Neural Cellular Automata Manifold

Alejandro Hernandez Ruiz[1] Armand Vilalta[2] Francesc Moreno-Noguer[1]
[1]Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Spain
[2]Barcelona Supercomputing Center, Spain

Abstract

Very recently, the Neural Cellular Automata (NCA) has been proposed to simulate the morphogenesis process with deep networks. NCA learns to grow an image starting from a fixed single pixel. In this work, we show that the neural network (NN) architecture of the NCA can be encapsulated in a larger NN. This allows us to propose a new model that encodes a manifold of NCA, each of them capable of generating a distinct image. Therefore, we are effectively learning an embedding space of CA, which shows generalization capabilities. We accomplish this by introducing dynamic convolutions inside an Auto-Encoder architecture, for the first time used to join two different sources of information, the encoding and cell's environment information. In biological terms, our approach would play the role of the transcription factors, modulating the mapping of genes into specific proteins that drive cellular differentiation, which occurs right before the morphogenesis. We thoroughly evaluate our approach in a dataset of synthetic emojis and also in real images of CIFAR-10. Our model introduces a general-purpose network, which can be used in a broad range of problems beyond image generation.

1. Introduction

Reproduction of multi-cellular organisms entails generating entire bodies from a single cell. Complex organisms also require to create different types of somatic cells and spatially arrange them to form the different tissues while ensuring temporal stability. These three aspects, cellular differentiation, morphogenesis and cell-growth control are the pillars of developmental biology. Computational methods are a key ingredient of the developmental biology study, up to the point that the term "morphogene" itself was coined by Turing [23] decades before its empirical demonstration.

In this context, we model these fundamental processes using novel dynamic neural network architectures in combination with neural cellular automata (NCA) [19]. We evaluate the proposed approach on the synthetic

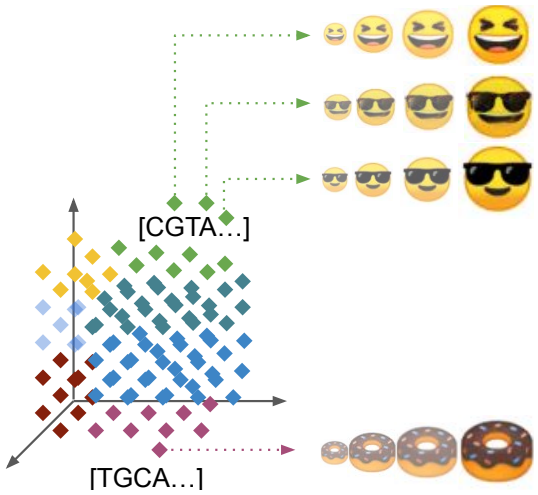

Figure 1. **A manifold of Neural Cellular Automata.** Our model encodes a manifold of programs in a DNA-like encoding. The encodings are transformed into NCAs parameters, whose behavior is able to reconstruct (or "grow") a desired target image from a pixel seed.

`NotoColorEmoji` [6] and real `CIFAR-10` [16] image datasets. In both cases our model is able to "grow" the images with a low error (see Fig. 2).

Most complex organisms have their DNA protected inside a nucleus. DNA expression outside the nucleus is carried out by the cellular machinery and regulated via Transcription Factors (TF). Many TF are involved in defining an organism's spatial arrangement. Most of them are Morphogenes, soluble molecules that can diffuse and carry signals via concentration gradients. This biological model inspired our network architecture at the macro-scale (see Fig. 3). In our network, we create a vector encoding where common information for all cells is stored, as in DNA. The expression of such encoding by the "cellular machinery" is modulated by a Parameter Predictor (PP), with a similar role to that of TF. The "cellular machinery" (NCAM's Dynamic Convolutions) receives two sources of information: one from its DNA-encoding through the Parameter Predictor and another from the gradients of Morphogenes in its surrounding envi-

