Artificial Life and Digital Evolution

Artificial Life (biological life is difficult!)

"Artificial life attempts to understand the essential general properties of living systems by synthesizing life-like behavior in software, hardware and biochemicals" [1]

- **Soft artificial life**
  - Digital simulations of life-like processes and organisms
- Hard artificial life
  - Hardware implementations (e.g. soft robots) → co-evolution of controllers and morphology (physical structure)
  - Robotic swarms
- Wet artificial life
  - Synthesizes living systems out of biochemical substances → synthesize an artificial cell

Not (necessarily) an engineering effort

We want to discover behaviors, get answers to questions, design useful models

We do not (necessarily) want to solve a given problem within a certain margin of error

Pioneering Artificial Life

First thing first: C. G. Langton, *Artificial Life [2]* (1989)
"By **extending** the empirical foundation upon which biology is based **beyond the carbon-chain of life** that has evolved on Earth, **Artificial Life** can contribute to theoretical biology **by locating "life as we know it" within** the larger picture of **"life as it could be."**" [2]

Does Artificial Life contain or is it contained within "life as we know it" ?

John von Neumann (of course) → Cellular Automata

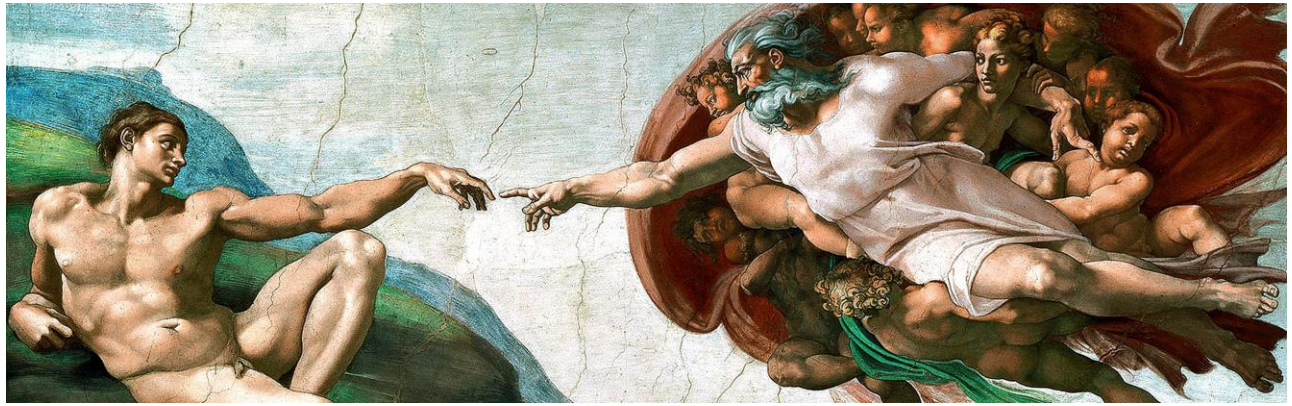
Norbert Wiener → information theory, homeostatis

John Holland → genetic algorithms

What are the origins of (biological) life?

If we want to study/understand life we first need to generate it (?)

- An **iterative** process driven by **evolution** (infinite loops are powerful)
- Initial conditions
  - how life *started*
- (biological) Evolution algorithm
  - how life *progressed* and became so *diverse*
  - *How modern organisms have descended from ancient ancestors*

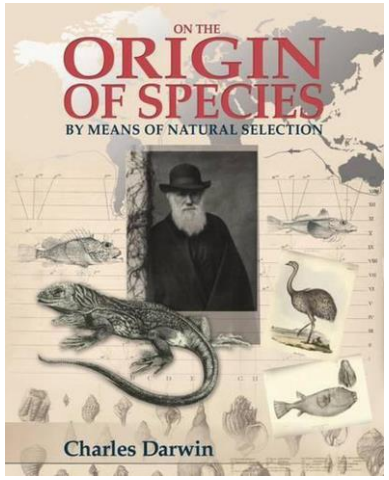


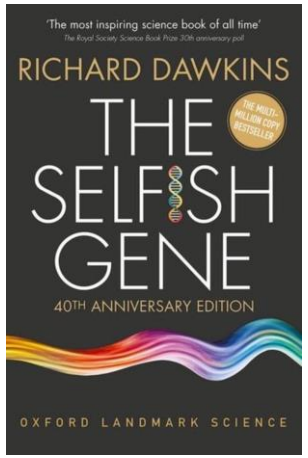
Is this the only way to simulate life?

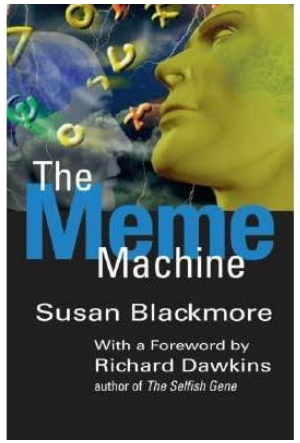
Evolution in biological life

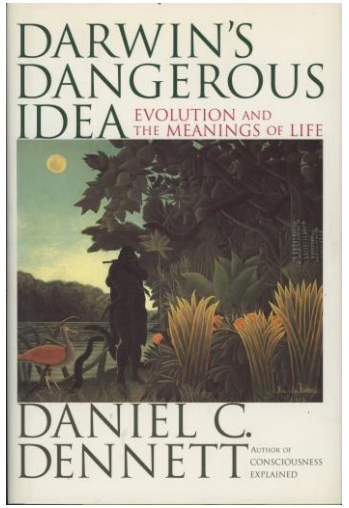
https://evolution.berkeley.edu/

- Charles Darwin's «On the Origin of Species» (1859)
  - Seminal book for evolutionary biology
- Richard Dawkins' «The Selfish Gene» (1976)
- Daniel Dennett's «Darwin's Dangerous Idea» (1995)
- Susan Blackmore's «The Meme Machine» (1999)









Evolution by natural selection

Necessary and sufficient conditions for *evolution by natural selection*

- Differential reproduction → not all individuals of the population reproduce at the same rate
  - e.g. due to some property of the environment that favors a certain trait
  - Survival of individuals that are *fit enough* (not only the fittest!)
- Heredity → traits can be passed on to the next generation
- Variation → different traits in a population

Advantageous traits survive in the population, while others disappear → **convergence to a homogenous population?**

Cool things about evolution

- **Punctuated equilibria**
  evolution can also happen “fast”
  Eldredge, N. & Gould, S.J., "Punctuated equilibria: an alternative to phyletic gradualism" (1972) pp 82-115 in "Models in paleobiology", edited by Schopf, TJM Freeman, Cooper & Co, San Francisco.
- **Adaptation**
  a feature favored by natural selection becomes increasingly important for a given task
- **Exaptation**
  finding a different use for a trait previously developed for a different use

What about diversity?

- Random Mutation
  - «create» new genes
- Recombination (… sex!)
  - cross-over, genetic shuffling
- Natural selection → changes by fitness
  - Fitness = how good a genotype is at transmitting its genes, compared to another genotype
  - Sexual selection, artificial selection
- **Divergent process** → it does not collapse, rather it creates novelty

Digital evolution

Life originated from an iterative process driven by evolution

- Initial conditions, environment
- (biological) Evolution algorithm

If we can simulate both we can “create” artificial life

Initial conditions

- We need to *approximate* the initial conditions (an environment, a population composed by multiple individuals)
  - Is the simulation of a hurricane a hurricane?
- Complex, chaotic system: any approximation will lead to large differences in future states
- Unless there exists a Laplace's demon we don't really have other options..

Evolutionary algorithms

Since the 1950s [7, 8], to solve **optimization problems**

We already start from a well-defined process: evolution by natural selection

- which may or may not be a sufficiently accurate approximation of the real process

We can simulate the necessary and sufficient conditions

- Variation
- Differential reproduction
- Heredity

Evolutionary plant (1957)

Not the one you are thinking of… [8]

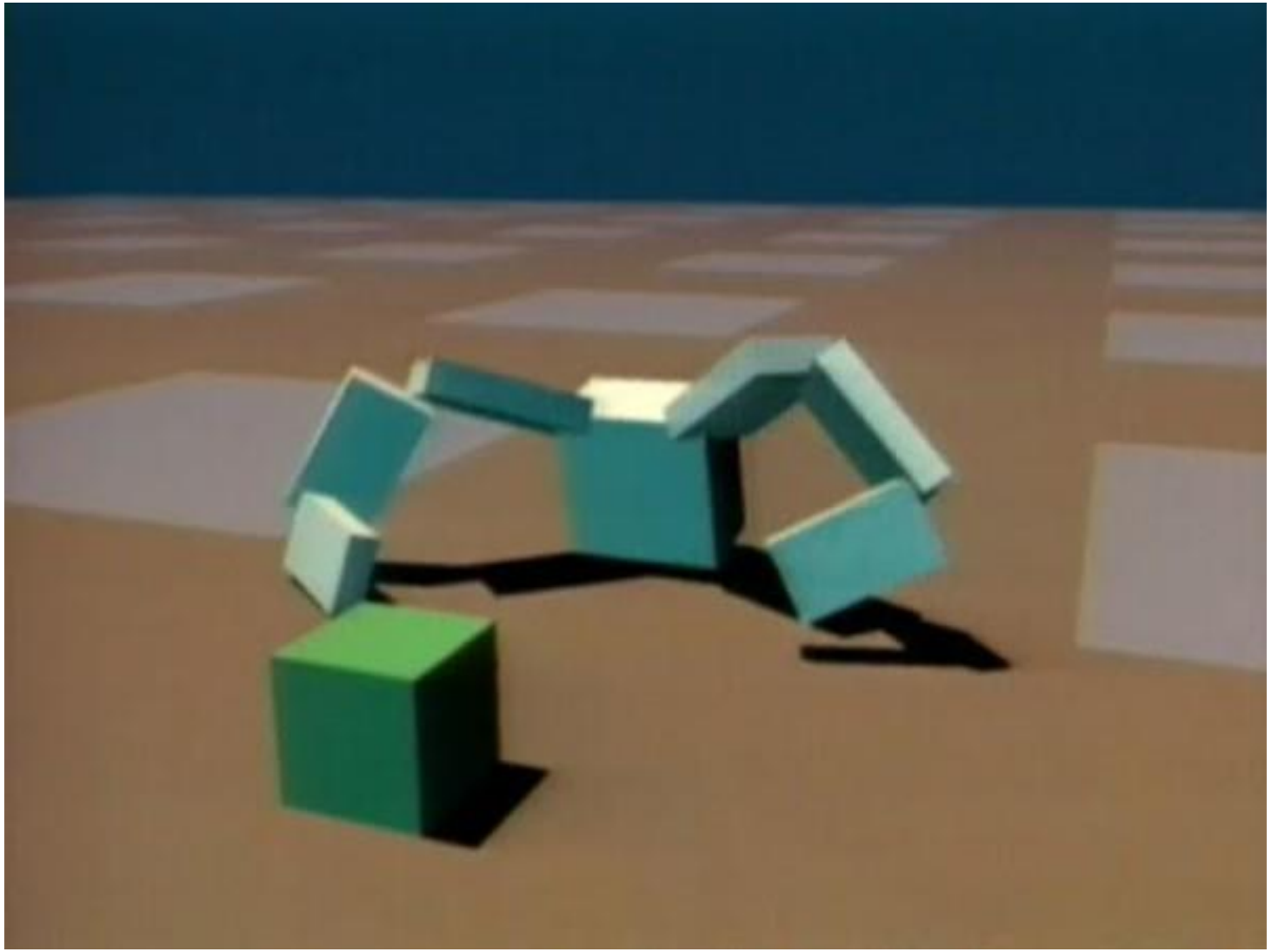
The information coming from the experiment was recorded by writing in chalk on an ordinary blackboard. Alternatively, wax pencil on a white plastic board or magnetic letters and numbers on a steel board could have been used. The essential thing is that it should be a simple matter to erase or remove one number and replace it by another. The

Skeleton of EAs

- Initialize **population** P_0 of **individuals** with N features
  - Features expressed in domain-specific language (bit-string, or real-valued numbers, graphs…)
- **Evaluate** fitness of population P_0
- For each step *t=1, 2, …:*
  - Select **parents** from P_0 (e.g., based on the fitness)
  - **Reproduction** and **recombination** of features to get **children**
  - Apply **mutation** to features (e.g., at random, adaptive mutation rate)
  - **Evaluate** fitness of new population
  - **Select** subset of K **survived** individuals to get P_t (e.g., random sample weighted by fitness)

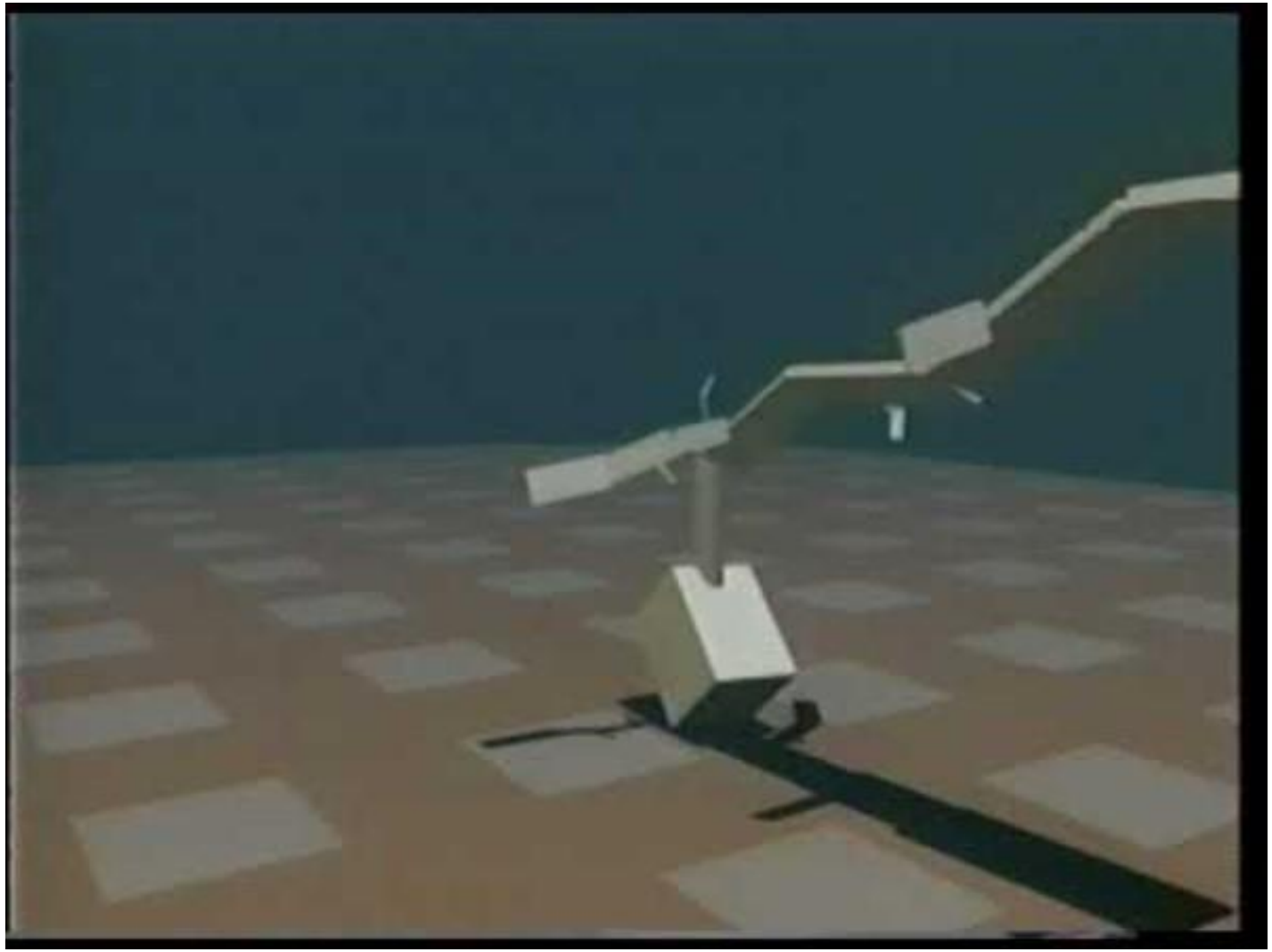
EAs details/variants

- Do the **parents survive**? Do we replace K parents with K children?
- Focus on regions where fitness is high → exploit
  - **Hill-climbing** in the fitness space
- Select parents based only on the **relative rank** of the fitness
- Number of children proportional to the difference between average and individual fitness
- **Different mutation rates** for each individual's feature
- Recombination can average features, or can take the maximum/minimum
  - Crossover recombination – insert a *cut point* and switch features



Evolving digital creatures

- Creatures that can move in the space and learn to walk / swim / jump [4, 5, 6] (1993-1994)
- https://www.karlsims.com/evolved-virtual-creatures.html



Why walk at all?

Different types of Evolutionary Algorithms

Same structure, different applications/low-level details [9]

- **Evolutionary Strategies**
  - randomly mutate + evaluate + *recombine*
  - Evolutionary programming: randomly mutate + evaluate
- **Genetic Algorithms**
  - Bit string input representation for *genotypes*
  - Mutation is rare, recombination is the main factor
  - Fitness is relevant also to select parents
- Genetic Programming
  - Individual is a set of instructions (e.g., tree)
  - Genetic algorithms for code generation

Genetic programming

Evolving code that «fix» an existing code [11]

Fixing = passing unit tests.

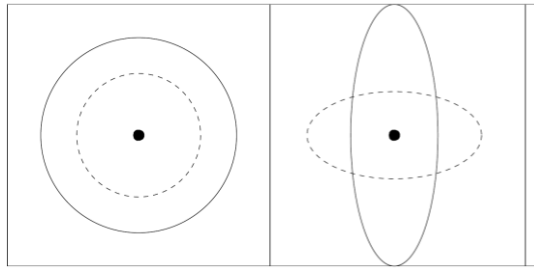
- Fixing sorting
  - Well… empty list is sorted
- Producing an output that matches the one in a text file
  - Evolution can cheat by deleting the target text file
- Bloating: code is full of “junk”
  - reduces the probability of a mutation

Evolution strategies 1/2 [14]

- **Stochastic** search algorithm to maximize $f: S \subseteq R^n \rightarrow R$
  - S is the **search space**
  - **Black-box** optimization, derivative-free: no information on the agent, the state function, no gradient etc..
  - Only info on points in the search space and their evaluation
- Start with a pool of individuals, evaluation + recombination + mutation
- Individual at beginning of simulation $x^0 \in R^n$
  - Phenotype, object or **decision parameters**
- A set of **strategy** («endogenous») **parameters** σ^0 for each individual
  - Mutation rate of decision parameters
- **Metric:** #evaluations required to converge *and* goodness of the final solution
  - computing the fitness can cost a lot!

