Learning Graph Cellular Automata

D. Grattarola[1,4] · L. Livi[2] · C. Alippi[1,3]

[1] Università della Svizzera italiana; [2] University of Manitoba; [3] Politecnico di Milano; [4] grattd@usi.ch

1. Cellular Automata

Cellular automata (CA) are lattices of stateful cells with a transition rule that updates the state of each cell as a function of its neighbourhood configuration. By applying this **local rule** synchronously over time, we see interesting dynamics emerge.

Fig. 1: A 2-dimensional CA, the Game of Life

2. Graph Cellular Automata

Graph CA (GCA) are a generalisation of typical CA that only preserve the idea of locality:

- Arbitrary neighbourhoods (cells in a graph);
- State space is a generic vector space;
- Transition rule τ is a function of neighbours $\mathcal{N}(i)$:

$$\tau(\mathbf{s}_i) : \{\mathbf{s}_i\} \cup \{\mathbf{s}_j, \mathbf{e}_{ji} \mid j \in \mathcal{N}(i)\} \mapsto \mathbf{s}'_i,$$

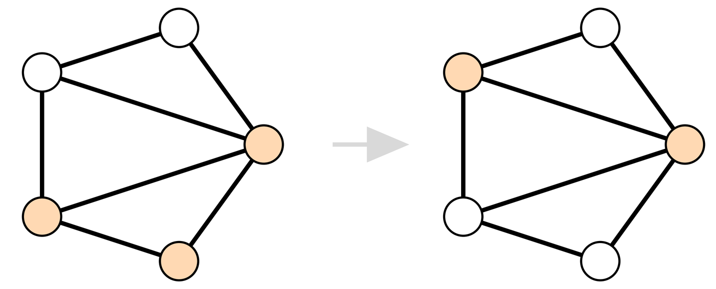

Fig. 2: Transition of a GCA

3. Graph Neural Cellular Automata

Key idea: use graph neural networks (GNNs) as learnable transition rules

$$\mathbf{s}'_i = \gamma\left(\mathbf{s}_i, \sum_{j \in \mathcal{N}(i)} \phi\left(\mathbf{s}_i, \mathbf{s}_j, \mathbf{e}_{ji}\right)\right).$$

Enables the design of rules by specifying desired behaviour.

Previous work: exclusively focused on regular grids, using convolutional neural networks [1–6]

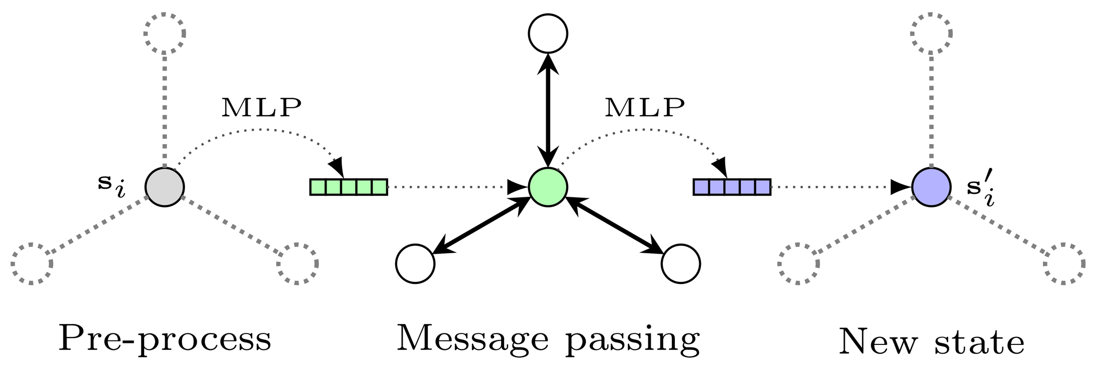

Fig. 3: Achitecture of the GNCA.

Universality results: we extend the previous results of Gilpin [4] to implement any M-state GCA rule:

- MLP for one-hot encoding states;
- Message-passing for pattern matching (use edge attributes as lookup keys).

4. Voronoi GCA

Voronoi GCA: binary states, random Delaunay graph.

