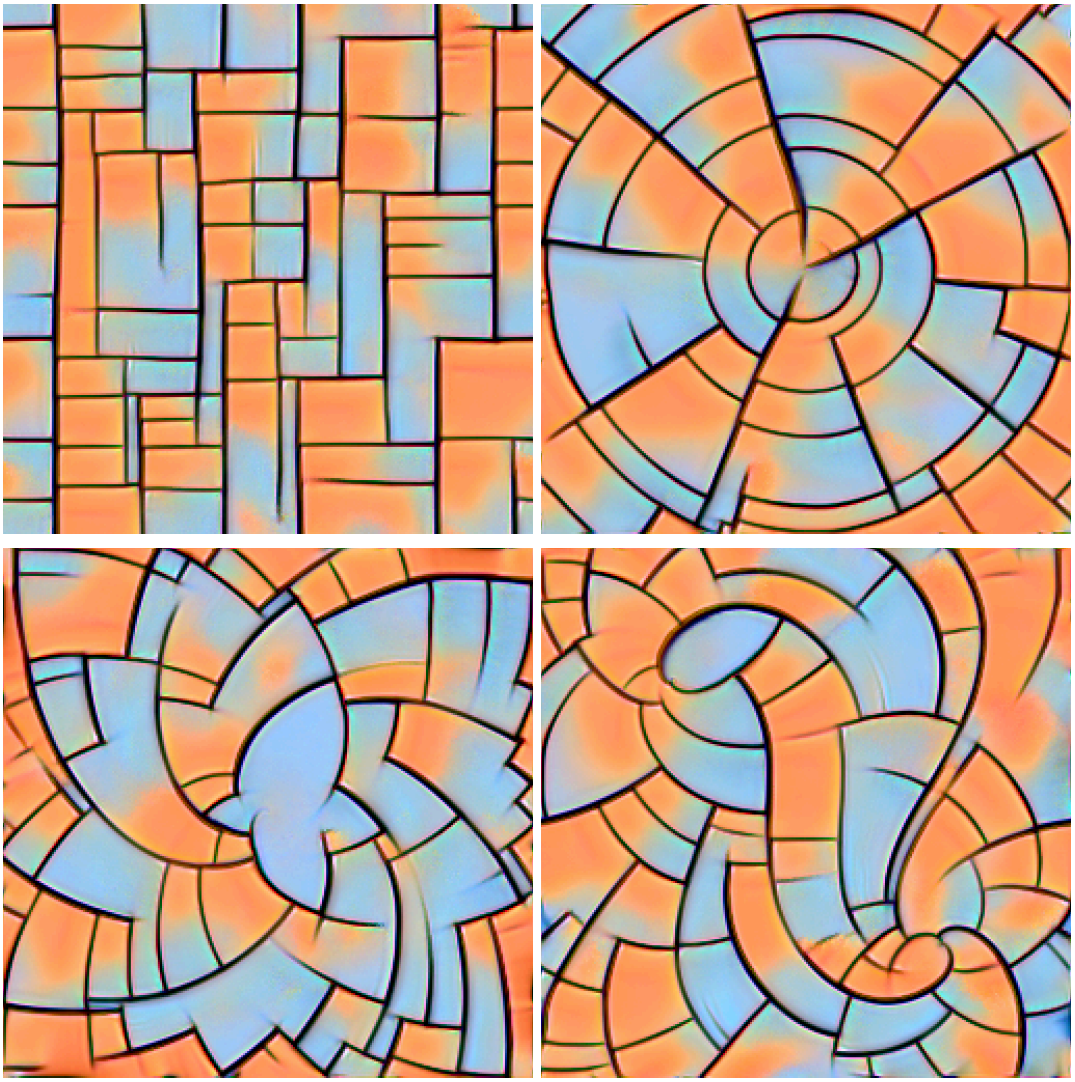
Hexagonal grid convolutional filters.