Evolution Strategies 2/2

- Mutation = **adding random noise** (e.g. Gaussian)
  - Isotropic Gaussian (1 parameter –variance)
  - 1 parameter for each dimension (n parameters, Gaussian with diagonal covariance)
  - Gaussian efficiently explore the search space in all directions



- **Mixing number** ρ: how many parents per offspring
  - $\rho = 1$ → cloning
- **Random** selection of parents
- Recombination – many possibilities
  - E.g., selecting features *j* from parent *i* (at random), average (multi-recombination), weighted average…
  - What about strategy parameters?

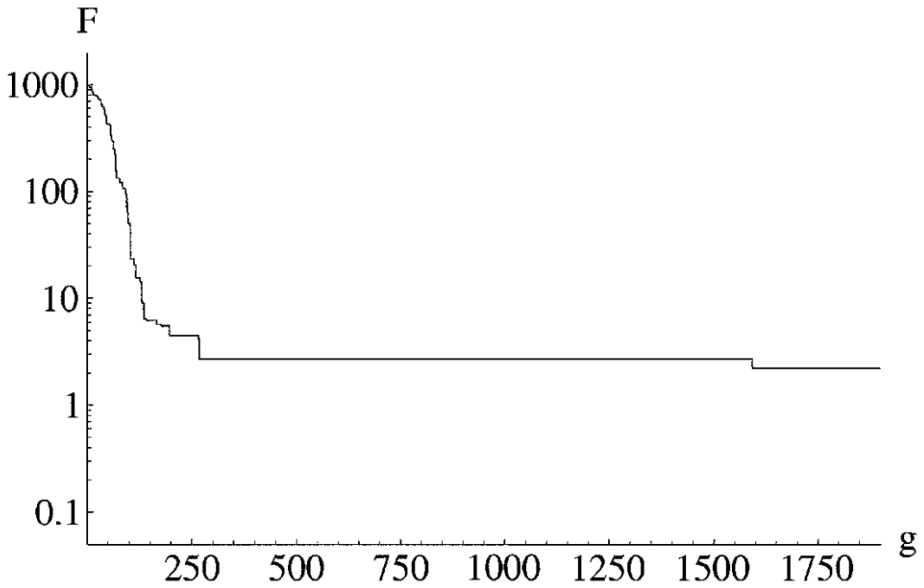
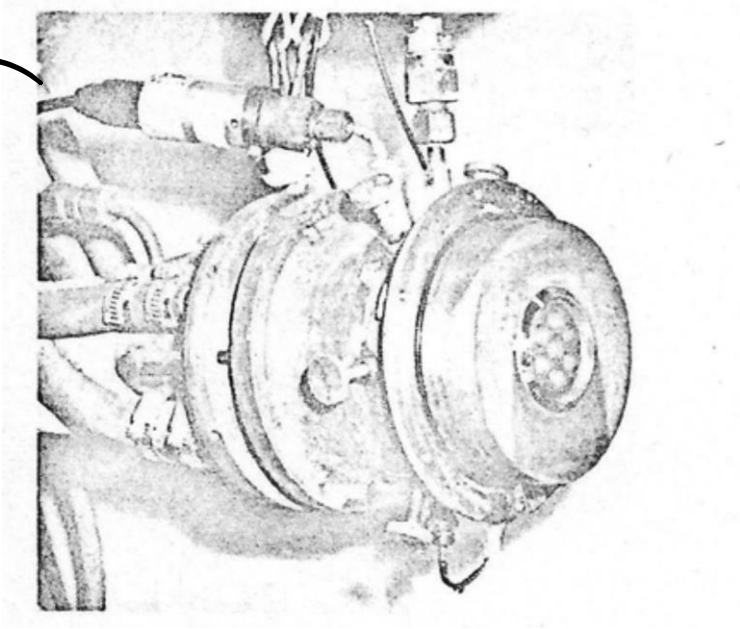


Figure 8. Evolutionary dynamics of a $(1 + 1)$-ES minimizing a sphere model. The mutation operator uses isotropic Gaussian mutations with *constant* mutation strength $\sigma = 1$ throughout the whole ES run. After a period of good performance the ES has lost its evolvability. Therefore, the mutation strength must be adapted appropriately.

Self-adaptive ES: adapting strategy parameters

Earli(est) results

tasks, e.g., the design of a 3D convergent-divergent hot water flashing nozzle (Schwefel, 1968; Klockgether and Schwefel, 1970). In order to be able to vary the length of the nozzle and the position of its throat, gene duplication and gene deletion was mimicked to evolve even the number of variables, i.e., the nozzle diameters at fixed distances. The perhaps optimal, at least unexpectedly good and so far best-known shape of the nozzle was counterintuitively strange, and it took a while, until the one-component two-phase supersonic flow phenomena far from thermodynamic equilibrium, involved in achieving such good result, were understood.



$(\mu / \rho, \lambda)$-ES: an example

- Population $\boldsymbol{X^g} \in R^{\mu,n}$
- Parameter *step size* $\boldsymbol{\sigma^g} \in R^{\mu}$ (same for all features, Isotropic Gaussian)
- Select parents at random $\boldsymbol{P^g} \in R^{\rho,n}$
- Generate λ offsprings $(k = 1, \dots, \lambda)$
  - $\sigma_k^{g+1} = recombine(\boldsymbol{\sigma^g}|\boldsymbol{P^g})\ e^{\epsilon_k \sim N(0,1)}$ ← recombination + mutation of step sizes
  - $\boldsymbol{x_k^{g+1}} = \boldsymbol{recombine}(\boldsymbol{P^g}) + N(0, \sigma_k^{g+1})$ ← recombination + mutation of individuals
  - Evaluate fitness
- Keep **best** μ offsprings out of all λ offsprings
  - μ, λ hyper-parameters («exogenous»)
- Parents are always discarded

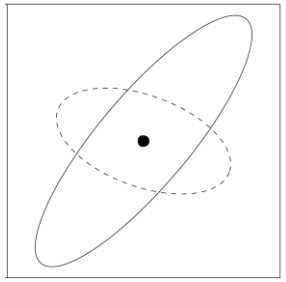
$(\mu / \rho + \lambda)$-ES

- **Offsprings also compete with parents**
  - best for large finite spaces / combinatorial problems
- Preserve the best so far (**elitist**)

> Had the $(\mu+1)$-ES already been laughed at because it makes use not only of the so far best individual to produce an offspring, but also of the second best, even the worst out of μ parents, one can imagine that the $(\mu+\lambda)$-ES was welcomed as a further step into a wrong direction, because it does not make immediate use of new information gathered by the next offspring. Instead, it delays the selection decision until all λ descendants are born. The step from the plus to the comma version finally seemed to climb to the top of mount nonsense in so far as now even better intermediate solutions can get lost and be replaced by worse ones. [15]

CMA-ES [45]

- Correlation matrix C
  - $\boldsymbol{C} = \boldsymbol{M}^{\boldsymbol{T}}\boldsymbol{M}$ → rotation matrix
  - $\frac{n(n-1)}{2}$ additional strategy parameters
  - Allows rotations of the ellissoids
- Population sampled from multivariate Gaussian $N(\boldsymbol{m}, \boldsymbol{C}^{\boldsymbol{g}})$
  - $\boldsymbol{x}_{\boldsymbol{k}}^{\boldsymbol{g+1}} = \boldsymbol{m}^{\boldsymbol{g}} + \sigma^{g} N(\boldsymbol{0}, \boldsymbol{C}^{\boldsymbol{g}})$
- CMA-ES updates $\boldsymbol{m}, \boldsymbol{C}, \sigma$
  - Section 3 of this tutorial: https://arxiv.org/abs/1604.00772
  - Quite involved, but very effective



CMA-ES: update mean

- Weighted average of μ individuals out of λ candidates (offsprings)
  - $\boldsymbol{m}^{\boldsymbol{g+1}} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{\mu}} w_i \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{g+1}}$
  - Weights are positive and sum to 1
  - Larger weights to better offspring

CMA-ES: update C (and step size)

- **Intuition:** Reproduce previous updates, but give priority to successful ones
  - C^{g+1} has a larger probability of reproducing successful x^g
  - Larger probability to generate meaningful update directions
- Assume current generation is enough to estimate C ($\sigma^g = 1$)
- Step 1: estimate previous C from samples
  - $\boldsymbol{C_\lambda} = \frac{1}{\lambda}\sum_{i=1}^{\lambda}\left(\boldsymbol{x_i^{g+1}} - \boldsymbol{m^g}\right)\left(\boldsymbol{x_i^{g+1}} - \boldsymbol{m^g}\right)^T$
  - unbiased estimator of $\boldsymbol{C^g}$ → reproduce previous updates
- Step 2: improve C with a weighted average from best offsprings (same as for **m**).
  - $\boldsymbol{C_\mu^{g+1}} = \frac{1}{\mu}\sum_{i=1}^{\mu} w_i \left(\boldsymbol{x_i^{g+1}} - \boldsymbol{m^g}\right)\left(\boldsymbol{x_i^{g+1}} - \boldsymbol{m^g}\right)^T$
  - $\boldsymbol{C^{g+1}} = \left(1 - c_\mu\right)C^g + \mathrm{c}_\mu \mathbf{C_\mu^{g+1}}$
- We can include more generations in the update (*evolution path)…*
  - Update of the step size takes on a similar form

The CMA-ES (Evolution Strategy with Covariance Matrix Adaptation)

Consider $P^{(t)} = \mathcal{N}\left(\boldsymbol{\mu}^{(t)},\ {\sigma^{(t)}}^2 \boldsymbol{C}^{(t)}\right)$ where $\boldsymbol{\mu}^{(t)} \in \mathbb{R}^n$, $\sigma^{(t)} \in \mathbb{R}_+$, $\boldsymbol{C}^{(t)} \in \mathbb{R}^{n \times n}$

- $\boldsymbol{\mu}^{(t)} \rightarrow \boldsymbol{\mu}^{(t+1)}$: Maximum likelihood update, i.e. $P(\boldsymbol{x}^{(t)}_{\text{selected}} | \boldsymbol{\mu}^{(t+1)}) \rightarrow \max$
- $\boldsymbol{C}^{(t)} \rightarrow \boldsymbol{C}^{(t+1)}$: Maximum likelihood update, i.e. $P(\frac{\boldsymbol{x}^{(t)}_{\text{selected}} - \boldsymbol{\mu}^{(t)}}{\sigma^{(t)}} | \boldsymbol{C}^{(t+1)}) \rightarrow \max$, under consideration of prior $\boldsymbol{C}^{(t)}$ (otherwise $\boldsymbol{C}^{(t+1)}$ becomes singular).
- $\sigma^{(t)} \rightarrow \sigma^{(t+1)}$: Update to achieve conjugate perpendicularity, i.e. conceptually $(\boldsymbol{\mu}^{(t+2)} - \boldsymbol{\mu}^{(t+1)})^{\mathrm{T}} {\boldsymbol{C}^{(t)}}^{-1} (\boldsymbol{\mu}^{(t+1)} - \boldsymbol{\mu}^{(t)}) / {\sigma^{(t+1)}}^2 \rightarrow 0$

http://www.cmap.polytechnique.fr/~nikolaus.hansen/searchandcmaslides.pdf

ES for RL [17]

- Gaussian policy π_θ, $\theta \sim p(\theta|\psi) = N(\psi, \sigma^2 I)$
- Discounted reward $F(\theta|\psi)$ → $argmax_\psi F(\theta|\psi)$
- Reparameterize $\bar{\theta} = \theta + \epsilon\sigma$: $E_{\theta \sim p_\psi} F(\theta) = E_{\epsilon \sim N(0,1)} F(\theta + \epsilon\sigma) = E_{\epsilon \sim N(0,1)} F(\bar{\theta})$
- Score function estimator for gradient of expectations (REINFORCE-like approach)

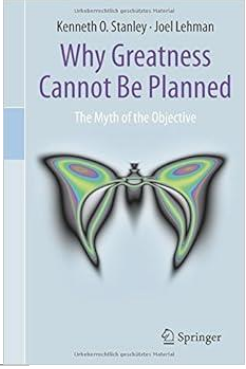
$$\nabla_\psi \mathbb{E}_{\theta \sim p_\psi} F(\theta) = \mathbb{E}_{\theta \sim p_\psi} \{F(\theta) \nabla_\psi \log p_\psi(\theta)\}$$

- $\boldsymbol{\nabla_\theta E_\epsilon F(\theta + \epsilon\sigma) = \frac{1}{n\sigma} \sum_\epsilon F(\theta + \epsilon\sigma)\epsilon}$
  - Use Monte Carlo and $\nabla \log p(\bar{\theta}) = \nabla \log N(\theta, \sigma^2 I)$, computed in $x = \bar{\theta}$

Algorithm 1 Evolution Strategies

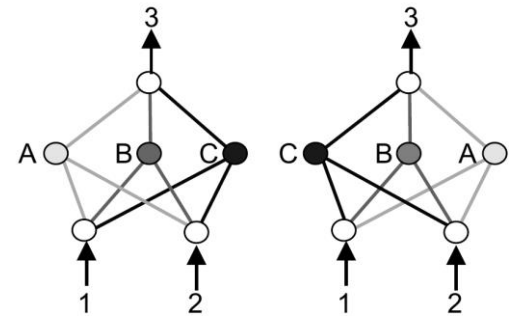
Input: Learning rate α, noise standard deviation σ, initial policy parameters θ_0
for $t = 0, 1, 2, \ldots$ **do**
  Sample $\epsilon_1, \ldots \epsilon_n \sim \mathcal{N}(0, I)$
  Compute returns $F_i = F(\theta_t + \sigma\epsilon_i)$ for $i = 1, \ldots, n$
  Set $\theta_{t+1} \leftarrow \theta_t + \alpha \frac{1}{n\sigma} \sum_{i=1}^{n} F_i \epsilon_i$
end for

ES for RL [17] - Algorithm



Neuroevolution with NEAT [24a]

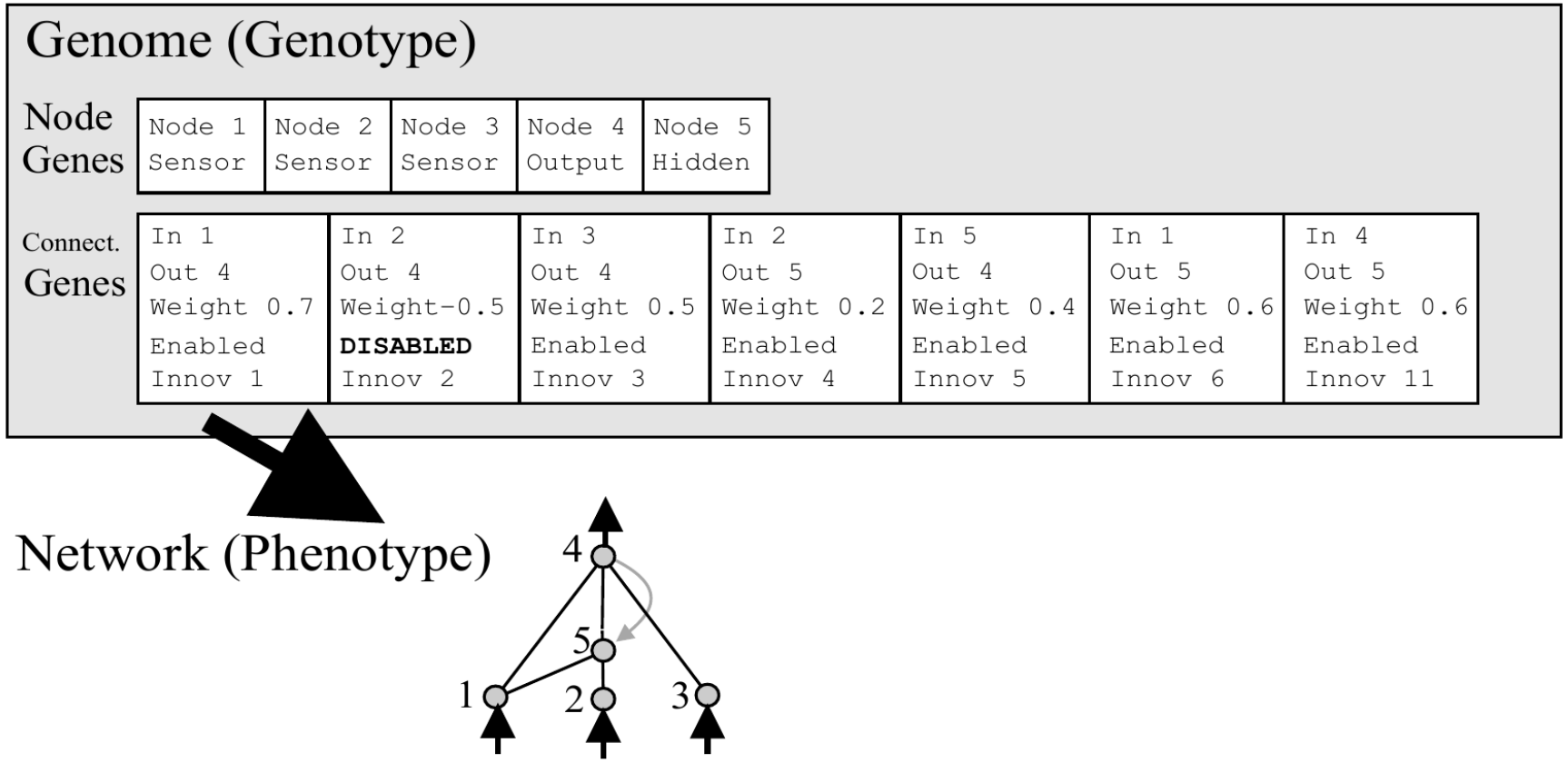
- NeuroEvolution through Augmenting Topologies with genetic algorithms
  - Evolve *topology* + weights vs. evolving weights for a fixed topology
- Starting from minimal solutions (architectures)
  - Let the evolution do the work, no fitness engineering
- **Direct encoding**: all node and connections in the genome
  - Indirect encoding: encode rules on how to build networks → introduce (un)wa
  - Linear encoding → allows gene alignment for crossover
    - Competing conventions/ Permutations problem

