A UNIFIED SUBSTRATE FOR BODY-BRAIN CO-EVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

A successful development of a complex multicellular organism took millions of years of evolution. The genome of such a multicellular organism guides the development of its body from a single cell, including its control system. Our goal is to imitate this natural process using a single neural cellular automaton (NCA) as a genome for modular robotic agents. In the introduced approach, called *Neural Cellular Robot Substrate* (NCRS), a single NCA guides the growth of a robot and the cellular activity which controls the robot during deployment. We also introduce three benchmark environments, which test the ability of the approach to grow different robot morphologies. We evolve the NCRS with covariance matrix adaptation evolution strategy (CMA-ES), and covariance matrix adaptation MAP-Elites (CMA-ME) for quality diversity and observe that CMA-ME generates more diverse robot morphologies with higher fitness scores. While the NCRS is able to solve the easier tasks in the benchmark, the success rate reduces when the difficulty of the task increases. We discuss directions for future work that may facilitate the use of the NCRS approach for more complex domains.

1 INTRODUCTION

Multicellular organisms are made of cells that can divide into many, and specialize in controlling and maintaining the body, sensing the environment, or protecting from external threats. Such features were acquired by evolution from the first living cell. After millions of years, colonies of unicellular organisms appeared, and were essential to the development of multicellular organisms with cellular differentiation. Developmental biologists study how this growth and specialization of cells occur in an organism by being coordinated through their genetic code (Slack & Dale, 2021).

The field of artificial life tries to create life-like computational models by following the same rules of biological life, such as decentralized and local control (Langton, 2019). One of the sub-fields of artificial life, artificial development (Harding & Banzhaf, 2009; Doursat et al., 2013), focuses on modelling or simulating cell division and differentiation. The techniques applied in this sub-field are based on indirect encoding of developmental rules. Thus, being analogous to the genome of a biological organism describing its phenotype. This type of encoding occurs in nature, and it facilitates the scaling of an organism because the information in the genome is much smaller than in the resulting phenotype. This is referred to as genomic bottleneck (Zador, 2019; Variengien et al., 2021), and it implies that the genetic code of an organism compresses all the information to grow and maintain its body, and even a complex brain in some species.

One of the simplest computational models of artificial life or dynamical systems is a cellular automaton (CA) (Wolfram, 2002). A CA can be described as a universe with discrete space and time, which is governed by local rules without any central control. Such a discrete space is divided in a regular grid of cells and can possess any number of dimensions. The most common CAs have one or two dimensions, and their most well-known versions are, respectively, elementary CA (Wolfram, 2002) and Conway's Game of Life (Rendell, 2002). Both have cells with binary states, but other CA can have many discrete states or continuous ones. In 1940s, the first CA was introduced by Ulam and von Neumann (Topa, 2011). Von Neumann aimed to produce self-replicating machines, and Ulam worked on crystal growth. In 2002, a CA with rules defined by an artificial neural network was described (Li & Yeh, 2002). Nowadays, this type of approach is called neural cellular automaton (NCA). In 2017, Nichele et al. (2017) presents a NCA that has developmental features

that were learned through neuroevolution using a method called compositional pattern-producing network. Recently, Mordvintsev et al. (Mordvintsev et al., 2020) introduced a differentiable NCA, which possesses growth and regeneration properties.

In their work, a NCA is trained through gradient descent to grow a colored image from one active "seed" cell.

In evolutionary robotics, co-evolution of morphology and control has the inherent challenge of optimizing two different features in parallel (Bhatia et al., 2021). It also presents scalability issues when it deals with modular robots (Yim et al., 2007). Our goal is to implement an approach where the optimization happens in just one dynamical substrate with local interactions. Here we introduce such a system, a *Neural Cellular Robot Substrate* (NCRS), in which a single NCA grows the morphology of an agent's body and also controls how that agent interacts with its environment.

The NCA has two phases (Fig. 1). First is the developmental phase, in which the robot's body is grown, including where to place its sensors and actuators. In the following control phase, copies of the same NCA are running in each cell of the agent, taking into account only local information from neighboring cells to determine their next state. The optimization task thus entails figuring out how to transmit information from the robot's sensors to its actuators to perform the task at hand.

We also introduce a virtual environment with three benchmark tasks for evaluating the NCRS' capacity of designing a robot and then controlling it. Two benchmarks consist in growing and controlling a robot to approach a light source (Fig. 1b and Fig. 2a).

The third task challenges the robot to carry a ball to a target area. In this benchmark, a second type of rudimentary eye is added, so the robot can differentiate the ball and the target area (Fig. 2b).

The main contribution of this work is the introduction of a single neural cellular automaton that first develops an agent's body from a "seed" and then is able to control it during deployment. While the solved benchmark domains are relatively simple, the unified substrate for both body and brain opens up interesting future research directions, such as opportunities for open-ended evolution (Stanley, 2019).

