REAL-X—Robot Open-Ended Autonomous Learning Architecture: Building Truly End-to-End Sensorimotor Autonomous Learning Systems

Emilio Cartoni, Davide Montella, Jochen Triesch, *Member, IEEE*, and Gianluca Baldassarre

***Abstract*—Open-ended learning is a core research field of developmental robotics and AI aiming to build learning machines and robots that can autonomously acquire knowledge and skills incrementally as infants. The first contribution of this work is to highlight the challenges posed by the previously proposed benchmark "REAL competition" fostering the development of truly open-ended learning robots. The benchmark involves a simulated camera-arm robot that: 1) in a first "intrinsic phase" acquires sensorimotor competence by autonomously interacting with objects and 2) in a second "extrinsic phase" is tested with tasks, unknown in the intrinsic phase, to measure the quality of previously acquired knowledge. The benchmark requires the solution of multiple challenges usually tackled in isolation, in particular exploration, sparse-rewards, object learning, generalization, task/goal self-generation, and autonomous skill learning. As a second contribution, the work presents a "REAL-X" architecture. Different systems implementing the architecture can solve different versions of the benchmark progressively releasing initial simplifications. The REAL-X systems are based on a planning approach that dynamically increases abstraction and on intrinsic motivations to foster exploration. Some systems achieves a good performance level in very demanding conditions. Overall, the REAL benchmark is shown to represent a valuable tool for studying open-ended learning in its hardest form.**

***Index Terms*—Autonomous robot, benchmark, competition, intrinsic motivation, open-ended learning, planning, simulation.**

I. Introduction

THE SATISFACTION of biological and social needs represents a major drive for animal learning. Beyond these drives, learning in humans and other animals with more sophisticated cognition is also guided by drives such as *curiosity* and *intrinsic motivations*. These motivations allow them to acquire knowledge and skills that can be later used to accomplish biological and social needs [1], [2], [3]. In the last two decades, several algorithms have been proposed to reproduce such processes in artificial intelligent machines and robots. This endows machines and robots with the capabilities for *open-ended learning*, that is, the capacity to autonomously acquire knowledge and skills without relying on prewired reward functions, tasks, or goals [4], [5]. The autonomous acquisition of knowledge and skills in an open-ended fashion has been studied under different headings. The field of *developmental robotics* [6], [7] has developed algorithms for autonomous learning based on *intrinsic motivations* (e.g., [8], [9], [10], [11], [12], [13], [14], [15], and [16]). More recently, also machine learning and robotics have started to propose systems to face the challenges of open-ended learning (see Section V), in particular based on reinforcement learning algorithms [17], [18].

An important trend[1] within both fields has been the use of intrinsic motivations not to directly learn skills but rather to guide the self-generation or discovery of *goals*, namely internal representations of the world that might drive the agent's actions to realize these world states [19], [20], [21], [22]. The idea is that the autonomous setting of goals can support open-ended leaning as it allows the autonomous generation of tasks which in turn can drive the acquisition of the skills directed to pursue the goals. An increasing number of works thus focuses on the development of agents able to autonomously form new goals and learn the associated skills for realizing these goals [22], [23], [24], [25], [26], [27], [28]. New goals may be defined based on the saliency of world states [8], the change of states [26], [29], eigenoptions [30], density models [31], entropy [32], or variational inference [33]. While these are important developments, at present we still do not have systems able to undergo a truly open-ended learning process.

One way to promote the development of such systems is the proposal of benchmarks and competitions that facilitate the comparison of alternative approaches and systems. Some existing competitions face issues relevant for robotic open-ended learning, but all seems to lack some key elements. For example, the *AutoML for Lifelong Machine Learning*[2] competition focuses on the acquisition of an increasing amount

Manuscript received 15 January 2022; revised 30 September 2022 and 3 December 2022; accepted 25 February 2023. Date of publication 24 May 2023; date of current version 11 December 2023. This work was supported by the European Union Horizon 2020 Research and Innovation Programme through the Project Goal-Robots—Goal-based Open-Ended Autonomous Learning Robots (http://www.goal-robots.eu/) under Grant 713010. The work of Jochen Triesch was supported by the Johanna Quandt Foundation. *(Corresponding author: Gianluca Baldassarre.)*

Emilio Cartoni, Davide Montella, and Gianluca Baldassarre are with the Institute of Cognitive Sciences and Technologies, National Research Council, 00185 Rome, Italy (e-mail: gianluca.baldassarre@istc.cnr.it; davidelmontella@gmail.com; emilio.cartoni@istc.cnr.it).

Jochen Triesch is with the Frankfurt Institute of Advanced Studies, 60438 Frankfurt, Germany (e-mail: triesch@fias.uni-frankfurt.de).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCDS.2023.3270081.

Digital Object Identifier 10.1109/TCDS.2023.3270081

[1]Whole projects have been dedicated to this, see for example the EU project *GOAL-Robots—Goal-based Open-ended Autonomous Learning*: www.goal-robots.eu.

[2]http://automl.chalearn.org/life-long-learning

of input-output data furnished externally, but it does not involve embodied systems interacting with a physical world to actively generate new experiences. *Animal-AI Olympics*[3] is focused on simulated animal-like robots, but these are tested with tasks defined through specific reward functions. The *ICDL MODELbot Challenge*[4] is focused on autonomous developmental processes, but it is not conceived as a benchmark allowing quantitative comparisons among competing approaches.