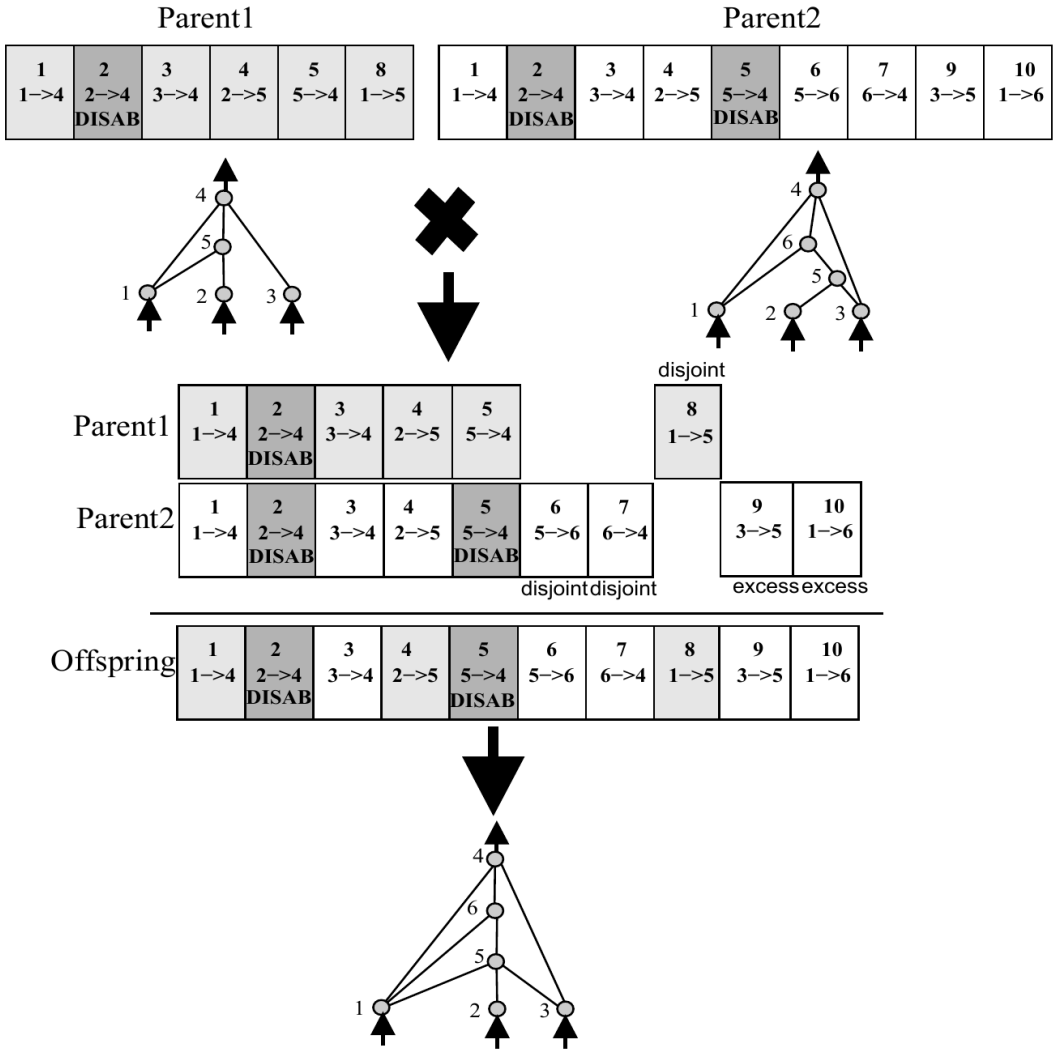


[A,B,C]
X[C,B,A]

Crossovers: [A,B,A] [C,B,C]
(both are missing information)

NEAT's direct encoding





NEAT's mutation

- Change both connections and nodes
  - **Randomly perturb** existing connections
  - **Create** a new randomly-weighted connection
  - **Add node** by splitting one existing connection (in-weight=1, out-weight=previous weight)
    - Previous connection is disabled, not destroyed
- Increasing **global Innovation number** at each structural mutation
  - Allows to to track «gene» origin
  - Used in crossover to align genes with same innovation numbers (same original structure)

NEAT's speciation

- **Compete** with your own **niche** only
  - Topological innovation gives rise to a new species
- **Protects** recently-added networks
  - They have little hope of performing well
  - Species with many individuals are penalized (avoid take over)
- **Define species:** innovation number can be used to compute the distance in the topology space
  - $\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \Delta W$, $\delta_t \rightarrow$ compatibility threshold
  - E, D = excess/disjoint genes, N = #genes
  - ΔW = average weight difference of *matching* genes
- Fitness for an individual i of species s is the average distance to the other fitnesses values in the same niche
  - $f_i^s = \frac{f_i}{|s|}$

NEAT for Mario Kart

https://github.com/nicknlsn/MarioKart64NEAT

Tierra [11], Avida [13]

- Digital exploration of **ecological dynamics**
- Tierra: Software programs (creatures) that compete for CPU time and live in RAM space
- Avida: Software has dedicated resources, it can acquire more by *performing* tasks (e.g. binary multiplication)
- Fitness = living long enough (+ doing something)
  - Tierra evolved right away parasitism, immunity to parasitism, circumvention of immunity, hyper-parasitism, obligate sociality, cheaters exploiting social cooperation, and primitive forms of sexual recombination
  - “Hyper-parasites stole the CPUs of parasites, exhibiting energy parasitism

Evolvability

- **Evolvability: adaptive evolution**
  - Acquiring and developing new features, new abilities to survive and reproduce
  - Design genomes that have potential for further evolution (for phenotypical variation)
  - Small mutation rate → not enough variation
  - Large mutation rate → system collapses (mutations are rarely beneficial)
- Most systems, like Tierra, sooner or later reduce their evolvability
- Still no agreement on a standard evaluation protocol

Fostering evolvability

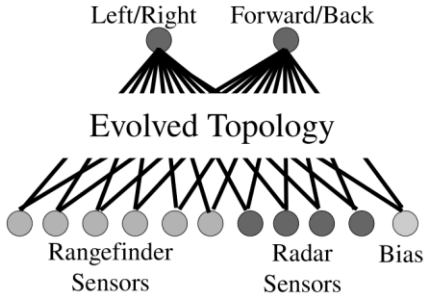
- Static fitness function can penalize individuals with large evolvability
  - Exploration is not always beneficial
  - Maximizing task-based fitness ≠ increasing evolvability
- More evolvable organisms → stronger to environment change
- Indirectly promote evolvability
  - Change the fitness function over time
  - Use several fitness functions
  - **Novelty Search** [20, 21] → prefer novel behaviors
  - Mass extinction (reset)
- ***Open-ended evolution (beyond objective-specific fitness) remains a challenge***

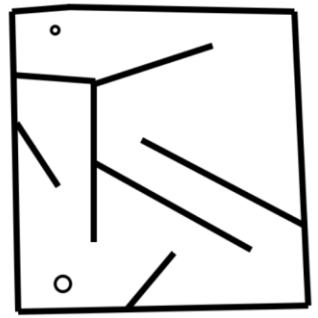
Novelty Search [20, 21]

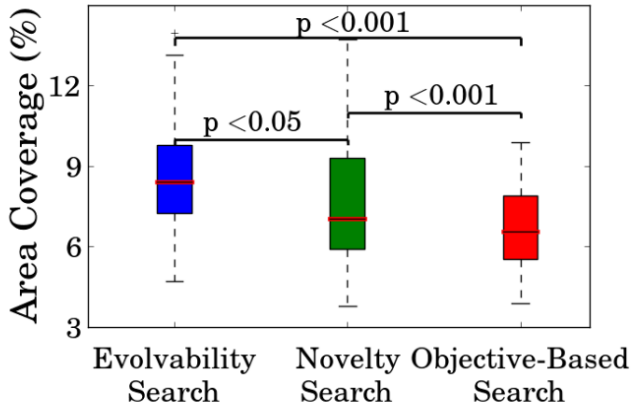
- The quest for novel behaviors will give rise to some good behaviors
  - In a biped locomotion task, falling in many different ways leads to walking
- In an RL task, reaching novel states implies advancing (e.g., Atari)
  - Novelty is an indirect way to create ever complex behaviors
  - Is it the most efficient one?
- **Novelty-based fitness** computed wrt archive of *past novel behaviors*
  - Novel behaviors are considered as *stepping stones* for future individuals (we want to keep them)
  - Novelty = sparseness in the behavior space (k-NN)
- **Co-evolution dynamics**: the fitness includes behaviors from multiple individuals
- It avoids deceptive fitness (no performance-oriented fitness here)
  - E.g. reaching a dead end close to the maze goal
- Combine it with NEAT → grow *novel* neural networks
- Still able to solve a given problem, but in many different ways!

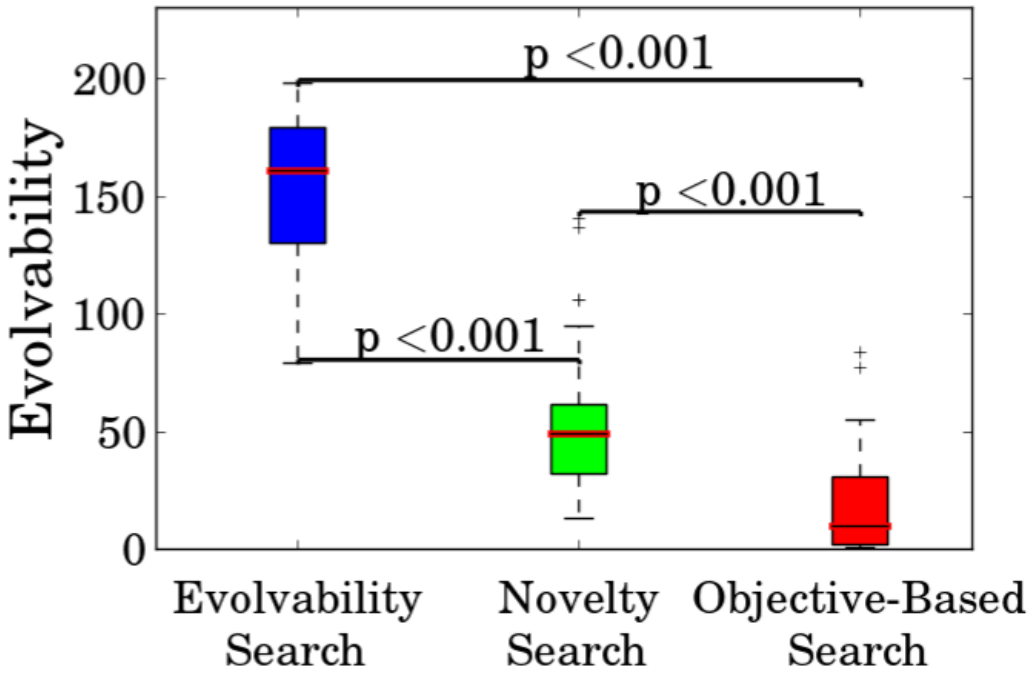
Evolvability Search [18]

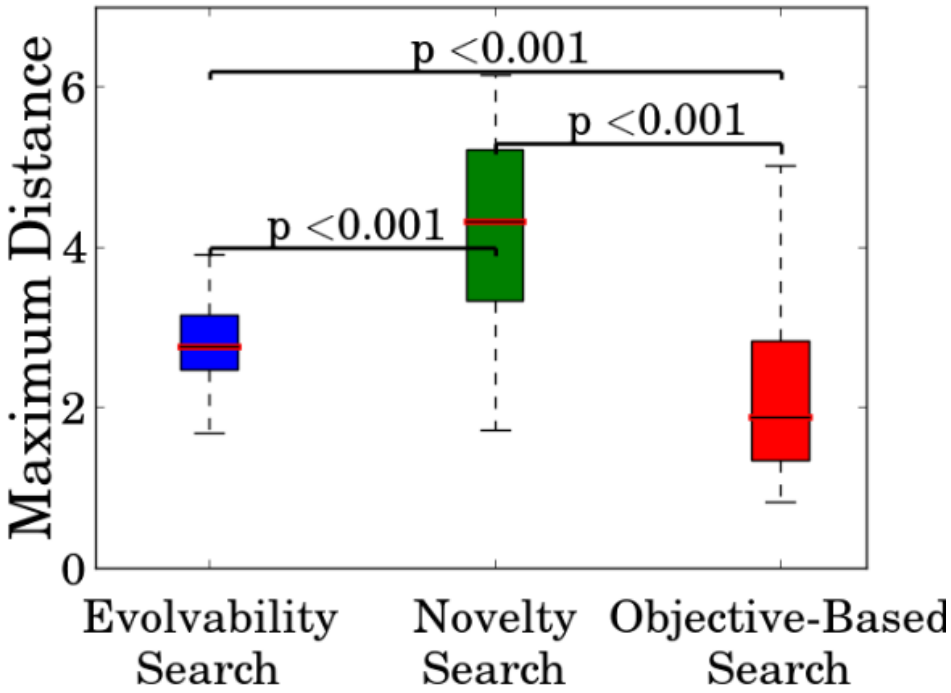
- **Select offsprings** based on a ***direct*** **measure of evolvability**
  - Assumption: evolvability ←→ phenotypical diversity
  - Novelty and evolvability are connected
  - Together with NEAT → evolvable neural networks
- **Create *fake* offsprings**, measure their evolvability
  - Select parents that produced more phenotipical diversity in offsprings
  - **Unique behaviors measured by a distance function (+ threshold)**
- Maze navigation w. neural network controller
  - Reach the goal
  - **Unique behaviors = cartesian coordinates**
  - Different coordinates → taking different routes → novelty
- Evolvability: generalization to a new maze
  - without further adaptation
  - How many grid cells do they cover in a new maze
- Expensive, due to evaluations, but effective
  - Also when compared w. objective-based evolution











Evolvability Search performance

Evolvability ES [16]

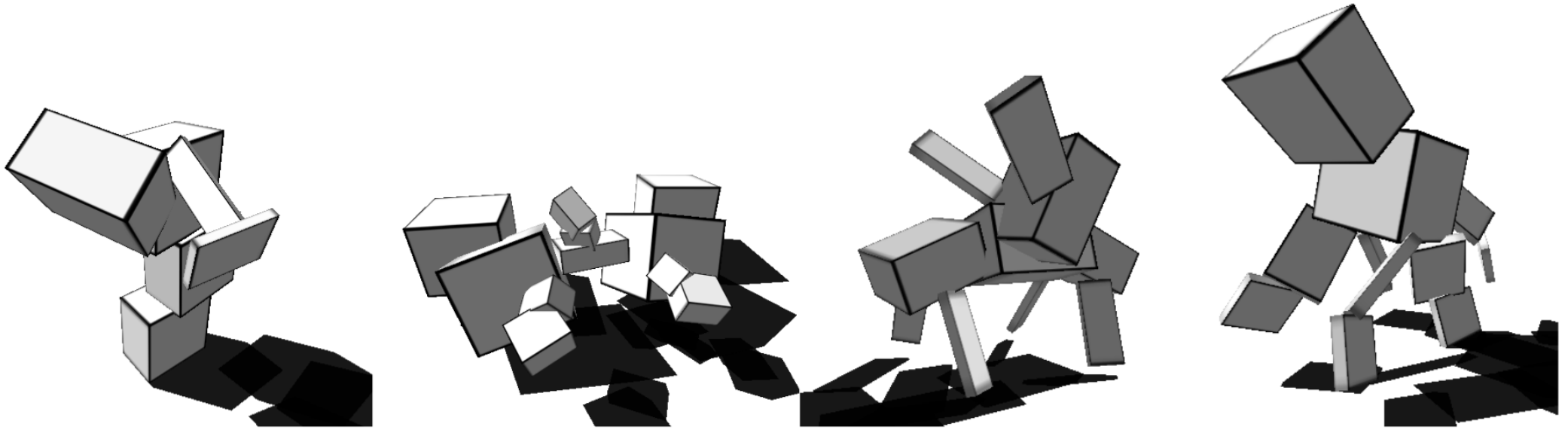
- All similar, but not the same…
  - Novelty Search →drives individuals to novel behaviors
  - Evolvability Search → drives offsprings of individuals to novel behaviors (individuals whose mutation + recombinations are likely to result in novel behaviors)
- Evolvability ES → scaling Evolvability Search to large problems
  - Reduce its computational cost due to many offsprings evaluation
  - **Use ES to avoid offsprings evaluation**
- Use a population distribution π (e.g. isotropic Gaussian)
  - Consider phenotype/behavior diversity by computing *variance (MaxVar-EES)* or *entropy (MaxEnt-EES)*
- Optimize through gradient descent **a loss function expressing evolvability**
  - $J(\theta) = \sum_j E_{z\sim\pi_\theta}\left[\left(B_j(z) - \mu_j\right)^2\right], \quad \mu_j =$ population mean of component j of behavior
  - B represents the behavior (e.g., final maze position)

Quality Diversity [22]

- Novelty search finds the optimal solution *indirectly*
  - Still limited by the «optimization» perspective
- **QD finds a maximally different set of solutions for the same problem** (Multimodal function optimization)
  - Divergent optimization (move away from visited regions)
  - Novelty Search with Local Competition (NSLC), MapElites
- Behavioral Characterization → the space of possible behaviors is equally important
  - Find best behavior for each **niche** in which the behavior is partitioned
  - BC is a vector storing info about individual's actions/structure during evaluation
  - BC can be more or less aligned with the task to be solved (e.g. behavior = position in the maze)

NSLC [23]

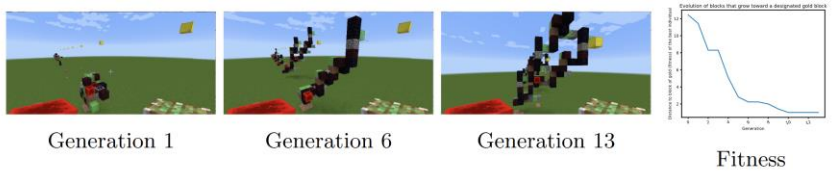
- Competition within a niche → select best, but different behaviors

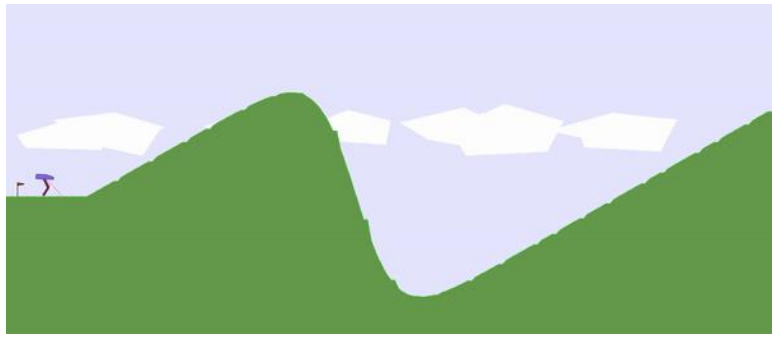


(a) Hopper (b) Crab (c) Quadruped (d) Tailed Quadruped

Figure 7: **Diverse competent morphologies discovered within a typical *single* run of local competition.** Various creatures are shown that have specialized to effectively exploit particular niches of morphology space. These creatures were all found in the final population of a typical run of local competition. The hopper (a) is a unipedal hopper that is very tall, (b) is a heavy short crab-like creature, and (c) and (d) are distinct quadrupeds. Creature (c) drives a large protrusion on its back to generate momentum, and (d) has a tail for balance.