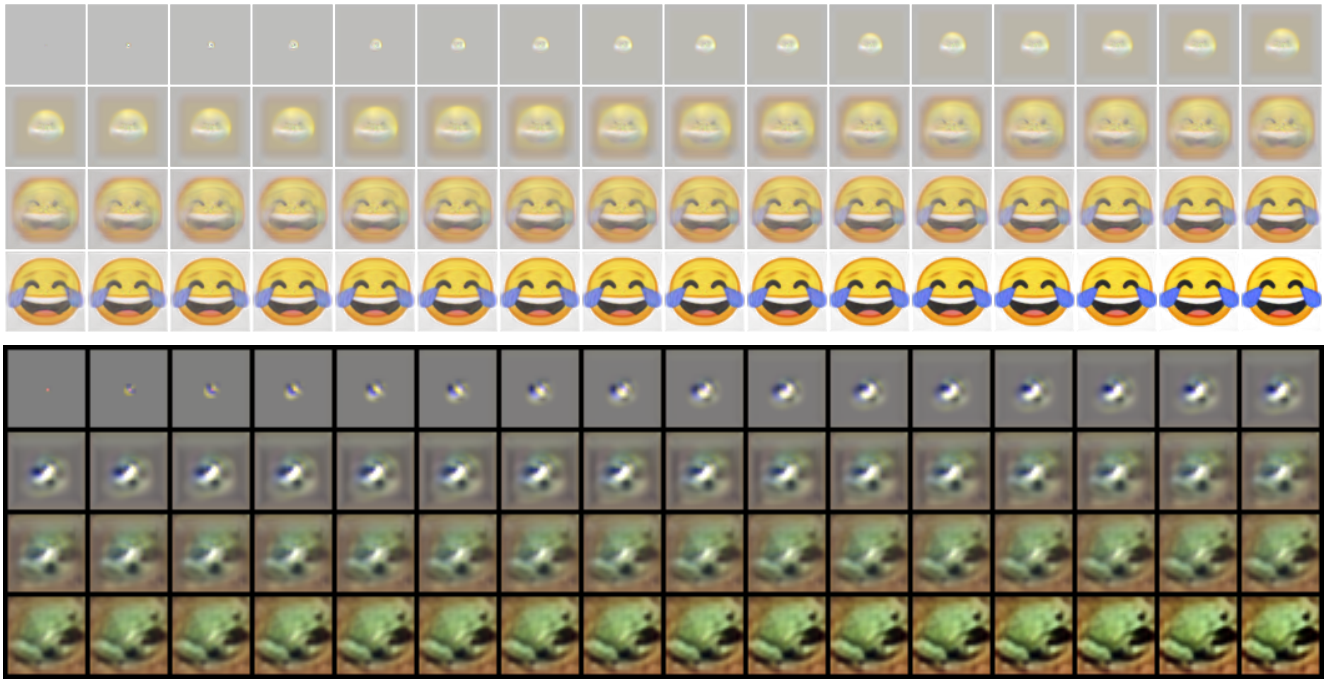

Figure 2. **Growth process** step by step from pixel seed image. Top, sample from the full `NotoColorEmoji` dataset. Bottom, sample from the full `CIFAR-10` dataset.

ronment. Both are combined to model cell differentiation, producing the "phenotype" (color) and other Morphogenes (invisible channels) that drive the expression of neighboring cells.

In this work, we aim to learn not only a single model suitable for one specific phenotype, but a single model suitable for different species with thousands of phenotypes. For this purpose, we introduce a novel Auto-Encoder architecture with dynamic convolutions, which is able to learn an embedding space of NCAs suitable for thousands of images. The proposed model is trainable in an end-to-end manner, and exhibits consistent generalization capabilities, in relation to the produced images.

In our "genetic engineering" experiments (Sec.4) we show that it is possible to produce sensible new images from CA defined by new codes based on existing samples in the dataset. For the sake of similarity with the biological model, we also included an encoding conceptually similar to that of DNA, i.e. a vector encoding on a base of four possible categorical values similar to our DNA's cytosine, guanine, adenine, and thymine bases (the well-known "CGAT" sequences).

In the biological domain, the model we propose could be useful to solve problems such as those faced by [21], when it was not possible to completely model a plant's stomata complexity due to difficulties to obtain CA's function from noisy real data. Its applications can spread among many other biological domains where CA-modeling has been previously used, from tumor growth processes [12] up to infection dynamics [4].

Although biologically inspired, our model is capable of dealing with any vectorial data, making it useful in a broad range of problems. Having an embedding space of CA opens the possibility to create new CA that can generate unseen behaviors based on the recombination of features from learned CA samples.

The main contributions of this work are:

- Introducing a new type of model that can learn a space of programs in the form of Cellular Automata, which are capable of producing desired target patterns.(see Sec.3)
- Showing that the learned program's space has generalization capabilities on the results the programs produce.(see genetic engineering experiments in Sec.4)
- Showing that the embedding space is capable of learning and representing up to 50.000 different programs simultaneously with only 512 real-valued dimensions.(see CIFAR experiments in Sec.4)
- Introducing a new fully dynamic network architecture, which generates CA's parameters from NN output.(see Sec. 3.3)
- The architecture proposed is trainable end-to-end, without need of pre-training any part, for datasets of different sizes and characteristics.
- Extending the capabilities of the original NCA [19] from RGBA to RGB images, and potentially to any

type of vectorial data.

- Demonstrate how to build a categorical embedding space similar to DNA, capable of encoding the same information that a continuous embedding space while achieving high robustness to random mutations.

2. Related

Cellular Automata (CA) is a model based on a grid representation of the world. Each cell of the grid is an automaton or program which perceives the environment (i.e. its state and the state of the neighboring cells) and updates its state according to a fixed rule, generally a mathematical function. The rule is the same for every cell and does not change over time. CA models can be universal [2], which means that they can be programmed to emulate any other system without changing its underlying construction. Despite its expressive power, the main shortcoming of CAs is that given any specific rule, it is impossible to predict its behavior. This means that to understand its behavior, each rule has to be tried with multiple starting states, and its performance needs to be observed for many iterations. The Neural Cellular Automata (NCA), recently proposed by [19], is a class of Cellular Automata which uses an artificial neural network as update function, so the the NN parameters can be learned to obtain a desired behavior. Even more, the specific behavior is learned without prescribing the intermediate states, but instead by minimizing a loss towards a goal pattern, i.e. an image. NCA approach is conceptually different from other image generation techniques since it targets to create a complex program, which in turn is capable of generating the final image. Its characteristics are accordingly different, being, for instance, able to recover corrupted patterns. In this work we also address the requirement of a visibility channel of the original NCA [19], allowing it a more general usability on RGB images, or any kind of data in vectorial form.