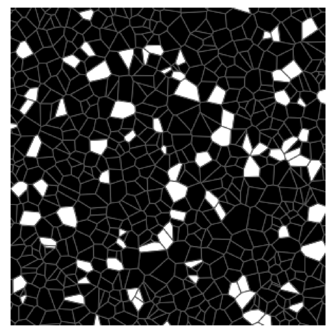

Fig. 4: Voronoi GCA

Outer-totalistic rule depends on the density ρ_i of alive neighbours:

$$\tau(\mathbf{s}_i) = \begin{cases} \mathbf{s}_i, & \text{if } \rho_i \leq \kappa \\ 1 - \mathbf{s}_i, & \text{if } \rho_i > \kappa. \end{cases}$$

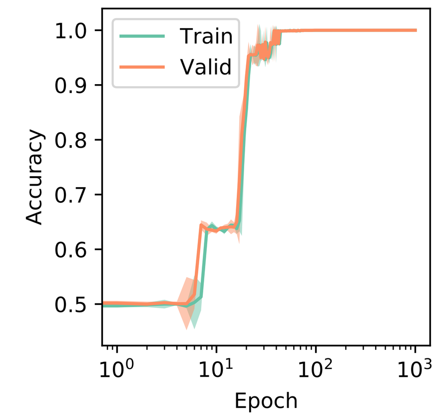

Fig. 5: Accuracy

5. Boids GCA

Boids GCA [7]: continuous multidimensional states, dynamic graph.
Results: the GNCA learns to imitate the flocking behaviour.

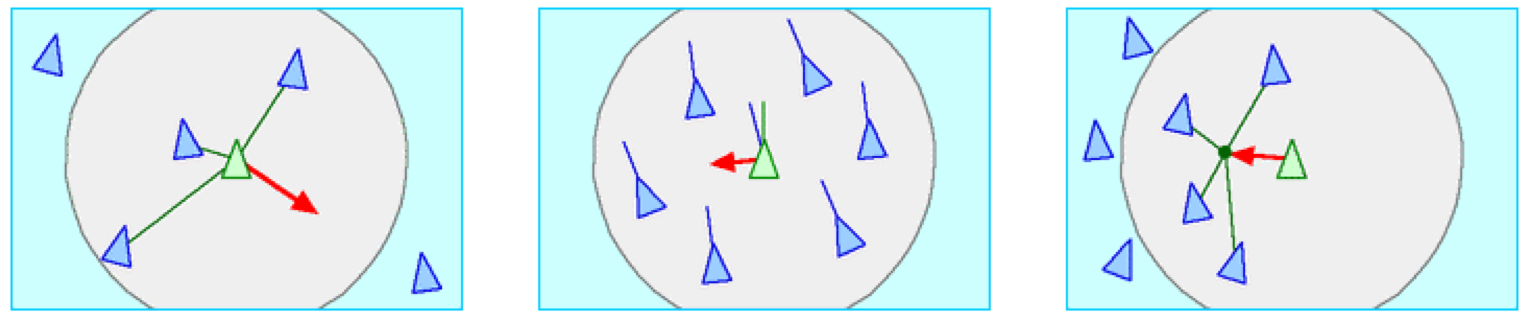

Fig. 6: Separation, alignment, cohesion

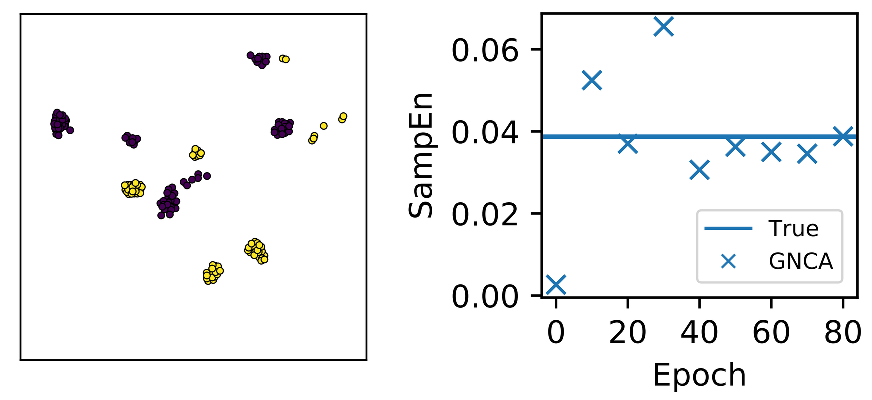

Fig. 7: GNCA flocks and complexity

6. Morphogenesis

GCA: point clouds, states are coordinates.
Task: converge to a desired target state.
Results: the GNCA learns stable rules most times. Sometimes, it oscillates around target.

