Growing Adaptive-Energy Neural Cellular Automata

Anonymous Authors[1]

Abstract

Multicellular organisms grow, maintain, and regenerate their bodies in a self-organizing manner, through local interactions with neighboring cells and the environment. The recently developed Neural Cellular Automata (NCA) model has shown that this process can be simulated by using a Neural Network to learn local rules that grow a desired pattern. In NCA, each cell has the same constant probability of being updated (fire rate) at every time step. In this paper, we present the Adaptive-Energy Neural Cellular Automata model (AdaptNCA) that addresses some limitations of regular NCA. The main change is a cell-wise fire rate, determined by an additional neuron in the network. The fire rates at each step are interpreted as the square root of the cell's energy expenditure, and the model is trained to minimize the total energy spent during growth. We show that AdaptNCA can have better persistence and damage regeneration capabilities than an equivalent NCA model, reaching pattern reconstruction losses up to 95% and 87% lower, respectively.

1. Introduction

The self-organizing capabilities of multicellular organisms have puzzled developmental biologists for a long time. Morphogenesis (Turing, 1952) (the biological process through which an organism takes its shape) and cell differentiation are accomplished using very limited local information, such as chemical concentrations, electrical potentials, and their gradients. This emergence of complexity from local interactions can also be observed computationally in Cellular Automata.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

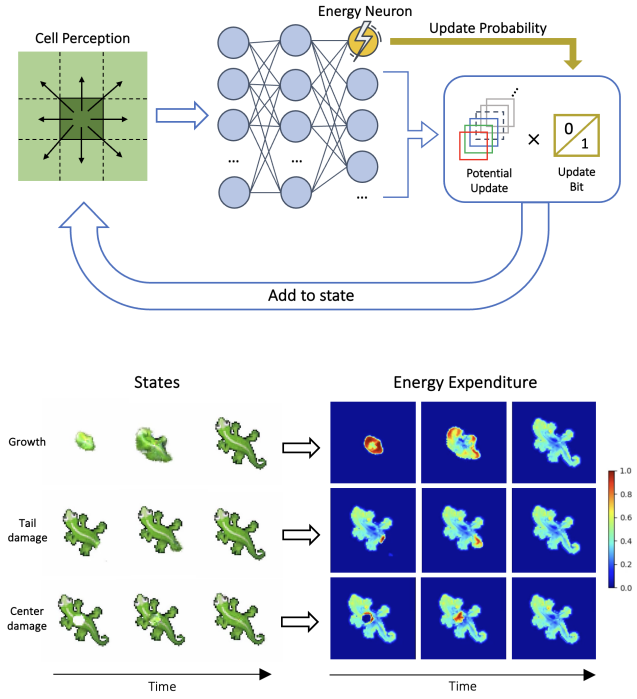

Figure 1. **Top**: AdaptNCA architecture. The cell update probability is computed from the deep neural network's hidden state by an additional neuron (in yellow), and interpreted as the square root of the cell's energy expenditure. **Bottom**: AdaptNCA states throughout time and different damage locations. Despite not having been shown damage during training, AdaptNCA can identify damaged locations and increase cell activity in the surrounding cells.

1.1. Cellular Automata

Cellular Automata (CA) were initially introduced by John von Neumann (Sarkar, 2000) and later popularized with Conway's Game of Life (Gardner, 1977). These are systems composed of a grid of cells, where a set of local rules is repeatedly applied to each cell, based on the states of the cell and its neighbors. Despite their relatively simple nature, CA have found utility in many different scientific fields, from computational fluid dynamics (Chopard & Masselot, 1999) to biological modelling (Ermentrout & Edelstein-Keshet, 1993). It has also been used in more applied subjects, such as the modelling of urban expansion (Losiri et al., 2016), public

transportation (Ding & Huang, 2010) and manufacturing (Barragán-Vite et al., 2018).

Multicellular organisms are highly complex systems, where cells can have very subtle and hard-to-define interactions with each other, and their states (e.g. concentrations of various chemicals) take continuous values. These characteristics are out of the range of regular CA, where local rules are hard-coded, and thus have to be relatively simple, and the state space is discrete. To overcome these limitations, Neural Cellular Automata (NCA) were introduced (Mordvintsev et al., 2020).

1.2. Neural Cellular Automata

In NCA, a cell's state is a 16-channel vector of continuous values, and the local rules are learned by a small neural network, with the objective of growing a desired pattern. The network's input is the cell state and its local directional gradients, and the output is an incremental update to the cell state vector. At each time step, a cell has a probability $p = 0.5$ of being updated. This asynchronicity between cells removes the effect of an external global clock, making NCA more akin to a biological system. It has also been shown that asynchronous NCA are more well-behaved than synchronous ones (Niklasson et al., 2021). This is because, during training, each cell is exposed to neighbors at different time steps, preventing overfitting to a single time step.