Conditioning Models. Conditioning models use information either from the same network (e.g. squeeze and excitation block [10]), from a joint network (e.g. style network [13, 14]) or from a completely independent source (e.g. language embedding [3]), to scale the internal representation channel-wise. The key part is to inject into the network the all the information it needs to perform its task. The conditioning is not always considered an input in the standard manner, but rather transformation on it. We could have used the conditioning approach to embed the NCA into our model, but this would have changed the original architecture by adding the conditioning layer to the CA. Such layer would also add an external source of information at each step, which would change the definition of a CA. Because our goal is to preserve as much as possible the architecture of a CA model, this approach was not ideal to us.

Dynamic Convolutions. Instead of using conditioning to inject the information, we opted for the dynamic convolutions, in which the weights of the convolutional kernel are specifically computed for each sample. In previous works that use dynamic convolutions [15, 11], the architecture forks in two paths from the input image: one path computes the kernels while the other process the input through some fixed convolutions before feeding applying the previously computed dynamic kernels. In contrast, in our approach, the kernel weights are generated from an encoding which is completely independent (and different) from the image to be processed by them. Indeed, the image to be processed by our dynamic convolutions is the pixel seed, which is the same for all targets.

Dynamic Networks. The formulation of our architecture makes the decoder a fully dynamic network, in which the weights are computed on the fly for each sample. In this perspective, it is related to the deep fried transform[26] one shot learners[1] and the HyperLSTM[7]. The goal in these works was to classify images or text, which is a very different task from ours. To the best of our knowledge, our model is the only using a fully convolutional and fully dynamic network architecture in a generative setting.

3. Neural Cellular Automata Manifold

We already described in the introduction the biological inspiration of our model. Next, we move forward to its detailed formulation. The Neural Cellular Automata Manifold (NCAM) uses dynamic convolutions to model a space of NCAs. This space is learned through an Auto-Encoder framework. Formally, our model is defined by the following equations:

$$
\begin{aligned}
\boldsymbol{I}^t &= \{\boldsymbol{C}_{ij}^t\} \quad \forall i,j \in \boldsymbol{I} \\
\boldsymbol{C}_{ij}^t &= f(\boldsymbol{C}_{ij}^{t-1}, \boldsymbol{M}_{kl}^{t-1}, \kappa(\boldsymbol{e}^I, \boldsymbol{\theta}), \theta_{LF}) \quad \forall (k,l) \in \boldsymbol{\epsilon}_{ij} \\
\boldsymbol{M}_{ij}^t &= g(\boldsymbol{C}_{ij}^{t-1}, \boldsymbol{M}_{kl}^{t-1}, \kappa(\boldsymbol{e}^I, \boldsymbol{\theta}), \theta_{LF}) \quad \forall (k,l) \in \boldsymbol{\epsilon}_{ij} \\
\kappa(\boldsymbol{e}^I, \boldsymbol{\theta}) &= \mathcal{P}(\mathcal{D}(\boldsymbol{e}^I, \boldsymbol{\theta}_{\mathcal{D}}), \boldsymbol{\theta}_{\mathcal{P}}) \\
\boldsymbol{\theta} &= \{\boldsymbol{\theta}_{\mathcal{P}}, \boldsymbol{\theta}_{\mathcal{D}}\} \\
\boldsymbol{\epsilon}_{ij} &= (\{i-n_x, \ldots, i+n_x\}, \{j-n_y, \ldots, j+n_y\})
\end{aligned}
\tag{1}
$$

where $\boldsymbol{I}^t$ is the image generated at step t, $\boldsymbol{C}_{ij}^t$ is the color vector (RGB or RGB-Alpha) of the pixel in position (i,j), $\boldsymbol{M}_{ij}^t$ is the corresponding vector of "Morphogenes" (i.e. invisible channels in the grid), $\boldsymbol{\epsilon}_{ij}$ are indices of the neighborhood of the cell (i,j) which extend n_x, n_y positions to each side in x and y axis, and $\boldsymbol{e}^I$ is the vector encoding the image. $f(\cdot)$ and $g(\cdot)$ are the functions implemented as an NCA to predict the colors and "Morphogenes", respectively. $\kappa(\boldsymbol{e}^I, \boldsymbol{\theta})$ is the function that predicts the weights of the NCA from the encoding $\boldsymbol{e}^I$ and its learned parameters $\boldsymbol{\theta}$, which is the composition of the functions learned by the DNA-decoder $\mathcal{D}(\cdot)$