Mine/Evo craft [24], POET [25]



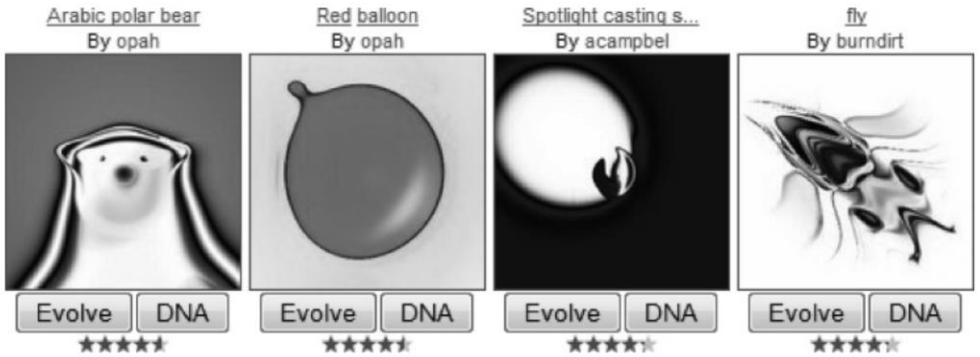


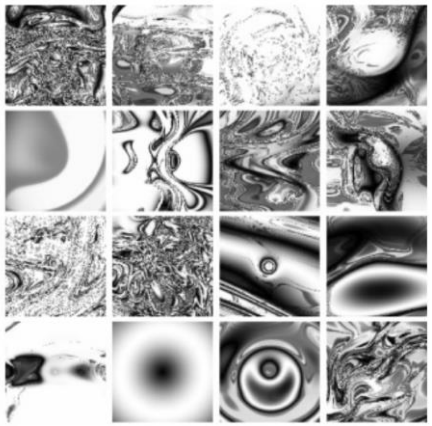
- Evocraft: an open-ended evolution environment for Minecraft
  - https://evocraft.life/
  - Study of objective-based evolution, evolvability, divergence…
  - Simple Python API
- POET: evolve a curriculum of both *problems* and *solutions*
  - Environment-agent pair → adapt agent to environment through ES
  - Environments need to be solved by at least one agent, an agent needs to solve at least one environment → coevolution leads to curricula
  - Attempt transfer to current environments → discover new types of solutions

Applications

Interactive and collaborative evolution

- **Picbreeder:** difficult to analytically express fitness function
  - Interaction with humans! → https://nbenko1.github.io/#/evolve (a remake)
- NEAT algorithm + human evaluation
- CPPN (compositional pattern producing networks): return HSB (or RGB) for each pixel. The input are the (x,y) coordinates.
  - Inductive bias on Gaussian shapes

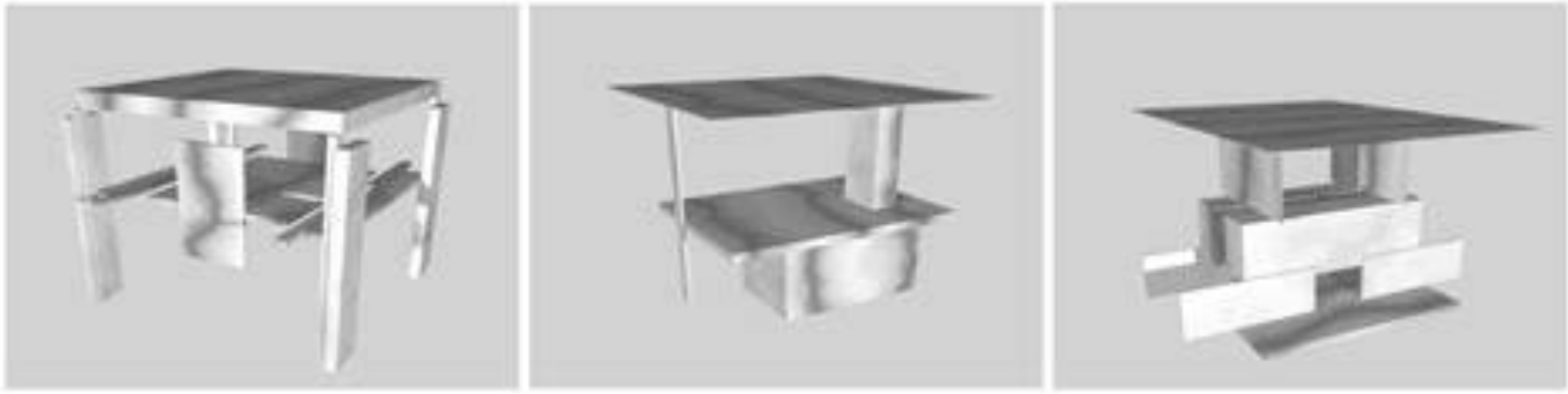




Evolution of Art and Design…

- **Generic Evolutionary Design**
- Evolving a table (designers thought it was not possible in the ’90s)

One of the design has been used and sold



…And Music

- 2000: Record label J13 Records . Contract with Universal Music
  - Mainly dance tracks
- Keep it a secret! No telling outside that was computer-generated music
- https://www.deezer.com/us/artist/10904338
  - Probably it?

Innovation Engine

- **Grand vision:** unlabeled innovation, evolution driven towards novel behaviors
  - Novel = not recognized by a DNN
- Use a deep neural network to estimate a novelty-based fitness function (Novelty search)
  - Applied to images, as in Picbreeder
  - CPPN-encoded images evolved with NEAT
- Novelty search on the distance between pre-trained AlexNet output probabilities
- Some results presented at one exhibit at the University of Wyoming Art Museum

Diversifying ImageNet

EvoJax

- A library for evolution of neural networks
- Parallelization on TPUs/GPUs
- Hard to say whether or not it will continue to be maintained
- https://github.com/google/evojax?tab=readme-ov-file#sister-projects
  - Some alternatives are listed

ALife social network!

- From Ken Stanley et al.
- Open endedness
  - Continuous innovation without bound
- No match through mutual friends, match through mutual interests
- Extraction of *topics* from posts, view more interesting topics (to you)
  - A % of exploration for new topics
- No «consensus», to avoid complete convergence

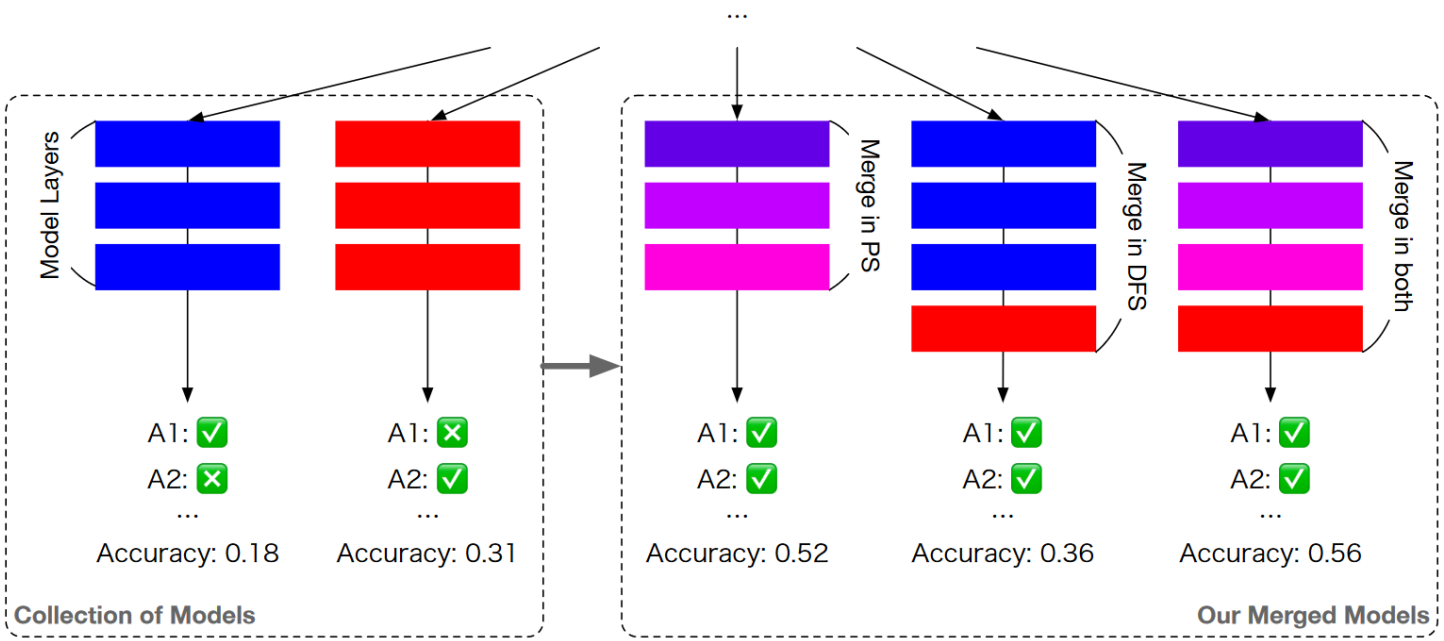
https://www.heymaven.com/about

A detour with LLMS

- LLMs are pre-trained on large corpora and then fine-tuned for specific purposes
- Model merging → combine skills without additional training
  - weights average (flat local minima are «compatible»)
  - Task vectors allow arithmetic (finetune – pretrain = task-specific vector)
  - Frankenmerging → stack layers from different models (in *mergekit* library) →handcraft required
- Neural Architecture Search
  - Use evolution to generate architectures
  - Requires to train each candidate solution → very expensive
- Can we automate this process?

Merging LLMs through evolution [44]

- Sakana AI → startup on Collective AI (w. David Ha, former Google)
- CMA-ES to evolve best model merging strategies (TIES-Merging + DARE)
- Mixing parameters (PS)
- Mixing layers (DFS)
- Mix both
  - Apply PS
  - Apply DFS on the resulting models
- Fitness = average accuracy
  - Task-based, no evolvability

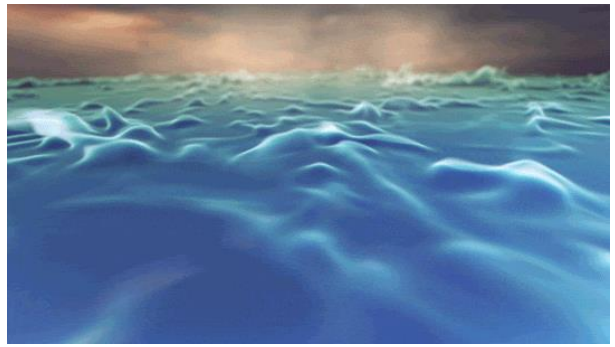


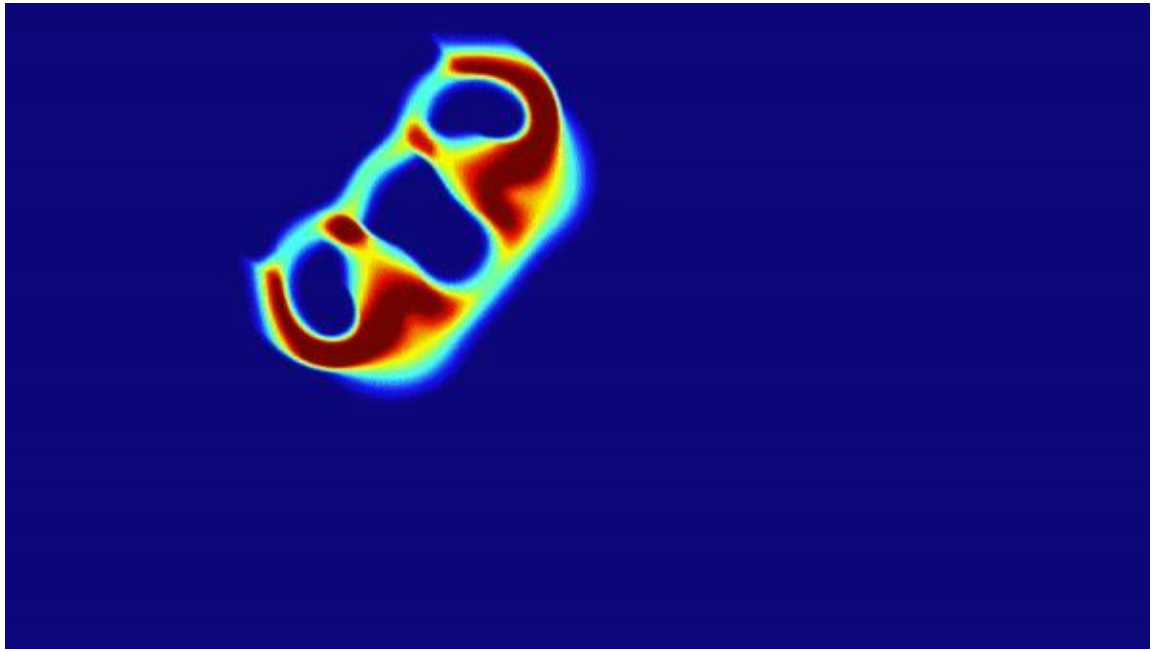
Artificial Life and Adaptive Complex Systems

Complex systems…

- System of systems
- «Simple» parts that interact
- Bottom-up «control»
  - No central unit
- Emergent behavior

- Santa Fe Institute
  https://www.santafe.edu/
- Complexity Explorer
  https://www.complexityexplorer.org/





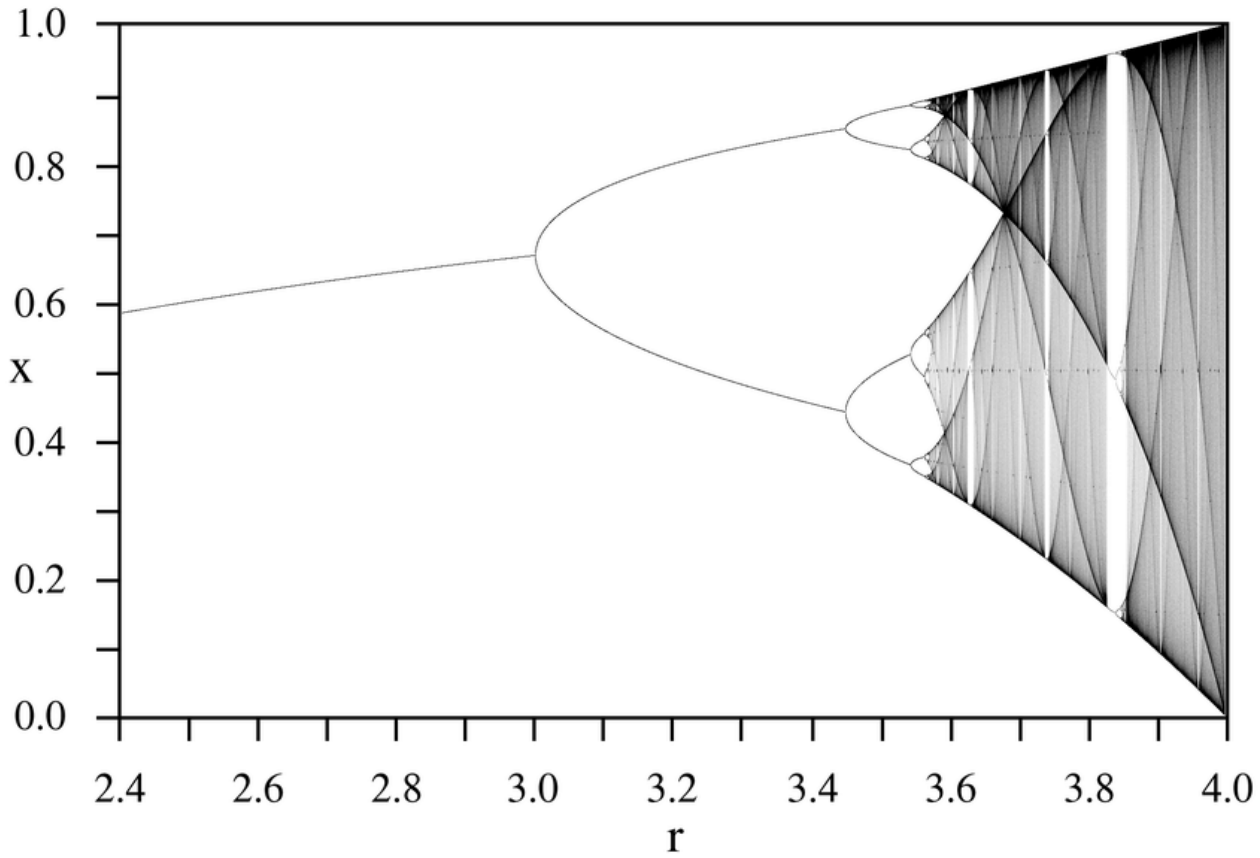
… for Artificial Life

Adaptive Complex Systems

- **Evolution of complexity!** (from prokaryotic life to eukaryotic life and then some)
- Open-ended evolution
  - yet again
- Self-organization (emergent)
- Chaotical dynamics
- “Comparing data from different artificial and natural evolving systems suggests that there are qualitatively different classes of evolutionary dynamics, and no known artificial system generates the kind of evolutionary dynamics exhibited by the biosphere” [1]

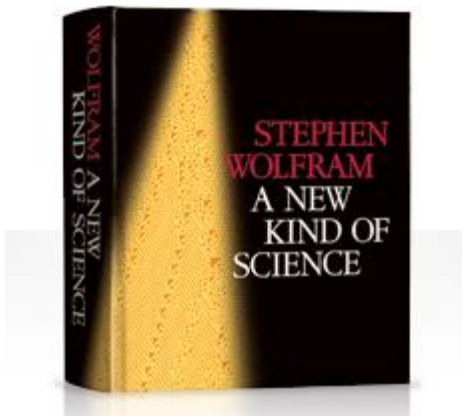
Are complex systems really complex?