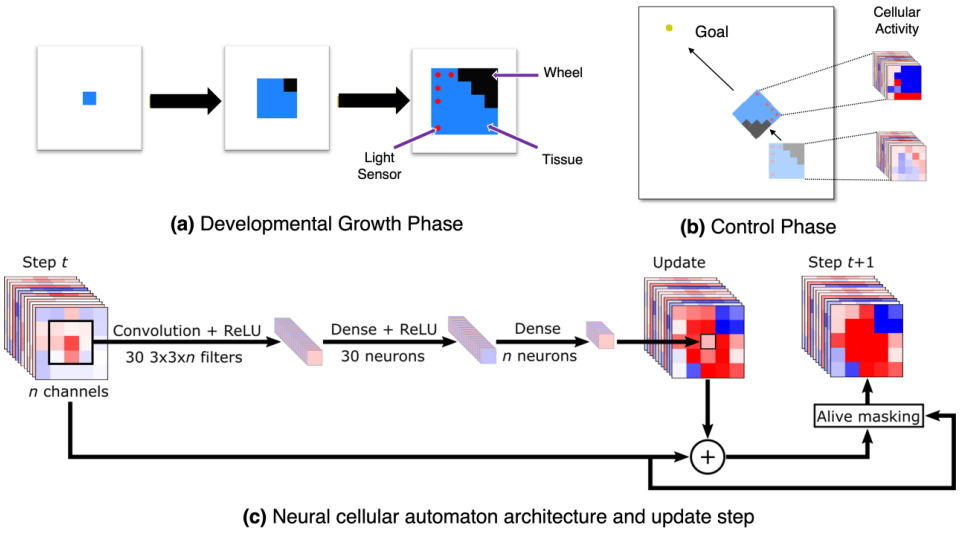


Figure 1: Neural Cellular Robot Substrate (NCRS) In the developmental phase (a), the robot is grown from an initial starting seed, guided by a neural cellular automaton (c). Once grown, the same neural cellular automaton determines how the signals propagate through the robot's morphology during the control phase (b).

2 Related Work

The co-design of robot bodies and brains has been an active area of research for decades (Medvet et al., 2021; Sims, 1994; Komosiński & Ulatowski, 1999). Brain and body co-design stands for

producing a control policy and a morphology for a robotic system. To the best of our knowledge, it is always done by using one method to define the body and another one to model the behavior of the agent. An example is the work of Lipson & Pollack (2000), where the same genome directly encodes the robot's body and the artificial neural network for control. A method that uses a genetic regulatory networks to define separately a body and an artificial neural network was introduced by Bongard & Pfeifer (2003), and named as artificial ontogeny. The evolved robots are able to locomote and push blocks in noisy environments. A recent work by Bhatia et al. (2021) presents several virtual environments and also an algorithm for brain and body co-design with separated description methods for the morphology and control. In comparison with NCRS, our co-design algorithm consists of only one neural cellular automaton, which is known to have self-organizing and self-repair properties (Mordvintsev et al., 2020; Horibe et al., 2021; Variengien et al., 2021).

The growing NCA (Mordvintsev et al., 2020) is one of the first examples of self-organizing and self-repair systems that use differentiable models as rules for cellular automata. Before that, NCA models were typically optimized with genetic algorithms (Nichele et al., 2017). After the work on growing NCA, other neural CAs were introduced, including methods optimized without differentiable programming. There exist other generative methods for growing 3D artefacts and functional machines (Sudhakaran et al., 2021), for developing soft robots (Horibe et al., 2021). Moreover, a NCA was used as a decentralized classifier of handwritten digits (Randazzo et al., 2020).

The developmental phase of our approach is similar to the generative method with NCA for level design trained with CMA-ME in the work of Earle et al. (2021). Morphology design is also present in other works (Hejna III et al., 2021; Talamini et al., 2021; Kriegman et al., 2018). The control phase is based on the NCA for controlling a cart-pole agent introduced by Variengien et al. (2021), but their NCA is trained using a reinforcement learning algorithm named deep-Q learning and the communication between NCA and environment happens in predefined cells. Our approach, NCRS, unifies these two methods by having two phases. The first phase that is generative, and the second one that is an agent's policy.

3 Robot design

The modular robot used in our three environments consists of some squared body parts with size one. Each cell in the cellular automaton defines the modules or body parts of the agent. If there is no body part defined in a cell, no information is passed through it.

The simplest body part of the robot is the body and brain tissue. This module allows communication with other parts, and it gives volume and a bit of weight to the robot. In the environments, it is rendered as a blue square. The sensors are the information inputs to the neural cellular automaton, which is controlling the agent. The equation that calculates the sensor activity s regarding the distance to the light, ball or target is expressed by

$$s = e^{-distance/playfield}. \tag{1}$$

Where the distance between the objects is normalized by the size of the playing field $playfield$. The equation equation 1 is also used to calculate the fitness score, but it uses the center or origin of the cellular automaton grid. The values of the sensor activity or fitness score are between 0 and 1, where 1 means no distance and exponentially decaying to 0 when distance increases. The agent can have two types of sensors. One is the light or ball sensor, which is a red circle on top of a tissue module. The other one is the target area sensor represented as a green circle. This last sensor is only used in the environment for carrying a ball to a target area. The actuator of this agent is a wheel. Its orientation is always vertical during initialization of the benchmark environment. The wheel rotates upwards and downwards relative to the initial angle of the robot. The maximum speeds for each of those directions are, respectively, +1 and -1. Fig. 1a illustrates how each body part is rendered in the environment when it is initialized with no rotation.