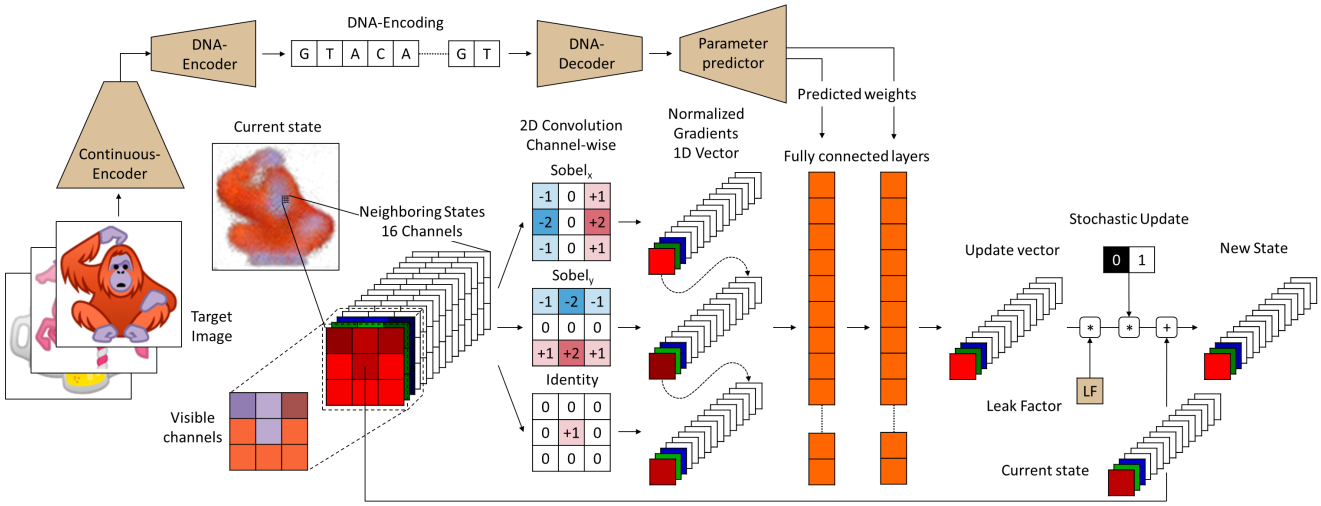

Figure 3. **Model overview.** Beige elements contain trainable parameters while orange layers use only predicted parameters. See Fig.4 for details of the architecture.

and the Parameter Predictor $\mathcal{P}(\cdot)$. The learned parameters are $\boldsymbol{\theta}_{\mathcal{P}}$, $\boldsymbol{\theta}_{\mathcal{D}}$ and θ_{LF}, the Leak Factor (see Sec. 3.1).

In order to train this model, we could simply feed it with arbitrary codes, compute the reconstruction error for corresponding target images, and back-propagate the error to learn the parameters. However, we found it more sensible to learn embedding codes that can exploit similarities between images. We, therefore, decided to use an Auto-Encoder architecture [17] at the macro level, which learns to map the set of inputs to a latent space and reconstructs them back. The Auto-Encoder consists of an Encoder and a Decoder (see Fig. 4). The Encoder is composed of two main components: a continuous encoder, that maps the input to a continuous variables encoding; and a DNA-encoder, that transforms this encoding to a categorical variables encoding. The Decoder's structure is symmetrical, mapping first the DNA-encoding to a continuous encoding which, in turn, feeds the parameter predictor block. From an auto-encoder perspective, the NCA should be considered the third part of the decoder since it is the responsible for finally providing the reconstructed images. From a NN perspective, we can see the NCA as a stack of Residual CNN blocks sharing weights or a "Recurrent Residual CNN".

3.1. Architecture Details

The architecture of the net is based on different types of residual blocks [8]. The smallest building blocks are of 3 types: Convolutional Block 3x3 (CB3), Convolutional Block 1x1 (CB1) and Fully Connected Block (FCB) (see details in Fig. 4). Unlike most of previous works [8, 25, 22], the blocks do not modify input/output representation dimensionality, neither in space resolution or number of filters. While these characteristics can be useful for classification problems, where information needs to be condensed through processing steps, our intuition is that this is not desirable in a reconstruction problem since detail information is lost.

A significant peculiarity of CB1 and CB3 is the expansion of the number features in the internal layer by a factor of 4 and 2, respectively. We consider that this increase of dimensionality of the embedding space allows for a better disentanglement of the data manifold. Notice that CB1 needs to be used in combination with CB3 to introduce neighbourhood information. This detail is opposed to previous architectures [8, 25, 22] that reduce the number of filters in the inner layer of the block, while increasing dimensionality in the short-cut path.

A specific detail of the architecture is the use of a Leak Factor (LF) as an aid for training stability. LF is a single learnable parameter that regulates how much each successive step contributes to the output. A low value of LF encourages the network to retain most of the knowledge from previous steps, avoiding to distort the image too abruptly at any given step. After the network has learnt the basics of the problem this scenario is less likely, thus the network can learn to let more and more information to leak through each step. LF is constrained between 10^{-3} and 10^{3}, initialized at 10^{-1}.