1.3. NCA Limitations

This setup allows for a system with capabilities similar to a multicellular organism, such as robustness to noise and moderate self-regeneration. However, it has some limitations that separate it from an accurate representation of a biological system.

1.3.1. Anisotropy

When computing the state gradients to feed into the neural network, Sobel filters in the x and y directions are used. This leads to a pattern that always grows in the same orientation because it is dependent on the orientation of the space itself. In the real world, this would be equivalent to an organism growing only when facing North. Isotropic NCA (IsoNCA) are introduced in Mordvintsev et al. (2022) to address this limitation. By using structured seeds instead of single seeds, and computing the loss on FFTs of polar transforms instead of on the raw cell states, the authors were able to grow asymmetric patterns in a fully symmetric perception field.

1.3.2. Unbounded State Updates

In NCA, a cell's incremental state update is the output of an activation-free fully-connected layer, with the hidden state as the input. That is, each incremental channel update is a linear combination of the hidden state values. The lack of activation creates unbounded state updates and might lead to positive feedback loops in the system, creating instabilities.

1.3.3. Unbounded Weight Gradients

With a constant learning rate, the neural network's weight gradients can be interpreted as the amount of change in the weights from one batch to the next. Paired with the unbounded state updates, this could also lead to instabilities in training. Besides, from a biological perspective, it is known that the change in the DNA between generations is a very slow and gradual process. Although the evolutionary process that leads to the changes in a population's DNA over time is much different from the training of a neural network, bounding the weight gradients would make NCA training more biologically grounded.

1.3.4. Constant Cell Fire Rates

In the original implementation of NCA, every cell has the same constant fire rate (probability of updating at a given step) of $p = 0.5$. While this stochasticity in the updates is necessary for robust NCA (Niklasson et al., 2021), it is a global parameter that is shared between all cells, in an otherwise local system (besides the neural network itself). Furthermore, as we present below, it might not be biologically accurate to use this method.

1.4. Contributions of this work

We propose an interpretation of the fire rate as the square root of a cell's energy expenditure for a certain time step. During morphogenesis, a tissue's cells expend energy at different rates and spatial distributions. Cells in the periphery consume more energy for mitosis, allowing for tissue growth, while cells in the interior expend a smaller amount of energy by producing mechanical forces that hold the tissue together (Stooke-Vaughan & Campàs, 2018).

In the context of NCA, higher fire rates would indicate a more active cell, updating itself more frequently, while lower fire rates would indicate more dormant cells with less activity. This can be accomplished by introducing cell-wise fire rates that can change over time, depending on the state of each cell.

In this work, we plan to address some of the points made previously, by:

- Bounding cell updates by adding a $tanh$ activation before their computation.
- Clipping weight gradients to stabilize training and make the training process more akin to the evolution of a biological system.

- Introducing a learnable cell-wise fire rate, trained by adding an energy loss term.
- Showing that AdaptNCA have better persistence and damage regeneration capabilities when compared to an equivalent NCA model.

We will use the original NCA model as a base for this work.

2. Adaptive-Energy NCA

This section covers the AdaptNCA design and training. The model was implemented in PyTorch[1].

2.1. Model

The model architecture for the AdaptNCA model is shown in Figure 2. Most of the model characteristics described below are replicated from the original Growing Neural Cellular Automata work (Mordvintsev et al., 2020).

2.1.1. Grid and States

Cells are defined on a 72x72 regular grid. Each cell has a state vector

$$\mathbf{x} = (x_0 = R, x_1 = G, x_2 = B, x_3 = A, x_4, ..., x_{15})$$

where x_0 through x_3 correspond to RGBA values, and x_4 through x_{15} correspond to additional channels meant to represent other possible quantities, such as chemical concentrations or electrical potentials. The alpha value ($\alpha = A = x_4$) is interpreted as a cell's "vitality". If $\alpha > 0.1$, a cell and its neighbors are considered as "living" and can be updated. Other cells are considered "dead" and all their channels are explicitly set to 0 at each time step. The initial seed is a "living" black pixel, that is, RGB set to 0 and all other channels to 1: $\mathbf{x}^{(0)} = (0, 0, 0, 1, ..., 1)$.