Our model, trained in a purely square environment, works out of the box on a hexagonal grid! Play with the corresponding setting in the demo to experiment with this. Zooming in allows observation of the individual hexagonal or square cells. As can be seen in the demo, the cells have no problem adjusting to a hexagonal world and producing identical patterns after a brief period of re-alignment.



The same texture evaluated on a square and hexagonal grid, respectively.

Rotation



Mondrian pattern where the cells are rotated in various directions. Note that the NCA is not re-trained - it generalises to this new rotated paradigm without issue.

In theory, the cells can be evaluated on any manifold where one can define approximations to the Sobel kernel and the Laplacian kernel. We demonstrate this in our demo by providing an aforementioned “hexagonal” world for the cells to live in. Instead of having eight equally-spaced neighbours, each cell now has six equally-spaced neighbours. We further demonstrate this versatility by rotating the Sobel and Laplacian kernels. Each cell receives an innate global orientation based on these kernels, because they are defined with respect to the coordinate system of the state. Redefining the Sobel and Laplacian kernel with a rotated coordinate system is straightforward and can even be done on a per-cell level. Such versatility is exciting because it mirrors the extreme robustness found in biological cells in nature. Cells in most tissues will generally continue to operate whatever their location, direction, or exact placement relative to their neighbours. We believe this versatility in our model could even extend to a setting where the cells are placed on a manifold at random, rather than on an ordered grid.

Time-synchronization

Two NCAs running next to each other, at different speeds, with some stochasticity in speed. They can communicate through their shared edge; the vertical boundary between them in the center of the state space.

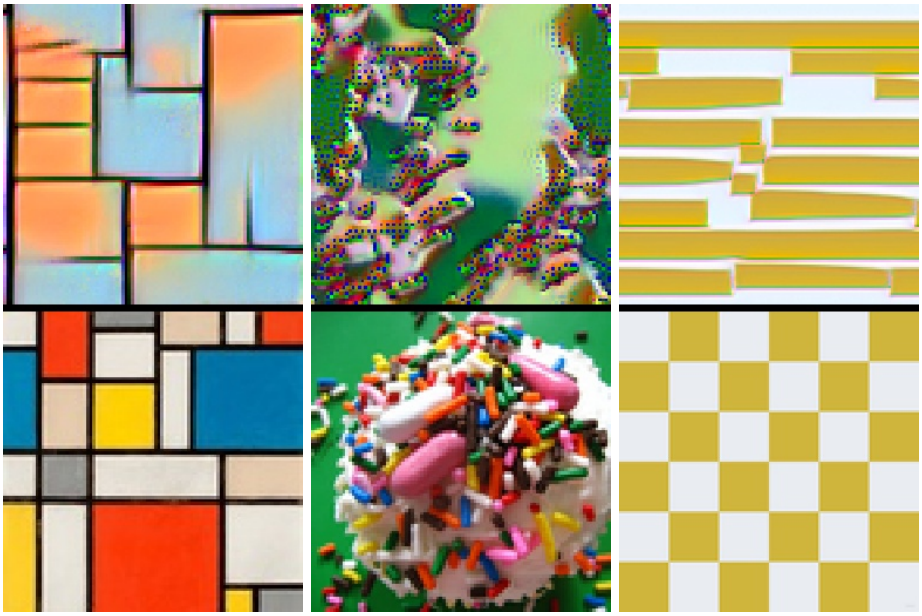
Stochastic updates teach the cells to be robust to asynchronous updates. We investigate this property by taking it to an extreme and asking *how do the cells react if two manifolds are allowed to communicate but one runs the NCA at a different speed than the other*? The result is surprisingly stable; the CA is still able to construct and maintain a consistent texture across the combined manifold. The time discrepancy between the two CAs sharing the state is far larger than anything the NCA experiences during training, showing remarkable robustness of the learned behaviour. Parallels can be drawn to organic matter self repairing, for instance a fingernail can regrow in adulthood despite the underlying finger already having fully developed; the two do not need to be sync. This result also hints at the possibility of designing distributed systems without having to engineer for a global clock, synchronization of compute units or even homogenous compute capacity.