Unlike many architectures that only take the spatial mean of the convolutional output tensor to feed FC layers [8, 25, 22], we use 3 additional slices. Being $(c{*}h{*}w)$ the dimensions of the output tensor, a spatial mean over h, w provides a (c) dimensional encoding of the image. A mean over h yields a $(c * w)$ dimensional vector, and doing similarly over the other 2 dimensions (c and w), we obtain the input to our FCB of $dimension = c + (c * h) + (c * w) + (h * w)$.

3.2. DNA-encoding

The latent vector embedding can be interpreted as the source code that dictates the behavior on the cells of each

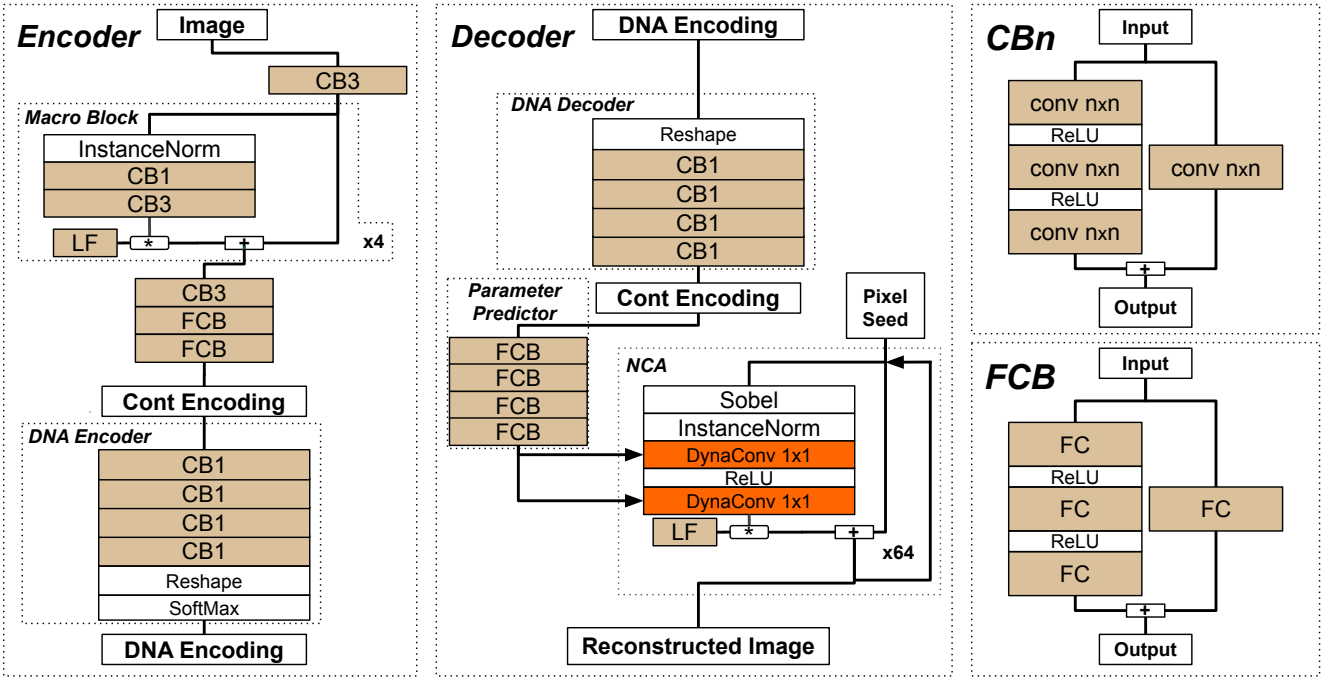

Figure 4. **Architecture Details.** Beige elements contain trainable parameters while orange layers use only predicted parameters. CBn blocks can be CB1 or CB3. All blocks share the same number of filters except the last blocks whose output matches the embedding or parameters dimensionality.

of the NCAs. This definition inspired us to think of a possible alternative representation of such source code: each value in the latent vector can be encoded into a categorical base, making our encoding compatible to that of the DNA. Notice that a simple quantization on the continuous variable would provide an ordinal discrete variable instead of a DNA-like categorical one. This encoding is dimensioned to handle the same 32 bits of information of the corresponding float variable, thus no additional bottleneck or expansion is introduced here.

As we can see in Fig. 4, the DNA-encoder is composed of 4 successive CB1 feeding a softmax layer to obtain the 4-categories DNA-embedding. The DNA-decoder follows a symmetrical structure, mapping back each "gene" to a continuous feature. These CB1 are all 1D convolutions, independently expanding each feature of the continuous encoding to a 16-features "gene". This independent processing of different variables makes no assumption about the "meaning" of each category for one variable in relation to its "meaning" in other variables, so the actual "meaning" depends only on latter interaction between corresponding continuous variables.