2.1.2. Cell Perception

To get information about a cell's 3x3 neighborhood, the state vector is concatenated with the directional gradients of each channel to create the perception vector $\mathbf{p}$ with $16 \times 3 = 48$ channels:

$$\mathbf{p}^{(t)} = (\mathbf{x}^{(t)}, Sobel_x * \text{Neighb}(\mathbf{x}^{(t)}), Sobel_y * \text{Neighb}(\mathbf{x}^{(t)}))$$

The Sobel filters used are

$$Sobel_x = \frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$Sobel_y = \frac{1}{8}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

2.1.3. Update Rule

The update vector for each cell is computed by passing the perception vector $\mathbf{p}$ through the main neural network (blue blocks in Figure 2), as in Eq. (1). The $tanh$ function is not present in the original Growing NCA work.

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + u^{(t)} \cdot \tanh(\text{ReLU}(\mathbf{p}^{(t)} W_0 + b_0) W_1) \quad (1)$$

In Eq. (1), W_0, b_0 and W_1 correspond to the main neural network weights. Their dimensions are presented in Table 1. $u^{(t)}$ corresponds to the update bit (Eq. (4)).

Table 1. AdaptNCA Model weights.

Name	Dimensions	#Parameters
W_0	(48, 128)	6,144
b_0	(128)	128
W_1	(128,16)	2,048
W_f	(128,1)	128
b_f	1	1
Total	-	8,449

2.1.4. Adaptive Fire Rates

In order to compute the fire rate $f^{(t)}$ at step t for each cell, the hidden state $\mathbf{h}^{(t)}$ is taken as the input to a separate neuron (orange blocks in Figure 2, with weights W_f and b_f) with sigmoid activation, as shown in Eqs. (2) and (3).

$$\mathbf{h}^{(t)} = \text{ReLU}(\mathbf{p}^{(t)} W_0 + b_0) \quad (2)$$

$$f^{(t)} = \sigma(\mathbf{h}^{(t)} W_f + b_f) \quad (3)$$

To compute the update bit $u^{(t)}$ for a single cell, a Gumbel-Softmax distribution (Jang et al., 2016) is used, as shown in Eq. (4). The Gumbel-Softmax distribution is a continuous approximation of a categorical distribution. This is necessary to make the categorical sampling process differentiable. An additional hyperparameter τ is introduced, corresponding to the temperature. For low temperatures ($\tau = 0.1$ to $\tau = 0.5$) the Gumbel-Softmax distribution approaches the categorical distribution but can lead to higher magnitude gradients. Higher temperatures make the Gumbel-Softmax distribution more uniform, but it can increase the training

[1]The code will be released upon publication of this paper.

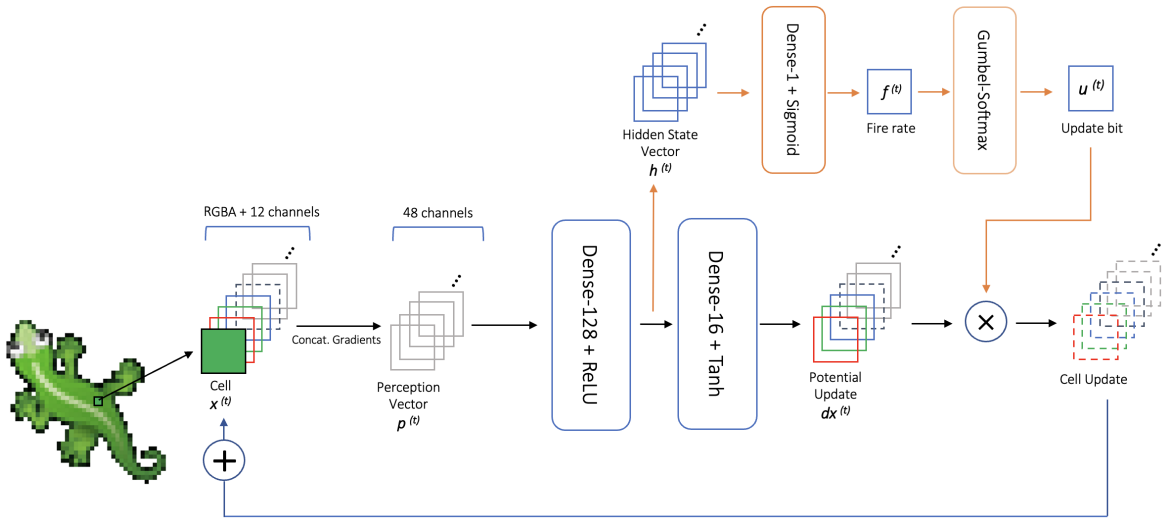

Figure 2. AdaptNCA model architecture. The hidden state vector is passed through an additional neuron to compute the cell's fire rate. An update bit is sampled from the fire rate using a Gumbel-Softmax distribution, and it determines if the cell will update. In the main network, a $tanh$ activation was added to the Dense-16 layer to restrain the update vector.

stability. In this work, the value $\tau = 0.1$ was always used, since we wanted to keep the distribution close to $B(1, f^{(t)})$, and gradient explosions were prevented by the gradient clipping.

$$u^{(t)} \sim \text{Gumbel-Softmax}(1, f^{(t)}; \tau) \approx B(1, f^{(t)}) \quad (4)$$

2.2. Training Configuration

The target pattern for AdaptNCA was the lizard *emoji*, shown in Figure 3. It is a grid of state vectors with RGBA values.

