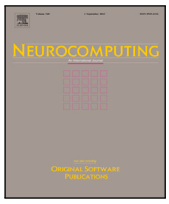
Neurocomputing 614 (2025) 128811

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



# Totipotent neural controllers for modular soft robots: Achieving specialization in body–brain co-evolution through Hebbian learning

Andrea Ferigo [a,1], Giovanni Iacca [a,*,1], Eric Medvet [b,1], Giorgia Nadizar [c,1]

[a] Department of Information Engineering and Computer Science, University of Trento, Italy
[b] Department of Engineering and Architecture, University of Trieste, Italy
[c] Department of Mathematics and Geosciences, University of Trieste, Italy

ARTICLE INFO

*Keywords:*
Hebbian learning
Neural plasticity
Voxel-based Soft Robots
Evolutionary Algorithms

ABSTRACT

Multi-cellular organisms typically originate from a single cell, the zygote, that then develops into a multitude of structurally and functionally specialized cells. The potential of generating all the specialized cells that make up an organism is referred to as cellular "totipotency", a concept introduced by the German plant physiologist Haberlandt in the early 1900s. In an attempt to reproduce this mechanism in synthetic organisms, we present a model based on a kind of modular robot called Voxel-based Soft Robot (VSR), where both the body, *i.e.*, the arrangement of voxels, and the brain, *i.e.*, the Artificial Neural Network (ANN) controlling each module, are subject to an evolutionary process aimed at optimizing the locomotion capabilities of the robot. In an analogy between totipotent cells and totipotent ANN-controlled modules, we then include in our model an additional level of adaptation provided by Hebbian learning, which allows the ANNs to adapt their weights during the execution of the locomotion task. Our in silico experiments reveal two main findings. Firstly, we confirm the common intuition that Hebbian plasticity effectively allows better performance and adaptation. Secondly and more importantly, we verify for the first time that the performance improvements yielded by plasticity are in essence due to a form of *specialization* at the level of single modules (and their associated ANNs): thanks to plasticity, modules specialize to react in different ways to the same set of stimuli, *i.e.*, they become functionally and behaviorally different even though their ANNs are initialized in the same way. This mechanism, which can be seen as a form of totipotency at the level of ANNs, can have, in our view, profound implications in various areas of Artificial Intelligence (AI) and applications thereof, such as modular robotics and multi-agent systems.

## 1. Introduction

One of the most striking properties of multi-cellular organisms is the fact that they typically originate from a single cell that, starting from a single common "code", the genome, develops into a multitude of structurally and functionally *specialized* cells. The ability of non-reproductive cells (called *stem cells*) to produce, by repeated divisions, *all* the specialized cells that make up an organism is generally referred to as cellular "totipotency", a concept introduced by the German plant physiologist Haberlandt in the early 1900s [1], and later on further elaborated in several works [2–4].

Thanks to this mechanism, stem cells have the ability to develop into any kind of cells of the body, from the muscular fibers to the different kinds of blood cells, and, ultimately, the neurons in the nervous system. Of note, this kind of cellular specialization is not the only key to the success of multi-cellular life forms. In fact, *structurally* specialized cells can also become *behaviorally* specialized, *i.e.*, they can *adapt* their behavior during their lifetime. For instance, muscle cells adapt their behavior based on the kind of training they undergo (e.g., endurance vs. strength training) [5]; white blood cells memorize the menaces to our system and adapt their behavior to properly contrast them [6]; neurons rearrange their connections through learning, and generate new synapses through the synaptogenic process [7].

In the light of these observations, it is clear that both structural and behavioral adaptation – at various spatial scales, from cells to body – is a key element for the success of an organism. Not only that: adaptation acts also at various temporal scales, playing a crucial role both *during* the lifetime of an organism and *beyond* it, through evolution. Eiben and Hart have recently captured the synthesis of the adaptation during

* Corresponding author.
*E-mail addresses:* andrea.ferigo@unitn.it (A. Ferigo), giovanni.iacca@unitn.it (G. Iacca), emedvet@units.it (E. Medvet), giorgia.nadizar@phd.units.it (G. Nadizar).
[1] All authors contributed equally to this work and their names appear in alphabetical order.

https://doi.org/10.1016/j.neucom.2024.128811
Received 8 January 2024; Received in revised form 12 August 2024; Accepted 31 October 2024
Available online 9 November 2024

and beyond the individual lifetime with one concept: "if it evolves it needs to learn" [8]. Of note, while this concept has been introduced specifically in the field of morphologically evolving robot systems [9], we can argue that it may apply also to other domains, including the biological one, where there is an interplay between evolution and adaptation/learning.

In this work, we further extend this concept, by specifically focusing on the interplay between *specialization*, *learning*, and *evolution* (in particular, body–brain co-evolution). More specifically, we attempt to reproduce the totipotency mechanism in a special kind of (simulated) modular robot, called Voxel-based Soft Robots (VSRs) [10], whose body, *i.e.*, the arrangement of voxels, and brain, *i.e.*, the ANN controlling each module, are subject to a co-evolutionary process aimed at optimizing the locomotion capabilities of the robot. The modularity of these robots provides, in fact, an excellent opportunity for replicating *in silico*, and studying, the specialization mechanism: we hypothesize that, in the specific case of *modular* evolving entities, the concept of "if it evolves it needs to learn" must be extended to the concept of "if it evolves and learns, it needs to specialize *at the level of its modules*". In essence, we draw an analogy between totipotent cells in living organisms and totipotent ANN-controlled modules, where specialization is achieved through Hebbian learning [11], which allows to adapt, *differently for each module*, the ANNs weights during the execution of the locomotion task.

To the best of our knowledge, this work is the first one to unify, into a single model, ① modularity (and hence **specialization** at the level of modules), ② **learning** (particularly, Hebbian learning), and ③ evolution (particularly, **body–brain co-evolution**). Previous works [8,12,13] already showed that learning is beneficial to the body–brain co-evolution of robots, possibly because through adaptation during their lifetime, robots under evolution explore a larger search space than the one which would be explored by non-plastic agents—a form of Baldwin effect [14,15]. Here, we take a step further and let learning operate *independently* on the components (*i.e.*, the modules) of the robot, where the various, different modules must cooperate to reach a common goal. Of note, this can be seen as an instance of collective intelligence [16,17]. In particular, since the modules in our robots are identical in terms of mechanical properties, structure, and initial weights of their ANN-based controllers, the use of Hebbian learning allows us to disentangle the effect of specialization (*i.e.*, verify if, and how, identical modules behave differently) from that of the co-evolution of the body and brain of the robots.

Moreover, we remark that Hebbian learning is a form of *unsupervised learning*: during their lifetime, the ANNs that control the modules of the robots are not driven by an externally dictated reward signal, but adapt according to emerging rules that are the result of the evolution. In this regard, we find through a set of ad hoc experimental analyses that the differentiation of the modules (w.r.t. their position and access to sensory information) in the body of the robot is crucial for driving adaptation through Hebbian learning.

The present paper is a natural prosecution of our ongoing research on VSRs, and it builds principally on our previous works [18,19]. In particular, in [18] we focused on the evolution of Hebbian ANN-based controllers, keeping the VSR morphologies fixed. The goal of that work was to compare Hebbian learning with the direct evolution of the weights of a feed-forward ANN, investigating primarily the ability to generalize and the robustness of the evolved Hebbian controllers. Whereas, in [19] we focused on the co-evolution of the body and controller of VSRs, with a direct encoding of the ANNs weights, co-evolved along with the parameters of the body. However, that work did not include any form of Hebbian learning in the ANNs. In essence, the present work combines [18,19] in that it investigates for the first time the co-evolution of body and controller (previously studied in [19]) *with* Hebbian learning (previously studied in [18]), focusing on the emergence of specialization as a consequence of both mechanisms.

The rest of the paper is structured as follows. In the next section, we briefly overview the related works. In Section 3, we present the methods used in this study. In Section 4, we discuss the experiments we performed and the main findings derived from our results. Finally, in Section 5 we conclude the work and suggest some possible future research directions.

## 2. Related works

In the following, we briefly review the literature related to the three main concepts mentioned earlier which are used in our work, namely ① specialization, ② learning, and ③ body–brain co-evolution.

### 2.1. Specialization

As discussed earlier, a totipotent cell is a stem cell that can differentiate into any other cell. Of note, neurons can further differentiate their behavior, a process called *neural differentiation* [20], hence becoming functionally specialized according to their position in the brain and, more in general, in the nervous system.

In the field of ANNs, different works analyzed a similar form of specialization at *neural level*, both in the case of a single task, e.g., object recognition [21], and in the case of a single ANN optimized for multiple tasks [22]. These works reached the same conclusion: after training, the ANN is divided into functionally specialized areas.

Beyond the ANN domain, another recent work [17] has shown that, in the context of Multi-Agent Reinforcement Learning (MARL) in which the goals of a group of agents are *aligned*, specialization emerges as an enabler for cooperation. Similar considerations have been made in [23], where authors found that, in the context of public games, social sanctioning can be a way to regulate specialization in a group of agents. Last, Kosak et al. [24] proposed a mobile multi-robot system where agents can autonomously reconfigure themselves, practically achieving a form of totipotency, both in silico and with real robots: the cited study focuses more on the hardware and system organization which allows for the reconfigurability of robotic agents, rather than on the adaptation mechanisms that actually make specialization emerge, as we do in our study.

In the context of modular robotics, Auerbach and Bongard [25] have focused on the specialization of different parts of the body of the agent for performing a task composed of two sub-tasks, highlighting how the modules would specialize for either or both sub-tasks. More recently, Whitman et al. [26] experimented with specialization on modular soft robots. Yet, this study differs from ours in that modules were designed by hand, and then assembled to achieve specialization patterns in the ensemble agent.

To summarize, while specialization has been widely studied in Multi-Agent Systems (MASs) and at the level modules in modular robots, with few other works studying instead the specialization at neural levels (*i.e.*, specialization of specific neurons – or groups thereof – within a given ANN), to the best of our knowledge no prior works have studied the specialization of a *single* ANN to multiple ANNs for the control of modular robots, which is instead a distinctive feature of our model. Note that this kind of specialization differs from the case where a single ANN learns to perform multiple tasks, which has the known drawback of leading to the so-called "catastrophic forgetting" effect [27–29], *i.e.*, whenever an ANN specializes for a new task, it tends to forget how to perform the previous tasks.