- Logistic map: $x_{t+1} = r\ x_t(1 - x_t)$
  - Pretty straightforward… right?
- From order to chaos
  - Sensitivity to initial conditions
- Self-similarity at multiple scales
  - Fractals
- Deterministic → can you predict the future?
  - If you know *exactly* the initial state



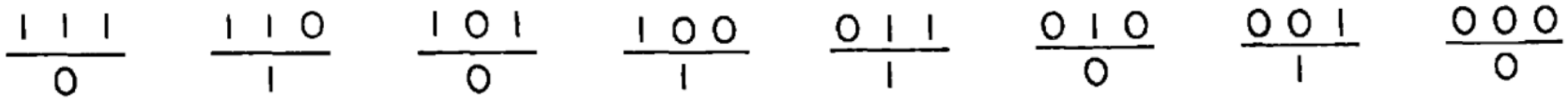
Cellular Automata [30]

- **Simple** mathematical model (dynamical system)
- Discrete **space**: 1/2D array of cells
  - discrete x,y coordinates
- Discrete **time**: simulation proceeds in time-steps
- 1 discrete variable per cell
- Each CA's cell evolves by looking at its neighbors (+ itself)
- Synchronous update
- **Update rules** dictates dynamics
- What about at the edge?
  - Cells can remain constant
  - Neighborhood can be defined differently
  - *Toroidal structures* (usually preferred)

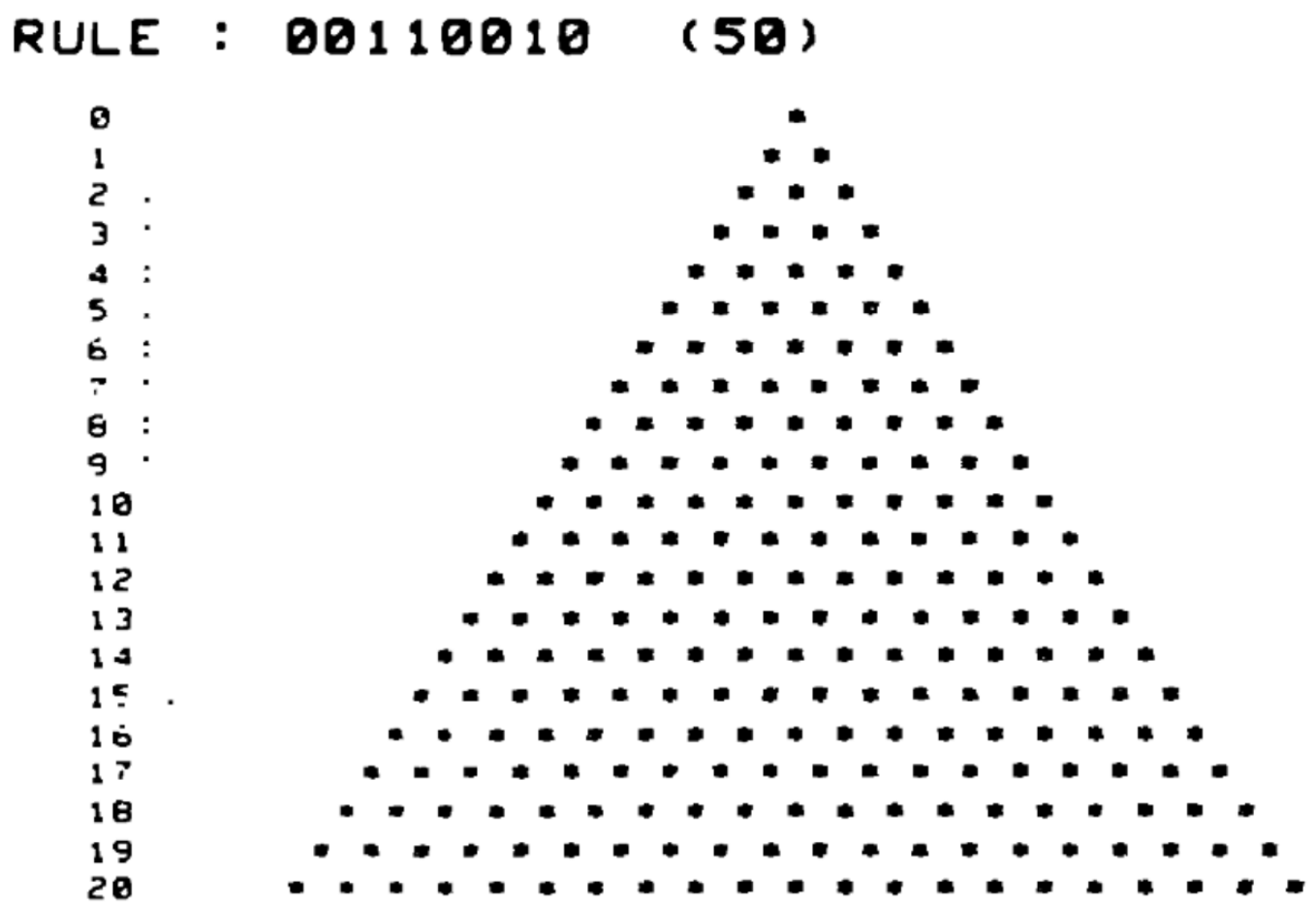


Elementary CA

- Binary-valued cells
- Update rule: $s_{t+1}^{j} = f(s_t^j, s_t^{j-1}, s_t^{j+1})$ → Boolean function
  - How many?



- 0-state should be mapped to zero → last digit needs to be 0
- Reflection symmetry → isotropy: “same behavior regardless of the direction”
  - 110 -> 011, 100 -> 001
- Admissible rules of the form $b_1 b_2 b_3 b_4 b_2 b_5 b_4 0$ → 32 possible rules
  - Still, they are named after their full binary/decimal number



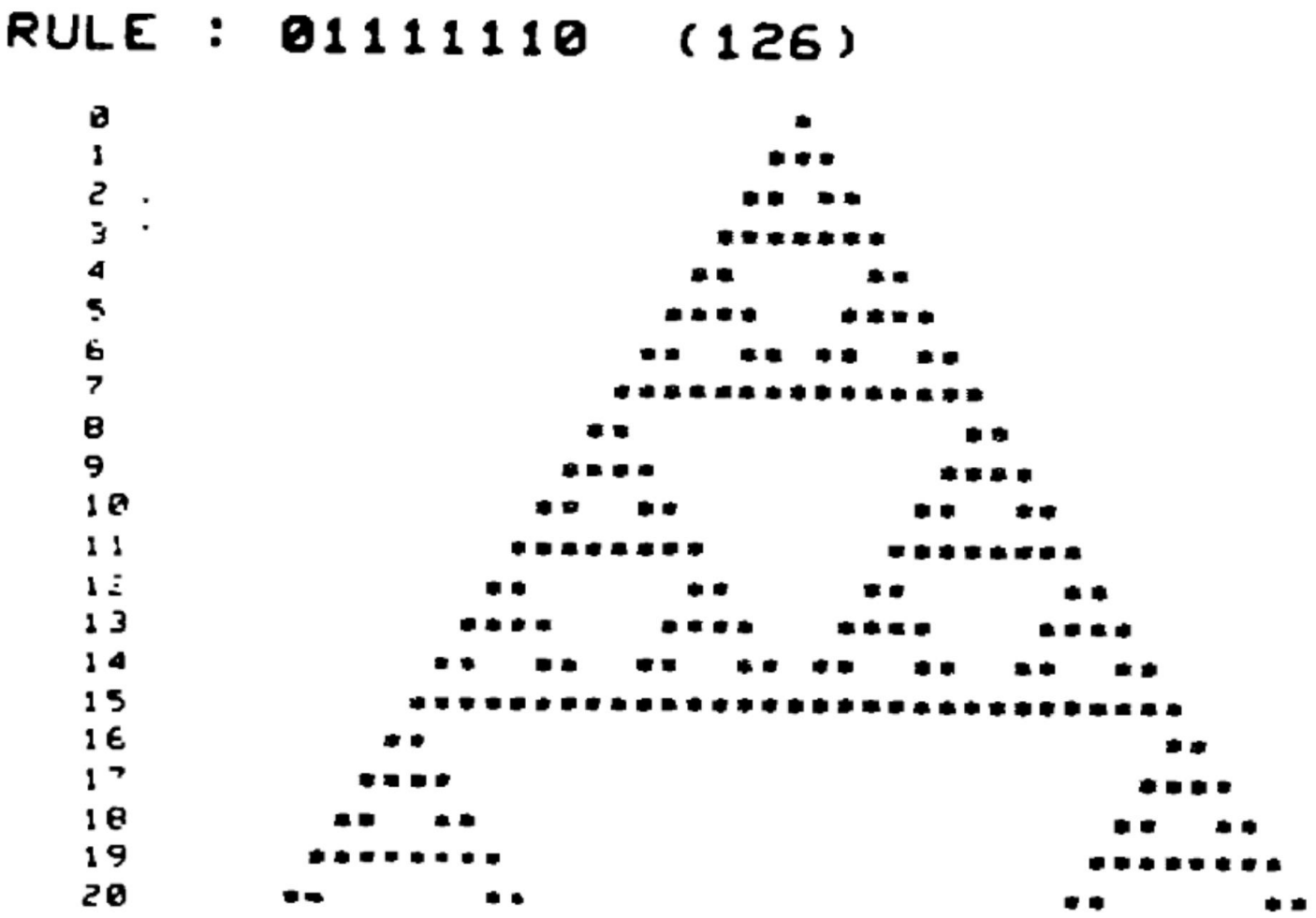
Evolution of an ECA

Some update rules are *additive (additive superposition)*

$$s_0 = t_0 \oplus u_0 \leftrightarrow s_n = t_n \oplus u_n$$

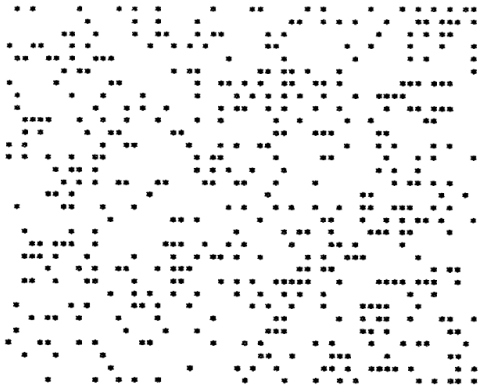
You can let the systems run separately and merge them later (or vice versa)

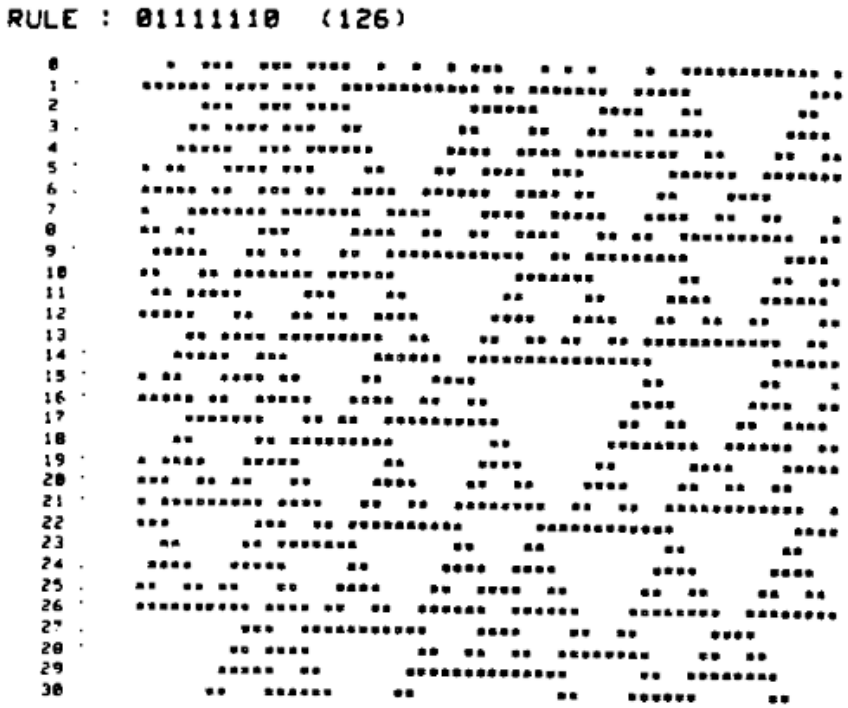
The operator $\oplus$ depends on the specific rule (e.g., modulo 2 addition for rule 90)



Evolution of an ECA (cont'd)

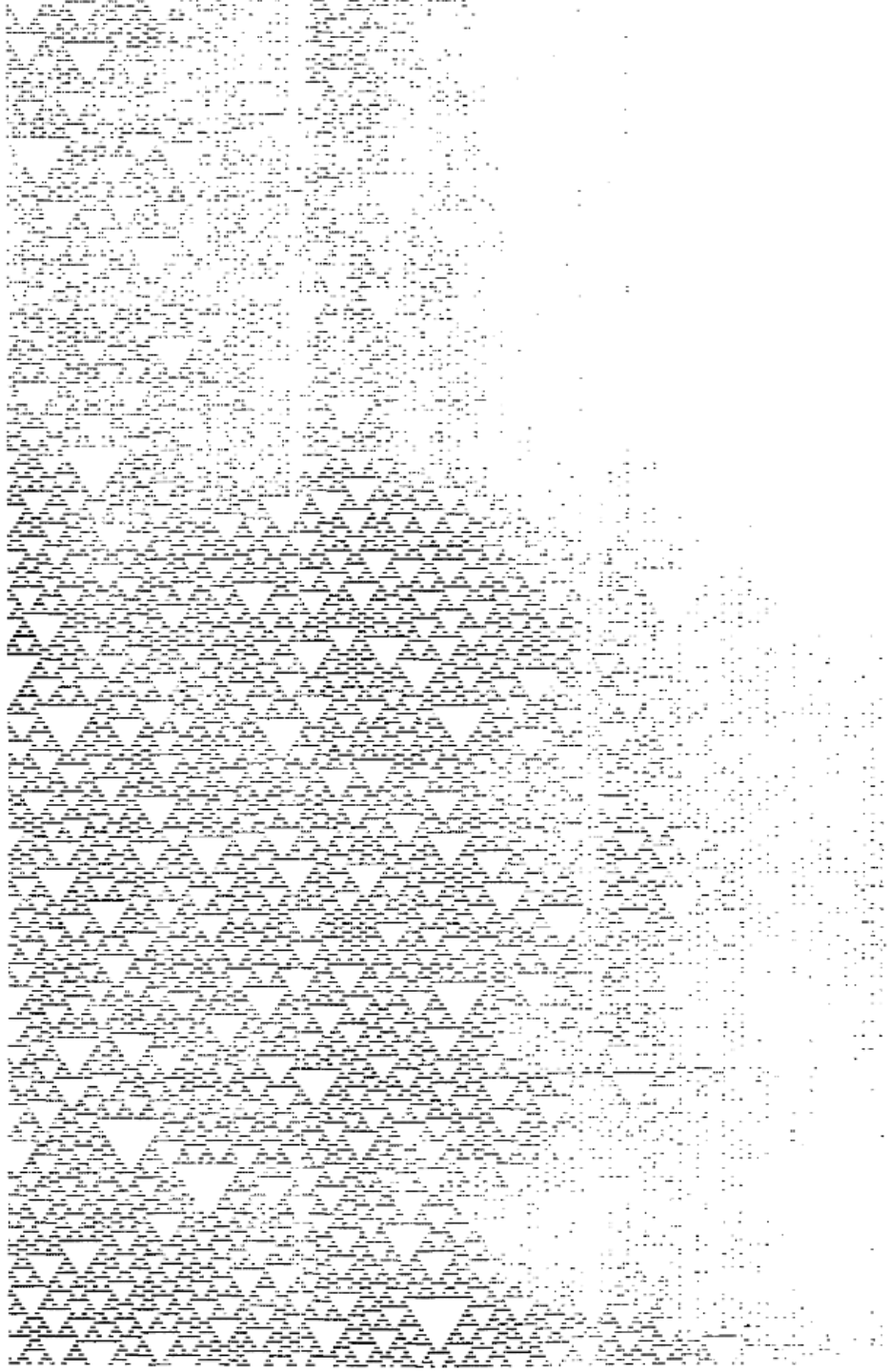
- Self-similarity at scale
  - Fractals
- One of the hallmarks of complex, nonlinear sytsems





Evolution of an ECA (cont'd)

- White noise as initial state
  - cells values are uncorrelated with each other
  - One parameter p describes the whole state
- Ordered system as a result
  - More parameters to describe it
- Self-organization properties
  - Another hallmark of complex nonlinear systems