Figure 3. Target pattern for the training of AdaptNCA

The training configuration is similar to the *growing* configuration in the original Growing NCA work. Every epoch, a batch of 8 initial seeds is grown for $s \sim U(64, 128)$ time steps, and s is the same for all seeds in a batch.

2.2.1. Loss function

After running for s steps, the final grid state and the total sum of the fire rates are used to compute the loss with Eq. (5)

$$L = \text{MSE}\left(\text{Grid}(\mathbf{x}_{RGBA}^{(s-1)}), \text{Grid}(\mathbf{x}^{target})\right) + \beta \sum_{t=0}^{s-1} \text{Grid}((f^{(t)})^2) \quad (5)$$

where Grid$(\cdot)$ represents all the values of a quantity in the grid. The interpretation of the loss function is as follows: the **reconstruction term** corresponds to the MSE loss between the final state's RGBA values and the target's, measuring how close the final state is to the target, and the **energy term** corresponds to the sum of all squared fire rates during the lizard's growth, seen as the total energy spent. This involves the introduction of an additional hyperparameter β that controls the relative weight of the energy term in the total loss. We argue that β has a clearer physical interpretation than choosing a fixed fire rate p. It indicates how much the organism should prioritize its energy expenditure. An environment with a high β value might be interpreted as having a lower available amount of energy and, as consequence, few resources. On the other hand, a lower β value allows the organism to focus more on its growth than energy expenditure control, similar to a growing environment rich in resources.

2.2.2. Training Hyperparameters

The models in this work were trained for 10,000 epochs, each consisting of one batch of size 8. The Adam optimizer was used with first and second moment β parameters of 0.5, and a weight decay parameter value of 10^{-5}. A learning rate of 2×10^{-3} was used with an exponential decay of 0.9999 per time step. Weight gradient norms were clipped to a maximum value of 1.

3. Results

3.1. Training

Six different AdaptNCA models were trained, with $\beta \in \{0, 2 \times 10^{-10}, 5 \times 10^{-10}, 1 \times 10^{-9}, 2 \times 10^{-9}, 5 \times 10^{-9}\}$.

In Figure 4 we see similar training curves for all β values. Lower β values lead to slightly lower reconstruction losses, due to the greater weight of this term in the overall loss.

In Figure 5, more pronounced differences appear between parameters. In general, training starts with an increase in energy loss, when the models prioritize the reconstruction loss term. The energy term reaches a peak, and then the models start to learn how to conserve energy by distributing it efficiently during growth. The epoch corresponding to this peak tends to be higher, the lower the β value is.

The increase in the prioritization of energy efficiency and distribution can be seen in Figure 6. By the final epoch, there's a higher energy density in the periphery of the lizard, since those are the regions of high cell proliferation, where most tissue grows from.

3.2. Persistence Tests

We define *persistence* as the ability of NCA to maintain its final state through time, when running it for a higher number of steps than seen in training. In persistence testing, the models ran for 5 times the maximum number of training steps, that is $5 \times 128 = 640$ steps.

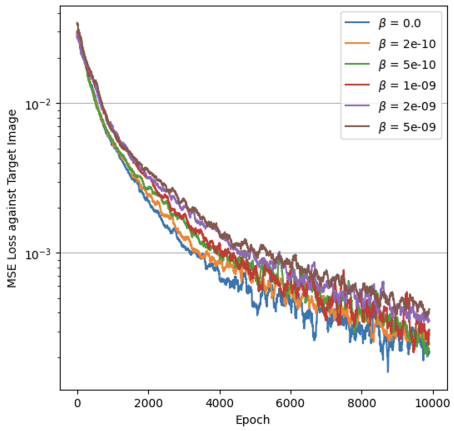

Figure 4. Reconstruction loss during AdaptNCA training for various values of the energy loss parameter β.