For these reasons, we have recently proposed a competition, now in its third edition (*REAL 2021 – Robot open-Ended Autonomous Learning*),[5] proposing a benchmark encompassing the major challenges of robotic open-ended learning [34], [35]). The core structure of the competition is based on two phases [34], [36]: 1) an intrinsic phase of learning and 2) an extrinsic phase of testing. In the first *intrinsic phase*, involving a very long period of learning during which no guidance is available in the form of rewards, tasks, goals, etc., the robot should autonomously interact with the environment to acquire as much knowledge and skills as possible to best solve any future tasks in the succeeding extrinsic phase. During the *extrinsic phase*, the knowledge that the robot acquired during the intrinsic phase is evaluated by asking it to solve a certain number of tasks (goals to be achieved) that are unknown during the intrinsic phase. Since during the intrinsic phase the robot is not given any guidance, the only element that it can exploit to learn knowledge useful to solve the tasks of the second phase is that the environment and object properties are the same in the two phases. The competition features two difficulty "Levels" in which the participants can take part: 1) an "Easy Level," allowing the use of some simplifications such as the availability of the position objects on the working space and the possibility of using parameterized motor primitives to control the robot arm and 2) a "Difficult Level," where the robot has to autonomously acquire end-to-end solutions directly mapping the image pixels and arm joint angles to the desired arm motor commands. The competition structure and the general setting is reviewed in more detail in Section II-A.

The benchmark poses extremely hard challenges, as revealed by the previous editions of the competition [34]. The first challenge is that the robot perceives only pixels, joint angles, and touch-sensor readings and so needs to autonomously understand from scratch the very concept of object and which specific objects populate the environment. Second, the robot does not have any information on how to control its several degrees of freedom to produce "relevant" effects on the environment, for example, to touch the objects, let alone to control them in any meaningful way. Third, and even worse, the robot has to face a formidable bootstrapping chicken-egg problem as to make sense of objects it should be able to physically interact with them; but at the same time to be able to interact with those objects it should first know them. This circular interdependence of the learning processes of sensory and motor abilities makes it very difficult to acquire them at the same time.

The REAL benchmark thus uncovers the challenges posed by open-ended-learning in the purest and hardest form. This is important for both the study of development in children, and for the design of autonomous robots.

Regarding the study of development, the challenges posed by the REAL benchmark reflect in a "pure form" (i.e., neglecting innate behaviors and experience acquired in the womb) the "great blooming, buzzing confusion" [37] that newborn babies have to face at birth when they are suddenly immersed in a flood of unstructured information from the senses, and have to control the many degrees of freedom of their bodies. This problem has been studied several times within the Developmental Robotics community, but very often it is simplified by furnishing the robots with some hardwired knowledge. Instead, the Difficult Level of the REAL challenge does not give the robot any prewired knowledge and any guidance for learning. In this respect, one might argue that the challenge is even too hard as the development of babies and children can rely on important forms of social scaffolding such as imitation, joint attention, social rewards, and teaching. In this respect, the REAL benchmark purposefully leaves out such forms of scaffolding for the sake of focusing on how much autonomous individual learning mechanisms can accomplish without using any form of external information and guidance.

In addition, the benchmark is also important to develop autonomous robots able to undergo open-ended learning to face real-life nonengineered human scenarios as those that might be encountered in homes or workplaces. These scenarios are particularly demanding as they pose challenges that cannot be foreseen at design time. For example, a robot-assistant able to help to tidy up a kitchen will need sufficient versatility to adapt to different kitchen layouts, kitchen objects etc. Thus, instead of coming out of the factory with predefined skills it should leverage intrinsic motivations and self-generated goals/tasks to acquire in the working environment a repertoire of skills that will be useful to accomplish human assigned tasks [27]. The application value of these types of algorithms cannot be overestimated. Indeed, a robot system able to undergo truly open-ended learning could in principle be employed in any different environments, with different robotic platforms, and to pursue any goal relevant for the users.

The first contribution of this article is to explain the actual nature and difficulty of the challenges posed by the REAL open-ended learning benchmark from a robotics perspective. In particular, the difficulties posed by the challenges are made evident by gradually introducing increasingly complex versions of the environment encompassing them, and by the different algorithms proposed to solve them. Indeed, as mentioned, open-ended learning poses multiple challenges, such as exploration to independently discover tasks/objectives and learn the policies to achieve them, interesting events that occur infrequently and thus cause sparse rewards, the need to learn from scratch how to represent and interact with objects, and the need to generalize all acquired knowledge to varying conditions. Moreover, these challenges are

[3]http://animalaiolympics.com
[4]https://icdl-epirob2019.org/modelbot-challenge
[5]https://www.aicrowd.com/challenges/real-robots-2020

made more difficult by the fact that they must be addressed simultaneously, as mentioned above. The REAL benchmark enables a systematic study of these challenges and their interactions. To this end, here we used the following "progressive" strategy. We first faced a simplified version of the REAL challenge with a basic REAL system, and then we progressively removed the simplifications and faced them through improved systems obtained by implementing the architecture components with enhanced algorithms. The simplifications we considered included: 1) *sensors:* the availability to the robot of information on the position of objects rather than on raw pixel images; 2) *actions:* the availability of a parameterized macro-action rather than joint angles; and 3) *environment:* the use of only one object rather than two or more.

The literature has proposed various systems having some features that might be relevant to face the REAL benchmark. These systems are briefly introduced here and then considered in more detail in Section V. Some systems [38], [39], [40], [41] focus on exploration using a reward based on prediction errors. However, all these systems use reward functions to guide the agent to execute a task and they do not learn a goal-conditioned policy that can be readily used to reach goal states: to face the REAL benchmark, they would need an added component that translates the goal images into suitable reward functions. In a subsequent work [42], this limitation is lifted by concurrently training an "achiever" component that embodies a goal-conditional policy using images. The systems presented in [43], [44], and [45] also focused on exploration, in this case using a variational autoencoder to generate target states and learn the skills to accomplish them. These systems can potentially be applied to the REAL benchmark but this is prevented by the limitations discussed below. The system proposed in [46] is also relevant as it trains a robotic agent in a pick and place scenario ("OpenAI FetchPickAndPlace") without using an external reward, but using an intrinsic reward based on the mutual information between the agent state and the environment state. However, in the used scenario the robot does not use a camera but has access to the position of objects.