In order to train our encoding, we add a biologically plausible noise replacing half of the letters by randomly drawn letters. Our intent is not to precisely mimic the mutation rate of DNA ($2.5 \times 10^{-8}/generation$ [20]) but to enforce a high redundancy in the encoding.

3.3. Dynamic Convolutions

Similarly to [15], for an image sample I, being $\boldsymbol{X}^I$ the input tensor of the convolution and $\boldsymbol{Y}^I$ the output tensor, we define the dynamic convolution as $\boldsymbol{Y}^I_n = \sum_m \kappa(\boldsymbol{e}^I)_{mn} \star \boldsymbol{X}^I_m$, where $\kappa(\boldsymbol{e}^I)_{mn}$ is the convolution kernel dynamically computed from the encoding for I, and (m, n) are the input and output channels of the convolution. Although dynamic convolutions are quite a novel idea, in our case they are the simplest way to achieve our goal, keeping the original NCA architecture unchanged but encapsulated in a meta-learning system.

3.4. Neural Cellular Automata Architecture

The last component of the decoder is an NCA. To reconstruct an image, the NCA starts from a pixel seed image (typically a blank image with a different pixel). Then, each cell in the grid recurrently perceives the environment and computes an update. After a fixed number of steps, the output is evaluated and the reconstruction error back-propagated. In [19], NCA's kernel weights are directly updated, but in our case the error is back-propagated further, through the NCA and to the parameter predictor, updating its weights instead of the NCA's. We can divide the NCA architecture in two main blocks, perception and update.

The **Perception Layer** was designed to propagate the gradients across the grid in the 16 channels composing it (the first four corresponding to the visible RGBA). This is

Figure 5. **Growth results.** First row is a random set of images from the full `NotoColorEmoji` dataset. Following rows are generated by different variants of NCAM codified as: CE: Continuous encoding, DNA: DNA-encoding; STO: Stochastic update, SYN: Synchronous update; 16-512: number of channels in NCA (16,32) - Dimensionality of the continuous embedding (256, 512, 1024) (DNA dimensionality is 16 times that of continuous).

achieved with manually defined Sobel filters (see Fig. 3), similarly to [19]. Since our network's architecture has only $ReLU$ activation functions, the activations have an unbounded positive value. Given the recurrent architecture of the NCA, during the training process, an exponential reinforcement can occur, leading activations and weights to ∞ and degenerate states. To avoid this problem, we apply instance normalization [24] on top of the Sobel filters.

To compute the **Update**, we use the NCA architecture, consisting on 2 pixel-wise dense layers (implemented as 1x1 convolutions). The update is modulated by the Leak Factor (the only trainable parameter in our NCA), analogous to the LF used in the Continuous Encoder. Finally, the update is stochastically applied on the current state with an independent probability of $p = 0.5$ for each cell.

4. Experiments

Datasets. `NotoColorEmoji` dataset [6]: 2924 images of synthetic emojis. This kind of images are very simple and with sharp edges, therefore it is relatively easy to visually assess the quality of the produced images. The original images are 128x128 pixels but in our experiments we downscaled them to 64x64 to reduce the computational requirements. `CIFAR-10` dataset [16]: 50.000 real 32x32 images from 10 categories: plane, car, bird, cat, deer, dog, frog, horse, ship, truck. The number of images and its variability make it challenging.

Continuous vs. DNA encoding. Generation results for these two sets of experiments are very similar, both visually (see Fig. 5, rows 1-2) and numerically in MSE (CE: 0.01591, DNA: 0.01633). Results show no clear reduction or increase on performance with the extra processing involved in the DNA encoding and decoding process. We consider that the proposed methodology achieves the desired objective of obtaining a categorical encoding equivalent to the continuous encoding commonly used in DNNs. Moreover, we applied a 50% chance of category error during the tests to showcase the remarkable robustness achieved by this encoding.

Stochastic vs. synchronous update. We consider that the stochastic update feature of the NCA, while being relevant in the biological analogy, may suppose an extra difficulty in the learning process. In these experiments we remove the stochastic update ($p = 0.5$) making it synchronous at every time step ($p = 1.0$). Notice that this change implies, on average, doubling the number of steps on the NCA processing. It also makes more reliable the expected state of neighboring cells. This modification reduced the error significantly in both scenarios, continuous and DNA encoding. Note that images generated with this approach succeed in reconstructing even the finest details (see Fig.5, rows 3-4). MSE is one order of magnitude below stochastic approaches (MSE: CE: 0.00199, DNA: 0.00262). With either kind of update, it is remarkable the absence of artifacts in the background of the emojis, given that we removed the alive masking mechanism used in [19], proving that our design is able to learn by itself the limits of the figure.