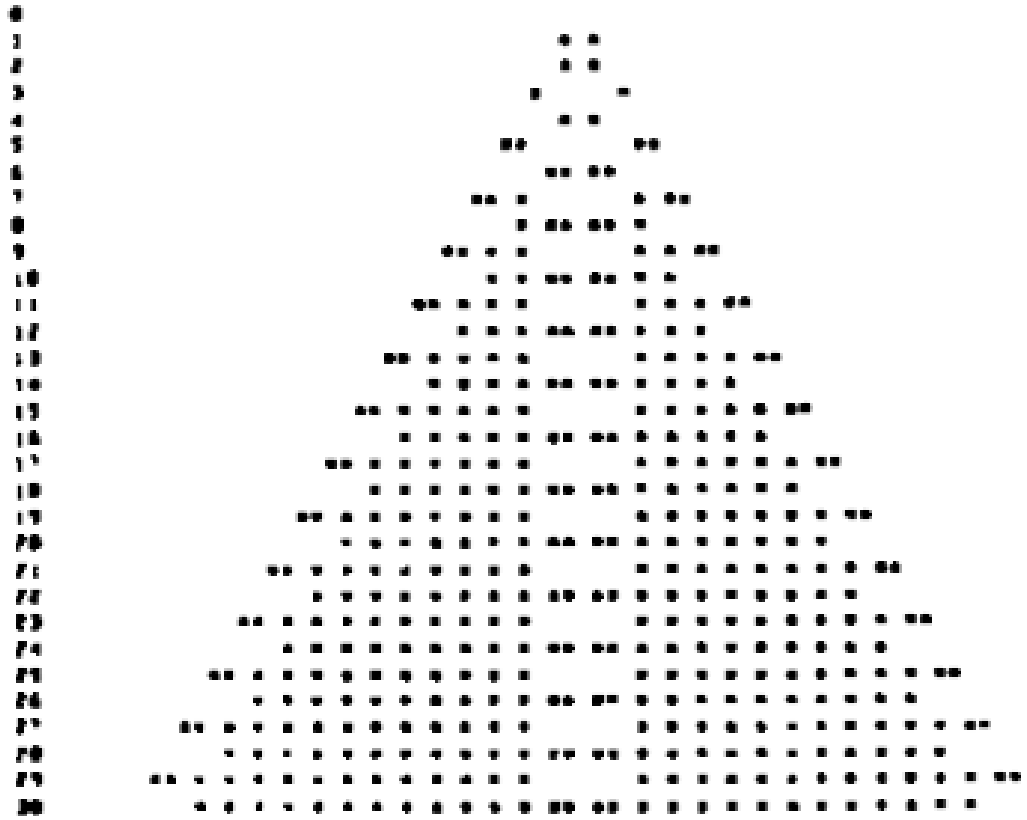


Evolution of an ECA (cont'd)

- 300 time-steps of evolution with rule 126
- Random initial state
  - P(x=1) = 0.5
- Study of global properties is also possible
  - Classical dynamical systems theory
  - Attractors, limit cycles etc…

Rules in ECA

- Type I
  - Evolution to static state
- Type II
  - Evolution to periodic structure
- Type III
  - Evolution to chaotic patterns -> leads to (pseudo) randomness
- Type IV
  - Evolution to complex patterns -> edge of chaos



CA with k states per cell

- Formation of protective membranes that shields their content from outside effects
- When two membranes collide they can destroy each other

Universal computer

- Wolfram's ECA 110
  - **Universal computation** → changes in input alone are sufficient to compute any computable function
  - Cfr. universal Turing machine
  - Not necessarily the simplest configuration
- Many, many variations of CA
- Non-uniform: different rules for different cells
- Non-local learning rules: different neighbors structure (e.g., a graph) → automata network
- Asynchronous updates
- Continuous time/space

Game of life

- **2D cellular automaton** with **2 values per cell** (0 – dead, 1 – alive)
- **8 neighbors** per cell (without self)
- Update rule 2,3/3 (environment/fertility):
  - Living cell: if it touches 2 or 3 live cells it stays alive, otherwise it dies (solitude, overpopulation)
  - Died cell: if it touches 3 live cells it comes back to life (birth)
- Emergence of “life-like” structures
  - Local attractors with several periods
  - Blinker, glider gun, spaceship….
  - https://conwaylife.com/wiki/
- Glider gun: continuously produce new cells
- https://playgameoflife.com/
- https://copy.sh/life/
- https://oimo.io/works/life/

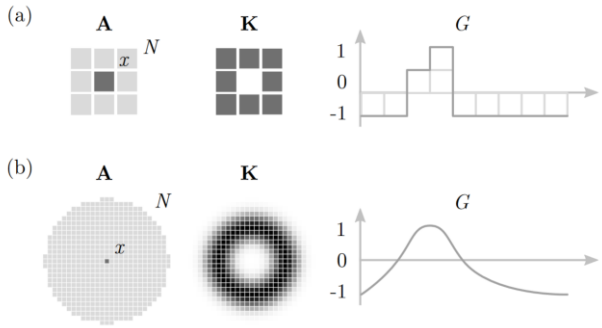
Discoveries of «life-like» structures

- Human-in-the-loop
- Genetic algorithms
  - Open-endedness
  - Novelty search
  - Evolvability search
  - …
- Definition of fitness function not trivial

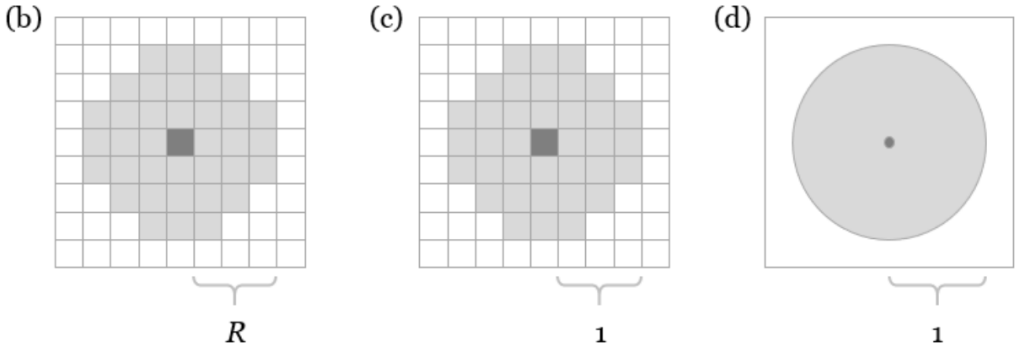
Extending GoL

- Reminiscent of Cellular Neural Networks
  - Analog, continuous computation showing emergent behavior
  - Abandoned after the GPU rise
  - Not easy to train (solving >=9 ODEs, or run costly genetic algorithms)
- $CA = (L, T, S, N, \phi)$
  - L → d-dimensional lattice (grid) (e.g., L=Z^2)
  - T → timeline (e.g., T=Z)
  - S → state set (e.g., {0, 1})
  - N → neighborhood (e.g., {-1, 0, 1}^{2})
  - ϕ → local update rule, Φ → global update rule
  - A^t → collection of states from all cells at time t, $A^t(x)$→ cell x
- $\Phi(A^0) = A^{0+\Delta t}$
  - e.g., binary cell, discrete lattice case: $A^{t+1}(x) = 1\ if\ living\_condition[\sum_{n \in N(x)} A^t(x+n)]$ else 0

Lenia [39]

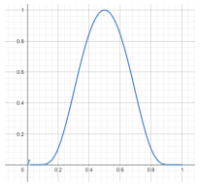


- $S = \{0, .., P\}$
- $N = \{x \in L \mid x : \left|\left|x\right|\right|_2 < R\}$ with range R (still discrete space)
- The resolution is discrete for space , time and state (Discrete Lenia)
  - $\frac{1}{R}, \frac{1}{T}, \frac{1}{P} \rightarrow 0$ → continuous space, time and state values (Continuous Lenia)
- Local update rule ingredients for Continuous Lenia:
  - Potential distribution
  - Growth distribution
  - Kernel

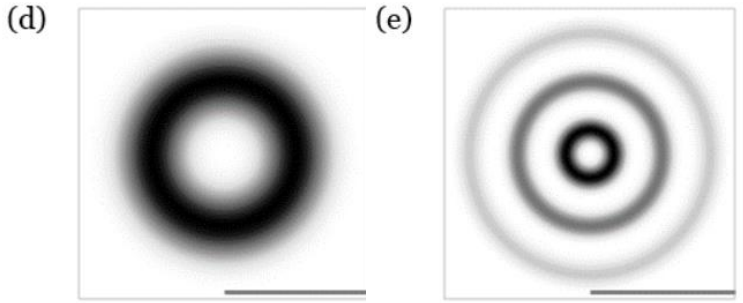


Potential distribution

- $U^t(x) = K \star A^T(x) = \int_{n \in N} K(n) A^t(x+n) dx^2$
  - $K: N \rightarrow S$ is a kernel that updates one cell based on the neighbors (brace yourself)
- Exponential **kernel core** with a given radius $K_C(r) = \exp\left(\alpha - \frac{\alpha}{4r(1-r)}\right), \alpha = 4$



  - Other forms are possible (rectangular, polynomial etc…)
- **B peaks** $\beta = (\beta_1, \ldots, \beta_B)$ of the concentric, equidistant rings
- **Kernel shell** $K_S(r; \beta) = \beta_{floor(Br)}\, K_C(Br)$
- **Normalized Kernel** $K(n) = \frac{K_S\left(||n||_2\right)}{|K_S|}\, (K \star A = \in [0,1])$
  - The kernel takes the polar distance (from the origin) as input



Growth distribution

- **Growth mapping**: $G(u;\mu,\sigma) = 2\exp\left(-\frac{(u-\mu)^2}{2\sigma^2}\right) - 1$
  - Unimodal, **growth center** μ and **growth width** σ
  - $G: [0,1] \to [-1,1]$
- Growth distribution: $G^t(x) = G\big(U^t(x)\big)$
- Update state $A^{t+\Delta t}(x) = A^t(x) + \Delta t\, G^t(x)$
  - Time advances of Δt
- Clip result in the [0, 1] range

$$\boldsymbol{A^{t+\Delta t} = clip_{[0,1]}(A^t + \Delta t\, G(K \star A^t))}$$

- As a note: convolution can be implemented efficiently with DFT
  - $K \star A = F^{-1}\big(F(K)\, F(A)\big)$ → convolutions are multiplications in FFT domain
  - DFT of K can be pre-computed

Extended Lenia

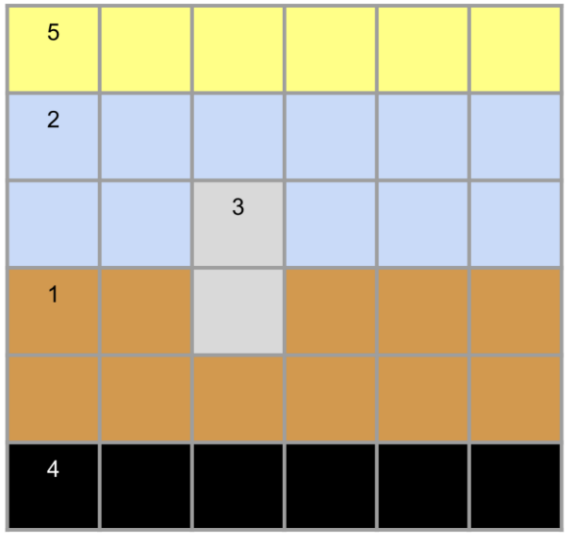
- Multi-dimensional
  - >= 3 → more stable forms, rarely moving
- Multiple kernels
  - Each has a relative radius $r_k R$
  - Each has a corresponding growth mapping
  - Chaoticity increases
- Multiple channels
  - Additional worlds interacting with each other by cross-channels convolutions
  - Each channel “specialize” for a role (e.g., nuclei vs. membranes)
- Evolutionary search is challenging
  - Filter by survival → forms that last (and do not fill entire space)
  - Human filter
  - Random changes from existing forms
  - Fitness function based on speed of movement or other desiderata

(Extended) Lenia results + Flow Lenia

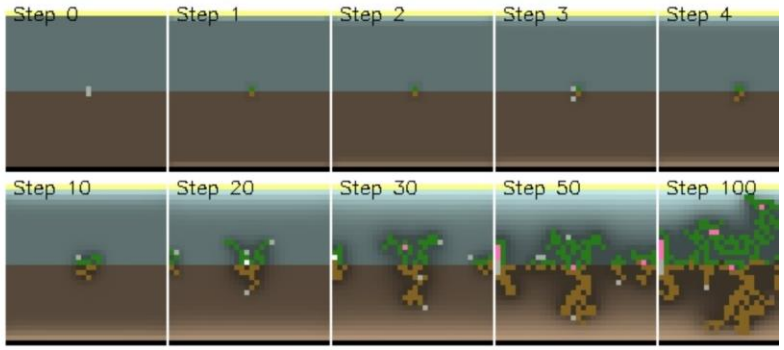
- https://chakazul.github.io/Lenia/JavaScript/Lenia.html
- Solitons = self-organizing structures
- Extended Lenia exhibits self-replication, which was missing in Lenia
  - Soliton splits in two and then the parts repel each other
- 3D Snake → eat and grow
- Coordination between channels
  - A soliton is the end result of multiple channels with different dynamics
  - Advanced behaviors may result from a change in the channel interaction, with the same genotype
- Flow-Lenia: mass conservation in Extended Lenia to facilitate search
  https://sites.google.com/view/flowlenia/videos

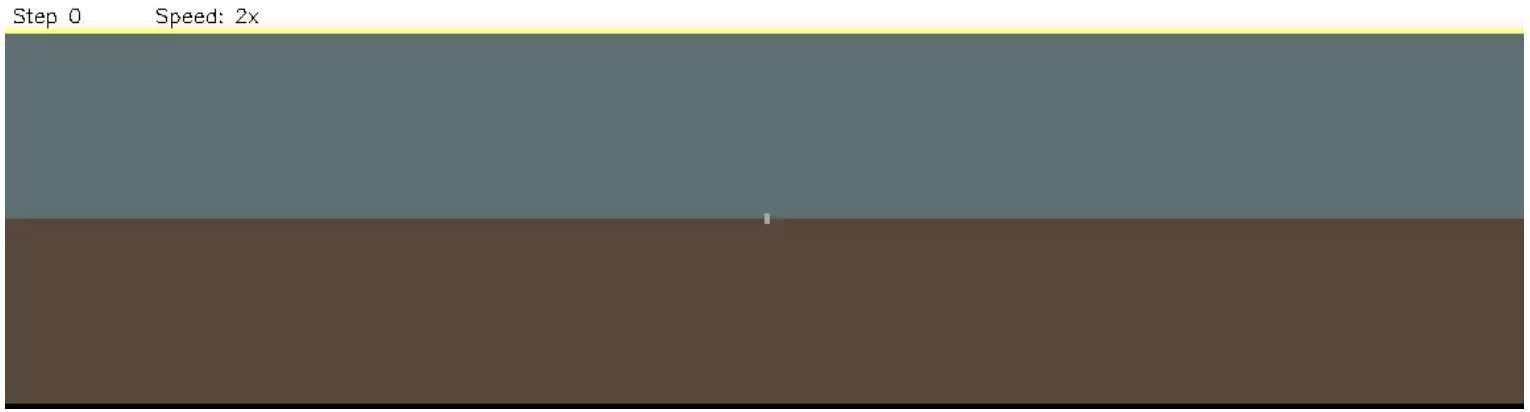
Biomaker CA – plant-based lifeforms

- Earth (1), Air (2), Seed Agent (3), Immovable (4), Sun (5)
  - Each cell perceives a local neighborhood
- Agent needs to acquire energy
  - From earth nutrients (through roots)
  - From air nutrients (through leaves)
  - Sun produces air nutrients, immovable produces earth nutrients
- Agent composed of different materials
- Reproduction through seed spawning
  - Seed also undergoes mutation



- Growing plant destroys earth/air (roots/leaves take space) and nutrients are finite
- Died plants return earth nutrients
- https://tinyurl.com/2x8yu34s





Biomaker CA

- Cells make actions (spawn, grow/specialize, move…)
  - Action depend on cell type
- Meta-evolution
  - Outer loop generates many environment, then ES optimization
  - Costly in terms of computational time
- Interactive evolution (Picbreeder style)
- https://google-research.github.io/self-organising-systems/2023/biomaker-ca/

Particle swarm ALife

- **Particle:** a pixel with *position*, *velocity* and *color* (randomly assigned at the beginning)
  - Different colors identify different families of particles
- Self-organizing, agent-based systems composed by families of many particles
  - e.g., flocking
- Update position based on *attraction/repulsion* force
  - Combination of attraction/repulsion among all families of particles
  - Red attracts red, yellow repels red, yellow does not interact with yellow → what is the final behavior?
- Collision detection can also be added

https://www.youtube.com/watch?v=0Kx4Y9TVMGg

https://www.ventrella.com/Clusters/

CA applications

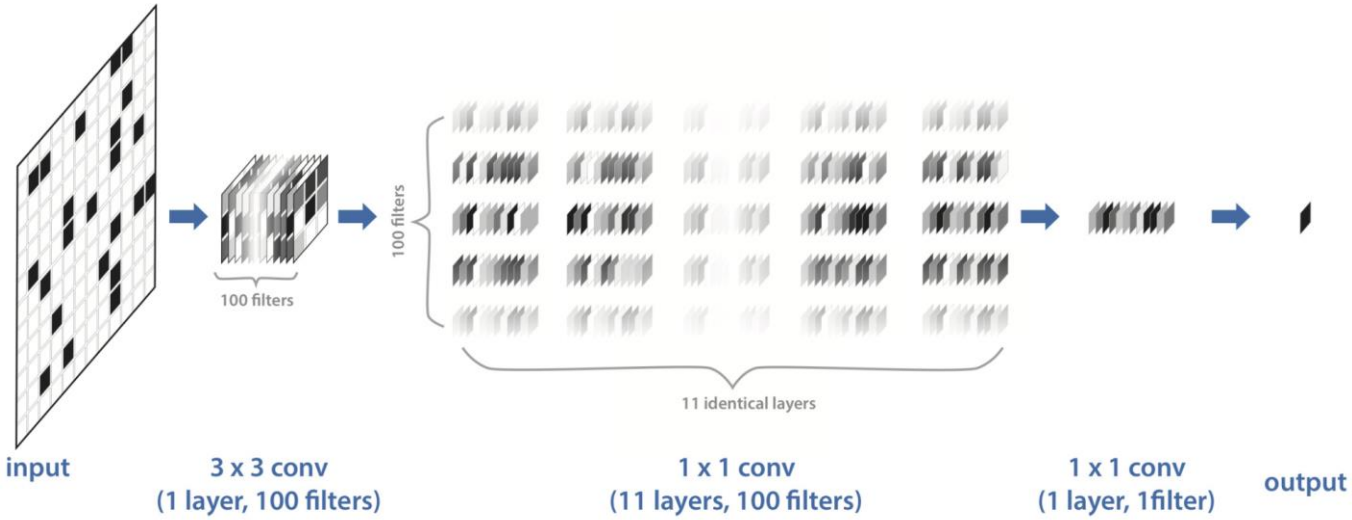
- Is it just a game? Beyond models of life processes and dynamical systems
- Non-uniform ECA for VLSI (circuit design)
  - Pseudo-random numbers generator for built-in self test of circuits
  - A similar use has been done in cryptography
- Image processing
  - edge detection, noise reduction
  - Image classifiers
  - Associative memories
  - **Segmentation**: quickly segmenting high-resolution images (less than 10k parameters) – challenging even for modern neural networks
- Text/image compression