4 Benchmark environments

To test the capacity of controlling the developed robot, we implemented three benchmark environments, which are: light chasing (LC), light chasing with obstacle (LCO), and carrying ball to target (CBT). They are environments where a modular robot equipped with simple light sensors and wheels

can be evaluated. In those environments, the size of the playing field and the distance between the objects are affected by the maximum size that the robot can have. Thus, larger the robot can be, bigger is the playing field. In our experiments, we use a robot and a neural cellular automaton grid with size 5×5. Because the possible maximum size of the robot is 5, the size of the playing field is 60. This is the value assigned to $playfield$ in equation 1.

The fitness score is calculated using the average score of 12 runs where the location of the agent, light, ball and target can differ for each run. The light or ball has some predefined regions to be initially placed.

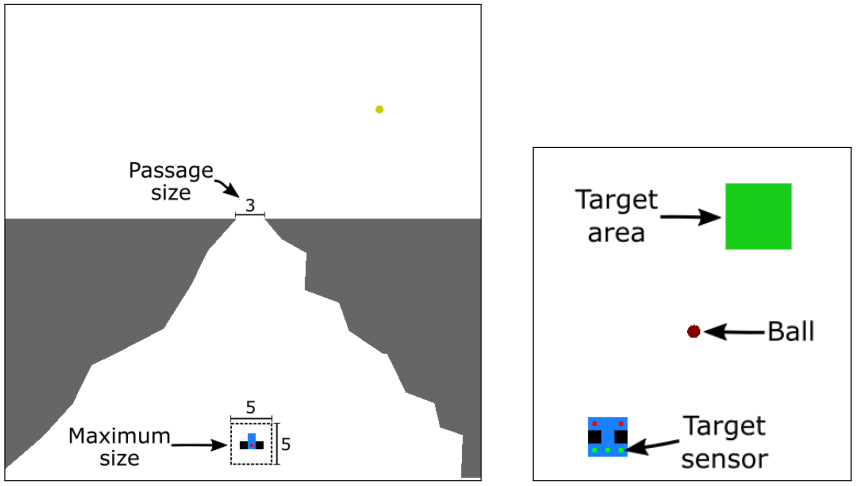
The benchmark environments are based on the implementation of the top-down racing environment in Open AI gym (Brockman et al., 2016). We use the pybox2d that is a 2D physics library in Python.

4.1 Light chasing

The light chasing (LC) environment is shown in Fig. 1b. The goal of the agent is to be closer to the light during the entire simulation. The agent starts in the middle of the playing field. One light is randomly placed around the region of one of the four corners of the playing field. The fitness score is calculated by the average distance between the center of the robot and the center of the light over all simulation time-steps, and a successful run means that this distance reached less than 10 times the module size.

4.2 Light chasing with obstacle

The light chasing with obstacle (LCO) environment is a more difficult version of the light chasing one. In this task, it is added an obstacle, which has a passage through it. The robot does not have sensors to this obstacle, then its morphology plays a bigger role in this benchmark. The passage width is calculated by the possible maximum size of the robot. If the robot can have up to 5×5 body parts, then the passage width would be the size of three body parts. Fig. 2a represents this environment with an obstacle, and the possible maximum size of the depicted robot is 5×5. Therefore, a possible successful body is also illustrated there. The robot is randomly initialized in the bottom of the playing field. An obstacle is procedurally generated with a target passage width and wall roughness. The obstacle has the shape of a funnel because there are no sensors to it, then this helps the robot to reach the passage depending on its body. The passage is randomly located in the horizontal axis and fixed in the vertical axis. The light is at the top and after the obstacle. The initial light location has four predefined regions in the horizontal axis, which are left, center-left, center-right and right. The fitness and success definition are the same as the light chasing task.



(a) Light chasing with obstacle task

(b) Carrying ball to target task

Figure 2: Extensions from the light chasing task. (a) It depicts the original size of the playing field, which is 60.

4.3 CARRYING BALL TO TARGET AREA

Among the three benchmark environments, the task to carry a ball to a target area is the most difficult one. Fig. 2b illustrates the carrying ball to target area (CBT) environment. For the control phase, the robot needs to move towards the ball, and then move to the target area without losing the ball during the transport. For the developmental phase, the body of the robot needs to be adequate to push or kick the ball to the target area, and properly placing the sensors of each type, so it can successfully locate ball and target area. The agent is located at the bottom in a random horizontal location. The ball is located in the middle of the vertical axis of the playing field, but it has the same four predefined regions as the light chasing with obstacle environment. The target is located at the top and its location in the horizontal axis is randomly defined.

Besides the sensor for the ball (or light for the other two environments), a new sensor type is added. With this, the robot can also receive the information where the center of the target area is located.

The fitness score of this environment is the average of the distance between robot and ball, and the distance between the ball and the center of the target area. Since they are distances used to calculate the fitness score, they are normalized using equation 1. The definition of success in this task means carrying the ball to the target, so it can have a distance less than ten times the module size of the robot.