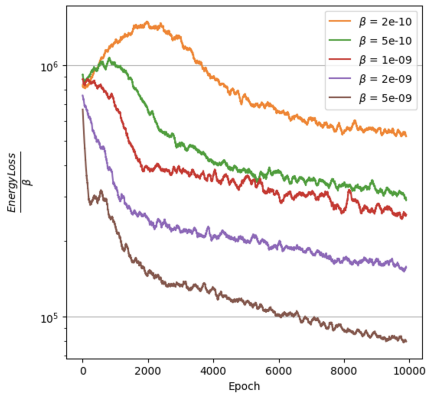

Figure 5. Energy loss divided by parameter β during AdaptNCA training for various values of the energy loss parameter.

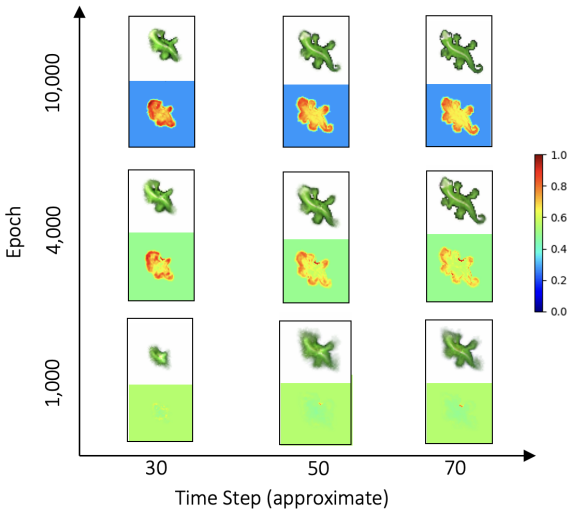

Figure 6. Evolution of states and fire rates during training, for AdaptNCA with $\beta = 2 \times 10^{-10}$.

The full time evolution of the reconstruction loss of the AdaptNCA models trained with different energy β parameters is shown in Figure 7. The average cell-wise fire rate of live cells is shown in Figure 8.

In Figure 7, the model with $\beta = 0$ (that is, with unrestricted fire rates) reaches a lower minimum loss but ends in a worse final state. Since this model set all fire rates to a value very close to 1, it's essentially synchronous, which has been found to lead to a more unstable NCA (Niklasson et al., 2021). The model with $\beta = 1 \times 10^{-9}$ achieves the lowest final loss, with a final lizard that shows very little degradation.

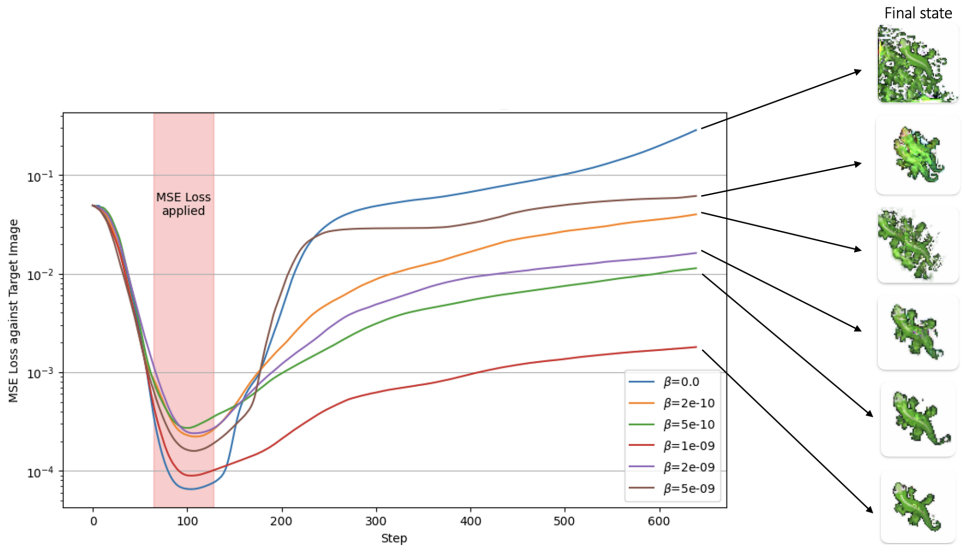

Figure 7. Persistence test: reconstruction loss of AdaptNCA through time for all β values. The red zone represents $U(68, 128)$, corresponding to the distribution of final steps during training. Values averaged over 80 trials for each model.