### 2.2. Learning

A well-established fact in the evolutionary theory is that learning *during* the lifetime, namely adapting the brain (*i.e.*, the controller) of an agent during the execution of a given task, is a crucial element to achieve improved results compared to approaches based on evolution alone, especially when facing complex scenarios. The importance of

learning has been acknowledged in both natural [14] and artificial evolution [30,31]. In both cases, learning essentially boils down to changing how the agent's brain elaborates the information received from the environment during the agent's lifetime (*i.e.*, in the case of artificial evolution, during the execution of the task). This capability is called *neural plasticity*, and it can be differentiated in structural or functional plasticity [32].

In the former case, plasticity affects the structure of the ANN, e.g., by pruning some connections [33,34], or developing the ANN from scratch [35].

In the functional case, plasticity affects only the parameters of the ANN, having its structure fixed. The parameters are usually changed through plasticity rules, for which the main mechanism used is Hebbian learning, that is the method we use in this work, see Section 3.2. It is also possible to indirectly encode the plasticity rules, like in Adaptive HyperNEAT [36], where a smaller ANN learns the weights and the plasticity rules for the main controller. Other approaches instead directly optimize the parameters of fixed plasticity rules [37–39] or completely evolve new functions [40]. Recently, two works have proposed to reduce the amount of plasticity rules/parameters in an ANN, namely by clustering rules [41] or by grouping rules by neuron [42].

Finally, some approaches take into consideration both structural and functional plasticity. For example, SBNN [43] takes inspiration from synaptogenesis by applying both a pruning algorithm and a plasticity mechanism to incrementally modify the ANN during the agent's life. Dresp-Langley [44] provides a survey exploring various biologically inspired learning models for robot control; we refer the reader to this work for further details and pointers on the topic.

### 2.3. Body–brain co-evolution

Body–brain co-evolution (where the body is typically described in terms of some mechanical parameters of the robot, while the brain has the form of an ANN) has been extensively studied in the evolutionary robotics field, under the premise that co-evolving (*i.e.*, co-optimizing) the brain and the body of a robot reduces the bias and effort of human-made design. This is particularly important in the field of modular robotics, where the morphological search space can be very wide [45] and non-trivial to explore. In particular, recent studies on the analysis of the fitness landscape of morphologically evolving robots [46] have noted the importance of choosing suitable representations for achieving high-quality results when jointly optimizing the body and brain.

Another advantage of co-evolving the brain and body of a robot is the capability to find a balance between the morphological and controller complexity [12,47,48], as opposed to evolving either the morphology or the controller alone [49,50]. Notably, Pagliuca and Nolfi [51] have highlighted how the superior performance of co-evolution often arises from the possibility of co-adapting the morphological traits to the control traits and vice versa. Yet, interestingly, Mertan and Cheney [52] have remarked how body–brain co-evolution might lead to premature convergence to sub-optimal solutions.

In fact, using a single, centralized, controller, as typically done in body–brain co-evolution studies, can lead to a mismatch between the structure of the ANN and the body [8]. One possible solution is the use of a *modular* controller, in which different controllers are instantiated to operate each module of the body, thus reducing the complexity of adapting a single controller to the new shape and the number of parameters to optimize [53,54]. This is the approach we take in the present work. On the other hand, to the best of our knowledge most body–brain co-evolution studies have focused on the offline optimization of the body, while none of these previous works applied lifetime learning during the task execution to perform online adaptation.

As a final note, it is important to remark that when a modular controller is used different input–output settings are possible. For instance, each module can take as input the inputs of the adjacent controller [55]. Or, it can take as input also the outputs of the adjacent controllers [28,56]. Some recent works have proposed using a self-attention mechanism [57]. Moving even further, Pontes-Filho et al. [58] proposed a modular controller based on Neural Cellular Automata, which could be used to first encode the body, during an initial developmental phase, and then control the agent behavior, during the working phase.

## 3. Methods

### 3.1. Voxel-based Soft Robots (VSRs)

VSRs are a class of modular soft robots, originally introduced in [10]. They are composed of an ensemble of elastic blocks, named *voxels*, which constitute the modules and can actively contract and expand w.r.t. their resting state. These actions, if properly synchronized among all the voxels, determine the VSR movement, in a way that is similar to what happens in the biological muscular tissue. A VSR is defined by two main components (the body and the controller), which we describe in deeper detail in the following two subsections.

In this work, we consider the simulated 2-D variant of VSRs initially proposed by Medvet et al. [59], in which simulation occurs in continuous space and discrete time (with a frequency $f_{\text{sim}} = 60\,\text{Hz}$). Hence, in our case, the voxels become squares, rather than cubic blocks, which can change area, rather than volume, over time.

Although we simulate VSRs in 2-D, we deem our results to be conceptually portable to the 3-D case, and, eventually, to the real world. In fact, there exist a few examples of physical implementations of these robotic systems. Starting from the seminal work by Hiller and Lipson [10], most realizations of VSRs have relied on pneumatic actuation [60–62], in conjunction with silicone-based materials. However, the latest frontiers of innovation in the field even encompass VSRs realized out of living matter [63].

#### 3.1.1. Body

The body of a VSR defines its external aspect, *i.e.*, its shape and its constituting material, and its sensory apparatus. We define the shape of a VSR as a polyomino, *i.e.*, a plane geometric figure formed by joining one or more equal squares edge to edge. Concerning the material constituting the VSR, its nature boils down to the way in which we model the voxels in the simulation. Here, following the model introduced in [59], we simulate the voxel elasticity with a number of spring–damper systems, with constraints on the maximum voxel length to prevent excessive deformation, linking four masses at the corners; we also use the masses to allow the welding of adjacent voxels into a single connected system.

Concerning the sensory apparatus of a VSR, we equip its voxels with sensors, which are *embodied*, and confer to each individual voxel both external environment perception and proprioception capabilities. Note that in this study the sensor apparatus is predefined, differently from previous works where it was subjected to evolutionary optimization [64,65]. At every time step $k$, each $j$th sensor reads a value $r_j^{(k)} \in [0, 1]$. In this study, we employ three types of sensors that collect various types of information about the VSR status and its relationship with the environment, namely:

1. **area sensors**, that perceive the ratio between the current area of the voxel they are placed in and its rest area;
2. **touch sensors**, that determine if a voxel is in contact with the ground ($r_j^{(k)} = 1$) or not, ($r_j^{(k)} = 0$);
3. **velocity sensors**, that measure the velocity of the voxel center of mass along the voxel $x$- and $y$-axes.

Hence, in total, the cardinality of the sensor inputs for each voxel is 4 (one value for the area sensor, one for the touch sensor, and two for the velocity sensors). For the area and velocity sensors, we rely on a soft normalization of the raw readings, based on the tanh function, followed

by re-scaling, to ensure that each sensor reading lies in $[0, 1]$. Lastly, in order to simulate sensor noise, we add a Gaussian noise with zero mean and $\sigma = 0.01$ to each reading.

The body of a VSR is also responsible for achieving movement. As previously anticipated, the motion of a VSR derives from the rhythmical contraction and expansion of its voxels, which is actively guided by a control signal, but also influenced by external forces exerted on the voxel, e.g., gravity. In our simulation, we model actuation with an instantaneous variation of the rest length of the spring–damper systems embedded in the voxel. The control signal dictating the actuation, $c_i^{(k)} \in [-1, 1]$ for each voxel $i$ at every time step $k$, is computed by the VSR controller, and instructs the voxel to contract to its maximum capability when $c_i^{(k)} = 1$ and to expand to its maximum capability when $c_i^{(k)} = -1$.

### 3.1.2. Controller

The controller of a VSR computes the control signals for the voxels to the extent of guiding their motion. Although simple open-loop controllers usually suffice for achieving elementary movements [66], given the availability of sensors placed along the VSR body, here we rely on a form of closed-loop controller. Namely, we consider the *distributed neural controller* proposed by Medvet et al. [67], which resorts to an ANN placed in each voxel to process the local sensory information and compute the local control signal. Moreover, to allow coordination of the individual ANN-based controllers, so that a form of *collective intelligence* can arise [16], we enable communication between adjacent voxels. Such communication occurs in terms of targeted message passing between neighbors: each voxel controller computes the local control signal, together with 4 numerical values to be passed to its neighbors (one value per neighbor), and receives as inputs the 4 local sensor readings along with the 4 communication values received by the adjacent voxels.

Formally speaking, each controller operates according to the following equation at each control time step $k$:

$$\left[c^{(k)}\ o_{\rightarrow}^{(k)}\ o_{\downarrow}^{(k)}\ o_{\leftarrow}^{(k)}\ o_{\uparrow}^{(k)}\right] = \mathrm{ANN}_{\boldsymbol{\theta}}\left(\left[\boldsymbol{r}^{(k)}\ i_{\uparrow}^{(k-1)}\ i_{\leftarrow}^{(k-1)}\ i_{\downarrow}^{(k-1)}\ i_{\rightarrow}^{(k-1)}\right]\right), \tag{1}$$

where $c^{(k)}$ is the local control signal, $o_{\rightarrow}^{(k)}$, $o_{\downarrow}^{(k)}$, $o_{\leftarrow}^{(k)}$, and $o_{\uparrow}^{(k)}$ are the communication values for the right, bottom, left, and top neighbors, $\boldsymbol{r}^{(k)}$ is the vector of local sensor readings, $i_{\uparrow}^{(k-1)}$, $i_{\leftarrow}^{(k-1)}$, $i_{\downarrow}^{(k-1)}$, and $i_{\rightarrow}^{(k-1)}$ are the communication values computed by the top, left, bottom, and right neighbors at the previous time step, and $\boldsymbol{\theta}$ are the parameters of the ANN—when a neighbor voxel is not present, we set to 0 the corresponding incoming communication value.

We report a graphical representation of the ANN controller embedded in a voxel in Fig. 1. In the example shown in the figure, for simplicity, we have $\boldsymbol{r} = [r_1\ r_2]$, such that $|\boldsymbol{r}| = 2$ (recall that, in our experiments, we instead have 4 sensor readings, hence $|\boldsymbol{r}| = 4$). As we can notice from both the figure and Eq. (1), the considered ANN has 5 outputs and $|\boldsymbol{r}| + 4 = 6$ inputs (in our experiments, we have instead $|\boldsymbol{r}| + 4 = 8$).

It should be noted that, as a consequence of the communication connecting adjacent ANNs, the overall controller of a VSR is recurrent, hence *stateful*, regardless of the types of the individual ANNs employed in all the voxels.