Another work [23] progressively trains an agent by using a network to predict states with "intermediate difficulty" to be reached, thus focusing on "competence" instead of "novelty" [47]. The system is not compatible with the REAL benchmark, as it requires the environment to be reset periodically during the training. Other relevant works [48], [49], [50], [51], [52], [53] are based on image-based planning and use a controller to execute plans formulated on the basis of latent state representations.

Among these systems, [48] and [52] do not seem directly applicable to the REAL scenario because they lack the mechanisms needed to support autonomous learning during the intrinsic phase as they rely on an external reward objective. Instead, the works presented in [49], [50], and [51], and also in [42], [43], [44], [45], and [53] mentioned above, seem to be applicable to the REAL challenge. This however would need to adapt their code (available only for [42], [43], [44], [49], [50], and [53]) to the benchmark scenario, a task that might be carried out in future work if it will be possible to solve the implementation issues presented in Section V.

The second contribution of this work is to propose a software *architecture*, called *REAL-X*, that can be used to implement different open-ended learning robot control *systems*. The different systems, named by differently specifying the "X" in "REAL-X," encompass different sets of solutions that can be used to face the different versions of the REAL benchmark. The REAL-X systems used to solve the benchmark highlight the nature and difficulty of the different challenges it poses. Moreover, they are the first robotic controllers to achieve a score well beyond chance level in different versions of the REAL benchmark, in particular by managing to autonomously learn from scratch to move objects closer to different goal positions on the table.

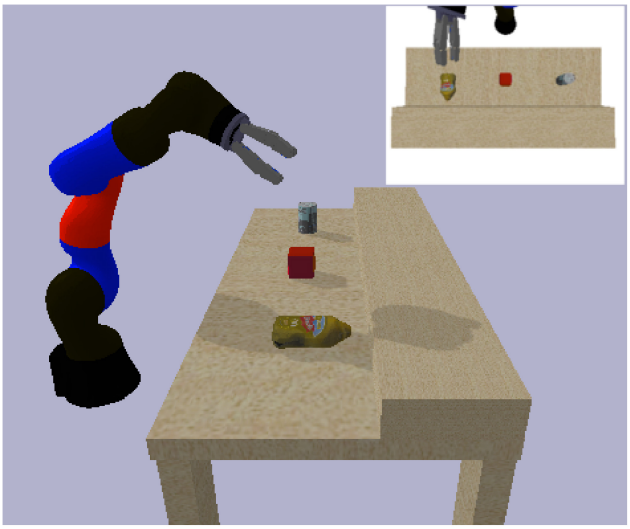
The REAL-X architecture has a modular design to facilitate incremental development and is in particular formed by three components: 1) *abstractor*: this component performs the abstraction of sensory inputs to learn relevant environment variables, for example related to the position and identity of objects; 2) *explorer*: this component generates the motor experience supporting the learning of goals and actions during the intrinsic phase; and 3) *planner*: this component formulates and executes action plans to accomplish the extrinsic goals. The tests of the different versions of REAL-X show that they exhibit coherent behavior and achieve a performance well above chance level with the simplifications, and also when these simplifications are progressively removed. In the following we present a thorough investigation of the challenges posed by REAL benchmark and possible solutions relying on the REAL-X architecture; in future work we plan to use the REAL benchmark and the architecture to further develop these solutions and systematically compare and integrate them with the systems and mechanisms reviewed above and in Section V.

The remainder of this article is organized as follows. Section II first presents more in detail the open-ended learning REAL benchmark and its objective function, and then presents the REAL-X architecture and its implementation in different systems created to solve increasingly complex versions of the scenario. Section III compares the performance of the different REAL-X systems and analyzes some aspects of their internal functioning. Section V reviews in more depth relevant previous systems on open-ended learning. Finally, Section VI draws the conclusions.

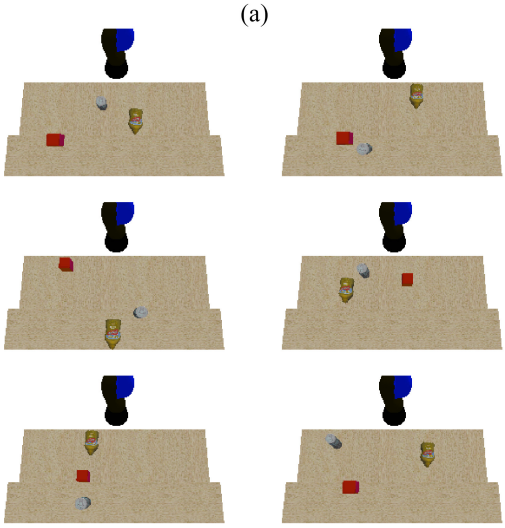
II. Methods

A. Scenario of the Open-Ended Learning Benchmark

The competition scenario is inspired by an assistant-robot scenario where the robot should help to tidy up a kitchen *(kitchen scenario)*. Specifically, the systems participating in the competition have to control a camera-arm-gripper robot manipulating some objects on a table simulated with the PyBullet physics engine and using the OpenAI Gym environment format [see Fig. 1(a)]. The robot is a 7-DoF Kuka arm coupled with a 2-DoF gripper. It stands in front of a table, which has a shelf and 1–3 objects: 1) a cube; 2) a tomato can; and 3) a mustard bottle. The perceptual space comprises



(a)



(b)

Fig. 1. (a) Environment encompassing the robot, the table with the shelf, and the three objects. The inset shows the environment as seen from the robot top-view camera. (b) Three examples of the 50 tasks used in the extrinsic phase: each row shows the initial object configuration (left) and the final one, that is, the *extrinsic goals* (right), for the different tasks.

the images from a fixed top-view camera, the proprioception of joint angles, and the gripper touch sensors; in a simplified version of the benchmark the robot directly perceives the positions of the objects. In the REAL-X systems discussed below, the touch sensors will not be used. The action space is the arm-gripper joint space; however, in a simplified version of the benchmark the robot can be controlled based on a parameterized macro-action. The environment also allows control of the arm in the Cartesian space combined with control of the gripper in the joint space. In the REAL-X systems discussed below we used either the macro-action or the whole-body joint control. The macro-action, as explained below, is limited to only "pushing actions." Instead, the other control modalities, in particular the whole-body joint control, allow the performance of any action, including grasping.