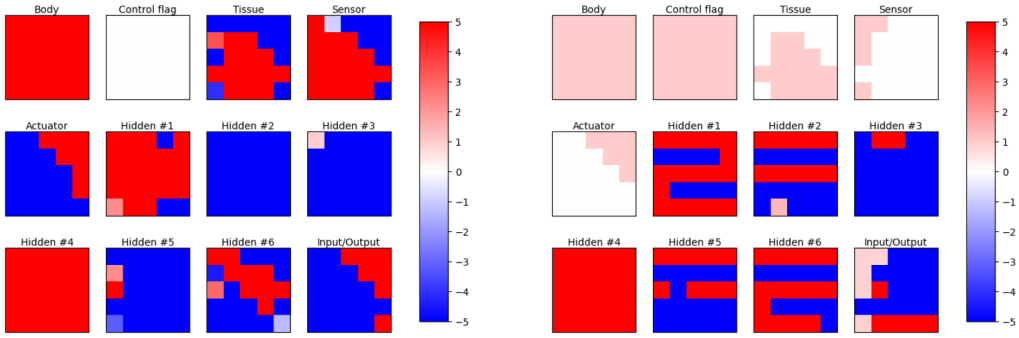
5 APPROACH: A UNIFIED SUBSTRATE

The robot is grown by a neural cellular automaton and then deployed in its particular environment. Importantly, in our approach the same NCA controls both the growth of the modular robot (Fig. 1a) and the robot itself (Fig. 1b). Therefore, it is a unified substrate for body-brain co-design, and called Neural Cellular Robot Substrate (NCRS). The architecture of NCRS is illustrated in Fig. 1c. When the growth process is finished, the channels responsible to define the body modules reflect the robot's morphology, then the NCA can observe and act in the environment using the cells assigned to the specific types of modules, which are sensors, wheels, and tissue.

The state of a cell is updated considering the eight surrounding neighbors and itself, then it forms a 3×3 neighborhood. The values of the nine cells with all the n channels are processed by a trainable convolutional layer with 30 filters of size $3 \times 3 \times n$. Followed by a dense layer of 30 neurons and another one with n neurons for the n channels of the neural CA. After all cells have been computed, the result of this process is added to the previous state of the neural CA, and then it is clipped to the range of $[-5, 5]$. This update is only valid to the cells that are considered "alive", which are the ones that have their value in the body channel greater than 0.1 and their neighbors. This architecture is very similar to the ones in self-classifying MNIST (Randazzo et al., 2020) and in self-organized control of a cart-pole agent (Variengien et al., 2021).

The channels have specific roles in the neural CA, as shown in Fig. 3. The number of channels n differs because of the different number of sensors in the types of benchmark tasks. The body channel is the one that indicates that there is a body part in that cell if its value is greater than 0.1. The neighbors of a body part are allowed to update their states because they are considered "growing". The next channel has fixed values and works as a control flag. When the neural CA is in developmental phase, all cells in this channel are set to zero. When it is in control phase, they are set to one. The following channels are responsible to define the type of the body part. The channel with the highest value is the one that specify the body part. In case of tie, the first channel is selected. The order of those channels are: body/brain tissue, light/ball sensor, target area sensor (if needed), and wheel. In this way, it can define a robot as depicted in Fig. 1a. Then, there are the hidden channels to support the computation in the neural CA. For all benchmark types, the neural CA contains six hidden channels. Finally, the input/output channel, which is the one that receives the values from the sensors and gives the values to the actuators (wheels).

The initial state of the neural CA is a "seed". The middle cell of the grid has the state set as one in the body channel, and the rest is zero. After a few time-steps in the developmental phase, all channels are updated except the control flag channel. This phase lasts for ten time-steps. The end of the developmental phase is represented in Fig. 3a. After development, the control phase starts. In this phase, the benchmark environment initializes with the developed robot body. For advancing one



(a) Final time-step of developmental phase

(b) Final time-step of control phase

Figure 3: Channels of the neural cellular automaton in different stages.

time-step in the environment, the NCA takes two time-steps for defining an action after receiving observations from the sensors. The body and body parts channels become fixed and their values are defined by the robot body. This is used to support the neural CA by identifying the cells with body modules, such as tissues, sensors and actuators. The cell is assigned the value one to the body channel if there is a body part and to the specific body parts channel. Fig. 3b shows this assignment for the identification of body parts during control phase. The robot designed by this NCA is depicted in Fig. 4a. The start of the control phase, the cellular activity of the hidden and input/output channels are set to one. In the input/output channel, only the input cells are fixed and their values come from the sensors.

In our neural CA, there is no noise. Even all "alive" cells are updated every time-step. This is done because the stochastic update or any other type of noise would affect the development of the robot body. After developmental phase, the same model could produce different types of robot body.

For our experiments, the neural cellular automaton has a grid of size 5×5. Therefore, it generates a body for an agent with the same size. Since this is a neural CA, the grid size does not affect the number of trainable parameters. The light chasing and light chasing with obstacle environments require just the light sensor. Therefore, the robot can have tissue, light sensor and wheel. This takes three body part channels. With one body, one control flag, six hidden, and one input/output channels, the total number of channels is 12. In this way, the number of trainable parameters is 4,572. In the carrying ball to target environment, the robot needs one additional sensor. Therefore, it adds one more channel. It results in a neural CA with 4,873 trainable parameters.

6 Training Methods

Our benchmark environments are complicated to deal using deep reinforcement learning due to the different number of inputs and outputs for the policy, as it was done for the cart-pole agent with NCA (Variengien et al., 2021). Therefore, we have chosen to use some derivative-free optimization methods. They are the covariance matrix adaptation evolution strategy (CMA-ES) (Hansen & Ostermeier, 1996) and covariance matrix adaptation MAP-Elites (CMA-ME) (Fontaine et al., 2020). The latter is used to add quality diversity to the former. Therefore, broadening the exploration of robot designs. For both training methods, we use the library CMA-ES/pycma (Hansen et al., 2019).