An NCA is evaluated for a number of steps. The surrounding border of cells are then also turned into NCA cells. The cells have no difficulty communicating with the "finished" pattern and achieving consistency.

An even more drastic example of this robustness to time asynchronicity can be seen above. Here, an NCA is iterated until it achieves perfect consistency in a pattern. Then, the state space is expanded, introducing a border of new cells around the existing state. This border quickly interfaces with the existing cells and settles in a consistent pattern, with almost no perturbation to the already-converged inner state.

Failure cases

The failure modes of a complex system can teach us a great deal about its internal structure and process. Our model has many quirks and sometimes these prevent it from learning certain patterns. Below are some examples.



Three failure cases of the NCA. Bottom row shows target texture samples, top row are corresponding NCA outputs. Failure modes include incorrect colours, chequerboard artefacts, and incoherent image structure.

Some patterns are reproduced somewhat accurately in terms of structure, but not in colour, while some are the opposite. Others fail completely. It is difficult to determine whether these failure cases have their roots in the parametrization (the NCA), or in the hard-to-interpret gradient signals from VGG, or Inception. Existing work with style transfer suggests that using a loss on Gram matrices in VGG can introduce instabilities [39], that are similar to the ones we see here. We hypothesize that this effect explains the failures in reproducing colors. The structural failures, meanwhile, may be caused by the NCA parameterization, which makes it difficult for cells to establish long-distance communication with one another.

Hidden states

When biological cells communicate with each other, they do so through a multitude of available communication channels. Cells can emit or absorb different ions and proteins, sense physical motion or “stiffness” of other cells, and even emit different chemical signals to diffuse over the local substrate [40].

There are various ways to visualize communication channels in real cells. One of them is to add to cells a potential-activated dye. Doing so gives a clear picture of the voltage potential the cell is under with respect to the surrounding substrate. This technique provides useful insight into the communication patterns within groups of cells and helps scientists visualize both local and global communication over a variety of time-scales.

As luck would have it, we can do something similar with our Neural Cellular Automata. Our NCA model contains 12 channels. The first three are visible RGB channels and the rest we treat as latent channels which are visible to adjacent cells during update steps, but excluded from loss functions. Below we map the first three principle components of the hidden channels to the R,G, and B channels respectively. Hidden channels can be considered “floating,” to abuse a term from circuit theory. In other words, they are not pulled to any specific final state or intermediate state by the loss. Instead, they converge to some form of a dynamical system which assists the cell in fulfilling its objective with respect to its visible channels. There is no pre-defined assignment of different roles or meaning to different hidden channels, and there is almost certainly redundancy and correlation between different hidden channels. Such correlation may not be visible when we visualize the first three principal components in isolation. But this concern aside, the visualization yields some interesting insights anyways.

Left: RGB channels of NCA. **Right:** Intensities of top three principal components of hidden states.

An NCA trained to excite **mixed4b_70** in Inception. Notice the hidden states appear to encode information about structure. “Threads” along the major diagonal (NW - SE) appear primarily green, while those running along the anti-diagonal appear blue, indicating that these have differing internal states, despite being effectively indistinguishable in RGB space.

In the principal components of this coral-like texture, we see a pattern which is similar to the visible channels. However, the “threads” pointing in each diagonal direction have different colours - one diagonal is green and the other is a pale blue. This suggests that one of the things encoded into the hidden states is the direction of a “thread”, likely to allow cells that are inside one of these threads to keep track of which direction the thread is growing, or moving, in.

Left: RGB channels of NCA. **Right:** Intensities of top three principal components of hidden states.

An NCA trained to produce a texture based on DTD image **cheqeuered_0121**. Notice the structure of squares - with a gradient occurring inside the structure of each square, evidencing that structure is being encoded in hidden state.