Effect of encoding dimensionality. We next evaluate our method under significant changes in the encoding dimensionality setting it to half (256) and double (1024) size. Notice that these dimensionalities refer to the continuous encoding, the actual DNA-encoding dimensions are 16 times larger. The experiments show (see Fig. 5, rows 5-6) that there is a significant quality degradation when the dimensionality is

Figure 6. **Growth results on reduced datasets.** For each subset of `NotoColorEmoji`, first row are random images while the second are generated using DNA-encoding and stochastic update.

reduced to half while there is no appreciable improvement when it is doubled (MSE: 256: 0.02302, 1024: 0.01584). Therefore, we consider that the embedding size is not a bottleneck for the problem at hand.

Smaller datasets. In order to assess the challenge that the size of the dataset and its visual complexity poses on the proposed method, we experiment on 4 different subsets of the `NotoColorEmoji` dataset, classified by the authors according to their visual appearance. `Chars` (96 images): emojis containing characters of the Latin set and a few symbols. They are simple shapes, usually with straight lines and few colors. `Emos` (103 images): round yellow faces showing emotions. `Heads` (858 images): images of person heads, typically showing different professions. All theses images include the full set of versions according to skin tone. `Variety` (684 images): images not present in other sets that are also visually different among them. It mainly includes animals, objects and food.

Results show that, as expected, problems have a growing level of difficulty in terms of MSE: `Chars`: 0.00976 < `Emos`: 0.01799 < `Heads`: 0.02439 < `Variety`: 0.05035 which can also be visually assessed on Fig. 6. The primary factor is the visual complexity or variety, not the size of the dataset. Linear simple shapes seem easier to generate than more rounded or intricate ones.

Ablation Study. A small ablation study using the `variety` dataset gives us the following results: Our proposed architecture achieves a MSE of 0.0504. w/o leak factor: 0.0597. w/o normalization 0.0627. w/o slices: 0.0670. These numbers could be further improved by training longer and tuning the training parameters. In our preliminary experiments we observe that the tendency of the error does not change, thus

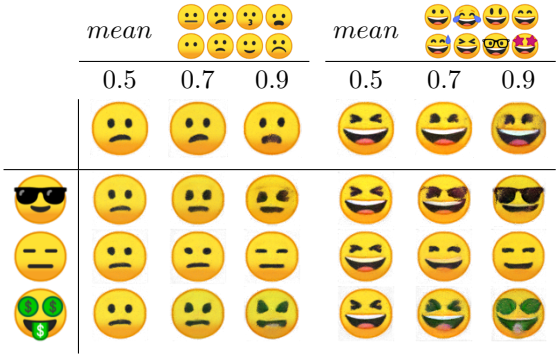

Figure 7. **"Genetic engineering" results.** On top: source images for each of two mean encodings. Just below: mean images generated by different thresholds (0.5, 0.7, 0.9): the higher, more common need to be the "genes". Three bottom rows: On the left: 3 original target images; On the right: different images generated when mean "genes" are injected to targets.

the ordering of the different options for the model remains the same over time.

Comparison to NCA. The method defined in [19] is a qualitative reference but it can not be a baseline since the original NCA model was designed to tackle a different problem, and the learned CA can only produce a single image. Nevertheless, using the code[1] provided by the authors to train a 64×64px CA, we obtain a MSE in the order of 0.001, comparable to our results with continuous encoding, but on a single image.

"Genetic engineering". We next play a little of "genetic en-

[1] https://colab.research.google.com/github/google-research/self-organising-systems/blob/master/notebooks/growing_ca.ipynb

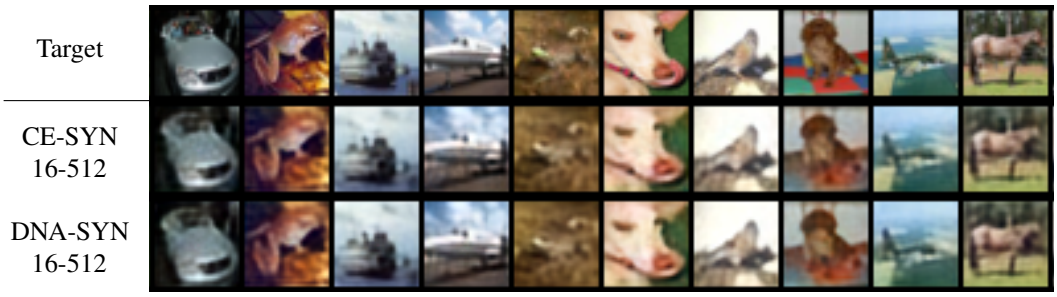

Figure 8. **CIFAR-10 results** using the best methods: Continuous and DNA Encoding with Syncronous update and baseline dimensionalities.

gineering", injecting part of the DNA-encoding from some images to others. To do so, we first generate a mean encoding of a group of source images that share a visual trait. We compute the mean of the 4 discrete features over the samples (i.e. DNA categories) and take the ones with values over a defined threshold. Notice that since they are produced by a softmax we can not have two values over 0.5. If none of the values is over the threshold we would have new category "none" which the DNA-decoder will ignore. With this encoding we generate the mean image. The parts of the mean encoding that are not "none" will substitute corresponding parts of target image encoding.