There are two training methods and three benchmark tasks. This gives a total of six different combinations. Each of these combinations were trained only one time.

The training process happens entirely on CPU. To speed up the evolution of the NCRS, robots with a design that would not work properly in the environment are not simulated. For the two light chasing environments, robots must have at least one light sensor, and two actuators. For the carrying ball to target, they must have one sensor of each type, and two actuators. The fitness scores of the failed designs are calculated according to the number of correct parts they have. For each correct body part, the fitness score increases 0.01.

To compare the quality diversity of CMA-ES and CMA-ME, we use the percentage of cells or feature configurations filled, and QD-score. They measure quality and diversity of the elites (Pugh et al., 2016). QD-score is calculated summing the fitness score of all elites and dividing it by the total number of possible feature configurations. Moreover, CMA-ES and CMA-ME have their elites stored, even though CMA-ES does not use elites during training.

6.1 Covariance matrix adaptation evolution strategy

CMA-ES is one of the best derivative-free numerical optimization methods for continuous domains (Fontaine et al., 2020). CMA-ES runs 20,000 generations for all environments. The initial mean is 0.0 for all dimensions, and the initial coordinate wise standard deviation (step size) is 0.01. The population size or the amount of solutions acquired to update the covariance matrix is 112. This number was selected by the number of available threads in the machine used to train, which contains 56 threads at 2.70GHz.

6.2 Covariance matrix adaptation MAP-Elites

CMA-ME is a variant of CMA-ES that adds the benefits of quality diversity from MAP-Elites (Mouret & Clune, 2015). The changes to CMA-ES are that there are emitters of CMA-ES being trained in a cycle, and there is a feature map to store one elite for each possible feature configuration. Since we have failed body designs, they do not produce an elite. When there is a successful robot design, the number of sensors, actuators, and body parts are used as features. If there is no elites or the current solution is better than the actual elite stored in the feature map, then the current solution is assigned to its feature configuration.

We use a slightly modified version of the CMA-ME with improvement emitters (Fontaine et al., 2020). We only restart an emitter when the number of elites are greater than the number of emitters and it is stuck for more than 500 generations. Being stuck means that the emitter could not find a better elite or an elite to an empty feature configuration in the map. When an emitter restarts, the mean used to initialize the CMA-ES is a random elite in the map.

CMA-ME is executed for 60,000 generations for all environments, except the light chasing environment with 67,446 generations because we forced it to stop a longer training and its best fitness score was already better than the one trained with CMA-ES. The initial mean and the initial coordinate wise standard deviation are the same as CMA-ES for all emitters. The population size is 128 because the CMA-ME training was executed in a computer with 128 threads at 2.9GHz.

7 Results

The training process took around 2.5 days for optimizing the NCA with CMA-ES. The evolution with CMA-ME took around 5.5 days. It is important to note that they do not have the same machine configuration, population size, and maximum number of generations.

Fig. 4 shows all robot designs that have the best fitness scores regarding their training method and task. All robots for the LC and CBT tasks fullfil the entire grid of cells, which is 5×5. Those environments do not have any constraint in the robot size, then we infer that the full grid of modules are easier to design and there are more computational resources for controlling the robot. Their fitness scores are presented in Table 1. This shows that CMA-ES and CMA-ME can reach almost the same fitness scores after training. However, CMA-ME has much less generations for the 15 emitters. This results in 4,000 generations per emitter. It is possible that if we run 20,000 generations per emitter, CMA-ME could reach much better final results than CMA-ES and with much more diversity. The history of the maximum fitness score per generation is depicted in Fig. 5.

The elites were saved for both CMA-ES and CMA-ME, then we can compare their quality diversity. In Table 2, the number of cells filled and QD-scores of all six methods and tasks combinations are presented. It is noticeable that CMA-ME provides much more quality diversity because of their bigger number of feature configurations and their QD-score. We can visualize it in Fig. 6. This shows a small part of the elites produced for the light chasing tasks with CMA-ES and CMA-ME. Nevertheless, it confirms those two quality diversity measurements because more cells are filled, and there are more cells with higher fitness score.

For testing the success of our six trained models, we run 100 times the simulation and the percentage of success is presented in Table 3. We can visualize some examples of those simulations in Fig. 7. The trained model with CMA-ES for the light chasing task got 92% of success rate with 0.58274 fitness score while the one trained with CMA-ME had 75% of success and 0.61481 of fitness score. This means that a higher fitness score does not indicate a more successful model for reaching the light. This can be observed in Fig. 7a-d for CMA-ES, and Fig. 7e-h for CMA-ME. We can see in Fig. 7g that the light is at the top-right corner and the robot goes to the top-left corner. This explains the 75% success rate of this NCRS because the light is at the top-right corner in 25 out of the 100 simulations. This model learned to move faster to the light in the other three corners, but it misses the one in the top-right corner. For the light chasing with obstacle task, the reason for the higher success rate of CMA-ES robot is that it is much thinner than the CMA-ME robot. Therefore, it is easier to pass through the passage. If we define success in LCO by passing the center of the body through the passage, then CMA-ES and CMA-ME had a success rate, respectively, of 77% and 45%. The NCRS did not learn to move to the light after passing through the obstacle. It just moves forward. Because of the difficulty of this task, we can consider the results for LCO were partially successful in general, and successful in body design. Fig. 7i-l and Fig. 7m-p show that. The task for carrying a ball to a target had no successful trained model. The robots for both training methods just moves forward and by chance it moves the ball to target. This can be seen in Fig. 7q-t and Fig. 7u-x.