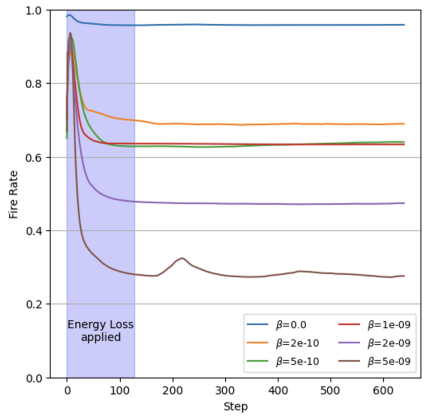

Figure 8. Average cell-wise fire rate (alive cells only) of AdaptNCA during persistence testing. The blue zone corresponds to the steps during which, in training, the fire rates were used for energy minimization. Values averaged over 80 trials for each model.

A peak of the average fire rates is observed in Figure 8 in the beginning, as the lizard is quickly growing into its general shape, corresponding to stages of high cell proliferation. Then, the fire rate stabilizes to a lower value, as the details of the lizard are being finished, such as the eyes and tail. The fire rates then stay constant for the rest of the evolution. This is because the state grid has reached relative stability, and since the fire rates are computed directly from the hidden states, these also do not vary after morphogenesis is completed. AdaptNCA models with $\beta = 1 \times 10^{-9}$ and $\beta = 5 \times 10^{-10}$ show very similar fire rate evolution profiles, despite the final state of the latter being considerably worse (Figure 7), showing that the slight decrease in β lead to a less ideal energy distribution among cells.

3.2.1. Equivalent NCA

We define an equivalent regular NCA to a certain AdaptNCA as the NCA with a constant fire rate p equal to the final average fire rate of the AdaptNCA. This NCA uses an identical main neural network architecture, and its updates are computed with Eq. (1), with the update bit $u^{(t)} \sim B(1, p)$. This is necessary in order to fairly compare the AdaptNCA and NCA models. The equivalent NCA are trained using the same training hyperparameters, but using only the reconstruction loss as the loss function. The original implementation of NCA, without $tanh$ activation or gradient clipping, was also trained in the same manner. In Table 2, the equivalent fire rates for the trained AdaptNCA models are presented. For $\beta = 5 \times 10^{-10}$ and $\beta = 1 \times 10^{-9}$, the final fire rates were very similar, therefore they'll be compared to the same NCA model.

Table 2. Equivalent NCA fire rates to the used AdaptNCA models.

AdaptNCA β	Equivalent NCA Fire Rate
0	0.958
2×10^{-10}	0.689
5×10^{-10}	0.634
1×10^{-9}	0.634
2×10^{-9}	0.471
5×10^{-9}	0.278

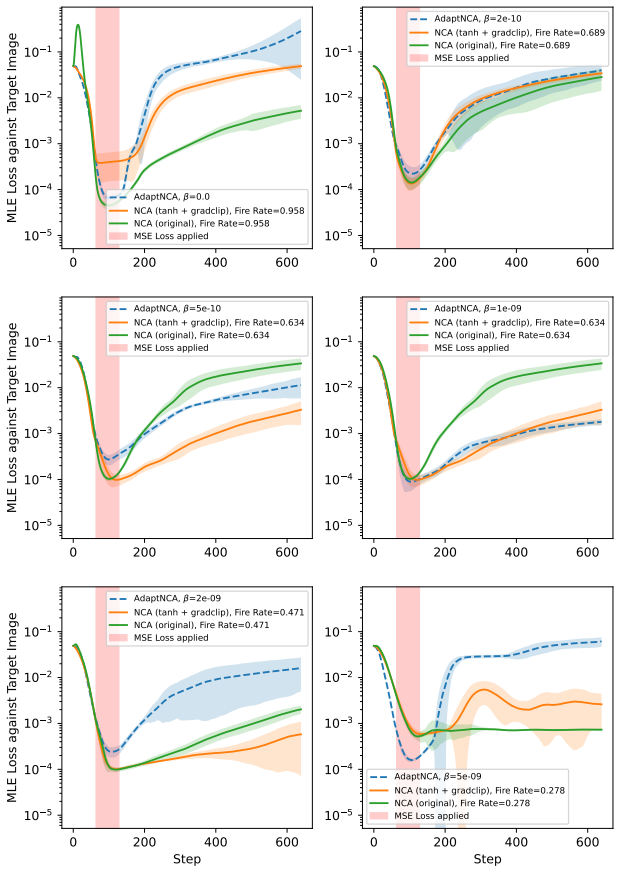

Figure 9. Persistence comparison between AdaptNCA and equivalent regular NCA. The red zone represents $U(68, 128)$, corresponding to the distribution of final steps during training. Values averaged over 80 trials for each model and error bands correspond to one standard deviation.

Table 3. Percentage differences in final reconstruction losses in the persistence test, compared to equivalent original NCA.