The kitchen scenario is inspired by a possible future real world use case for open-ended learning robots. In this regard, one of the main objectives of the competition is to translate this kind of scenario into a robotic benchmark for objectively measuring the capacity of robots for autonomous open-ended learning. To this purpose, a main feature of the benchmark is its organization into two phases [34], [36]. In the first *intrinsic phase*, the robot *autonomously* interacts with the environment for a long time during which it should acquire as much knowledge and skills as possible to best solve the tasks in a second "extrinsic phase." Here, the intrinsic phase lasts 15 000 000 simulation steps each lasting 5 ms, amounting to 20.83 h of simulated time. During the *extrinsic phase*, the acquired knowledge and skills are evaluated with tasks (here goals, intended as desired world states) unknown by the robot during the intrinsic phase. The extrinsic goals are drawn from the following classes of possible object configurations: 1) *2-D goals:* goals defined in terms of the configuration of 1–3 objects on the table plane, never close to each other, and with a fixed orientation; 2) *2.5-D goals:* goals defined in terms of the configuration of 1–3 objects set on the table plane and on the shelf, never close to each other, and with a fixed orientation; and 3) *3-D goals:* goals defined in terms of 1–3 objects set on the table plane and on the shelf, with any orientation and no minimum distance. Each goal involves a different starting configuration, which adheres to the same criteria as for the goal itself. Fig. 1(b) shows some examples of extrinsic goals. Crucially, in the extrinsic test the robot can still learn, but this is of little help given the limited time available to solve each task, *so the performance in the extrinsic phase can be considered an objective measure of the system's capacity to autonomously acquire knowledge during the intrinsic phase.* Importantly, during the intrinsic phase the robot is not given any knowledge: no tasks, reward functions, pretrained networks for object recognition, world models, abilities, or motor skills. The robot learning processes used in the intrinsic phase should hence be guided by algorithms supporting autonomous learning, for example intrinsic motivations and mechanisms for the self-generation of tasks or goals.

B. Objective Function of the Open-Ended Learning Benchmark

The overall objective of the robot participating in the competition is to find, *during the intrinsic phase*, the parameter vector θ^* that maximizes the expected reward collected *during the extrinsic phase*

$$\theta^* = \arg\max_{\theta} E_{g\sim\tau(g)}\left(E_{\pi(a|s,g,\theta)}R(g)\right) \tag{1}$$

where $R(g)$ are the total rewards obtained for goal g used in the extrinsic phase to evaluate the system, $g \sim \tau(g)$ is the distribution of possible tasks (goals) that can be posed in the environment, and $\pi(a|s, g, \theta)$ is the control policy, dependent on parameters θ, that the robot uses to select actions a in response to state s and the currently pursued goal g. The crucial feature of the benchmark is that the parameters θ must be learned during the intrinsic phase but are tested with goals g

during the extrinsic phase, but these goals are unknown during the intrinsic phase. The latter condition implies that the knowledge that the agent can pass from the intrinsic to the extrinsic phase must rely on the physics of the environment, the robot's body, and the objects, that remain the same in the two phases. The optimal policy $\pi(a|s, g, \theta^*)$ also depends on the time that the robot has to solve each goal g (see [54]), but for simplicity this issue is not considered here.

C. Metrics

During the extrinsic phase the system is asked to solve 50 tasks [Fig. 1(b)]. For each task: 1) the robot is shown a certain configuration of the 3 (or 2 or 1) objects in the environment ("overall goal") in which they are placed anywhere on the table plane or on the shelf: the objects can have any initial/final orientation and may touch/overlap; 2) the objects are then set in a different position and orientation in the environment; and 3) the robot is given 10 000 simulation steps (50 s) to bring the object(s) to the overall goal configuration.

The extrinsic-phase performance for an overall goal g is scored with the metric M_g

$$M_g = \sum_{o=1}^{n} \left[e^{-c||\mathbf{p}_o^* - \mathbf{p}_o||} \right] \tag{2}$$

where n is the number of objects (1, 2, or 3), $\mathbf{p}_o^*$ is the (x, y, z) position vector of the center of mass of object o in the overall goal, $\mathbf{p}_o$ is the position of the object at the end of the task after the robot attempts to bring it to the goal position, c is a constant ensuring that this part of the score will be 0.25 if the distance to the specific object goal position is 10 cm. Note that, for each object, the performance metric M_g ranges in $(0, 1]$, is equal to 1.0 if the object is exactly at the goal position, and decays exponentially with increasing distance from it. Placing all 3 objects exactly in the overall goal configuration yields a maximum score of 3.0. The total *Score M* is the average of the scores across the G ($G = 50$) goals

$$M = \frac{1}{G} \sum_{g=1}^{G} M_g. \tag{3}$$

For simplicity the score does not consider object orientation, but keeping track of orientation allows a finer object control and hence facilitates a higher final score, so it is implicitly rewarded.

D. Simplifications

We have designed different REAL-X systems to face different versions of the described benchmark. These versions feature different simplifications that were progressively released to move toward the full hardest challenge. In particular, the simplifications were implemented along three dimensions: perception, motor action, and number of objects. A fourth dimension involves the absence of environment reset at the end of "trials/rollouts," which is commonly used in reinforcement learning tasks but here is avoided. These simplifications are now considered in more detail.

1) Perception: In the first simplification, the robot was given the x, y, z positions of the objects. When this simplification was removed, the robot instead received a raw camera image of 320×240 RGB pixels. This is a challenging condition as it introduces a difficult "chicken and egg" problem: as to acquire information on objects the robot needs to act on them, but to learn to act on the objects the robot needs to know where the objects are in space.