Results in Fig. 7 show some interesting properties of the embedding. The lack of several features in the mean encoding causes no perturbation in the common traits and usually also provides a reasonable solution for the uncommon. The transfer process does not disrupt other common traits in target images. If the traits transferred collide with existing traits, they produce mixed shapes. We can observe that some traits that are not common in the source images are also transferred, suggesting some kind of trait dominance.

CIFAR-10 results. Finally, we report results on `CIFAR-10` dataset [16]. Note that in this case, our NCAM model is capable to generate up to 50K different images. In Fig. 2 we show an example of image generation for this case. On this dataset we only experimented with the synchronous update (best solution) since it is difficult to appreciate the level of detail required to visually evaluate the quality of the results (see Fig. 8). However, MSE values obtained are CE: 0.00717 DNA: 0.00720, perfectly comparable with those of `NotoColorEmoji`, which implies that the NCAM model also has enough capacity for this dataset.

5. Conclusions

Machine learning techniques are already considered critical for the study of epigenetic processes, meddling between genetic information and their expression, which hold immense promise for medical applications [9]. The model proposed here successfully simulates the main components of developmental biology, from DNA encoding to morphogenesis, at a high conceptual level. In our experiments, it is capable of reproducing almost 3.000 different emoji images, with a great level of detail, and up to 50.000 real images of the `CIFAR-10` dataset. Given its unique structure, capable of combining DNA-encoded and environment information, demonstrated scalability and robustness to noise, we consider it has potential in modeling genetic expression problems.

It is important to notice that the properties of the manifold learnt by the proposed Auto-Encoder will depend on the loss function used for training. In our work, we use MSE loss to learn to reproduce original images with high fidelity. If we were to use an adversarial loss [5], we likely would obtain visually plausible images of the same class but not an exact replica. We consider that the application of a reinforcement learning loss [18] could allow to produce a model driven by a fitness metric, such model would then be similar to genetic evolution. These simple adaptations open unfathomable use possibilities for the NCAM.

This work is a first step on creating embedding spaces to represent not only static information but behavioural patterns, effectively programs. Moreover, our experiments generate programs for multiple agents that, interacting through the environment, achieve the desired collective outcome. Given that our model is capable of dealing with any kind of vectorial data, we do not foresee any theoretical limitation to learn complex programs, to perform all sorts of different tasks, only from data and an approximate loss function. Finally, thanks to the NCAM embedding space, we can generate programs that produce new sensible novel behaviours unseen in the data.

Acknowledgments

This work has been partially funded by the Spanish government with the project HuMoUR TIN2017-90086-R and Maria de Maeztu Seal of Excellence MDM-2016-0656.

References

[1] Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In *Advances in neural information processing systems*, pages 523–531, 2016.

[2] Matthew Cook. Universality in elementary cellular automata. *Complex systems*, 15(1):1–40, 2004.

[3] Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville, and Yoshua Bengio. Feature-wise transformations. *Distill*, 2018. https://distill.pub/2018/feature-wise-transformations.

[4] Ramón ER González, Sérgio Coutinho, Rita Maria Zorzenon dos Santos, and Pedro Hugo de Figueirêdo. Dynamics of the hiv infection under antiretroviral therapy: A cellular automata approach. *Physica A: Statistical Mechanics and its Applications*, 392(19):4701–4716, 2013.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[6] Google. Noto emoji distribution, 2020.

[7] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] Lawrence B Holder, M Muksitul Haque, and Michael K Skinner. Machine learning for epigenetics and future medical applications. *Epigenetics*, 12(7):505–514, 2017.

[10] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[11] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *Advances in Neural Information Processing Systems*, pages 667–675, 2016.

[12] Yang Jiao and Salvatore Torquato. Emergent behaviors from a cellular automaton model for invasive tumor growth in heterogeneous microenvironments. *PLoS computational biology*, 7(12), 2011.

[13] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[14] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

[15] Benjamin Klein, Lior Wolf, and Yehuda Afek. A dynamic convolutional layer for short range weather prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4840–4848, 2015.

[16] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

[17] Yann LeCun. Modeles connexionnistes de l'apprentissage (connectionist learning models). *Doctoral dissertation*, 1987.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[19] Alexander Mordvintsev, Ettore Randazzo, Eyvind Niklasson, and Michael Levin. Growing neural cellular automata. *Distill*, 2020. https://distill.pub/2020/growing-ca.

[20] Michael W Nachman and Susan L Crowell. Estimate of the mutation rate per nucleotide in humans. *Genetics*, 156(1):297–304, 2000.

[21] David Peak, Jevin D West, Susanna M Messinger, and Keith A Mott. Evidence for complex, collective dynamics and emergent, distributed computation in plants. *Proceedings of the National Academy of Sciences*, 101(4):918–922, 2004.

[22] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[23] Alan Turing. The chemical basis of morphogenesis. *Sciences*, 237(641):37–72, 1952.

[24] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[25] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[26] Zichao Yang, Marcin Moczulski, Misha Denil, Nando de Freitas, Alex Smola, Le Song, and Ziyu Wang. Deep fried convnets. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1476–1483, 2015.