Fig. 3 shows how the channels progress through time. The hidden channels are predominantly different in their behavior for the developmental and control phases. We infer this is mainly due to the control flag channel which regulates these two phases. We can observe the different patterns that emerged in their final time-steps. From initial "seed" state to the state in Fig. 3a, we can see how the NCA behaves during 10 time-steps of the developmental phase. In Fig 3b, we can see the end of the control phase during its 200 time-steps (100 time-steps in the environment). We can still understand its behavior because the hidden and input/output channels were set to zero in the beginning of the control phase, and the body, control flag, tissue, sensor, and actuator channels were fixed according to the morphology of the robot.

8 Discussion and conclusion

Body-brain co-evolution is a challenging task (Bhatia et al., 2021). In this work, we develop three benchmark tasks for robot co-design and introduce a novel method by having a unified substrate as a genome with its own rules. This substrate is a single neural cellular automaton that works to develop and control a modular robot. This novelty opens up several possibilities in open-ended evolution (Stanley, 2019). The number of body parts and dimensions can increase over time with the progress of the generations. Evolution in multi-agent environments may also be applied, such as in PolyWorld (Yaeger et al., 1994).

The presented results were successful for the LC task, but our trained models presented some failures when increasing the difficulty of the tasks. This may be addressed by adjusting the fitness score to reflect the success conditions, as well as by applying curriculum learning. In future works, we plan to apply our method in the Evolution Gym (Bhatia et al., 2021), or in a modified version of VoxCraft (Liu et al., 2020) for 3D soft robots. Moreover, we aim at training and testing our approach for self-repair and robustness to noise.

References

Jagdeep Bhatia, Holly Jackson, Yunsheng Tian, Jie Xu, and Wojciech Matusik. Evolution gym: A large-scale benchmark for evolving soft robots. *Advances in Neural Information Processing Systems*, 34, 2021.

Josh C Bongard and Rolf Pfeifer. Evolving complete agents using artificial ontogeny. In *Morpho-functional Machines: The new species*, pp. 237–258. Springer, 2003.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

René Doursat, Hiroki Sayama, and Olivier Michel. A review of morphogenetic engineering. *Natural Computing*, 12(4):517–535, 2013.

Sam Earle, Justin Snider, Matthew C Fontaine, Stefanos Nikolaidis, and Julian Togelius. Illuminating diverse neural cellular automata for level generation. *arXiv preprint arXiv:2109.05489*, 2021.

Matthew C Fontaine, Julian Togelius, Stefanos Nikolaidis, and Amy K Hoover. Covariance matrix adaptation for the rapid illumination of behavior space. In *Proceedings of the 2020 genetic and evolutionary computation conference*, pp. 94–102, 2020.

Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of IEEE international conference on evolutionary computation*, pp. 312–317. IEEE, 1996.

Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. Cma-es/pycma on github. *Zenodo, doi*, 10, 2019.

Simon Harding and Wolfgang Banzhaf. Artificial development. In *Organic Computing*, pp. 201–219. Springer, 2009.

Donald J Hejna III, Pieter Abbeel, and Lerrel Pinto. Task-agnostic morphology evolution. *arXiv preprint arXiv:2102.13100*, 2021.

Kazuya Horibe, Kathryn Walker, and Sebastian Risi. Regenerating soft robots through neural cellular automata. In *EuroGP*, pp. 36–50, 2021.

Maciej Komosiński and Szymon Ulatowski. Framsticks: Towards a simulation of a nature-like world, creatures and evolution. In *European Conference on Artificial Life*, pp. 261–265. Springer, 1999.

Sam Kriegman, Nick Cheney, and Josh Bongard. How morphological development can guide evolution. *Scientific reports*, 8(1):1–10, 2018.

Christopher Langton. *Artificial life: proceedings of an interdisciplinary workshop on the synthesis and simulation of living systems*. Routledge, 2019.

Xia Li and Anthony Gar-On Yeh. Neural-network-based cellular automata for simulating multiple land use changes using gis. *International Journal of Geographical Information Science*, 16(4): 323–343, 2002.

Hod Lipson and Jordan B Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974–978, 2000.

S Liu, D Matthews, S Kriegman, and J Bongard. Voxcraft-sim, a gpuaccelerated voxel-based physics engine, 2020.

Eric Medvet, Alberto Bartoli, Federico Pigozzi, and Marco Rochelli. Biodiversity in evolved voxel-based soft robots. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 129–137, 2021.

Alexander Mordvintsev, Ettore Randazzo, Eyvind Niklasson, and Michael Levin. Growing neural cellular automata. *Distill*, 5(2):e23, 2020.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

Stefano Nichele, Mathias Berild Ose, Sebastian Risi, and Gunnar Tufte. Ca-neat: evolved compositional pattern producing networks for cellular automata morphogenesis and replication. *IEEE Transactions on Cognitive and Developmental Systems*, 10(3):687–700, 2017.

Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40, 2016.

Ettore Randazzo, Alexander Mordvintsev, Eyvind Niklasson, Michael Levin, and Sam Greydanus. Self-classifying mnist digits. *Distill*, 5(8):e00027–002, 2020.