2) Motor Control: In this simplification the robot used a parameterized macro-action to act on the objects. In particular, the macro-action first moved the (closed) end effector near the working plane, then along a segment trajectory parallel to the plane, and then back to a home position (arm straight up out of the camera sight). The use of the macro-action greatly facilitated hitting/pushing the objects during exploration. The parameters of the macro-action were the two (x, y) and final (x', y') extremes of the movement segment indicating positions of the end actuator on the plane. When this macro-action is usable, the robot needs "only" to learn the four parameters of the macro-action based on the objects' state. When this simplification is removed, the robot has to directly set the desired joint angles at each step. This poses a great challenge as in the initial phase of (random) exploration of the joint space the robot rarely touches the objects. While using a macro-action restricts the robot to push movements only, this is still a significant challenge as testified by similar push scenarios in many other benchmarking simulations, such as in OpenAI RoboGym, RLBench, and Meta-World [55], [56], [57]. However, some of the REAL-X systems we will present below can also work without this simplification and hence are not restricted to only push movements.

3) Number of Objects: The number of objects in the environment is another critical element determining the level of difficulty of open-ended learning. In the simplest case we consider only one object. In more challenging conditions we consider up to three. When the position and identity of objects is furnished to the robot, this does not represent a problem as the robot can focus on one object per time and so the situation becomes similar to the case involving only one object. However, in the most realistic condition where the robot perceives the environment through RGB images having more than one object represents a notable challenge as the robot needs to process the image pixels to make sense of the existence of different objects. Moreover, the presence of multiple objects can also make the planning and control more difficult as they can interact and interfere with each other.

4) Environment Resets: During the intrinsic phase, when the objects are moved by the robot they are left where they are, without being reset to their initial position periodically. This element, that at first sight might seem a secondary technical detail, turned out to be one of the most important aspects of the challenge (note how most reinforcement learning tasks make the reset assumption). Indeed, to develop the different components of REAL-X we often used the simplified condition where we reset the objects, although the experiments reported here do not report the results of this condition as it violates a fundamental element of autonomous open-ended learning (e.g., resetting a real environment would require another agent

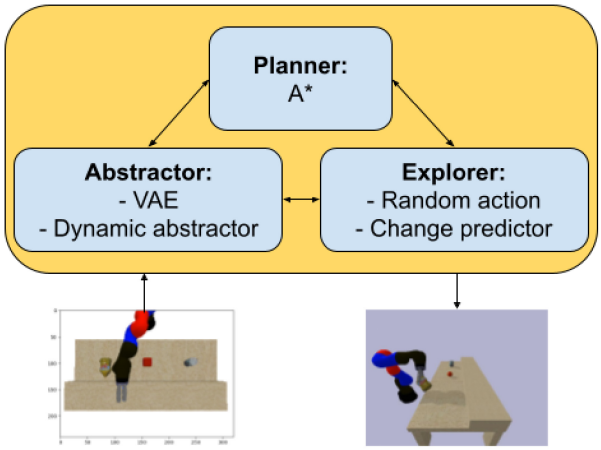


Fig. 2. General architecture of REAL-X, based on three components: Abstractor, Explorer, and Planner. The boxes indicate the specific solutions used here to implement the components.

to do so to support the robot learning). The challenge is that if objects are not reset to the same position/orientation after each action execution, then after each contact of the robot with the object the succeeding initial condition will change. This implies that initially the robot tends to face a sequence of new situations making it difficult to accumulate knowledge.

E. REAL-X Architecture

In this section we describe the REAL-X architecture, while in the following section we present the specific systems that were implemented through the architecture to face the different challenges emerging when progressively removing the simplifications of the REAL benchmark. The REAL-X architecture is formed by three main components (Fig. 2): 1) an *Explorer* module; 2) an *Abstractor* module; and 3) a *Planner* module. The processes performed by the architecture are indicated in Algorithm 1. We now consider the overall functions implemented by the architecture components and their features shared by all systems instantiating the architecture and explained in Section II-F.

1) Explorer: This component guides motor exploration of the environment during the intrinsic phase in order to maximize the acquisition of motor skills. Exploration is based on actions each lasting 1000 steps. Each action starts and ends in a "home" position involving the arm and gripper straight upward. In its basic version, the Explorer produces actions by generating random sets of the parameters of an action, either referring to the macro-action or to joint trajectories (explained below). On each action the components store the acquired knowledge in the form of an *action triplet* (s, a, s') containing: 1) the *precondition*, that is, the initial state s of the environment encoded in terms of image or positions of objects; 2) the *action parameters* a of the performed actions; and 3) the action *outcome*, that is, the state s' that follows the action (image/position of objects).

2) Abstractor: This component is in charge of implementing an abstraction of the perceptual states to support exploration, planning and action performance. The Abstractor is implemented in different ways in the different systems.

Algorithm 1: Benchmark and Architecture Functioning

```
Input: a set G of extrinsic goals, environment;
Output: global_performance
obs_pre = env.get_observation()
for i ← 1 to I        // intrinsic phase steps
do
    action ← explorer.select_next_action(obs_pre)
    obs_post ← env.step(action)
    transitions.add(obs_pre, action, obs_post)
    obs_pre ← obs_post
abstract_transitions = VAE(transitions)
da = dynami_abstractor(abstract_transitions)
planner = planner_setting(da, abstract_transitions)
foreach s_g ∈ G        // extrinsic phase goals
do
    goal_to_pursue ← s_g
    obs = env.get_observation()
    for i ← 1 to J      // extrinsic goal steps
    do
        action ← planner.plan(obs, goal_to_pursue)
        obs ← env.step(action)
    s_final ← obs
    performance_array.append(evaluate(s_final,s_g))
return(global_performance(performance_vector))
```

3) Planner: Planning processes implemented by this component can be used by some systems in the extrinsic phase to pursue each single extrinsic goal. Given an extrinsic goal, the component should be able to search sequences of action triplets that lead from the current state of the environment to a desired goal state. When a plan is found, the first action of the plan is performed and then replanning is repeated to choose the next action and deal with noisy outcomes caused by the previous action.