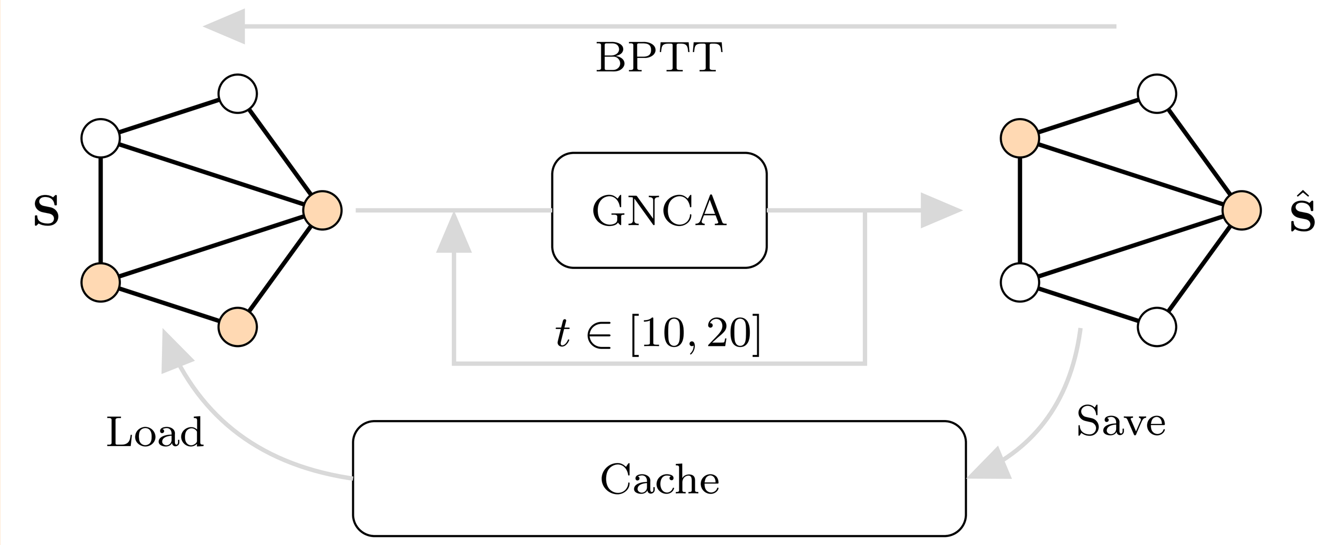

Fig. 8: Training GNCA with BPTT and a states cache.