In this work, we consider two variants of ANNs as controllers. ① The first controller that we consider is a MLP, *i.e.*, a fully connected feed-forward ANN where neurons are organized in layers and those in adjacent layers are fully connected. ② The second controller, instead, is a *plastic* MLP, in which Hebbian learning occurs during the lifetime of the agent, governed by a set of predefined rules.

For both controllers, we set the control frequency to 4 Hz, *i.e.*, the controller computes the control and communication values every 15 simulation time steps, to prevent high-frequency vibrating behaviors [68], which would not represent realistic motion patterns. Of note, both MLPs and hMLPs have been used previously as controllers for the kind of VSRs used in this work [18,69]. We keep the architecture of the ANN fixed, thus for the MLP the parameters vector $\boldsymbol{\theta}$ coincides with the vector of synaptic weights, whereas for the hMLP it corresponds to the vector of the parameters of the rules dictating the adaptation of the synaptic weights (more on this in Section 3.2).

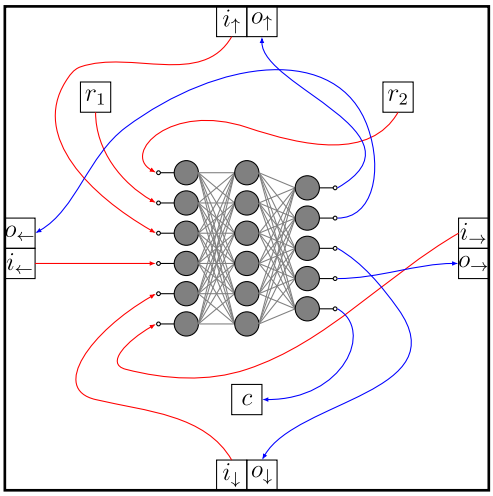



**Fig. 1.** Schematic of a voxel with an ANN controller within it.

For both the MLP and the hMLP, we instantiate a controller with the same $\boldsymbol{\theta}$ in all voxels. This means that for the static MLP, all voxels possess the same weights, whereas in the hMLP all voxels are governed by the same Hebbian learning rules; however, for the hMLPs, the synaptic weights can change *independently for each synapse*, depending on the signals traveling across synapses in each hMLP and triggering the activation of the rules. We remark that using the same parameters $\boldsymbol{\theta}$ for each controller is a key feature of our study: since voxels also share the same material and local sensors, all the modules of the modular robot are identical. This greatly increases the potential applicability of our results in a real scenario, as fabricating and deploying modular robots composed of identical modules is easier than assembling different modules that require careful placement in the overall body plan. As we will show in Section 4, this versatility is achieved without sacrificing the capability of the robot to perform its task, because modules do specialize during the life of the robot, thanks to the adaptation of the controller allowed by hMLPs.

## 3.2. Hebbian learning

Hebbian learning is a form of plasticity that allows an ANN to adapt, *i.e.*, to change its synaptic weights when processing a sequence of inputs. This adaptation is based on the idea put forward by Hebb [70] that the strength of a synapse should change depending on the strength of the activation of the two neurons it connects. Importantly, this form of adaptation is agnostic w.r.t. any overall measure of the degree to which the ANN is performing its task (*i.e.*, it does not use any reward signal), because it is based only on the local knowledge of each synapse, in particular the activation of the pre-synaptic and post-synaptic neurons. In these terms, Hebbian learning can be considered a form of *unsupervised learning*.

In detail, we used a form of Hebbian learning called the ABCD model (named after the four coefficients defining the update rule) [11] applied to an MLP. More specifically, at each time step $k$, the weight $w_{l,i,j}$ of a synapse connecting the $i$th neuron of the $l-1$-th layer with the $j$th neuron of the $l$th layer is updated as follows:

$$w_{l,i,j}^{(k)} = w_{l,i,j}^{(k-1)} + \eta\left(A a_{l-1,i}^{(k-1)} + B a_{j}^{(k-1)} + C a_{l-1,i}^{(k-1)} a_{l,j}^{(k-1)} + D\right), \tag{2}$$

where $\eta$ is the learning rate, $a_{l,j}^{(k-1)} = \tanh\left(\sum_i w_{l,i,j}^{(k-1)} a_{l-1,i}^{(k-1)} + w_{l,i,0}^{(k-1)}\right)$ is the post-synaptic activation value at the previous time step, and $a_{l-1,i}^{(k-1)}$ is

the pre-synaptic value at the previous time step. The role of the learning rate $\eta$ is to make the adaptation of the synaptic weights faster or slower: the larger its value, the faster the adaptation.

The $A$, $B$, $C$, and $D$ coefficients determine how each synaptic weight updates over time, starting from a $w^0_{l,i,j}$; these coefficients can hence be optimized to achieve a desired behavior. Based on previous works [18,39], we use a different set of ABCD coefficients *for each synapse* in the hMLP, instead of sharing the same four values across all the synapses. Moreover, we start each task execution by setting all weights to zero. It follows that the number of optimizable parameters (*i.e.*, $|\boldsymbol{\theta}|$) in an hMLP with $n$ synapses is $|\boldsymbol{\theta}| = 4n$; for reference, a MLP with the same architecture where the synaptic weights are directly optimized has a number of optimizable parameters of $|\boldsymbol{\theta}| = n$.

### 3.3. Body–brain co-evolutionary optimization of VSRs

As introduced before, we optimize at the same time the body and the brain of a VSR, to make the robot effective at solving a given task, in this case locomotion. For this purpose, we rely on an Evolutionary Algorithm (EA), described in detail in Section 3.3.2, that explores a space of candidate solutions, *i.e.*, VSRs differing in body and brain.

We take a similar approach to previous works which evolved concurrently the body and the brain of a VSR [19,71]. Namely, we represent each candidate solution as a numerical vector $\boldsymbol{g} \in \mathbb{R}^p$ which describes both the body and the brain of the corresponding VSR: we present this representation in detail in Section 3.3.1.

Finally, in Section 3.3.3 we describe the locomotion task for which we optimize the VSRs and the way we measure the degree to which they accomplish it, *i.e.*, their fitness used in the evolutionary process.

#### 3.3.1. VSR representation

As said, we represent a VSR as a numerical vector $\boldsymbol{g} \in \mathbb{R}^p$, called *genotype* in the context of the evolutionary optimization, which describes both its body and its brain. In particular, $\boldsymbol{g} = \left[\boldsymbol{g}_{\text{body}}\ \boldsymbol{g}_{\text{brain}}\right]$ is the concatenation of a vector $\boldsymbol{g}_{\text{body}}$ describing the body and a vector $\boldsymbol{g}_{\text{brain}}$ describing the brain. For the latter part, we simply set $\boldsymbol{g}_{\text{brain}} = \boldsymbol{\theta}$, *i.e.*, we directly encode the parameters of the ANNs constituting the brain: we recall that $\boldsymbol{\theta}$ contains the synaptic weights for the MLP and the values of the ABCD coefficients for the hMLP.

For the body, we use an indirect generative representation, as done in [71] and inspired by Cheney et al. [66]. Let $h \times w$ be the size of the largest body that can be represented: since a VSR body is a polyomino, it can be seen as a Boolean matrix $\boldsymbol{B} \in \{\text{true}, \text{false}\}^{h \times w}$ of $h \times w$ elements in which each element $b_{x,y}$ encodes the presence or absence of a voxel at position $x, y$. We use an ANN to fill $\boldsymbol{B}$ and directly encode the parameters $\boldsymbol{\theta}$ in $\boldsymbol{g}_{\text{body}}$. Namely, we use an MLP with 2 inputs and 1 output and compute each $b_{x,y}$ as:

$$b_{x,y} = \begin{cases} \text{true}, & \text{if } \text{MLP}_{\boldsymbol{\theta}}\left(\left\lfloor \frac{x}{w} \right\rfloor, \left\lfloor \frac{y}{h} \right\rfloor\right) \geq \tau \\ \text{false}, & \text{otherwise}, \end{cases} \tag{3}$$

where $\tau$ is the median value given as output by the MLP when applied to all the elements of the matrix. Finally, to ensure that the body is a proper polyomino, we only consider the largest portion of adjacent 'true' values in $\boldsymbol{B}$.

#### 3.3.2. Evolutionary Algorithm (EA)

To optimize the body and the brain of a VSR, *i.e.*, to search in the space $\mathbb{R}^p$ of vectors describing the VSR, we use a simple EA that has already been used in previous related works [19,71]. In particular, we use a form of ES that, starting from an initial population of candidate solutions, *i.e.*, vectors in $\mathbb{R}^p$, iteratively generates new solutions from the best ones for a predefined number of iterations. Our ES, shown in Algorithm 1, works as follows.

First, we build a population $P$ of $n_{\text{pop}}$ solutions by sampling $U(-1, 1)^p$, *i.e.*, by randomly setting each element of each vector to a value in $[-1, 1]$. Then, at each iteration: (1) we select the best quarter of $P$ solutions, *i.e.*, the $\frac{n_{\text{pop}}}{4}$ numerical vectors with the largest fitness, as parents $P_{\text{parent}}$, (2) we compute their element-wise mean $\boldsymbol{\mu}$, (3) we generate $n_{\text{pop}} - 1$ new solutions, each one by adding some Gaussian noise ($N(0, \sigma)^p$ in Algorithm 1) to $\boldsymbol{\mu}$, and, finally, (3) we set the new population $P$ for the next iteration by adding the previous best solution to the newly generated solutions. Finally, after $n_{\text{evals}}$ fitness evaluations, *i.e.*, after $n_{\text{evals}}$ solutions have been generated and evaluated, we take the solution with the largest fitness in the final population $P$ as the optimized solution $\boldsymbol{g}^\star$.

**Algorithm 1** The simple ES used for evolutionary optimization. The function $\text{BEST}(P, n)$ returns the best $n$ solutions, in terms of fitness, of a population $P$.

```
function EVOLVE()
    P ← ∅                                              ▷ population initialization
    while |P| < n_pop do
        P ← P ∪ {U(−1, 1)^p}
    end while
    c ← |P|
    while c ≤ n_evals do                               ▷ iterations
        P_parents ← BEST(P, ⌊|P|/4⌋)
        μ ← 1/|P_parents| Σ_{g ∈ P_parents} g
        P′ ← BEST(P, 1)                                ▷ add previous best solution
        while |P′| < n_pop do                          ▷ build and add new solutions
            P′ ← P′ ∪ {μ + N(0, σ)^p}
            c ← c + 1
        end while
        P ← P′
    end while
    return BEST(P, 1)
end function
```

#### 3.3.3. Locomotion task

We measure the performance of the VSRs in a locomotion task, a traditional task in evolutionary robotics [72–74]. In the locomotion task, the VSR has to move as fast as possible along the positive $x$-direction on a flat terrain. We use this task mainly for two reasons. Firstly, as said this is a common task in the evolution robotics field. Secondly, this task can offer useful insights, but at a reasonable computational cost.