The chequerboard pattern likewise lends itself to some qualitative analysis and hints at a fairly simple mechanism for maintaining the shape of squares. Each square has a clear gradient in PCA space across the diagonal, and the values this gradient traverses differ for the white and black squares. We find it likely the gradient is used to provide a local coordinate system for creating and sizing the squares.

Left: RGB channels of NCA. **Right:** Intensities of top three principal components of hidden states.

An NCA trained to excite **mixed4c_208** in Inception. The visible body of the eye is clearly demarcated in the hidden

states. There is also a “halo” which appears to modulate growth of any solitons immediately next to each other. This halo is barely visible in the RGB channels.

We find surprising insight in NCA trained on Inception as well. In this case, the structure of the eye is clearly encoded in the hidden state with the body composed primarily of one combination of principal components, and an halo, seemingly to prevent collisions of the eye solitons, composed of another set of principal components.

Analysis of these hidden states is something of a dark art; it is not always possible to draw rigorous conclusions about what is happening. We welcome future work in this direction, as we believe qualitative analysis of these behaviours will be useful for understanding more complex behaviours of CAs. We also hypothesize that it may be possible to modify or alter hidden states in order to affect the morphology and behaviour of NCA.

Conclusion

In this work, we selected texture templates and individual neurons as targets and then optimized NCA populations so as to produce similar excitations in a pre-trained neural network. This procedure yielded NCAs that could render nuanced and hypnotic textures. During our analysis, we found that these NCAs have interesting and unexpected properties. Many of the solutions for generating certain patterns in an image appear similar to the underlying model or physical behaviour producing the pattern. For example, our learned NCAs seem to have a bias for treating objects in the pattern as individual objects and letting them move freely across space. While this effect was present in many of our models, it was particularly strong in the bubble and eye models. The NCA is forced to find algorithms that can produce such a pattern with purely local interaction. This constraint seems to produce models that favor high-level consistency and robustness.

This article is part of the Differentiable Self-organizing Systems Thread, an experimental format collecting invited short articles delving into differentiable self-organizing systems, interspersed with critical commentary from several experts in adjacent fields.

← PREVIOUS ARTICLE
Self-classifying MNIST Digits

NEXT ARTICLE →
Adversarial Reprogramming of Neural Cellular Automata

Acknowledgments

We'd like to thank Sam Greydanus for especially thoughtful proofreading and giving extensive feedback throughout the article. We also extend our gratitude to the other reviewers; Maximilian Otte, Aleksandr Groznykh, and Smitty van Bodegom. Finally, we would like to acknowledge the continued support from Blaise Agüera y Arcas and Dominik Roblek, without whom this work wouldn't be possible.

Author Contributions

Research: Alexander proposed using Neural CA for texture synthesis and feature visualization and prototyped the TF and PyTorch implementations. Eyvind refined the implementation and performed most of the article experiments.

Demos: Alexander implemented the WebGL NCA engine. Eyvind implemented the demo UI.

Writing and Diagrams: Eyvind wrote most of the article text and created most of the diagrams and videos. Michael provided the biological context for the article. Alexander and Ettore contributed to the content.

An Aside: Solitons and Lenia

The motion of waves propagating through a medium can be described using the classical wave equation. The equation below defines the change of some quantity u (be it the surface height map of a body of water, the position of a vibrating string, etc.) with respect to the laplacian of u. c ends up being the propagation speed of the wave.

$$\ddot{u} = c^2 \nabla^2 u$$

One can imagine waves to come either as a single wave, or as a larger mixture of waves of different frequencies (a phenomenon referred to as a group or a packet). Physical phenomena, however, are rarely as structured and regular as we would like them to be. To describe the propagation of real-world waves in most physical media, such as waves in water, or sound, we must use somewhat more complex partial differential equations. Many of these systems, with the notable exception of light, share a property that waves of different frequencies will travel at different speeds. "Speed" in such a context is a tricky thing to define - however in this case we are referring to the speed of any localized quantity of energy - how fast its peak travels in space. In the above, classic, wave equation this corresponds to c. However, in the real world, when waves of different frequencies have different speeds, groups of waves are no longer cohesive and will experience "dispersion": the envelope of the group of waves will change shape over time and potentially not remain cohesive. Even in wave groups with dispersive properties, it is possible to find solutions to their partial differential equations where nonlinearities in the propagation and interaction between waves will counteract the dispersive properties of the wave group. This phenomenon, while lacking a strict definition, is called a "soliton". It describes a wave packet which retains its shape during propagation.