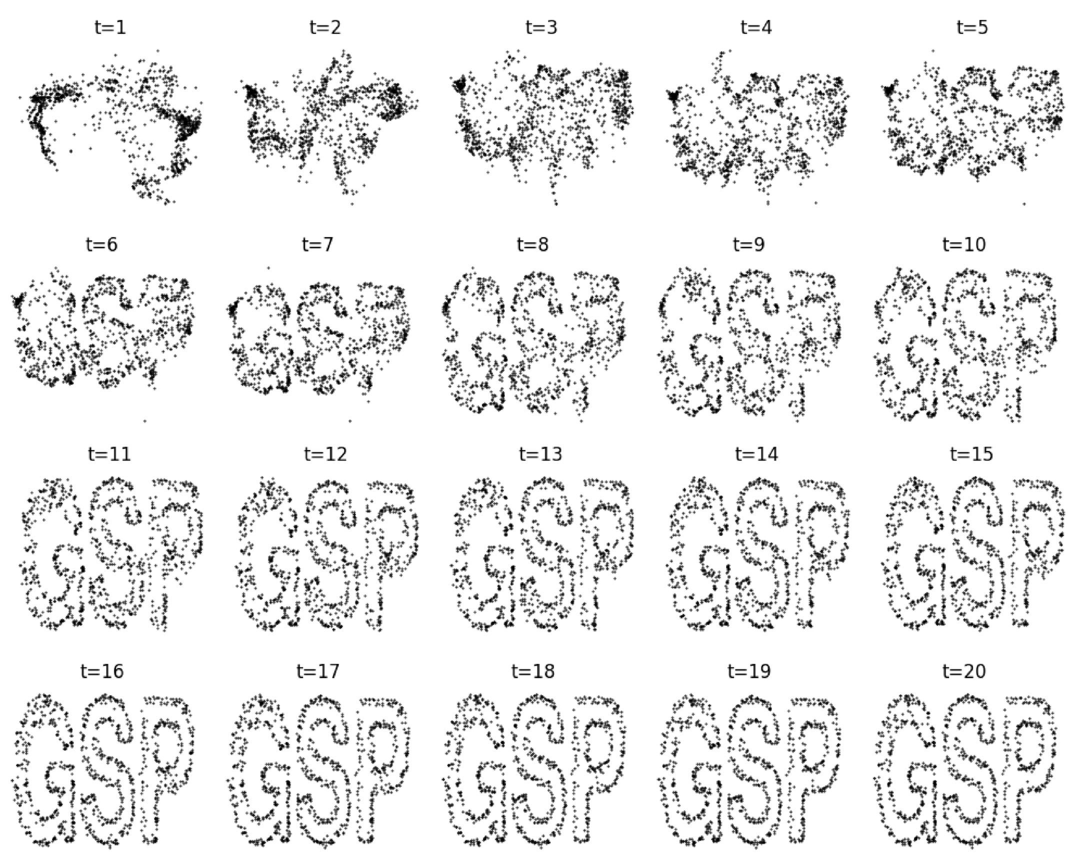

Fig. 9: Convergent GNCA

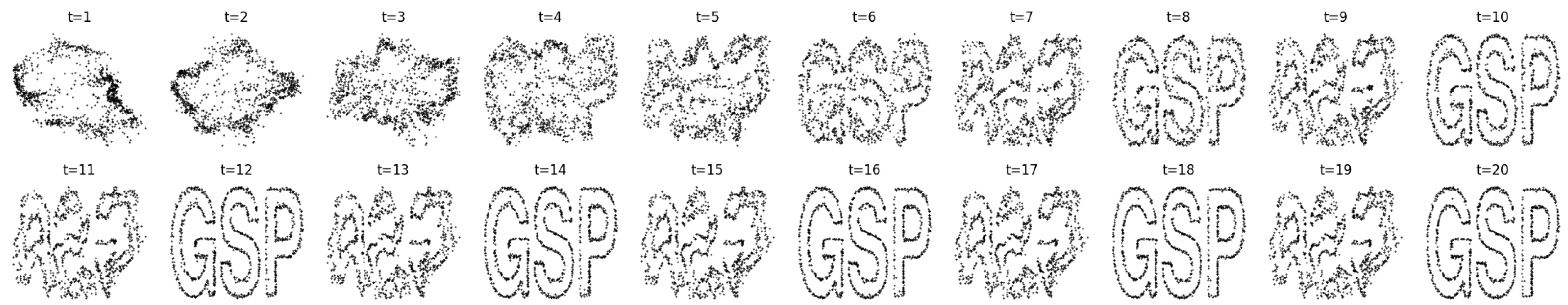

Fig. 10: Oscillating GNCA

References

[1] N Wulff and J A Hertz. Learning cellular automaton dynamics with neural networks. *Neural Information Processing Systems*, 1992.

[2] Wilfried Elmenreich and István Fehérvári. Evolving self-organizing cellular automata based on neural network genotypes. In *International Workshop on Self-Organizing Systems*, 2011.

[3] Stefano Nichele et al. Ca-neat: evolved compositional pattern producing networks for cellular automata morphogenesis and replication. *IEEE Transactions on Cognitive and Developmental Systems*, 2017.

[4] William Gilpin. Cellular automata as convolutional neural networks. *Physical Review E*, 2019.

[5] Alexander Mordvintsev et al. Growing neural cellular automata. *Distill*, 2020.

[6] Ettore Randazzo et al. Self-classifying mnist digits. *Distill*, 2020.

[7] Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. *Annual Conference on Computer graphics and interactive techniques*, 1987.

Code: github.com/danielegrattarola/GNCA · **ArXiv:** arxiv.org/abs/2110.14237
Blog: danielegrattarola.github.io/posts/2021-11-08/graph-neural-cellular-automata