To measure the degree of accomplishment of the task, we use the average velocity of the VSRs in a simulation lasting $t_{\text{final}}$ time steps, computed considering the center of mass of the VSR and discarding the initial $t_{\text{init}}$ time steps in which the robot might exhibit a transitory behavior.

## 4. Experimental evaluation

We performed an experimental evaluation aimed at answering the following two research questions:

**RQ1** Does Hebbian learning have a positive impact on the body–brain co-evolution? Are the resulting VSRs more effective than those without Hebbian learning?

**RQ2** Why is Hebbian learning beneficial? Does its effectiveness depend on how hMLP-based controller leverage specialization?

To address these questions we conducted a two-fold analysis. First, we performed the body–brain co-evolution (as detailed in Section 3.3) of several VSRs, optimizing them for the task of locomotion, and comparing the performance obtained by MLPs and hMLPs (**RQ1**). Then, we examined the evolved VSRs and their controllers, looking for motivations for the difference in performance between those equipped with

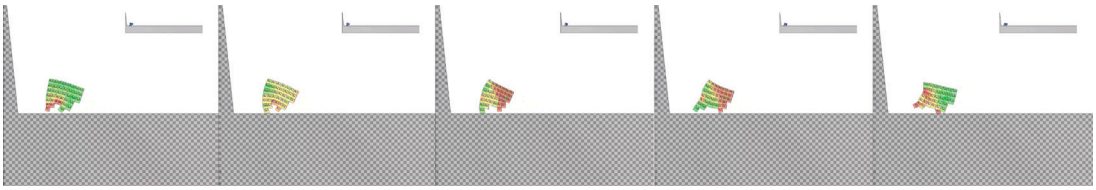


**Fig. 2.** Screenshots of 2D-VSR-Sim at different frames of the simulation, showing an example of VSR made of 37 voxels performing locomotion on a flat terrain. Each voxel's color depends on its action (red indicates contraction, green expansion, and yellow rest).

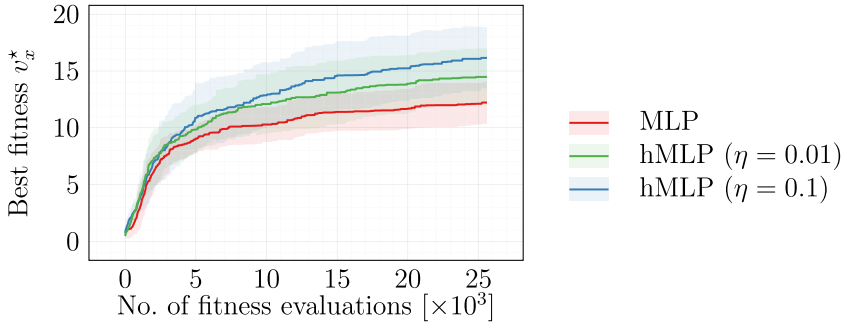



**Fig. 3.** Median ± standard deviation (across 30 runs) of the velocity $v_x^\star$ of the best individual during the evolution. VSRs equipped with hMLPs are in general faster than those equipped with MLPs.

MLPs and those using hMLPs; in particular, we looked for signs of specialization of the controllers embedded in different voxels (**RQ2**).

All the experiments were executed using the 2D-VSR-Sim simulator [59][2] for the simulation of the robots and JGEA [75][3] for the evolutionary optimization, leaving the parameters to the default values unless otherwise specified. For the sake of visualization, we provide a screenshot of 2D-VSR-Sim showing a simulated VSR in Fig. 2.

### 4.1. RQ1 : *effectiveness of Hebbian controllers*

We performed 30 independent runs of the EA described in Algorithm 1 with $n_{\text{pop}} = 40$, $n_{\text{evals}} = 25\,000$, and $\sigma = 0.35$. We set the values of these parameters, and those of the following parameters specific to the task, based on the results obtained in previous experiments with this kind of robots [18,71]. Concerning the task, we set $t_{\text{final}} = 180\,\text{s}$ and $t_{\text{init}} = 5\,\text{s}$. Concerning the representation, we set $w = h = 10$ (*i.e.*, the evolved VSRs are at most $10 \times 10$ voxel large) and we used, for both the MLP and hMLP controllers, a fixed architecture with one single hidden layer with the same number of input neurons: the latter choice resulted in having $(4+4+1)\times(4+4)+(4+4+1)\times 5 = 117$ synapses (given $|\boldsymbol{r}|+4$ inputs, +1 for the bias, *i.e.*, $w_{l,i,0}$, and 5 outputs) in both types of ANN, hence giving $p = |\theta| = 117$ parameters to optimize for the MLP and $4\times 117 = 468$ for the hMLP. We used the same architecture also for the MLP employed in the indirect representation for the body, resulting in other $(2 + 1) \times 2 + (2 + 1) \times 1 = 9$ parameters. Therefore, overall we performed the body–brain optimization in $\mathbb{R}^{126}$ for the MLP case and in $\mathbb{R}^{477}$ for the hMLP case.

Since the parameter $\eta$ (the learning rate, see Eq. (2)) plays a key role in determining the adaptation of the hMLP-based controllers, we performed the experiments with two values: 0.01 and 0.1. We chose these two values as they are typically employed in similar studies, as, e.g., [41] or [18], and we deem them representative of two different learning dynamics (*i.e.*, a slower and a faster learning process).

Fig. 3 summarizes the results of these experiments. It shows the fitness $v_x^\star$ of the best individual in the population during the evolution, one line for each of the three variants: MLP (that we used as a comparison baseline), hMLP with $\eta = 0.01$, and hMLP with $\eta = 0.1$.

The first observation we draw from Fig. 3 is that the adaptation enabled by Hebbian learning is beneficial to the body–brain co-evolution of VSRs. The robots equipped with hMLPs are on average faster, regardless of the value of $\eta$ than those equipped with MLPs. We remark that this result is obtained with the same EA (and the same number of fitness evaluations) and notwithstanding a much larger search space in the case of hMLP ($\mathbb{R}^{477}$ vs. $\mathbb{R}^{126}$). For each pair of variants, we performed a statistical significance test, namely a one-sided Mann Whitney U rank test (after having verified the adequate hypotheses) with the null hypothesis that the distribution of $v_x^\star$ for the first variant is statistically lower than or equal to the distribution of $v_x^\star$ for the second variant. We obtained a $p$-value of 0.0007 for MLP vs. hMLP ($\eta = 0.01$), $5 \times 10^{-7}$ for MLP vs. hMLP ($\eta = 0.1$), and 0.01 for hMLP ($\eta = 0.01$) vs. hMLP ($\eta = 0.1$), meaning that all differences are statistically significant.

We attempted to explain the better effectiveness of robots using Hebbian learning in terms of body–brain coupling. As already observed by Nadizar et al. [76], finding a unique controller for each module of a modular robot while, at the same time, searching for a good body is hard [77]. This is even harder for modular soft robots, because softness makes the body more important in determining the behavior, and hence the brain more sensible to body changes [68]. However, we speculate that the ability of hMLP to adapt the synaptic weights depending on what the ANN processes allows the initially identical hMLPs to become different in different voxels, hence better coping with the variability of the body during the evolution. We discuss this speculation more in detail in Section 4.2.

To gain more insights into the results summarized in Fig. 3, we analyzed more in detail the behaviors of the 30 best VSRs obtained for each of the three variants. In particular, we computed their average speed in different subsequent phases of their "life", which, we recall, lasts 180 s: namely, four periods lasting 30 s. The result of this analysis is summarized in Fig. 4 which shows, in the form of box plots, the distribution of the average speed $v_x$ for the three variants in the four phases.

There are two observations we can draw from Fig. 4. First, VSRs equipped with hMLPs do learn. That is, they adapt, improving, during their life: the average speed $v_x$ in the first 30 s is clearly lower than their speed in later phases; some much lighter improvement is also visible between 30–60 s and 60–90 s. This form of adaptation is not visible in VSRs equipped with MLPs.

[2] The code is available at https://github.com/ericmedvet/2dhmsr.

[3] The code is available at https://github.com/ericmedvet/jgea.

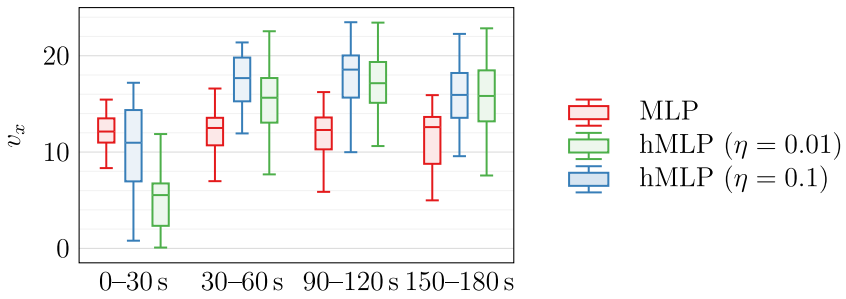



**Fig. 4.** Distributions (across 30 runs) of the average velocity of the best robots during four consecutive intervals in the task (each lasting 30 s). VSRs equipped with hMLPs do learn: they are faster after 30 s than at the beginning. With a lower learning rate $\eta$, they take longer (w.r.t. greater $\eta$) to become equally fast.

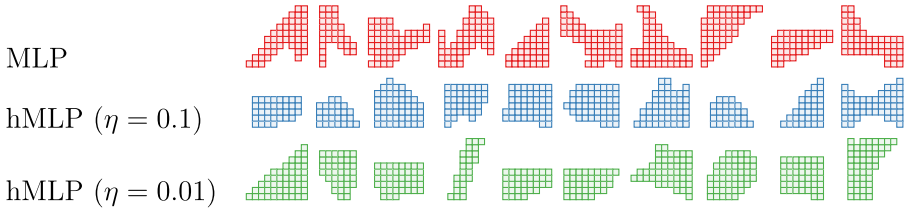



**Fig. 5.** Samples of bodies evolved with the three variants (for seeds 1 to 10). There are no apparent differences between variants in terms of the shape of evolved bodies.