F. Increasingly Sophisticated REAL-X Systems

We now consider the different systems we used to instantiate the REAL-X architecture. These systems were implemented to face increasingly difficult versions of the REAL benchmark. See Table I for a summary of the different systems and the different conditions of the benchmark. Note that the character "_" was used to indicate the different conditions (*REAL_X*) of the real *REAL benchmark*, whereas the character "-" was used to indicate the different *systems* (*REAL-X*) implementing the *REAL-X architecture*.

1) REAL-R—Random: To have a baseline performance, we compared all systems tested in all conditions with *REAL-R*, a system facing the extrinsic goals by producing random actions. These actions were produced by generating action parameters with a uniform probability distribution.

2) REAL-D—Dynamic Abstractor, REAL-T—Threshold: The REAL-D system was used to face the simplest version of the benchmark, REAL_OM, where the robot was given the positions of objects and it could use the macro-action.

TABLE I
PERFORMANCE OF THE DIFFERENT ARCHITECTURE IMPLEMENTATIONS, NAMED REAL-X (WITH A DIFFERENT "X" FOR DIFFERENT SYSTEMS), IN DIFFERENT CONDITIONS, NAMED AS REAL_X (WITH DIFFERENT X FOR DIFFERENT CONDITIONS). THE UPPER PART OF THE TABLE REPORTS THE PERFORMANCE OF THE DIFFERENT SYSTEMS IN THE DIFFERENT CONDITIONS (MEAN SCORE AND STANDARD ERROR OF THE MEAN OVER 30 REPETITIONS OF THE TEST). THE LOWER PART OF THE TABLE INDICATES THE MEANING OF THE ACRONYMS USED IN PLACE OF THE X IN THE NAMES OF THE SYSTEMS OR THE CONDITIONS. *: BEST RESULT AFTER RUNNING WITH DIFFERENT THRESHOLDS. NA: THE RESULT IS NOT AVAILABLE AS THE PLANNER WAS NOT ABLE TO RETURN A PLAN BEFORE RUNNING OUT OF MEMORY

REAL-X systems	REAL_X benchmark conditions		
	REAL_OM	**REAL_IM**	**REAL_IJ**
REAL-R	$0.021 \pm .002$	$0.021 \pm .002$	$0.038 \pm .002$
REAL-T	$0.212 \pm .006$*		
REAL-D	$0.216 \pm .004$	NA	
REAL-LD		$0.222 \pm .005$	$0.139 \pm .007$
REAL-ILD		$0.234 \pm .007$	$0.149 \pm .006$

Letters in place of the 'X' in the system acronym 'REAL-X':
R - Control system performing random actions in the extrinsic phase
T - Manually optimised fixed threshold (no Dynamic Abstractor)
D - Dynamic Abstractor (to support planning)
L - Latent variables (of the VAE) used instead of pixel images
I - Intrinsic motivation (to guide exploration)

Letters in place of 'X' in the benchmark acronym 'REAL_X':
O - Objects' position as input
I - Images as input
M - Macro-action as control output
J - Joints as control output

A first innovation of REAL-D is represented by a "dynamic abstractor (DA)," implementing the Abstractor component, allowing the performance of planning at levels of abstraction decided autonomously by the system (Fig. 3). The motivating idea of the DA is that the Planner should be able to reuse the actions of those triplets, even if the current state is not exactly one of starting states (s) of those triplets, or even if the goal state is not exactly one of the outcome (s') states. To define when a certain environment state s_i can be considered equivalent to an s or s' state of a triplet, we define a series of increasing thresholds that define different abstraction levels. These thresholds are constructed by considering the differences caused by the experienced actions, considering in particular the minimum and maximum differences. We reasoned that it does not make sense to make a distinction between states that are nearer than the minimum difference that the agent has experienced between each variable, since it has no actions that can move it through states at such a resolution. Conversely, the highest abstraction should not "merge" states that are farther than the maximum experienced difference since, with such a high threshold, even the action which caused the largest difference would not bring the agent to a different state. The DA algorithm thus works in detail as follows (see also Algorithm 2).

The output of the DA is an $L \times V$ matrix, where L is the number of desired abstraction levels (here 200), and V is the number of state variables to consider (e.g., the x-y position of an object; or the latent variables of an autoencoder used to abstract images, as explained below). For each level of abstraction, the DA gives V thresholds, one for each variable. This $L \times V$ matrix is used later by the Planner to decide which states are considered equivalent to each other at a given level of the abstraction: two state vectors whose difference is less than the thresholds specified for each variable v at a certain abstraction level l are considered equal, i.e., $s_1 \simeq s_2$ if $|s_1[v] - s_2[v]| < DA[l, v] \quad \forall v \in V$. To compute the levels of abstraction, the absolute differences $|s - s'|$ of each action triplet are first computed and then ranked, independently for each variable. From these ranked differences, 200 differences,

Algorithm 2: DA Threhold Matrix Generation

```
Input: (s, a, s') triplets, L levels of abstractions;
Output: abstractions distances matrix
foreach (s, a, s') ∈ triplets do
    difference_vector ← |s − s'|
    difference_matrix.append(difference_vector)
for v ← 1 to V                          // V variables
do
    sort_by_column(difference_matrix, v)
for l ← 1 to L     // L levels of abstraction
do
    t ← 0                                // T triplets
    for v ← 1 to V do
        abstractions[l, v] ← difference_matrix[t, v]
    t ← t + T / (L - 1)
```