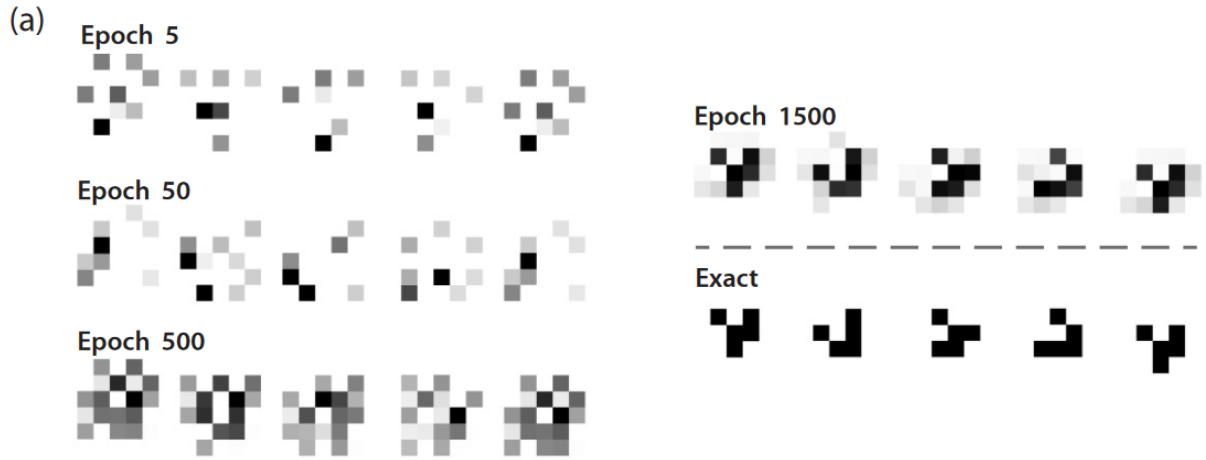
Image segmentation via CA, I

- The temporal information can be relevant in computer vision as well
  - Videos, temporal consistency (e.g. MRIs): CNN + RNN
- Each CA cell has *continuous* state $s \in R^d$, associated with an RGB pixel of the image
- 9-cell neighborhood: for each neighbor consider both RGB and cell state information
- *Learn* transition function / update rule
  - $\mathrm{T}(s) = U(s) + s$, U is a CNN (3 layers: 1 3x3 conv and then 2 1x1 convs)
  - Only the first layer considers local interaction
- Random initial state $S_0 \sim$ N(0,I)
- $S_t = T^t(S_0, img) = (T \circ T \circ \cdots \circ T)(S_0, img)$
  - Same input image at each time step
- The proposed increment to the state update is accepted cell-wise with p=0.5
- Train CNN + readout layer with cross-entropy at each step

Image segmentation via CA, II

- BPTT for all steps
  - T>20 often required → expensive
- BPTT for K<<T steps (loss measured at a given step T')
  - With probability p reuses images/state for the next unroll, with (1-p) reset state and choose another image
  - Robustness to image changes, more efficient
- Cell state never changes after many steps. Even though the image changes.
  - **Reset gate**: $r(s)N(0,I) + \big(1 - r(s)\big)(U(s) + s)$
  - $r(s) = \sigma\big(W\ H(s)\big),\ H(s)$→ pre-output activations
- For high-resolution images → reduce image resolution (learned conv) → convert CA state back to high-resolution (learned transpose conv)



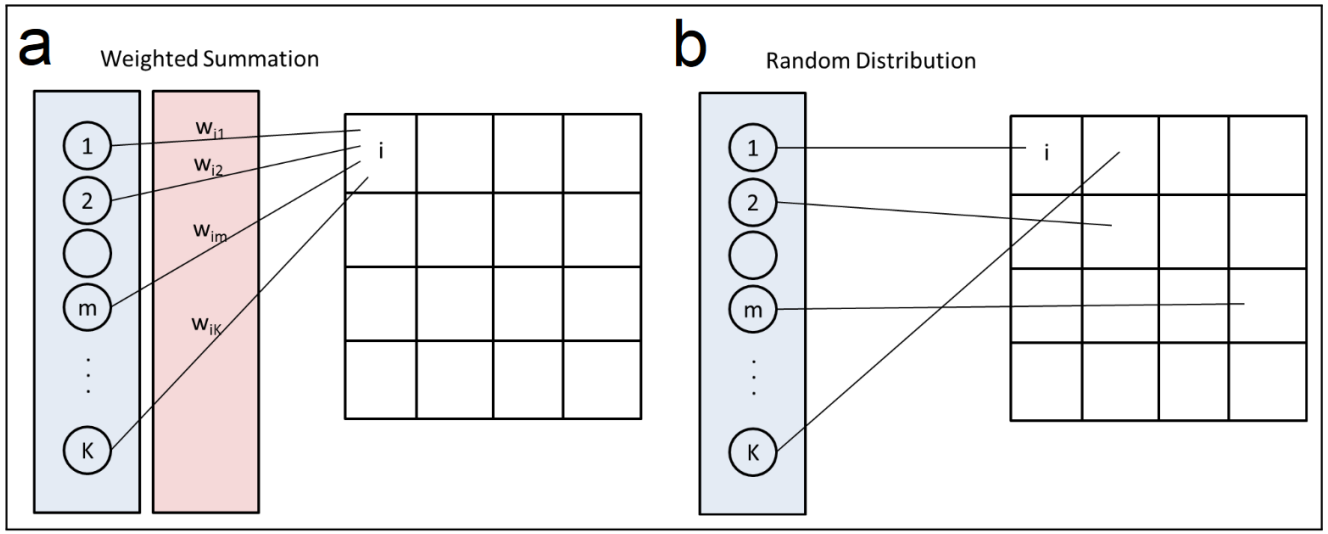


CA and Artificial Neural Networks [36]

- CAs «are» recurrent CNNs (and viceversa)
  - You can use one to represent the other
  - Train a CNN to learn the update rule
  - Apply the same CNN over time steps

Reservoir Computing and CA

- Why not use CAs as reservoirs?
- **Input encoding**
  - Weighted summation (then binarized)
  - Random distribution
- Execute R different CAs for T steps
  - non-linear, edge-of-chaos rules
  - Turing completeness
- Concatenate all T states
- Process with a readout

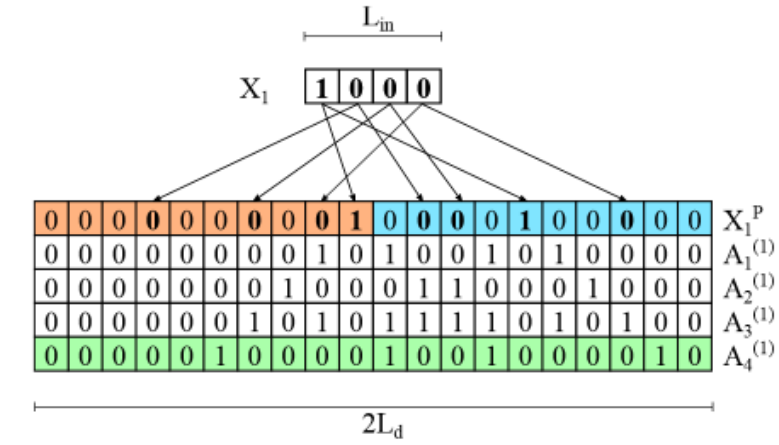


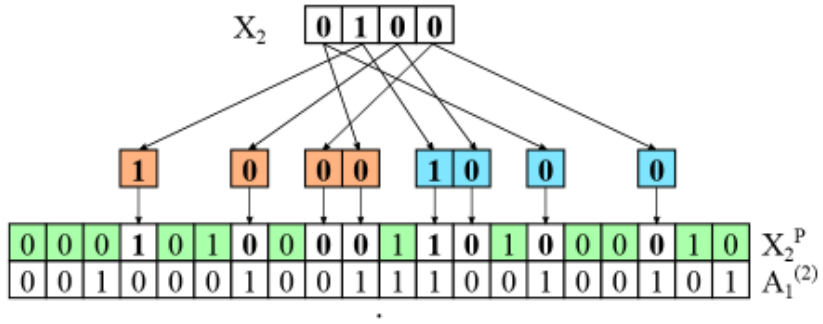
Reservoir Computing and CA

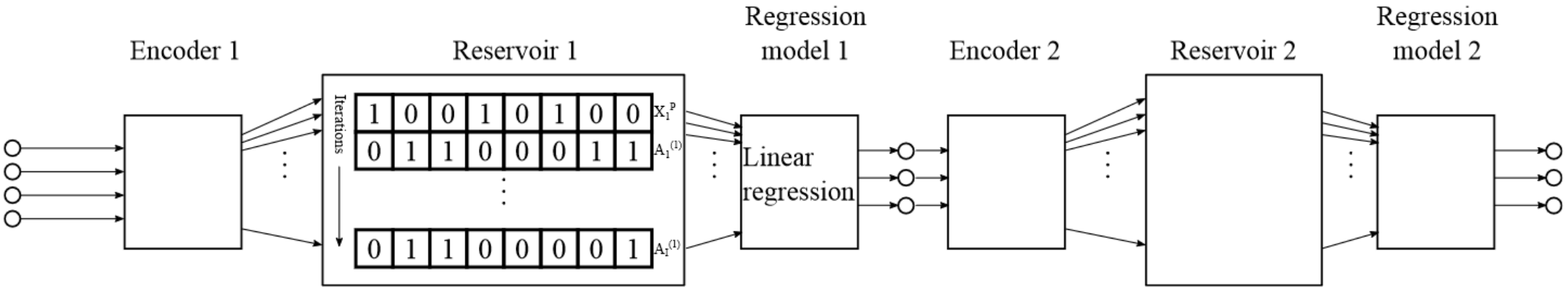
- Nice memory properties
- Symbolic manipulation (Conceptor-style)
  - The *additivity* property! Suppose A and B are processed through two reservoirs
  - $A \vee B = C_A \oplus C_B = C_A \oplus C_{B-A}$
  - Useful for binary CAs, since the configuration of 1 represents a given object (e.g. A)
- Much more efficient than traditional reservoir computing (bitwise operation for evolution)
  - Neuromorphic / hardware implementation
  - Readout matrix multiplication = sum
  - Symbolic combination for free (Conceptors are more expensive)

Tackling time series

- ECA as reservoir computing layer (wrapped boundaries)
- Input encoding through diffusion
  - Random mapping only initialized once
- New time step is mapped to an initial state which is the final state on previous time step
  - Many mappings avoid losing too much information
  - This is able to manage temporally-correlated information
- Train readout offline






Encoder 1
Reservoir 1
Regression model 1
Encoder 2
Reservoir 2
Regression model 2
Iterations
1 0 0 1 0 1 0 0
0 1 1 0 0 0 1 1
0 1 1 0 0 0 0 1
Linear regression

Going deep

CA, Graphs, RNNs, CNNs

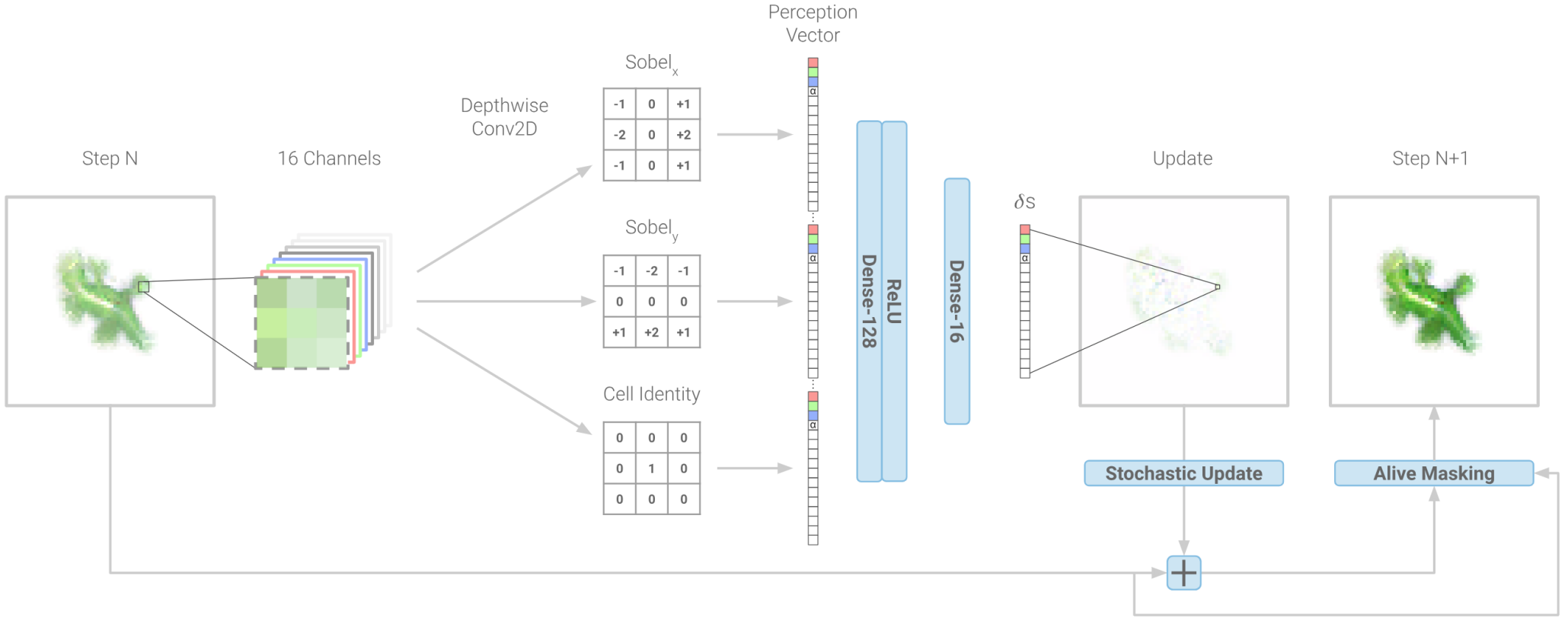
- The formulation of CAs is flexible
- Strong links with RNNs and CNNs (cfr. Lenia)
- Strong links with graphs → change in neighborhood structure

Can we do something to adapt via *learning?*

Neural Cellular Automaton I [42]

https://distill.pub/2020/growing-ca/

- Morphogenesis study: how an organism gets its «shape»
  - self-organization
  - Robust to perturbations
  - How to know what to build? When to stop?
- **Cell:** real-valued vector of 16 numbers
  - First 3 channels are RGB coordinates
  - Fourth channel is «alpha»
- $\alpha > 0.1$ → *living* cell
- $\alpha \leq 0.1$ → growing cells
- Dead cells: no living cell in the neighborhood → state set to 0
- NCA environment: [height, width, 16], 3x3 neighborhood


Step N
16 Channels
Depthwise Conv2D
Sobel$_x$
-1 0 +1
-2 0 +2
-1 0 +1
Sobel$_y$
-1 -2 -1
0 0 0
+1 +2 +1
Cell Identity
0 0 0
0 1 0
0 0 0
Perception Vector
Dense-128
ReLU
Dense-16
δs
Update
Step N+1
Stochastic Update
Alive Masking

Neural Cellular Automaton II

- **Perception phase**
  - Neigborhood convolution (fixed kernel, Sobel filter)
  - Perception vector: real-valued vector of 48 numbers: 16 (cell state) + 16 (Sobel x) + 16 (Sobel y)
- **Update rule**
  - A small «neural network» (8k parameters): 1x1 conv, pooling, ReLU etc…
  - It returns the state *update:* $s_{t+1} = s_t + \Delta s_t$
  - Shared across all cells
  - Randomly applied with per-cell probability of 50%
- Iteratively apply the update rule. Loss at the end of all steps (BPTT)
- Start from a single living cell and *grow* it

Neural Cellular Automaton in action

Adding stability

- Make the target pattern an *attractor*
  - Use larger T → computationally expensive
- Use a pool of states and learn to grow the pattern from each state
  - Pool gets updated over time with intermediate states