Energy β	AdaptNCA	NCA (tanh+gradclip)
0	+5356.2%	+837.1%
2×10^{-10}	+40.8%	+20.9%
5×10^{-10}	-66.5%	-90.2%
1×10^{-9}	**-94.7%**	-90.2%
2×10^{-9}	+697.9%	-71.5%
5×10^{-9}	+8290.0%	+256.9%

In Figure 9, each AdaptNCA model is compared to its NCA equivalent (both original and modified) with a persistence test. In Table 3, the changes in final reconstruction loss relative to the original NCA are shown. The modified NCA (with tanh activation and gradient clipping) outperform AdaptNCA for most β parameters. However, the model

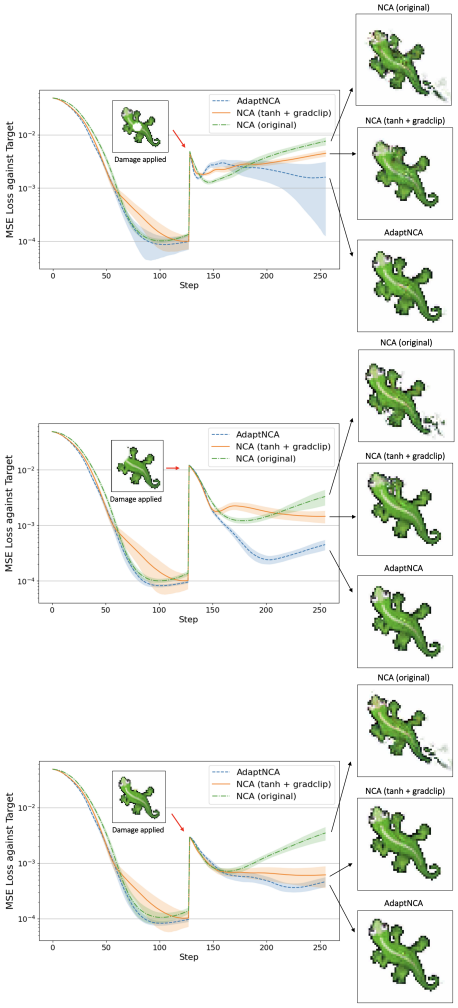

Figure 10. Regeneration comparison between AdaptNCA with $\beta = 1 \times 10^{-9}$ and equivalent original and modified NCA with fire rate $p = 0.634$. Top: center damage. Middle: head damage. Bottom: leg damage. Values averaged over 80 trials for each model and error bands correspond to one standard deviation.

that outperforms equivalent original NCA by the highest margin is AdaptNCA with $\beta = 1 \times 10^{-9}$, with a loss decrease of approximately 95%.

3.3. Regeneration Tests

In this section, the self-regeneration capabilities of the AdaptNCA model with the best persistence relative performance ($\beta = 1 \times 10^{-9}$) is compared to its equivalent NCA ($p = 0.634$). The models grow for 128 steps, after which some damage is inflicted on a region of the lizard's body by setting all state values in that region to 0. The models then grow for an additional 128 steps.

Figure 10 depicts the reconstruction loss for damage in the center, leg, and head regions. The regeneration of the

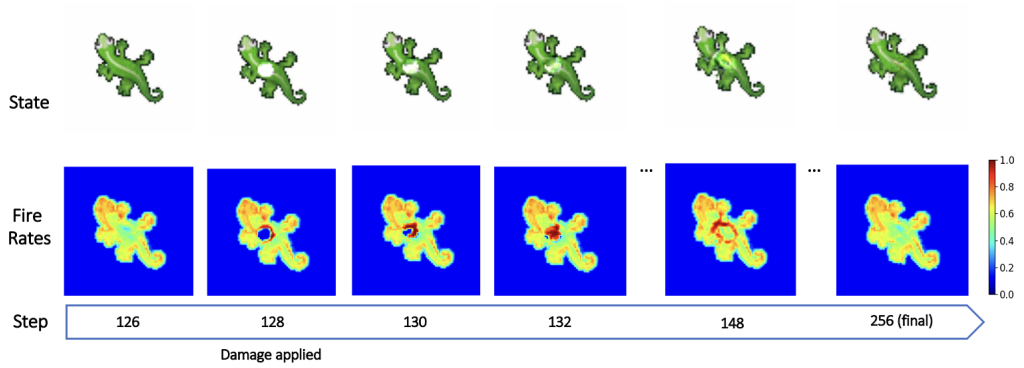

Figure 11. Regeneration of center region of AdaptNCA with energy parameter $\beta = 1 \times 10^{-9}$.

center region is the hardest, since during the training process, lizards without a center are never observed, although ones without legs or head appear during the growth phase. As can be seen in Table 4, AdaptNCA with $\beta = 1 \times 10^{-9}$ show superior regeneration capabilities compared to equivalent NCA, both original and modified. This results in a decrease of final reconstruction losses of up to 87%.