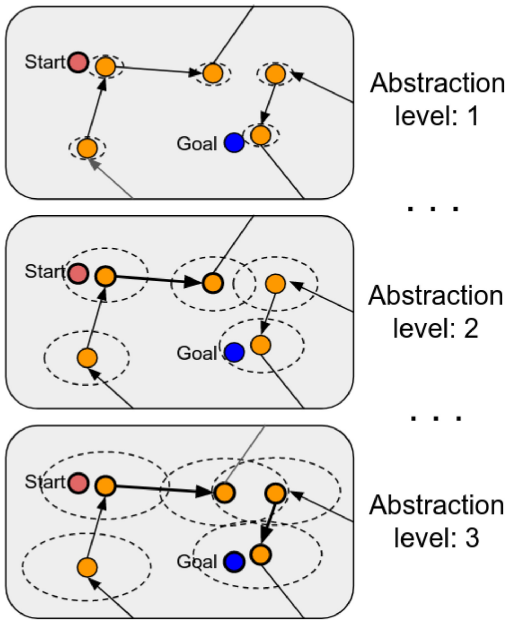


Fig. 3. Dynamic abstraction: scheme of functioning. In each of the three rectangles, a different level of abstraction is represented. The yellow circles in each rectangle represent the states experienced during the intrinsic phase, the red circle represents the current position of the object seen in the environment, and the blue circle represents the position of the object in the goal image. The arrows are the experienced actions that connect the different states. The dashed ovals represent the boundary within which a certain state (at the center of each oval) is considered to be the same as the states within the boundary. The oval shape of the dashed circles suggests that at each level of abstraction each state dimension can have different thresholds. The bold circles and arrows represent the plan portion that the algorithms managed to find up to a certain level of abstraction: only the last abstraction level allows finding a plan linking the start and goal states.

one for each abstraction level, are then selected for each variable, starting from the lowest rank up to the maximum rank, at equal intervals. The lowest level of abstraction thus involves V thresholds representing the minimum differences found, for each state variable, between all the starting-outcome state couples of the triplets; the maximum level of abstraction uses instead the largest differences found. This process allows the DA to work in an unsupervised fashion for any domain and requires only a single parameter establishing the granularity of the different abstraction levels (how many levels are desired). The DA can then be used to pursue a given extrinsic goal based on the planning processes described below.

In this system, the Planner is based on the A* algorithm having a maximum depth of ten actions. The heuristic used for the A* search was the outcome-goal distance based on the L-1 norm. The planning process works in close integration with the DA mechanism. The states are initially abstracted at "abstraction level 1": if a plan is found, its first action is executed, otherwise the planning process tries to find a plan with the succeeding higher level of abstraction ("abstraction level 2," "abstraction level 3," etc.) until a plan is found. If the DA reaches the maximum level of abstraction without finding a plan, the goal is aborted as deemed not reachable with the current action triplets. By using the DA, planning is able to generalize the triplet preconditions or triplet outcomes to increasingly different environment states. A higher abstraction, however, comes at the cost of less accurate actions.

To test the utility of the DA in this condition (REAL_OM) we also tested a system, called REAL-T, where we manually set a *fixed* abstraction threshold that was optimised manually. It was not possible to test REAL-D system with images as the A* Planner was not able to return a plan before running out of memory.

3) REAL-LD—Latent Variables and Dynamic Abstractor: The REAL-LD system was used to face the version of the benchmark where the robot had to use raw images although it could still use the macro-action, the REAL_IM condition. The Explorer and Planner components were as in REAL-D. The Abstractor was instead enhanced to handle RGB images with an additional abstraction process run at the end of the intrinsic phase. To this purpose, the precondition and outcome images were first processed with the *OpenCV MOG2* computer vision algorithm filtering everything that does not change much in different images, in our case, the background behind the objects. The algorithm does this in a fully autonomous way based only on the images collected during the intrinsic phase. We filtered the background because we assume that the robot is interested in learning about *things that it can change with its actions*, and so the static areas of the image are not interesting for it. Next, a variational autoencoder (VAE [58]) is trained with the background-filtered precondition and outcome images of the triplets. Based on this training, the VAE can extract a compact representation of images encoded by the activation pattern of the VAE bottleneck (latent variables). The latent state representations are used in place of the images in all planning processes during the extrinsic phase. REAL-LD was an important system that allowed to focus on studying the challenges posed by a perception based on raw-pixel images without the complications of the motor aspects. The system was thus also tested with two objects achieving lower performance than with 1 object but higher than random (0.095 versus 0.060 of REAL-R, not further reported).

REAL-LD was also tested in the REAL_IJ condition involving not only raw-pixel but also the direct control of joints. To face this condition, the macro-action was substituted with a control method that did not make specific assumptions about how to act in the environment. In particular, an action was implemented as a sequence of via points and directly used to control the arm-gripper joints (based on the robot's PIDs). In particular, the action was generated as follows. The whole action still lasted 1000 steps in total. First a random number of via points (from 1 to 10) was generated with a uniform distribution. Then each via point was generated by randomly sampling the joint angles from a uniform distribution.

4) REAL-ILD—Intrinsic Motivation Exploration, Latent Variables, Dynamic Abstractor: The REAL-ILD system worked as REAL-LD but its Exploration component was enhanced with a mechanism directed to improve the efficiency of exploration during the intrinsic phase based on a new intrinsic motivation mechanism (see Algorithm 3). This mechanism was in particular used to increase the likelihood that the actions generated for exploration touched the objects. The mechanism is based on a neural-network predictor that takes as input the current state and a planned action (parameters), and predicts if the action will lead to a *significant change* of the state, for example because an object has been moved. The predictor is trained every 500 000 steps based on the information acquired that far. A VAE is also trained every 500 000 steps to feed suitable representations of the images to the predictor, with the data collected that far. The predictor is then used before performing each exploration action to select the actions that have a high chance to cause a change (hit the objects). In particular, the system generates up to 1000 actions and then performs the first one that is predicted to cause a change or a random one if no such action is found.

III. Results

Table I reports the scores of all REAL-X systems tested with their respective versions of the benchmark. The performance and some aspects of the internal functioning of the systems are now analyzed in detail. A video of the performance of REAL-LD is available on https://youtu.be/kl26SyGAy_M (REAL_IM condition) and on https://youtu.be/nriO73Sftq0 (REAL_IJ condition).