Second, while, as expected, the variant with $\eta = 0.01$ results in slower learning, the final speed achieved by hMLP-equipped robots is approximately the same, regardless of the value of $\eta$. As a consequence, the difference between the two lines of Fig. 3 corresponding to hMLP variants can be explained more in terms of speed of learning than in terms of actual search effectiveness.

Regarding the bodies, in Fig. 5, we show a representative subset of the evolved solutions (namely, those corresponding to seeds 1 to 10). We can visually observe that there are no apparent differences. For example, we can observe that the "stair" shape appears in all three settings (the 5-th for MLP, the 9-th for hMLP ($\eta = 0.1$), and the first hMLP ($\eta = 0.01$)).

We further analyzed the bodies using the same descriptors presented in [19]. In particular, we measured the *elongation* and the *compactness* of the bodies. Intuitively, the former measures the ratio between the width and height of the body, while the latter describes how much the body presents empty spaces. We performed a two-sided Mann–Whitney U rank test that confirmed no statistical differences for the descriptors used[4].

In the next section, we discuss an additional set of experiments and analyses we carried out in order to understand the reasons behind the better performance of hMLP-equipped robots w.r.t. those not exploiting Hebbian learning.

### 4.2. RQ2 : *specialization in Hebbian controllers*

Having observed a neat performance increase with the introduction of Hebbian learning, we conducted additional analyses to investigate the deep causes of such an improvement. In fact, we speculated that, since Hebbian learning confers each voxel the capability to learn based on its experience, this could lead to specialization, which in turn could facilitate the evolution and enhance the overall VSR performance.

To test our hypothesis, we performed two experiments on the evolved VSRs: an online one and an offline one. In the former one, we observed the hMLP embedded in the voxels and analyzed them while working inside the robots; in the latter, we pulled the hMLP out of the voxels and then performed further analyses. Namely, we detached the hMLPs from the simulator and studied their behavior using artificially generated inputs. For simplicity, we performed these experiments on a subset of 10 (out of 30) VSRs evolved in the experiment described in the previous section, due to the significant computational cost needed for the analysis. Namely, we selected those corresponding to seeds 1 to 10 (*i.e.*, those for which the body is shown in Fig. 5).

#### 4.2.1. *Online analysis*

Our first analysis aimed at verifying that: (**H1**) learning is actually occurring in Hebbian controllers, and (**H2**) there is some form of specialization in the voxels guided by the learning process.

By "learning is actually occurring" we mean that Hebbian learning is guiding the weights towards values that are able to lead the VSR to the accomplishment of its task, even if no additional learning is performed; this is in contrast with an alternative hypothesis according to which the VSRs runs successfully only because of the dynamics induced by Hebbian learning. By specialization, we mean that voxels ANNs become *functionally different* from one another after the learning process.

To test these two hypotheses (**H1** and **H2**), for each of 10 + 10 VSRs equipped with hMLP (10 for each value of $\eta$), we proceeded as follows. First, we performed a simulation of 60 s where we let the VSR learn according to its evolved Hebbian rules. Then, we considered three different re-assessment conditions, namely:

1. **With learning**: In this case, for each seed, we cloned the resulting VSR (with its learned weights) and we re-assessed it in another 60 s simulation, where we allowed the VSR to continue learning.
2. **Without learning**: As in the previous setting, but in this case we disabled Hebbian learning in the new 60 s simulation, *i.e.*, we "froze" all the weights to their values reached after the initial learning phase.
3. **Homogeneous controller**: In this case, for each seed we copied the weights of one of the voxels ANNs into all the other ANNs, obtaining a homogeneous controller. We repeated this evaluation for each of the voxels ANNs, hence obtaining as many clones as the number of voxels. Each of these clones was then re-assessed in another 60 s simulation, disabling Hebbian learning.

We show the results of this online analysis in Fig. 6, where we report, for each of the evolved VSRs (one per subplot, divided by $\eta$ values on the rows), the velocities $v_x$ achieved in each of the three

[4] Some selected videos of the VSRs performing the locomotion task are available at https://tinyurl.com/4ye6nkpu.

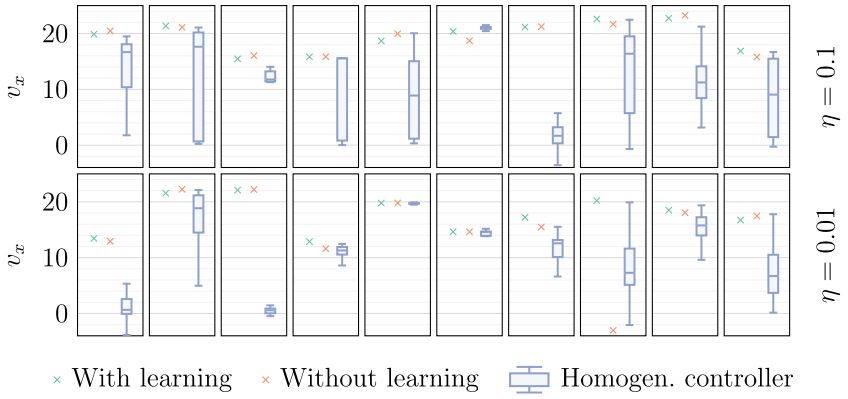



**Fig. 6.** Velocities measured during the re-assessment of the evolved VSRs: one row per learning rate $\eta$, one column per VSR (seeds 1 to 10), one color per re-assessment condition.

aforementioned cases. Note that, while in the first two settings ("With learning" and "Without learning") we have one re-assessment per each original VSR (*i.e.*, for each seed), hence one single value per subplot, in the third setting ("Homogen. controller") we obtain a distribution of velocities for each seed, with a number of values equal to the number of voxels constituting the VSR (hence, a boxplot).

By comparing the first two velocities in each plot (*i.e.*, those achieved with the Hebbian learning enabled and disabled, respectively) we can reach a conclusion regarding the occurrence of actual learning. In fact, in all but one case ($\eta = 0.01$, 8-th robot) the two values are close to each other, which means that, after the initial learning phase, the weights of each ANN have converged to values that are able to properly control the VSR, and Hebbian learning is not necessary anymore (although not detrimental). Hence, we can confirm the occurrence of *actual* learning (**H1**).

Moving on to our second hypothesis (**H2**), which is the one regarding specialization, we can draw some conclusions by looking at the rightmost distribution of each subplot. Among these distributions, we can identify two main scenarios: one with eminent spread and another one with less spread. In the first case (largely spread distributions), the velocities usually extend from very low values, corresponding to almost idle VSRs, to values comparable to those achieved without any controller alteration. These distributions hint that some voxels ANNs have specialized so much that they are not suitable for controlling any other voxel, leading to a significant performance drop if they are employed in a homogeneous controller. However, the tops of the distributions indicate that for some other voxels, specialization has occurred less strongly, making the local ANNs versatile enough to be able to effectively control every voxel in the VSR.

Concerning the latter case, the one with denser distributions, we can still identify cases with high and low specialization: when the distribution is centered at high-velocity values we can conclude that specialization is low, and hence all the ANNs are versatile, whereas when the distribution is centered at low velocities we deduce that specialization is high, and almost every ANN can only control the voxel it was evolved for. We remark that we never enabled learning when re-assessing the VSRs in the homogeneous controller condition: in fact, the rationale was to measure the impact of an ANN being forced to operate in different conditions w.r.t. the one it had (possibly) specialized for. If we had enabled learning, the ANN might have had a chance to re-adapt to the new conditions, possibly making this experiment's results harder to interpret.

Summarizing, we can confirm that in the hMLPs some of the voxels have specialized, although the specialization phenomenon is not uniform, neither across VSRs, nor across the voxels of a single VSR, and even though all voxels are governed by the same Hebbian rules. In fact, it could be the case that the amount of specialization and its uniformity across a VSR depend not only upon the particular Hebbian rules configuration achieved at the end of the evolution, but also on the particular experiences undergone by each voxel (and hence by each hMLP embedded in it). To further investigate this matter, we performed an offline analysis, described in the following section.

#### 4.2.2. Offline analysis

Building upon the findings of the previous section, we speculated that specialization in a voxel could be driven by its "experience", *i.e.*, by the stimuli to which a given hMLP is subjected while it is learning, in conjunction with the set of evolved learning rules. To test this hypothesis we conducted an offline analysis, aimed at: (a) assessing how different the stimuli of different voxels are and at (b) evaluating how diverse the hMLPs become in terms of responses to the same set of stimuli.

As a preparatory phase for our analysis, we gathered the $10+10+10$ VSRs obtained with the experiment of Section 4.1 (again, in particular, those related to seeds 1 to 10), in this case both with MLP and hMLP controllers, and simulated them for 60 s. During these simulations, at each control time step, we collected and saved the input vector of each ANN of the VSR. Moreover, for the VSRs controlled by hMLPs, we also saved a clone of the VSR with its achieved weight configuration at $t_s \in \{0.25, 15, 30, 45\}$. Following this data collection phase, we proceeded with the offline analysis.

To tackle the first point (difference in stimuli), we started by examining the collected inputs, *i.e.*, the various stimuli to which the different ANNs are subjected. We recall that at each time step $k$ the input vector of an ANN, $\boldsymbol{x}^{(k)} \in [0,1]^8$, is defined as the concatenation of the sensor readings ($\boldsymbol{r}^{(k)} \in [0,1]^4$) and the communication values coming from its four direct neighbors:

$$\boldsymbol{x}^{(k)} := \left[\boldsymbol{r}^{(k)}\ i_{\uparrow}^{(k-1)}\ i_{\leftarrow}^{(k-1)}\ i_{\downarrow}^{(k-1)}\ i_{\rightarrow}^{(k-1)}\right]. \tag{4}$$

To evaluate how different the experiences of the voxels of the same VSR are throughout their lifetime, we relied on the inputs distance matrix $\boldsymbol{D}$. For each VSR, we computed $\boldsymbol{D}$ as:

$$\boldsymbol{D} = d_{i,j} = \frac{1}{n_{\text{ts}}} \sum_{k=0}^{n_{\text{ts}}} \left\| \boldsymbol{x}_i^{(k)} - \boldsymbol{x}_j^{(k)} \right\|_2, \tag{5}$$

where $n_{\text{ts}} = 240$ corresponds to the total amount of control time steps performed in a 60 s simulation, and $\boldsymbol{x}_i^{(k)}$ indicates the input vector of the $i$th voxel at the $k$th control time step. In plain words, we introduced a matrix where each element quantifies the average distance between the inputs of each pair of ANNs of a VSR. By observing these matrices for all the considered VSRs, we can draw some conclusions regarding how diverse the inputs of different ANNs are.