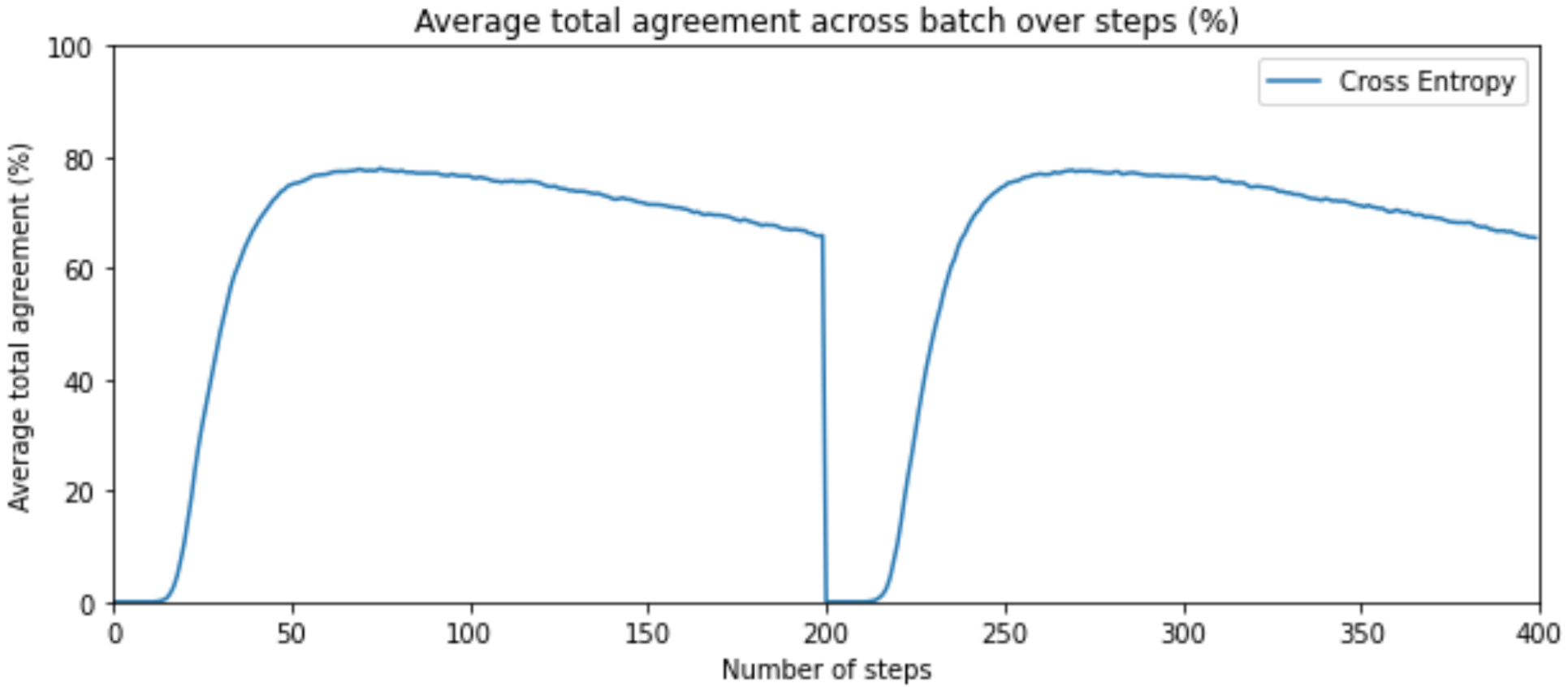
Regenerate after trauma

- Damage pool states before using
  - Zero-out random circular region

Self-classifying MNIST

https://distill.pub/2020/selforg/mnist/

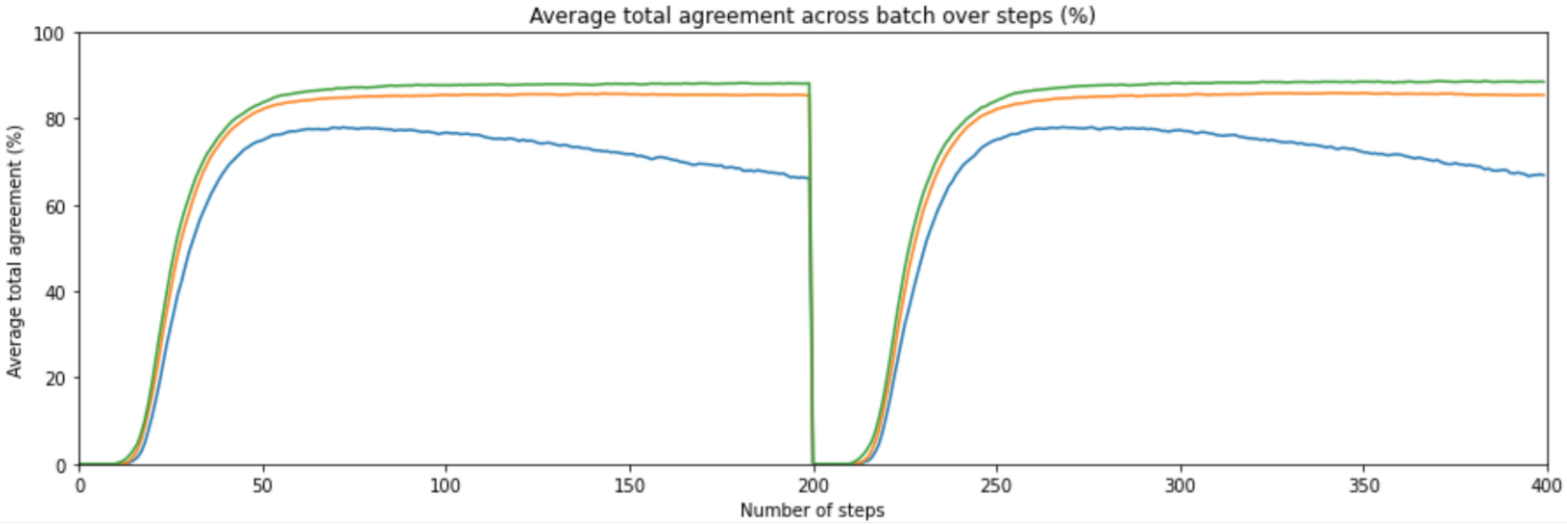
- NCA for classification
  - Each cell predicts the class
  - Agreement on the final prediction
- Input channels are immutable (not trained)
- 19 trained channels
  - Last ten channels represent class probabilities
  - Trainable 3x3 conv (for neighborhood)
  - Only 25k parameters


Average total agreement across batch over steps (%)
Cross Entropy
Average total agreement (%)
Number of steps
0
20
40
60
80
100
0
50
100
150
200
250
300
350
400

Agreement

Cross-entropy problems

- Cross-entropy always returns update signal
  - Indefinitely push logits towards $\pm \infty$
  - No reference scale across cells
- One-hot target + cell-wise MSE loss



Getting there

- An environment where multiple agents can interact
- Agents do not all play the same role
- Agents adapt (learn) *during* their lifetime
- Evolution drives the agents towards «interesting» behaviors
- Inner-loop and outer-loop like in meta learning
  - The outer loop is non differentiable
- In a continual fashion

References 1

[1] M. A. Bedau, “Artificial life: organization, adaptation and complexity from the bottom up,” *Trends in Cognitive Sciences*, vol. 7, no. 11, pp. 505–512, Nov. 2003, doi: 10.1016/j.tics.2003.09.012. **A useful review**

[2] C. G. Langton, *Artificial Life: Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*. USA: Addison-Wesley Longman Publishing Co., Inc., 1989. **Artificial Life term introduced**

[3] T. S. Ray, “An Evolutionary Approach to Synthetic Biology: Zen and the Art of Creating Life,” *Artificial Life*, vol. 1, no. 1_2, pp. 179–209, Oct. 1993, doi: 10.1162/artl.1993.1.1_2.179. **Virtual creatures**

[4] K. Sims, “Evolving 3D Morphology and Behavior by Competition,” *Artificial Life*, vol. 1, no. 4, pp. 353–372, Jul. 1994, doi: 10.1162/artl.1994.1.4.353. **Virtual creatures**

[5] Sims K. Evolving virtual creatures. In: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques. ACM; 1994. p. 15–22 **Virtual creatures**

[6] Cheney N, MacCurdy R, Clune J, Lipson H. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In: Proceedings of the Genetic and Evolutionary Computation Conference. ACM; 2013. p. 167–174 **Virtual creatures (soft robots)**

[7] J. S. F. Barker, “Simulation of Genetic Systems by Automatic Digital Computers,” *Aust. Jnl. Of Bio. Sci.*, vol. 11, no. 4, pp. 603–612, 1958, doi: 10.1071/bi9580603. **Genetic algorithms: one of the first**

References 2

[8] G. E. P. Box, “Evolutionary Operation: A Method for Increasing Industrial Productivity,” *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 6, no. 2, pp. 81–101, 1957, doi: 10.2307/2985505. **Genetic algorithms: one of the first**

[9] W. M. Spears, K. A. De Jong, T. Bäck, D. B. Fogel, and H. de Garis, “An overview of evolutionary computation,” in *Machine Learning: ECML-93*, P. B. Brazdil, Ed., Berlin, Heidelberg: Springer, 1993, pp. 442–459. doi: 10.1007/3-540-56602-3_163. **Survey on evolutionary computation**

[10] Forrest S, Nguyen T, Weimer W, Le Goues C. A genetic programming approach to automated software repair. In: Proceedings of the Genetic and Evolutionary Computation Conference; 2009. p. 947–954. **Genetic Programming**

[11] Ray, T. S. (1992). An approach to the synthesis of life. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), Proceedings of Artificial Life II (p. 371). Reading, MA: Addison-Wesley **Tierra (good luck finding the paper)** https://tomray.me/tierra/

References 3

[12] J. Lehman *et al.*, “The Surprising Creativity of Digital Evolution: A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities.” arXiv, 2024. http://arxiv.org/abs/1803.03453 **Anecdotes of surprising outcomes of Digital Evolution**

[13] C. Ofria and C. O. Wilke, “Avida: A Software Platform for Research in Computational Evolutionary Biology,” *Artificial Life*, vol. 10, no. 2, pp. 191–229, 2004, doi: 10.1162/106454604773563612. **Avida**

[14] Ellefsen KO, Mouret JB, Clune J. Neural modularity helps organisms evolve to learn new skills without forgetting old skills. PLoS Computational Biology. 2015;11(4):e1004128 **continual learning of edible food**

[15] I. Rechenberg, *Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart, Germany: Frommann-Holzboog, 1973. **Evolution Strategies (yes, it is in German). In case you can‘t read it you can take a look at** H.-G. Beyer and H.-P. Schwefel, “Evolution strategies – A comprehensive introduction,” Natural Computing, vol. 1, no. 1, pp. 3–52, Mar. 2002, doi: 10.1023/A:1015059928466.

[16] A. Gajewski, J. Clune, K. O. Stanley, and J. Lehman, “Evolvability ES: scalable and direct optimization of evolvability,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, in GECCO ’19. New York, NY, USA: Association for Computing Machinery, Jul. 2019, pp. 107–115. doi: 10.1145/3321707.3321876. **Evolvability ES**

[17] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, “Evolution Strategies as a Scalable Alternative to Reinforcement Learning.” arXiv, Sep. 07, 2017. doi: 10.48550/arXiv.1703.03864.

[18] H. Mengistu, J. Lehman, and J. Clune, “Evolvability Search: Directly Selecting for Evolvability in order to Study and Produce It,” in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, in GECCO ’16. New York, NY, USA: Association for Computing Machinery, Jul. 2016, pp. 141–148. doi: 10.1145/2908812.2908838. **Evolvability Search**

References 4

[19] J. Lehman and K. O. Stanley, “Improving evolvability through novelty search and self-adaptation,” in *2011 IEEE Congress of Evolutionary Computation (CEC)*, New Orleans, LA, USA: IEEE, Jun. 2011, pp. 2693–2700. doi: 10.1109/CEC.2011.5949955.

[20] J. Lehman and K. O. Stanley, “Abandoning Objectives: Evolution Through the Search for Novelty Alone,” *Evolutionary Computation*, vol. 19, no. 2, pp. 189–223, Jun. 2011, doi: 10.1162/EVCO_a_00025. **Novelty search**

[21] J. Lehman and K. O. Stanley, “Improving evolvability through novelty search and self-adaptation,” in *2011 IEEE Congress of Evolutionary Computation (CEC)*, New Orleans, LA, USA: IEEE, Jun. 2011, pp. 2693–2700. doi: 10.1109/CEC.2011.5949955.

[22] J. K. Pugh, L. B. Soros, and K. O. Stanley, “Quality Diversity: A New Frontier for Evolutionary Computation,” *Front. Robot. AI*, vol. 3, Jul. 2016, doi: 10.3389/frobt.2016.00040. **quality diversity**

[23] J. Lehman and K. O. Stanley, “Evolving a diversity of virtual creatures through novelty search and local competition,” in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, in GECCO ’11. New York, NY, USA: Association for Computing Machinery, Jul. 2011, pp. 211–218. doi: 10.1145/2001576.2001606. **NSLC**

[24] D. Grbic, R. B. Palm, E. Najarro, C. Glanois, and S. Risi, “EvoCraft: A New Challenge for Open-Endedness,” in *Applications of Evolutionary Computation*, P. A. Castillo and J. L. Jiménez Laredo, Eds., Cham: Springer International Publishing, 2021, pp. 325–340. doi: 10.1007/978-3-030-72699-7_21. **Evocraft**

[24a] K. O. Stanley and R. Miikkulainen, “Evolving Neural Networks through Augmenting Topologies,” *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, Jun. 2002, doi: 10.1162/106365602320169811. **NEAT**

[25] R. Wang, J. Lehman, J. Clune, and K. O. Stanley, “Paired Open-Ended Trailblazer (POET): Endlessly Generating Increasingly Complex and Diverse Learning Environments and Their Solutions.” arXiv, 2019. http://arxiv.org/abs/1901.01753 **POET**

[26] R. Wang *et al.*, “Enhanced POET: Open-ended Reinforcement Learning through Unbounded Invention of Learning Challenges and their Solutions,” in *Proceedings of the 37th International Conference on Machine Learning*, PMLR, Nov. 2020, pp. 9940–9951, 2024 https://proceedings.mlr.press/v119/wang20l.html

References 5

[27] Secretan J, Beato N, D'Ambrosio DB, Rodriguez A, Campbell A, Folsom-Kovarik JT, et al. Picbreeder: A case study in collaborative evolutionary exploration of design space. Evolutionary Computation. 2011;19(3):373–403. **Picbreeder**

[28] Nguyen A, Yosinski J, Clune J. Understanding innovation engines: Automated creativity and improved stochastic optimization via deep learning. Evolutionary Computation. 2016;24(3). **Innovation Engine**

[29] D. Ha and Y. Tang, "Collective intelligence for deep learning: A survey of recent developments," *Collective Intelligence*, vol. 1, no. 1, p. 26339137221114874, Aug. 2022, doi: 10.1177/26339137221114874. **Survey on collective intelligence**

[30] S. Wolfram, "Statistical mechanics of cellular automata," *Rev. Mod. Phys.*, vol. 55, no. 3, pp. 601–644, Jul. 1983, doi: 10.1103/RevModPhys.55.601.

[31] J. von Neumann, "The general and logical theory of automata," in *Cerebral mechanisms in behavior; the Hixon Symposium*, Oxford, England: Wiley, 1951, pp. 1–41. **first appearance of cellular automata**

[32] M. Gardner, "Mathematical Games," *Scientific American*, vol. 223, no. 4, pp. 120–123, Oct. 01, 1970. **Conway's Game of Life description**

References 6

[33] K. Bhattacharjee, N. Naskar, S. Roy, and S. Das, “A survey of cellular automata: types, dynamics, non-uniformity and applications,” *Nat Comput*, vol. 19, no. 2, pp. 433–461, Jun. 2020, doi: 10.1007/s11047-018-9696-8.

[34] M. Sandler, A. Zhmoginov, L. Luo, A. Mordvintsev, E. Randazzo, and B. A. y Arcas, “Image segmentation via Cellular Automata.” arXiv, Aug. 12, 2020. doi: 10.48550/arXiv.2008.04965.

[35] O. Yilmaz, “Reservoir Computing using Cellular Automata.” arXiv, Oct. 01, 2014. doi: 10.48550/arXiv.1410.0162.

[36] W. Gilpin, “Cellular automata as convolutional neural networks,” *Phys. Rev. E*, vol. 100, no. 3, p. 032402, Sep. 2019, doi: 10.1103/PhysRevE.100.032402.

[37] Oslo and Akershus University College of Applied Sciences, S. Nichele, M. S. Gundersen, and Norwegian University of Science and Technology, “Reservoir Computing Using Nonuniform Binary Cellular Automata,” *ComplexSystems*, vol. 26, no. 3, pp. 225–246, Oct. 2017, doi: 10.25088/ComplexSystems.26.3.225.

[38] S. Nichele and A. Molund, “Deep Reservoir Computing Using Cellular Automata.” arXiv, Mar. 08, 2017. doi: 10.48550/arXiv.1703.02806.

[39] Hong Kong and B. W.-C. Chan, “Lenia: Biology of Artificial Life,” *ComplexSystems*, vol. 28, no. 3, pp. 251–286, Oct. 2019, doi: 10.25088/ComplexSystems.28.3.251. **Lenia**

[40] B. W.-C. Chan, “Lenia and Expanded Universe,” presented at the ALIFE 2020: The 2020 Conference on Artificial Life, MIT Press, Jul. 2020, pp. 221–229. doi: 10.1162/isal_a_00297. **Extended Lenia**

References 7

[41] E. Randazzo and A. Mordvintsev, “Biomaker CA: a Biome Maker project using Cellular Automata.” arXiv, Jul. 18, 2023. Accessed: Mar. 22, 2024. [Online]. Available: http://arxiv.org/abs/2307.09320 **Biomaker CA**

[42] A. Mordvintsev, E. Randazzo, E. Niklasson, and M. Levin, “Growing Neural Cellular Automata,” *Distill*, vol. 5, no. 2, p. e23, Feb. 2020, doi: 10.23915/distill.00023. **Neural Cellular Automata**

[43] E. Randazzo, A. Mordvintsev, E. Niklasson, M. Levin, and S. Greydanus, “Self-classifying MNIST Digits,” *Distill*, vol. 5, no. 8, p. e00027.002, Aug. 2020, doi: 10.23915/distill.00027.002.

[44] T. Akiba, M. Shing, Y. Tang, Q. Sun, and D. Ha, “Evolutionary Optimization of Model Merging Recipes.” arXiv, Mar. 19, 2024 http://arxiv.org/abs/2403.13187

[45] N. Hansen and A. Ostermeier, “Completely Derandomized Self-Adaptation in Evolution Strategies,” *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001, doi: 10.1162/106365601750190398.