Recall the idea (touched upon in Growing Neural Cellular Automata) that a grid of communicating NCAs can be thought of as a finite difference approximation of a partial differential equation in both time and space. Several of the patterns we render in the pattern-generation experiment, as well as in the inception experiment, consist of well-defined structures such as circles or polygons. We consider such structures to be functionally equivalent to solitons and refer to them as such. Such a classification is inspired by B. Chan's reference to solitons in "Lenia." He defines them as solid, self-maintaining structures which arise in the continuous approximation of the Game of Life.

Attribution

The "mouse" and "swipe" icons in the demo are licensed under CC-BY.

Footnotes

1. We use NCA to refer to both *Neural Cellular Automata* and *Neural Cellular Automaton*. [↩]
2. Degenerate in this case refers to the biological concept of degeneracy. [↩]
3. Perhaps an apocryphal claim, but at the very lowest level every zebra will be unique. Ourp point is - "zebra stripes" as a concept in human understanding refers to the general structure of a black and white striped pattern and not to a specific mapping from location to colour. [↩]
4. See liar paradox. In principle, gene regulatory networks can express paradoxical behaviour, such as that expression of factor A represses the expression of factor A. [14] One result of such a paradox can be that a certain factor will oscillate with time. [↩]
5. For a brief definition of gram matrices, see here. [↩]

References

1. Growing Neural Cellular Automata [link]
   Mordvintsev, A., Randazzo, E., Niklasson, E. and Levin, M., 2020. Distill, Vol 5(2), pp. e23. DOI: 10.23915/distill.00023
2. Image segmentation via Cellular Automata [PDF]
   Sandler, M., Zhmoginov, A., Luo, L., Mordvintsev, A., Randazzo, E. and Arcas, B.A.y., 2020. arXiv [cs.CV].
3. Self-classifying MNIST Digits [link]
   Randazzo, E., Mordvintsev, A., Niklasson, E., Levin, M. and Greydanus, S., 2020. Distill, Vol 5(8). DOI: 10.23915/distill.00027.002
4. Differentiable Image Parameterizations [link]
   Mordvintsev, A., Pezzotti, N., Schubert, L. and Olah, C., 2018. Distill, Vol 3(7). DOI: 10.23915/distill.00012
5. The chemical basis of morphogenesis [link]
   Turing, A.M., 1952. Philosophical transactions of the Royal Society of London. Series B, Biological sciences, Vol 237(641), pp. 37–72. Royal Society. DOI: 10.1098/rstb.1952.0012
6. Turing patterns in development: what about the horse part? [link]
   Marcon, L. and Sharpe, J., 2012. Current opinion in genetics & development, Vol 22(6), pp. 578–584. DOI: 10.1016/j.gde.2012.11.013
7. A unified design space of synthetic stripe-forming networks [link]
   Schaerli, Y., Munteanu, A., Gili, M., Cotterell, J., Sharpe, J. and Isalan, M., 2014. Nature communications, Vol 5, pp. 4905. DOI: 10.1038/ncomms5905
8. On the Formation of Digits and Joints during Limb Development [link]
   Hiscock, T.W., Tschopp, P. and Tabin, C.J., 2017. Developmental cell, Vol 41(5), pp. 459–465. DOI: 10.1016/j.devcel.2017.04.021
9. Modeling digits. Digit patterning is controlled by a Bmp-Sox9-Wnt Turing network modulated by morphogen gradients [link]
   Raspopovic, J., Marcon, L., Russo, L. and Sharpe, J., 2014. Science, Vol 345(6196), pp. 566–570. DOI: 10.1126/science.1252960
10. Pattern formation mechanisms of self-organizing reaction-diffusion systems [link]
    Landge, A.N., Jordan, B.M., Diego, X. and Müller, P., 2020. Developmental biology, Vol 460(1), pp. 2–11. DOI: 10.1016/j.ydbio.2019.10.031
11. Bioelectric gene and reaction networks: computational modelling of genetic, biochemical and bioelectrical dynamics in pattern regulation [link]