Paul Rendell. *Turing Universality of the Game of Life*, pp. 513–539. Springer London, London, 2002. ISBN 978-1-4471-0129-1. doi: 10.1007/978-1-4471-0129-1_18. URL https://doi.org/10.1007/978-1-4471-0129-1_18.

Karl Sims. Evolving 3d morphology and behavior by competition. *Artificial life*, 1(4):353–372, 1994.

Jonathan MW Slack and Leslie Dale. *Essential developmental biology*. John Wiley & Sons, 2021.

Kenneth O Stanley. Why open-endedness matters. *Artificial life*, 25(3):232–235, 2019.

Shyam Sudhakaran, Djordje Grbic, Siyan Li, Adam Katona, Elias Najarro, Claire Glanois, and Sebastian Risi. Growing 3d artefacts and functional machines with neural cellular automata. *arXiv preprint arXiv:2103.08737*, 2021.

Jacopo Talamini, Eric Medvet, and Stefano Nichele. Criticality-driven evolution of adaptable morphologies of voxel-based soft-robots. *Frontiers in Robotics and AI*, 8:172, 2021.

Pawel Topa. Network systems modelled by complex cellular automata paradigm. *Cellular automata-simplicity behind complexity. InTech, Europe, China*, pp. 259–274, 2011.

Alexandre Variengien, Sidney Pontes-Filho, Tom Eivind Glover, and Stefano Nichele. Towards self-organized control: Using neural cellular automata to robustly control a cart-pole agent. *Innovations in Machine Intelligence*, 1:1–14, Dec 2021. doi: 10.54854/imi2021.01.

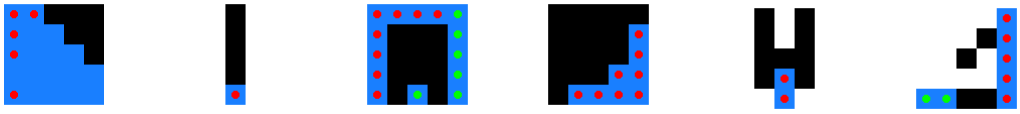
Stephen Wolfram. *A new kind of science*, volume 5. Wolfram media Champaign, IL, 2002.

Larry Yaeger et al. Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or poly world: Life in a new context. In *SANTA FE INSTITUTE STUDIES IN THE SCIENCES OF COMPLEXITY-PROCEEDINGS VOLUME-*, volume 17, pp. 263–263. Citeseer, 1994.

Mark Yim, Wei-Min Shen, Behnam Salemi, Daniela Rus, Mark Moll, Hod Lipson, Eric Klavins, and Gregory S Chirikjian. Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robotics & Automation Magazine*, 14(1):43–52, 2007.

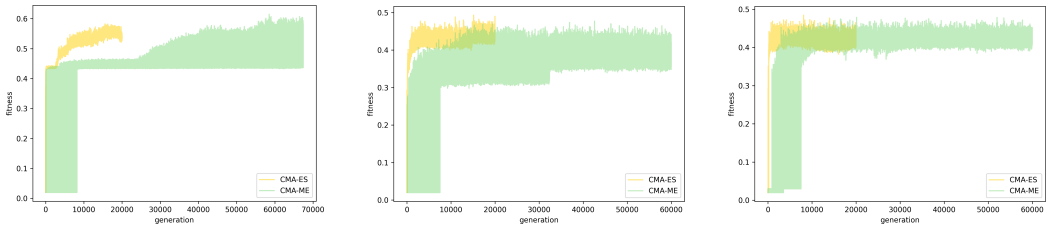
Anthony M Zador. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature communications*, 10(1):1–7, 2019.

A APPENDIX



(a) CMA-ES - LC (b) CMA-ES - LCO (c) CMA-ES - CBT (d) CMA-ME - LC (e) CMA-ME - LCO (f) CMA-ME - CBT

Figure 4: Robot designs with best fitness scores for the tasks of light chasing (LC), light chasing with obstacle (LCO) and carrying ball to target (CBT).



(a) Light chasing (b) Light chasing with obstacle (c) Carrying ball to target

Figure 5: Maximum fitness score through generations.

Table 1: Best fitness score after training in the tasks of light chasing (LC), light chasing with obstacle (LCO) and carrying ball to target (CBT)

	CMA-ES	CMA-ME
LC	0.58274	0.61481
LCO	0.49295	0.47723
CBT	0.48445	0.47884

Table 2: Elites stored during training for the light chasing (LC), light chasing with obstacle (LCO) and carrying ball to target (CBT)

	CMA-ES		CMA-ME	
	Cells filled	QD-score	Cells filled	QD-score
LC	67.58%	0.29530	89.57%	0.40152
LCO	17.88%	0.06069	61.80%	0.19841
CBT	58.10%	0.22957	93.82%	0.37996