We report all the computed matrices in Fig. 7, grouped by controller type (MLP and the hMLP with the two different values for $\eta$) on the rows and VSRs on the columns. In order to ease the visualization and the discovery of patterns in the matrices, we ordered the voxels according to their *connectivity*, *i.e.*, the presence or absence of neighbor

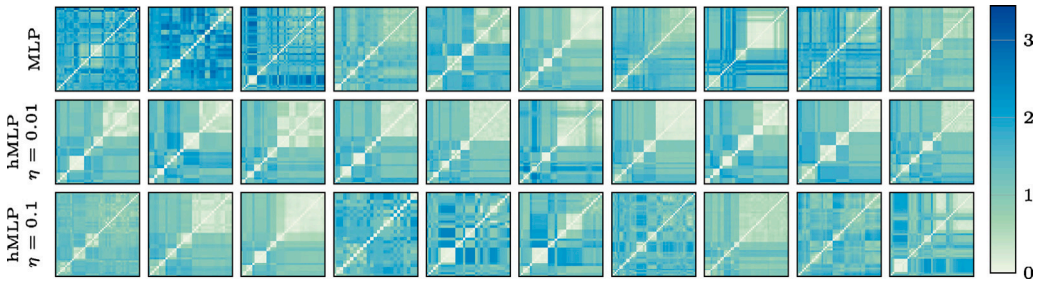



**Fig. 7.** Average pairwise distance between inputs observed by the ANNs of each VSR (grouped by controller type on rows, and VSR on columns (seeds 1 to 10), during a simulation of 60 s. The ANNs embedded in different voxels of the same VSR experience different inputs.

voxels on the four sides (with $2^4-1$ different values). As a consequence, each pair of voxels $i$ and $i+1$ considered when computing $\boldsymbol{D}$ are either of the same types in terms of connectivity (*i.e.*, they have the same free and connected sides), or usually differ by at most one connected side (e.g., the $i$th voxel has all sides connected while the $(i+1)$-th has the top side free).

Given this representation choice, we can clearly highlight a checked pattern in the matrices of Fig. 7, which hints that similar connectivity corresponds to similar experiences in the voxels. This likely descends from the fact that sharing the same connectivity usually corresponds to playing the same *role* in the robot: for instance, "back" voxels will only have the upper side free, whereas "core" voxels will have all sides connected. In fact, square-colored blocks indicate that the average distance in the input space is approximately the same between pairs of voxels belonging to the same two connectivity families, *i.e.*, having the same role. As a further confirmation, we can note that the checks are lighter (*i.e.*, the distance is smaller) around the diagonals, where the distance is computed between voxels of the same type.

If we look at the various matrices in a comparative manner, we can also notice that for some VSRs the distances are larger (*i.e.*, the map is darker in some points) than for others. This does not appear to be correlated with learning, as we can see some darker maps on all three rows. Instead, we believe that this descends from the diverse gaits achieved by the VSRs, which might determine weaker or stronger differences in the inputs seen by the voxels.

Given these results, we can conclude that different voxels do indeed experience different stimuli during their lifetime and that these differences depend mostly on two factors, namely: (1) the role of the voxel, expressed by its connectivity, and the (2) the gait achieved by the VSR.

To proceed with evaluating how diverse the learned ANNs become in terms of response to the same stimuli, we started by identifying a set of representative inputs. To this end, we partitioned the space of collected inputs into $n_c = 25$ clusters[5] with the K-Means clustering algorithm and selected the $n_c$ resulting centroids as representative stimuli for the later analysis.

Having obtained $n_c$ representative inputs, we employed them to compute the specialization matrix $\boldsymbol{S}$ of all the saved VSRs. We computed the specialization matrix $\boldsymbol{S}$ of a VSR as:

$$\boldsymbol{S} = s_{i,j} = \frac{1}{n_c} \sum_{\boldsymbol{x}_c \in C} \left\| \mathrm{ANN}_i(\boldsymbol{x}_c) - \mathrm{ANN}_j(\boldsymbol{x}_c) \right\|_2, \tag{6}$$

where $C$ indicates the set of $n_c$ previously computed centroids, and $\mathrm{ANN}_i$ indicates the ANN pertaining to the $i$th voxel. Simply put, each element of $\boldsymbol{S}$ quantifies the average distance in the output space across some representative inputs for the two considered voxels. Clearly, the larger the average distance, the more functionally different the voxels: *i.e.*, for the same inputs, they produce different outputs. As for the input distance matrices $\boldsymbol{D}$, we ordered the voxels according to their connectivity when computing each $\boldsymbol{S}$, to ease the visual discovery of patterns.

We show the specialization matrices for all the considered VSRs in Fig. 8. We organize the figure in two sub-figures, one per value of $\eta$—we do not compute this matrix for VSRs equipped with MLP-based controllers, as their ANNs are all the same by design. Within each subfigure, every column corresponds to a given VSR with the ANNs sampled at different control time steps (reported on the left of each row).

We start our analysis by focusing on each column of the figure, to assess how the functional difference of the ANNs of a VSR develops throughout its lifetime. We can neatly notice that all columns start with very light matrices, corresponding to no functional differences, which become darker as time passes. This is a clear indicator of specialization: even though all the ANNs are initialized in the same way, they specialize to react in different ways to the same set of stimuli, *i.e.*, they become functionally and behaviorally different.

Comparing the various maps, we can note that specialization does not occur in the same way in all cases. In fact, some maps become darker quickly, whereas some others remain lighter even after 45 s. This confirms our previous deduction about the non-uniformity of specialization across different individuals. However, in general, the matrices of the first subfigure appear to achieve darker colors than those of the second subfigure, meaning that a higher value of $\eta$ may lead to a stronger specialization. This naturally descends from Eq. (2), as the parameter $\eta$ regulates the intensity of the weights alteration performed with Hebbian learning.

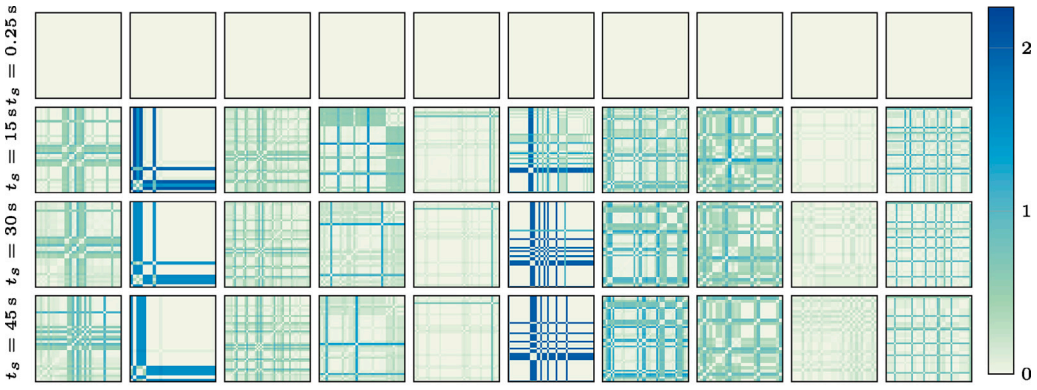
Focusing on the final rows of Fig. 8, we observe the same checked patterns detected in Fig. 7. As before, this is an indicator that specialization is strongly tied to the connectivity of a voxel. Anyway, here the patterns are denser than before, meaning that in some cases there are functional differences even within the same connectivity family.

To summarize, we have confirmed that Hebbian learning induces specialization as a form of behavioral and functional difference in the resulting ANNs. Moreover, we have highlighted a threefold entanglement between (1) the role and connectivity of a voxel, (2) its experience, and (3) its specialization.

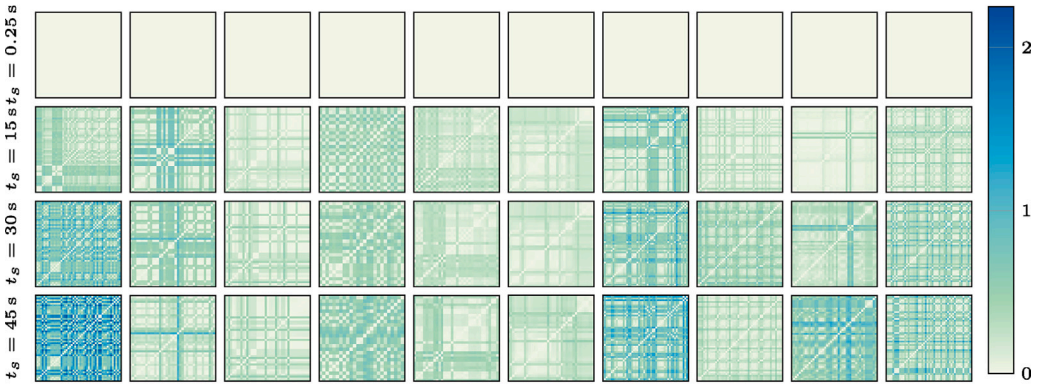
## 5. Conclusions and broader impact

We considered the case of the concurrent optimization of the body and the brain of Voxel-based Soft Robots (VSRs), a kind of modular robot in which the collective behavior of several identical modules allows the robot to perform complex tasks. We optimized both the body, *i.e.*, the way the modules are organized together, and the brain, *i.e.*, the parameters of an ANN embedded in each module, through an EA. For the ANNs constituting the brain, we adopted Hebbian learning, a form of unsupervised learning according to which the synaptic weights of the ANN change during the life of the agent depending on the signals running over the synapses. We found experimentally that robots employing Hebbian learning are more effective than those without this form of plasticity. Moreover, through extensive experiments, we found

[5] We repeated the analysis with various values of $n_c$, observing negligible differences in the final outcomes.




(a) hMLP ($\eta = 0.1$)




(b) hMLP ($\eta = 0.01$)

**Fig. 8.** Specialization matrices $\boldsymbol{S}$. The ANNs embedded in different voxels of the same VSR become functionally and behaviorally different over time.

that: (1) robots with Hebbian learning do learn, *i.e.*, they retain their abilities even if the plasticity is disabled, provided this is done after a long enough learning period; (2) Hebbian learning results in specialization, *i.e.*, modules which are initially identical become different after learning, with their ANNs processing differently the same inputs—in brief, we observed a first form of *totipotency* in artificial organisms and in the ANNs controlling them; (3) specialization is mostly related to the role of the modules, namely, their position, in the overall body of the robot.