Pietak, A. and Levin, M., 2017. Journal of the Royal Society, Interface / the Royal Society, Vol 14(134). DOI: 10.1098/rsif.2017.0425

12. Turing-like patterns can arise from purely bioelectric mechanisms [link]
Brodsky, M.. Draft. DOI: 10.1101/336461

13. Dissipative structures in biological systems: bistability, oscillations, spatial patterns and waves [link]
Goldbeter, A., 2018. Philosophical transactions. Series A, Mathematical, physical, and engineering sciences, Vol 376(2124). DOI: 10.1098/rsta.2017.0376

14. Gene networks and liar paradoxes [link]
Isalan, M., 2009. BioEssays: news and reviews in molecular, cellular and developmental biology, Vol 31(10), pp. 1110–1115. DOI: 10.1002/bies.200900072

15. Texture Synthesis Using Convolutional Neural Networks [PDF]
Gatys, L.A., Ecker, A.S. and Bethge, M., 2015. arXiv [cs.CV].

16. The chemical basis of morphogenesis. 1953 [link]
Turing, A.M., 1990. Bulletin of mathematical biology, Vol 52(1-2), pp. 153–97; discussion 119–52. DOI: 10.1007/BF02459572

17. Pattern formation by interacting chemical fronts [link]
Lee, K.J., McCormick, W.D., Ouyang, Q. and Swinney, H.L., 1993. Science, Vol 261(5118), pp. 192–194. DOI: 10.1126/science.261.5118.192

18. Complex patterns in a simple system [link]
Pearson, J.E., 1993. Science, Vol 261(5118), pp. 189–192. DOI: 10.1126/science.261.5118.189

19. Very Deep Convolutional Networks for Large-Scale Image Recognition [PDF]
Simonyan, K. and Zisserman, A., 2014. arXiv [cs.CV].

20. Adam: A Method for Stochastic Optimization [PDF]
Kingma, D.P. and Ba, J., 2014. arXiv [cs.LG].

21. Describing Textures in the Wild [PDF]
Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S. and Vedaldi, A., 2013. arXiv [cs.CV].

22. The texture lexicon: Understanding the categorization of visual texture terms and their relationship to texture images [link]
Bhushan, N., Rao, A.R. and Lohse, G.L., 1997. Cognitive science, Vol 21(2), pp. 219–246. Wiley. DOI: 10.1207/s15516709cog2102_4

23. Re-membering the body: applications of computational neuroscience to the top-down control of regeneration of limbs and other complex organs [link]
Pezzulo, G. and Levin, M., 2015. Integrative biology: quantitative biosciences from nano to macro, Vol 7(12), pp. 1487–1517. DOI: 10.1039/c5ib00221d

24. Embryonic Development and Induction [link]
Speman, H., 1938. The American Journal of the Medical Sciences, Vol 196(5), pp. 738. DOI: 10.1097/00000441-193811000-00047

25. Communication, Memory, and Development [link]
Grossberg, S., 1978. Progress in Theoretical Biology, pp. 183–232. Elsevier. DOI: 10.1016/b978-0-12-543105-7.50012-9

26. WaveFunctionCollapse [link]
Gumin, M.. Github.

27. Texture Networks: Feed-forward Synthesis of Textures and Stylized Images [PDF]
Ulyanov, D., Lebedev, V., Vedaldi, A. and Lempitsky, V., 2016. arXiv [cs.CV].