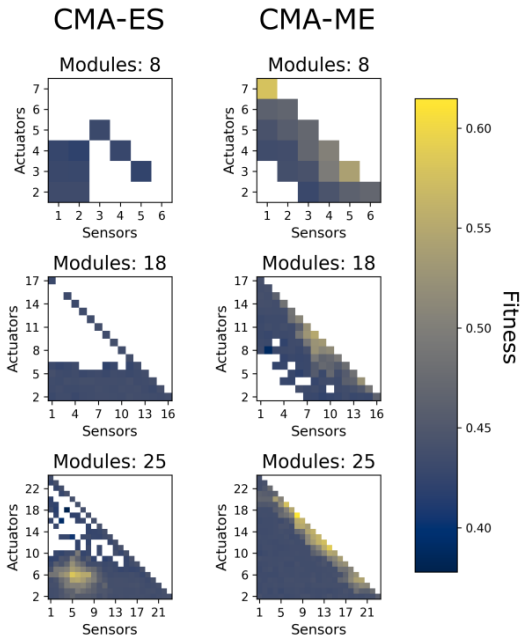
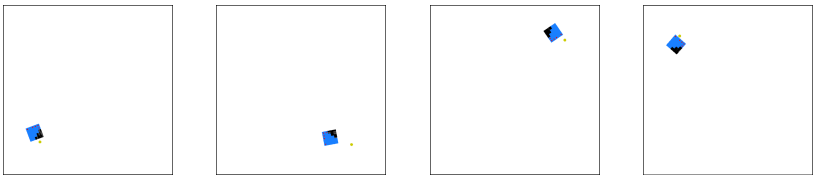


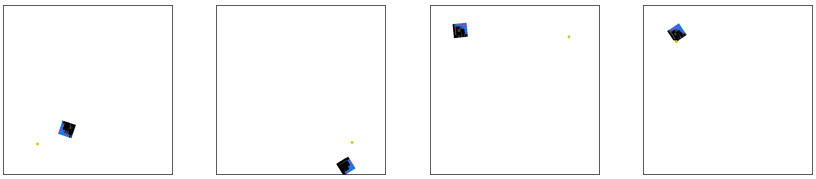
Figure 6: Selected elites trained in the light chasing environment. Axes and subplots indicate the number of components.

Table 3: Testing success percentage over 100 runs for the tasks of light chasing (LC), light chasing with obstacle (LCO) and carrying ball to target (CBT)

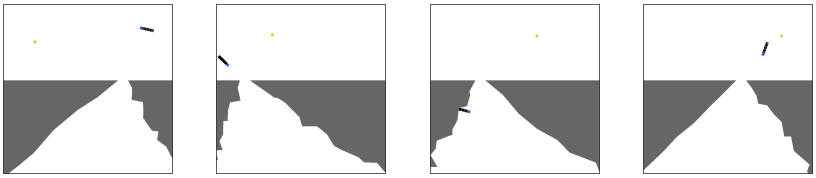
	CMA-ES	CMA-ME
LC	92%	75%
LCO	20%	8%
CBT	1%	2%



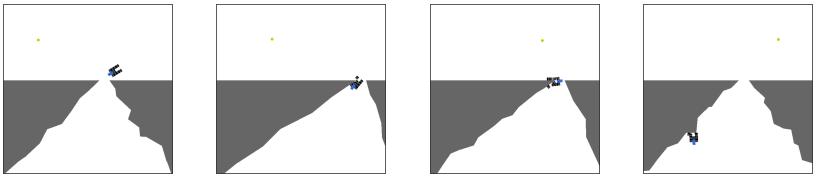
(a) CMA-ES - LC #1 (b) CMA-ES - LC #2 (c) CMA-ES - LC #3 (d) CMA-ES - LC #4



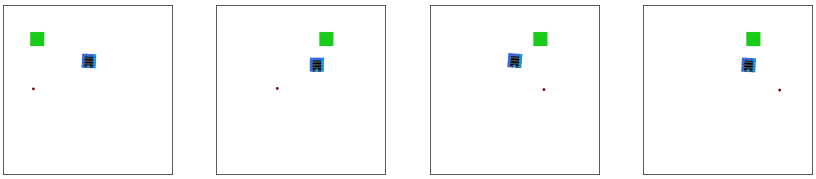
(e) CMA-ME - LC #1 (f) CMA-ME - LC #2 (g) CMA-ME - LC #3 (h) CMA-ME - LC #4



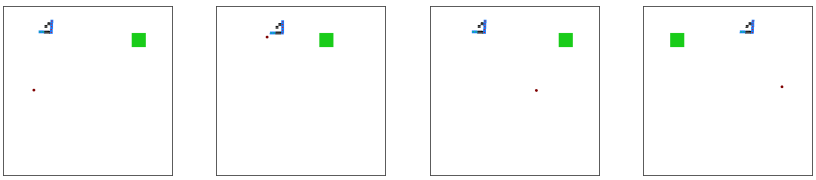
(i) CMA-ES - LCO #1 (j) CMA-ES - LCO #2 (k) CMA-ES - LCO #3 (l) CMA-ES - LCO #4



(m) CMA-ME - LCO #1 (n) CMA-ME - LCO #2 (o) CMA-ME - LCO #3 (p) CMA-ME - LCO #4



(q) CMA-ES - CBT #1 (r) CMA-ES - CBT #2 (s) CMA-ES - CBT #3 (t) CMA-ES - CBT #4



(u) CMA-ME - CBT #1 (v) CMA-ME - CBT #2 (w) CMA-ME - CBT #3 (x) CMA-ME - CBT #4

Figure 7: Last time-step where the robot is fully visible of the best NCRS trained with CMA-ES and CMA-ME in the environments for light chasing (LC), light chasing with obstacle (LCO) and carrying ball to target (CBT).