The proposed approach, in which we achieve specialization of modules through Hebbian learning, may have in our opinion broad implications in various areas of AI and applications thereof. Beyond modular robotics, which is the scope of the present work, this mechanism could be applied in principle to various autonomous complex systems composed of simple (but initially identical) modules, that require some form of adaptation at runtime. One possible example is represented by networked systems, such as Wireless Sensor Networks and other instances of the Internet of Things. In recent work, Yaman and Iacca [78] have attempted to apply embodied evolution in this kind of system, however adaptation through Hebbian learning-based specialization may be an interesting opportunity for future investigations.

Another potential field of application is that of MAS, particularly MARL, a growing area in AI [79–82], and the even more recent *explainable* MARL field [83,84]. To some extent, the modular robots studied in the present work can be seen as a form of MAS, with communication mediated both implicitly (through actuation and perception of adjacent voxels) and explicitly (through ad hoc communication signals). On the other hand, our model could be applied to other instances of MAS, including systems with more specific means of communication through, e.g., symbolic or sub-symbolic signals [85,86] or parameter sharing [87].

As a final note, it would be interesting to consider in our system also non-connectionist neural models, such as Spiking Neural Networks, for which recent works have established promising meta-learning plasticity-based frameworks [29,40].

## CRediT authorship contribution statement

**Andrea Ferigo:** Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization. **Giovanni Iacca:** Writing – review & editing, Resources, Methodology, Conceptualization. **Eric Medvet:** Writing – review & editing, Resources, Methodology, Conceptualization. **Giorgia Nadizar:** Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## References


[1] G. Haberlandt, Physiologische pflanzenanatomie, Leipzig, W. Engelmann, 1904, 1904, p. 666.

[2] S. Mitalipov, D. Wolf, Totipotency, pluripotency and nuclear reprogramming, Eng. Stem Cells (2009) 185–199.

[3] M.L. Condic, Totipotency: what it is and what it is not, Stem Cells Dev. 23 (8) (2014) 796–812.

[4] U. Kutschera, P.M. Ray, Forever young: stem cell and plant regeneration one century after haberlandt 1921, Protoplasma 259 (1) (2022) 3–18.

[5] D.C. Hughes, S. Ellefsen, K. Baar, Adaptations to endurance and strength training, Cold Spring Harbor Perspect. Med. 8 (6) (2018) a029769.

[6] W. Ratajczak, P. Niedźwiedzka-Rystwej, B. Tokarz-Deptuła, W. Deptuła, Immunological memory cells, Central Eur. J. Immunol. 43 (2) (2018) 194–203.

[7] A.G. Petzoldt, S.J. Sigrist, Synaptogenesis, Curr. Biol. 24 (22) (2014) R1076–R1080.

[8] A.E. Eiben, E. Hart, If it evolves it needs to learn, in: Genetic and Evolutionary Computation Conference Companion, ACM, New York, NY, USA, 2020, pp. 1383–1384.

[9] J. Luo, A.C. Stuurman, J.M. Tomczak, J. Ellers, A.E. Eiben, The effects of learning in morphologically evolving robot systems, Front. Robot. AI 9 (2022) 1–18.

[10] J. Hiller, H. Lipson, Automatic design and manufacture of soft robots, IEEE Trans. Robot. 28 (2) (2012) 457–466.

[11] T.H. Brown, E.W. Kairiss, C.L. Keenan, Hebbian synapses: biophysical mechanisms and algorithms, Annu. Rev. Neurosci. 13 (1) (1990) 475–511.

[12] E. Hart, L.K. Le Goff, Artificial evolution of robot bodies and control: on the interaction between evolution, learning and culture, Philos. Trans. R. Soc. B 377 (1843) (2022) 20210117.

[13] F. Pigozzi, F.J. Camerota Verdù, E. Medvet, How the morphology encoding influences the learning ability in body-brain co-optimization, in: Genetic and Evolutionary Computation Conference, ACM, New York, NY, USA, 2023, pp. 1045–1054.

[14] J.M. Baldwin, A new factor in evolution, Amer. Nat. 30 (354) (1896) 441–451.

[15] N. Le, Organic selection and social heredity: The original baldwin effect revisited, in: Artificial Life Conference Proceedings, MIT Press, 2019, pp. 515–522.

[16] G. Nadizar, E. Medvet, S. Nichele, S. Pontes-Filho, Collective control of modular soft robots via embodied Spiking Neural Cellular Automata, 2022, arXiv:2204.02099.

[17] E. Masquil, G. Hamon, E. Nisioti, C. Moulin-Frier, Intrinsically-motivated goal-conditioned reinforcement learning in multi-agent environments, 2022, arXiv:2211.06082.

[18] A. Ferigo, G. Iacca, E. Medvet, F. Pigozzi, Evolving Hebbian learning rules in voxel-based soft robots, IEEE Trans. Cogn. Dev. Syst. (2022) 1–11, online first.

[19] E. Medvet, A. Bartoli, F. Pigozzi, M. Rochelli, Biodiversity in evolved voxel-based soft robots, in: Genetic and Evolutionary Computation Conference, ACM, New York, NY, USA, 2021, pp. 129–137.

[20] B.-Y. Hu, J.P. Weick, J. Yu, L.-X. Ma, X.-Q. Zhang, J.A. Thomson, S.-C. Zhang, Neural differentiation of human induced pluripotent stem cells follows developmental principles but with variable potency, Proc. Natl. Acad. Sci. 107 (9) (2010) 4335–4340.

[21] K. Dobs, J. Martinez, A.J.E. Kell, N. Kanwisher, Brain-like functional specialization emerges spontaneously in deep neural networks, Sci. Adv. 8 (11) (2022) eabl8913.

[22] G.R. Yang, M.R. Joglekar, H.F. Song, W.T. Newsome, X.-J. Wang, Task representations in neural networks trained to perform many cognitive tasks, Nature Neurosci. 22 (2) (2019) 297–306.

[23] A. Yaman, J.Z. Leibo, G. Iacca, S.W. Lee, The emergence of division of labor through decentralized social sanctioning, 2022, arXiv:2208.05568.

[24] O. Kosak, C. Wanninger, A. Hoffmann, H. Ponsar, W. Reif, Multipotent systems: Combining planning, self-organization, and reconfiguration in modular robot ensembles, Sensors 19 (1) (2018) 17.

[25] J. Auerbach, J.C. Bongard, Evolution of functional specialization in a morphologically homogeneous robot, in: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, 2009, pp. 89–96.

[26] J. Whitman, M. Travers, H. Choset, Learning modular robot control policies, IEEE Trans. Robot. (2023).

[27] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al., Overcoming catastrophic forgetting in neural networks, Proc. Natl. Acad. Sci. 114 (13) (2017) 3521–3526.

[28] G. Nadizar, E. Medvet, K. Walker, S. Risi, A fully-distributed shape-aware neural controller for modular robots, in: Genetic and Evolutionary Computation Conference, ACM, New York, NY, USA, 2023, pp. 184–192.

[29] T. Zhang, X. Cheng, S. Jia, C.T. Li, M. ming Poo, B. Xu, A brain-inspired algorithm that mitigates catastrophic forgetting of artificial and spiking neural networks with low computational cost, Sci. Adv. 9 (34) (2023) eadi2947.

[30] G.E. Hinton, S.J. Nowlan, et al., How learning can guide evolution, in: Adaptive Individuals in Evolving Populations: Models and Algorithms, vol. 26, 1996, pp. 447–454.

[31] W. Li, E. Buchanan Berumen, L. Le Goff, E. Hart, M. Hale, M. De Carlo, R. Woolley, A. Winfield, J. Timmis, A. Eiben, et al., Evaluation of frameworks that combine evolution and learning to design robots in complex morphological spaces, IEEE Trans. Evol. Comput. (2023).

[32] C.A. Shaw, J.C. McEachern, J. McEachern, Toward a Theory of Neuroplasticity, Psychology Press, New York, NY, USA, 2001.

[33] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, A. Peste, Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks, J. Mach. Learn. Res. 22 (1) (2021).

[34] G. Nadizar, E. Medvet, H.H. Ramstad, S. Nichele, F.A. Pellegrino, M. Zullich, Merging pruning and neuroevolution: towards robust and efficient controllers for modular soft robots, Knowl. Eng. Rev. 37 (2022) e3.

[35] E. Najarro, S. Sudhakaran, S. Risi, Towards self-assembling artificial neural networks through neural developmental programs, in: Conference on Artificial Life, ALIFE '23, 2023, pp. 1–10.

[36] S. Risi, K.O. Stanley, Indirectly encoding neural plasticity as a pattern of local rules, in: International Conference on Simulation of Adaptive Behavior, Springer, 2010, pp. 533–543.

[37] T. Miconi, K. Stanley, J. Clune, Differentiable plasticity: training plastic neural networks with backpropagation, in: International Conference on Machine Learning, PMLR, Stockholm, Sweden, 2018, pp. 3559–3568.

[38] E. Najarro, S. Risi, Meta-learning through Hebbian plasticity in random networks, Adv. Neural Inf. Process. Syst. 33 (2020) 20719–20731.

[39] A. Yaman, G. Iacca, D.C. Mocanu, M. Coler, G. Fletcher, M. Pechenizkiy, Evolving plasticity for autonomous learning under changing environmental conditions, Evol. Comput. 29 (3) (2021) 391–414.

[40] J. Jordan, M. Schmidt, W. Senn, M.A. Petrovici, Evolving interpretable plasticity for spiking networks, Elife 10 (2021) e66273.

[41] J.W. Pedersen, S. Risi, Evolving and merging Hebbian learning rules: increasing generalization by decreasing the number of rules, in: Genetic and Evolutionary Computation Conference, 2021, pp. 892–900.

[42] A. Ferigo, E. Cunegatti, G. Iacca, Neuron-centric hebbian learning, 2024, arXiv preprint arXiv:2403.12076.

[43] A. Ferigo, G. Iacca, Self-building neural networks, in: Conference on Genetic and Evolutionary Computation Companion, ACM, New York, NY, USA, 2023, pp. 643–646.

[44] B. Dresp-Langley, From biological synapses to "intelligent" robots, Electronics 11 (5) (2022) 707.

[45] E. Zardini, D. Zappetti, D. Zambrano, G. Iacca, D. Floreano, Seeking quality diversity in evolutionary co-design of morphology and control of soft tensegrity modular robots, in: Genetic and Evolutionary Computation Conference, ACM, New York, NY, USA, 2021, pp. 189–197.