Table 4. Percentage differences in final reconstruction losses in the regeneration tests, compared to equivalent original NCA, for different damage regions. AdaptNCA with $\beta = 1 \times 10^{-9}$ and equivalent NCA with fire rate $p = 0.634$.

Region \ Model	AdaptNCA	NCA (tanh+gradclip)
Center	**-79.0%**	-41.0%
Leg	**-86.7%**	-82.4%
Head	**-86.4%**	-56.0%

The regeneration process can be seen in Figure 11. Damage to an area increases the fire rates of its surrounding cells as they try to quickly regenerate the damaged body part. After regeneration, fire rates return to their normal values. It is worth reiterating that all AdaptNCA were trained for *growth*, without any damage applied during training.

4. Future Work

There are several potential directions for further research and improvements. Experiments on different patterns with variable degrees of asymmetry might be conducted to replicate these results. To make the comparison with equivalent NCA more fair, one could consider forcing the average of the AdaptNCA fire rates to be constant throughout time. Furthermore, altering the design of the fire rate computation by using an incremental approach similar to the state updates would remove the fire rate's direct dependency on the hidden state. Finally, one could incorporate the fire rates into the perception vector and feed them back into the network at each step.

5. Conclusion

In this paper, we present Adaptive-Energy Neural Cellular Automata (AdaptNCA), an improved NCA model that addresses some of the limitations of the original NCA model (Mordvintsev et al., 2020). In AdaptNCA, cell state updates are bounded by adding a $tanh$ activation to the neural network, and weight gradient norms are clipped to improve training stability and biological accuracy. An adaptive cell-wise fire rate is computed by an additional neuron, and the fire rate is interpreted as the square root of the cell's energy expenditure. An energy term is added to the loss function, corresponding to the total energy spent by the model while growing. A Gumbel-Softmax distribution is used to generate the update boolean, as a differentiable approximation of a Bernoulli process. AdaptNCA trained with an energy parameter $\beta = 1 \times 10^{-9}$ are shown to have better persistence and self-regeneration capabilities than equivalent original NCA, achieving reconstruction loss values that are up to 95% and 87% lower, respectively.

We believe AdaptNCA could prove to be useful in constructing biologically and physically accurate morphogenesis models in developmental biology, especially if paired with more sophisticated cell proliferation and communication models.

References

Barragán-Vite, I., Seck Tuoh Mora, J. C., Hernandez-Romero, N., Marín, J. M., and Hernández-Gress, E. S. Distributed control of a manufacturing system with one-dimensional cellular automata. *Complex.*, 2018: 7235105:1–7235105:15, 2018.

Chopard, B. and Masselot, A. Cellular automata and lattice boltzmann methods: a new approach to computational

fluid dynamics and particle transport. *Future Generation Computer Systems*, 16(2):249–257, 1999. ISSN 0167-739X.

Ding, J. and Huang, H. A cellular automaton model of public transport system considering control strategy. *Journal of Transportation Systems Engineering and Information Technology*, 10(3):35–41, 2010. ISSN 1570-6672.

Ermentrout, G. B. and Edelstein-Keshet, L. Cellular automata approaches to biological modeling. *Journal of theoretical Biology*, 160(1):97–133, 1993.

Gardner, M. Mathematical games. *Scientific American*, 236 (5):128–138, 1977.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Losiri, C., Nagai, M., Ninsawat, S., and Shrestha, R. Modeling urban expansion in bangkok metropolitan region using demographic–economic data through cellular automata-markov chain and multi-layer perceptron-markov chain models. *Sustainability*, 8:686, 07 2016.

Mordvintsev, A., Randazzo, E., Niklasson, E., and Levin, M. Growing neural cellular automata. *Distill*, 2020. https://distill.pub/2020/growing-ca.

Mordvintsev, A., Randazzo, E., and Fouts, C. Growing isotropic neural cellular automata. *arXiv preprint arXiv:2205.01681*, 2022.

Niklasson, E., Mordvintsev, A., and Randazzo, E. Asynchronicity in neural cellular automata. In *ALIFE 2021: The 2021 Conference on Artificial Life*. MIT Press, 2021.

Sarkar, P. A brief history of cellular automata. *Acm computing surveys (csur)*, 32(1):80–107, 2000.

Stooke-Vaughan, G. A. and Campàs, O. Physical control of tissue morphogenesis across scales. *Current opinion in genetics & development*, 51:111–119, 2018.

Turing, A. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London Series B*, 237(641):37–72, 1952.