3D Neural Cellular Automata

October 29, 2021

Giorgio Becherini

Abstract

Natural organisms are able to develop into complex multicellular anatomies starting from a single egg cell and, to some degree, regenerate lost organs and tissues with a surprising level of accuracy. The work *Growing Neural Cellular Automata* (Mordvintsev et al., 2020) provides a method for simulating this type of cellular interaction on the 2-dimensional domain of images. This project aims to expand and apply the method to the 3-dimensional domain of voxelized volumes.

1. Introduction

A cellular automaton is a system composed by a number entities, the cells; they encode a set of rules that guide the generation/regeneration of a structure (a multicellular organism). Each cell receives signals only from its immediate surroundings and uses them to ”choose” an action to perform in order to build such structure.
In the proposed model, each cell is a voxel in a discretized 3D space that is able to ”perceive” its surroundings, i.e. the state of the adjacent voxels.
Given a desired target structure, the purpose of the model is to allow the automated learning of a function that encodes the set of rules needed to generate such target, starting from a single cell.
The code for the project along with some video demonstrations can be found at
https://github.com/gfgb/3DNeuralCellularAutomata

2. Method

As in (Mordvintsev et al., 2020), each cell state is represented by a 16-dimensional vector. The first 4 channels represent the RGBA signal, the remaining 12 ”hidden” channels are used by the learnable update rule as extra information. The alpha channel represents the state of growth of a cell; a cell is considered as ”growing” if there is a mature cell (alpha grater than 0.1) in its neighborhood.

Email: Giorgio Becherini <becherini.1478745@studenti.uniroma1.it>.

Deep Learning and Applied AI 2021, Sapienza University of Rome, 2nd semester a.y. 2020/2021.

Perception. The perception of the surroundings of a voxel cell is simulated by convolving the input volume with the 3-dimensional Sobel operator, composed by three 3x3 kernel, one for each direction.
The results of the convolution of each of the three kernels (fixed or trainable) or is a 16-dimensional vector which is concatenated with the others (plus the original 16-dimensional vector) into a single 64-dimensional *perception vector*.
The Sobel operator represents a coarse approximation of the gradient at a given point in the volume. As explained in (Mordvintsev et al., 2020), the choice of a fixed kernel is motivated by the fact that real life cells often rely *only* on chemical gradients to guide the organism development. However, using a trainable convolutional layer yields slightly better results, at the cost of a less truthful modelization of the biological process.

Update rule. The perception vectors are fed to a neural network comprised of two fully connected layers (used as a 1x1x1 convolutions), with ReLU nonlinearity between them. This network learns an update rule (shared among all the cells) and outputs an incremental update to the previous state.

Stochastic cell update. To simulate the lack of synchronization between the cells in the system (which is more truthful to how things work from a biological point of view), not every cell is updated at each step. This behavior is modeled using a ”per-cell dropout”, called *fire rate*, that sets the update values of random cells to zero.

3. Experiments and results

Each experiment has an input, called *seed*, that consists of a single cell, and a target, that is the volume that we want the model to be able to generate. We ask the model to perform such generation within a random number of steps in a given range. The error between the generated output and the target is computed as a voxel-wise L2 loss between their RGBA values.

Experiment 1: growing. The NCA is trained to learn the rules for generating the target volume. Although the model is able to generate the target, it generally shows an unstable behavior as the number of updates exceeds the number of steps that the model has been trained with: we need the model to stop updating the volume once the target shape is achieved.

Experiment 2: persisting. To overcome the instability problem, the obvious solution would be to let the training iterate for a much larger number of steps; however, this would increase the training time drastically. However, (Mordvintsev et al., 2020) propose a sampling pool strategy that has minimal impact on the training time. Instead of using the seed as input, a pool of seed states is defined and sampled by the model at training time to obtain an input. The pool is updated with ”incorrect” results that are outputted by the model. When the model samples an incorrect result, it will learn how to recover from that state. This increases the stability of the output over long periods of time.

Experiment 3: regenerating. By applying random ”damage” to an input (i.e. removing part of the cells) to some samples concurrently with the sampling pool strategy, we are able to impose the target volume as an *attractor* for a large number of states. This means that the model should be able to learn the rules for making damaged or incorrect states gravitate towards the correct one, thus allowing a state to regenerate into its desired shape.

During the training, the loss with respect to the target shape has followed a trend similar to that shown in Figure 1.

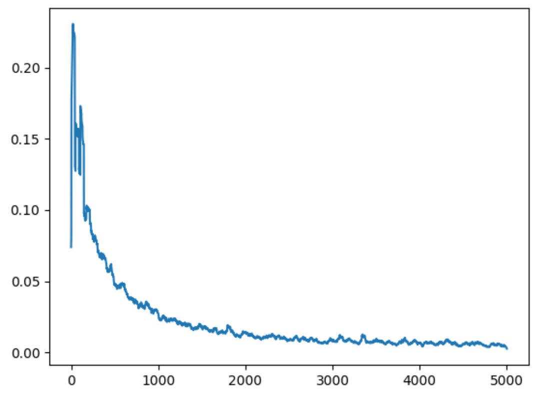

Figure 1. Typical loss during training. As more epochs are performed, the loss decreases until the model is not able to improve significantly. In this type of scenario we are not really worried about overfitting, since the function that we want the network to learn is specific to a single ”organism”.

4. Limitations

Expanding a neural cellular automaton to a third dimension comes at a very high computational cost.
In practice, given a resolution n for each dimension of the input, the output of each phase in 3D will be about n times bigger than the result of the same operations in 2D. Moreover, at each epoch, the network is executed a certain number of steps, and with it grows the size of the computational graph to be kept in memory.
This leaves very little room for manuver when choosing the resolution of the volumes. After various experiments, the best tradeoff I could find (with 16GB of RAM and 8GB of VRAM) is a resolution of 20x20x20=8000 voxels, a hidden dimension of 300 for the first fully connected layer and a maximum number of 32 steps at each epoch.
If a model trained with a small number of steps, there will be a bigger variation between input and output, allowing for less accurate results; however, 32 steps have proven to be enough for providing a reasonable generation/regeneration capability.

5. Results

A few video demonstrations can be found in the linked GitHub repository.

6. Conclusion

The method developed by (Mordvintsev et al., 2020) for growing neural cellular automata is applicable to a 3-dimensional domain and yields good results, although with big limitations in the resolution of the volume with current hardware.
Most attempts at obtaining a more efficient learning process would mean ”loosening” the requirement for a truthful modelization of the biological process of regeneration.
With this in mind, a possible way of increasing efficiency could be training on a different representation of the input. As an example, we could use a learnable Signed Distance Function as in (Park et al., 2019) to represent a surface in three dimensions. By doing a sparse sampling of the SDF, it could be possible to train a model using smaller inputs with respect to the full volume.

References

Mordvintsev, A., Randazzo, E., Niklasson, E., and Levin, M. Growing neural cellular automata. *Distill*, 2020. doi: 10.23915/distill.00023. https://distill.pub/2020/growing-ca.

Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.