[46] S.L. Thomson, L.K.L. Goff, E. Hart, E. Buchanan, Understanding fitness landscapes in morpho-evolution via local optima networks, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2024.

[47] Y. Jin, Y. Meng, Morphogenetic Robotics: An Emerging New Field in Developmental Robotics, IEEE Trans. Syst. Man Cybern. C (Appl. Rev.) 41 (2) (2011) 145–160.

[48] V. Vujovic, A. Rosendo, L. Brodbeck, F. Iida, Evolutionary developmental robotics: Improving morphology and control of physical robots, Artif. Life 23 (2) (2017) 169–185.

[49] S. Nolfi, D. Floreano, Synthesis of autonomous robots through evolution, Trends Cogn. Sci. 6 (1) (2002) 31–37.

[50] A. Hallawa, G. Iacca, C. Sariman, T. Rahman, M. Cochez, G. Ascheid, Morphological evolution for pipe inspection using Robot Operating System (ROS), Mater. Manuf. Process. 35 (6) (2020) 714–724.

[51] P. Pagliuca, S. Nolfi, The dynamic of body and brain co-evolution, Adapt. Behav. 30 (3) (2022) 245–255.

[52] A. Mertan, N. Cheney, Investigating premature convergence in co-optimization of morphology and control in evolved virtual soft robots, in: European Conference on Genetic Programming, Part of EvoStar, Springer, 2024, pp. 38–55.

[53] W. Huang, I. Mordatch, D. Pathak, One policy to control them all: Shared modular policies for agent- Agnostic control, in: International Conference on Machine Learning, ICML '20, JMLR.org, 2020, pp. 4455–4464.

[54] M.-K. Kvalsund, K. Glette, F. Veenstra, Centralized and decentralized control in modular robots and their effect on morphology, in: Conference on Artificial Life, ALIFE '2023, 2023, p. 49.

[55] A. Mertan, N. Cheney, Modular controllers facilitate the co-optimization of morphology and control in soft robots, in: Genetic and Evolutionary Computation Conference, ACM, New York, NY, USA, 2023, pp. 174–183.

[56] F. Pigozzi, E. Medvet, A. Bartoli, M. Rochelli, Factors impacting diversity and effectiveness of evolved modular robots, ACM Trans. Evol. Learn. 3 (1) (2023) 1–33.

[57] F. Pigozzi, Y. Tang, E. Medvet, D. Ha, Evolving modular soft robots without explicit inter-module communication using local self-attention, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2022, pp. 148–157.

[58] S. Pontes-Filho, K. Walker, E. Najarro, S. Nichele, S. Risi, A single neural cellular automaton for body-brain co-evolution, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2022, pp. 148–151.

[59] E. Medvet, A. Bartoli, A. De Lorenzo, S. Seriani, 2D-VSR-Sim: a simulation tool for the optimization of 2-D voxel-based soft robots, SoftwareX 12 (2020) 100573.
[60] S. Kriegman, A.M. Nasab, D. Shah, H. Steele, G. Branin, M. Levin, J. Bongard, R. Kramer-Bottiglio, Scalable sim-to-real transfer of soft robot designs, in: 2020 3rd IEEE International Conference on Soft Robotics, RoboSoft, IEEE, 2020, pp. 359–366.
[61] X. Sui, H. Cai, D. Bie, Y. Zhang, J. Zhao, Y. Zhu, Automatic generation of locomotion patterns for soft modular reconfigurable robots, Appl. Sci. 10 (1) (2020) 294.
[62] J. Legrand, S. Terryn, E. Roels, B. Vanderborght, Reconfigurable, multi-material, voxel-based soft robots, IEEE Robot. Autom. Lett. (2023).
[63] S. Kriegman, D. Blackiston, M. Levin, J. Bongard, A scalable pipeline for designing reconfigurable organisms, Proc. Natl. Acad. Sci. 117 (4) (2020) 1853–1859.
[64] A. Ferigo, G. Iacca, E. Medvet, Beyond body shape and brain: Evolving the sensory apparatus of voxel-based soft robots, in: European Conference on the Applications of Evolutionary Computation, vol. 12694, Springer, Cham, 2021, pp. 210–226.
[65] A. Ferigo, E. Medvet, G. Iacca, Optimizing the sensory apparatus of voxel-based soft robots through evolution and babbling, SN Comput. Sci. 3 (2) (2022) 1–17.
[66] N. Cheney, R. MacCurdy, J. Clune, H. Lipson, Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding, in: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, 2013, pp. 167–174.
[67] E. Medvet, A. Bartoli, A. De Lorenzo, G. Fidel, Evolution of distributed neural controllers for voxel-based soft robots, in: Genetic and Evolutionary Computation Conference, ACM, New York, NY, USA, 2020, pp. 112–120.
[68] E. Medvet, F. Rusin, Impact of morphology variations on evolved neural controllers for modular robots, in: Italian Workshop on Artificial Life and Evolutionary Computation, Springer, 2022, pp. 266–277.
[69] G. Nadizar, E. Medvet, S. Nichele, S. Pontes-Filho, An experimental comparison of evolved neural network models for controlling simulated modular soft robots, Appl. Soft Comput. 145 (2023) 110610.
[70] D.O. Hebb, The Organization of Behavior: A Neuropsychological Theory, Psychology Press, London, UK, 2005.
[71] A. Ferigo, L. Soros, E. Medvet, G. Iacca, On the entanglement between evolvability and fitness: an experimental study on voxel-based soft robots, in: Conference on Artificial Life, ALIFE '22, 2022, pp. 1–10.
[72] S. Nolfi, D. Floreano, Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines, MIT Press, Cambridge, MA, USA, 2000.
[73] J.C. Bongard, Evolutionary robotics, Commun. ACM 56 (8) (2013) 74–83.
[74] S. Doncieux, N. Bredeche, J.-B. Mouret, A.E. Eiben, Evolutionary robotics: what, why, and where to, Front. Robot. AI 2 (2015) 4.
[75] E. Medvet, G. Nadizar, L. Manzoni, JGEA: a modular java framework for experimenting with evolutionary computation, in: Genetic and Evolutionary Computation Conference Companion, 2022, pp. 2009–2018.
[76] G. Nadizar, E. Medvet, K. Miras, On the schedule for morphological development of evolved modular soft robots, in: European Conference on Genetic Programming, Part of EvoStar, Springer, Cham, 2022, pp. 146–161.
[77] H. Lipson, V. Sunspiral, J. Bongard, N. Cheney, On the difficulty of co-optimizing morphology and control in evolved virtual creatures, in: Artificial Life Conference Proceedings, MIT Press, Cambridge, MA, USA, 2016, pp. 226–233.
[78] A. Yaman, G. Iacca, Distributed embodied evolution over networks, Appl. Soft Comput. 101 (2021) 106993.
[79] L. Busoniu, R. Babuska, B. De Schutter, A Comprehensive Survey of Multiagent Reinforcement Learning, IEEE Trans. Syst. Man Cybern. C (Appl. Rev.) 38 (2) (2008) 156–172.
[80] L. Canese, G.C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Re, S. Spanò, Multi-agent reinforcement learning: A review of challenges and applications, Appl. Sci. 11 (11) (2021) 4948.
[81] K. Zhang, Z. Yang, T. Başar, Multi-agent reinforcement learning: A selective overview of theories and algorithms, in: Handbook of Reinforcement Learning and Control, Springer, 2021, pp. 321–384.
[82] A. Oroojlooy, D. Hajinezhad, A review of cooperative multi-agent deep reinforcement learning, Appl. Intell. 53 (11) (2023) 13677–13722.
[83] S. Kraus, A. Azaria, J. Fiosina, M. Greve, N. Hazon, L. Kolbe, T.-B. Lembcke, J.P. Muller, S. Schleibaum, M. Vollrath, AI for explaining decisions in multi-agent environments, in: AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 13534–13538.
[84] M. Crespi, A. Ferigo, L.L. Custode, G. Iacca, A population-based approach for multi-agent interpretable reinforcement learning, Appl. Soft Comput. (2023) 110758.
[85] J. Foerster, I.A. Assael, N. de Freitas, S. Whiteson, Learning to communicate with deep multi-agent reinforcement learning, in: Advances in Neural Information Processing Systems, vol. 29, 2016, pp. 1–13.
[86] Q.F. Lotito, L.L. Custode, G. Iacca, A signal-centric perspective on the evolution of symbolic communication, in: Genetic and Evolutionary Computation Conference, ACM, New York, NY, USA, 2021, pp. 120–128.
[87] X. Chu, H. Ye, Parameter Sharing Deep Deterministic Policy Gradient for Cooperative Multi-agent Reinforcement Learning, 2017, arXiv:1710.00336.

**Andrea Ferigo** is a Ph.D. student at the Department of Information Engineering and Computer Science of University of Trento, Italy, where he also completed his Bachelors and Masters degree in Computer Science in October 2018 and March 2021, respectively. His research interests include evolutionary robotics, self-growing and adapting models, and explainable artificial intelligence.

**Giovanni Iacca** is an Associate Professor in Computer Engineering at the Department of Information Engineering and Computer Science of the University of Trento, Italy. Previously, he worked as postdoctoral researcher in Germany (RWTH Aachen, 2017-2018), Switzerland (University of Lausanne and EPFL, 2013-2016), and The Netherlands (INCAS3, 2012-2016), as well as in industry in the areas of software engineering and industrial automation. He is currently co-PI of the PATHFINDER-CHALLENGE project "SUSTAIN" (2022-2026). Previously, he was co-PI of the FET-Open project "PHOENIX" (2015-2019). He has received two best paper awards (EvoApps 2017 and UKCI 2012). His research focuses on computational intelligence, distributed systems, and explainable AI applied e.g. to medicine. In these fields, he co-authored more than 180 peer-reviewed publications.

**Eric Medvet** received the degree in Electronic Engineering cum laude in 2004 and the Ph.D. degree in Computer Engineering in 2008, both from the University of Trieste, Italy, where he is currently an associate professor in computer engineering, the director of the Evolutionary Robotics and Artificial Life Lab, and the co-director of the Machine Learning Lab. His research interests include evolutionary robotics, artificial life, evolutionary computation, and applications of machine learning.

**Giorgia Nadizar** earned her Bachelor and Master in Electronics and Computer Engineering at the University of Trieste, Italy, in 2019 and 2021 respectively. She is currently pursuing her Ph.D. degree at the University of Trieste in the Evolutionary Robotics and Artificial Life Lab. Her research interests lie in the realm of Embodied AI and Explainable AI, and the intersection thereof.