28. TextureGAN: Controlling deep image synthesis with texture patches [PDF]
Xian, W., Sangkloy, P., Agrawal, V., Raj, A., Lu, J., Fang, C., Yu, F. and Hays, J., 2018. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE. DOI: 10.1109/cvpr.2018.00882

29. Interactive evolution of camouflage [link]
Reynolds, C., 2011. Artificial life, Vol 17(2), pp. 123–136. DOI: 10.1162/artl_a_00023

30. A parametric texture model based on joint statistics of complex wavelet coefficients [PDF]
Portilla, J. and Simoncelli, E.P., 2000.

31. Trainable Nonlinear Reaction Diffusion: A Flexible Framework for Fast and Effective Image Restoration [link]
Chen, Y. and Pock, T., 2017. IEEE transactions on pattern analysis and machine intelligence, Vol 39(6), pp. 1256–1272. DOI: 10.1109/TPAMI.2016.2596743

32. The evolutionary significance of butterfly eyespots [link]
Kodandaramaiah, U., 2011. Behavioral ecology: official journal of the International Society for Behavioral Ecology, Vol 22(6), pp. 1264–1271. Oxford University Press (OUP). DOI: 10.1093/beheco/arr123

33. Live Cell Imaging of Butterfly Pupal and Larval Wings In Vivo [link]
Ohno, Y. and Otaki, J.M., 2015. PloS one, Vol 10(6), pp. e0128332. DOI: 10.1371/journal.pone.0128332

34. Focusing on butterfly eyespot focus: uncoupling of white spots from eyespot bodies in nymphalid butterflies [link]
Iwata, M. and Otaki, J.M., 2016. SpringerPlus, Vol 5(1), pp. 1287. DOI: 10.1186/s40064-016-2969-8

35. OpenAI Microscope [link]
Schubert, L., Petrov, M., Carter, S., Cammarata, N., Goh, G. and Olah, C., 2020. OpenAI.

36. The neural origins of shell structure and pattern in aquatic mollusks [link]
Boettiger, A., Ermentrout, B. and Oster, G., 2009. Proceedings of the National Academy of Sciences of the United States of America, Vol 106(16), pp. 6837–6842. DOI: 10.1073/pnas.0810311106

37. Emergent complexity in simple neural systems [link]
Boettiger, A.N. and Oster, G., 2009. Communicative & integrative biology, Vol 2(6), pp. 467–470. DOI: 10.4161/cib.2.6.9260

38. Going deeper with convolutions [PDF]
Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1--9.

39. Stable and Controllable Neural Texture Synthesis and Style Transfer Using Histogram Losses [PDF]
Risser, E., Wilmot, P. and Barnes, C., 2017. arXiv [cs.GR].

40. Stem cell migration and mechanotransduction on linear stiffness gradient hydrogels [link]
Hadden, W.J., Young, J.L., Holle, A.W., McFetridge, M.L., Kim, D.Y., Wijesinghe, P., Taylor-Weiner, H., Wen, J.H., Lee, A.R., Bieback, K. and al., e., 2017. Proceedings of the National Academy of Sciences of the United States of America, Vol 114(22), pp. 5647–5652. DOI: 10.1073/pnas.1618239114

Updates and Corrections

If you see mistakes or want to suggest changes, please create an issue on GitHub.

Reuse

Citation

For attribution in academic contexts, please cite this work as

```
Niklasson, et al., "Self-Organising Textures", Distill, 2021.
```

BibTeX citation

```
@article{niklasson2021self-organising,
  author = {Niklasson, Eyvind and Mordvintsev, Alexander and Randazzo, Ettore and Levin,
Michael},
  title = {Self-Organising Textures},
  journal = {Distill},
  year = {2021},
  note = {https://distill.pub/selforg/2021/textures},
  doi = {10.23915/distill.00027.003}
}
```