A. REAL-D Versus REAL-R and REAL-T: Test in the REAL_OM Condition

The REAL-D system, tested in the REAL_OM condition, achieves a score of 0.216 while a random system REAL-R achieves a performance of 0.021.

The role of the DA for such result can be seen by comparing such performance to the one of REAL-T. We tried different values of the threshold, and only the best one led to a performance similar to the one of REAL-D, 0.212. In particular, we first tried with threshold values close to those

Algorithm 3: Intrinsic Motivation Exploration Based on the Change Predictor

```
Input: environment, action_space
Output: transitions
obs_pre ← env.get_observation()
for i ← 1 to I          // intrinsic phase steps
do
    // Bootstrap
    if i < 500 then
        action ← action_space.random_sample()
    else
        // Training every 500 actions
        if i mod 500 = 0 then
            abstract_transitions = VAE(transitions)
            max_difference = max(abst_pre - abs_post)
            dataset = ∅
            foreach pre,post ∈ abstract_transitions do
                if pre - post > max_difference * 0.01
                then
                    change = 1
                else
                    change = 0
                dataset.append(pre, action, change)
            network.train(dataset)
        // Action generation
        for n ← 1 to 1000 do
            action ← action_space.random_sample()
            change = network.predict(action)
            if change = 1 then
                break
    obs_post ← env.step(action)
    transitions.add(obs_pre, action, obs_post)
    obs_pre ← obs_post
```

most used by REAL-D while planning, that is 1 cm and 2 cm for, respectively, the x and y position of the object. The score, however, was only 0.141. We then tested lower values, (0.005, 0.01), which obtained a lower score (0.040), and higher values, (0.02, 0.04) and (0.04, 0.08), which obtained 0.212 and 0.150, respectively. This shows the capacity of the DA to automatically find a good level of abstraction compatible with the found actions.

To investigate how the DA works, Fig. 4 shows the number of effective actions available to the Planner (A*) for each level of abstraction. We consider an action to be "effective" if its precondition (starting state) is considered different from the outcome at the given abstraction level. At level 0, all actions are different, so the Planner has 15 000 actions available at its disposal (all the actions found in the intrinsic phase). However, it is unlikely that the starting state during the extrinsic phase is exactly the same as any precondition of the action triplets and it is also unlikely that the goal matches the outcome of

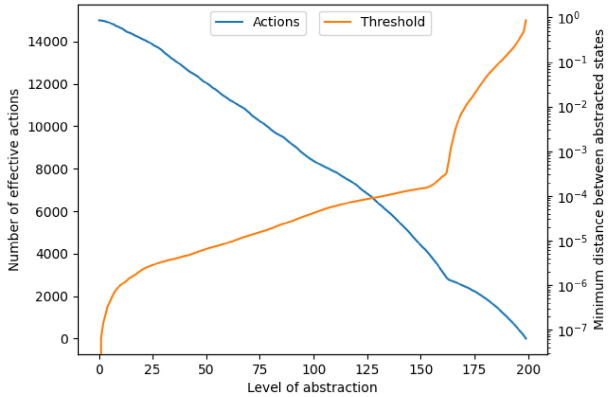


Fig. 4. REAL-D: Number of effective actions (left *y*-axis) at different levels of abstraction (*x*-axis). An action is effective if it changes the world from a state to another state that is considered different at the considered level of abstraction. The right *y*-axis measures the minimum distance between two states to be considered different at that abstraction level.

an action triplet. So at abstraction level 0, the Planner usually cannot find an action to start from, or an action that leads to the goal state. As the abstraction level is increased to higher levels, the minimum distance to consider two states as different rises (orange line in the figure), so an increasing number of actions are not effective as the precondition and outcome are no longer considered different. In particular, we can see that up to about the abstraction level 171, the distance is very small, less than 1 cm. When the distance is about 1 cm, only 2435 effective actions remain. This is because during the intrinsic phase most of the time the random macro-actions produced by the Explorer and performed in the environment miss the object and so the precondition-outcome difference is basically only due to the noise of the perceived object position. Indeed, only about one in six times the robot hits the object and displaces it by a distance higher than 1 cm. In the simulations, the Planner quickly filters out all the first abstraction levels where no solution is found and then starts finding workable solutions at the abstraction level 170 or higher. The combination of the Planner with the Dynamic Abstraction thus works effectively in providing the right abstraction level with which to plan by adapting to the experienced data and without the need of providing any preset threshold. In the given domain, this allows the system to automatically adapt to the failure to obtain significant effects with most actions and to the noise of the perceived object position.

B. REAL-LD: Test in the REAL_IM Condition

The REAL-LD system in the REAL_IM condition achieves an average score of 0.222, even slightly higher than the score of the REAL-D system in the simpler REAL_OM condition (0.216). This confirms that using the VAE to convert the images into a latent space, and then make plans on such a basis, performs just as well as receiving directly the object positions.

We thus investigate the quality of the latent variable representations. Fig. 5 shows how the Euclidean distance in the latent space correlates with the distance between the

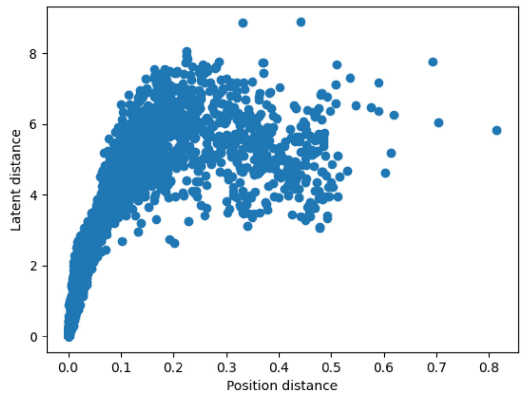


Fig. 5. REAL-LD in the REAL_IM condition: correlation between distances between randomly chosen couples of object locations (perceived through the RGB images), measured in both the real and the latent space. Spearman correlation $\rho = 0.75$, $p < 0.001$; Pearson correlation $r = 0.83$, $p < 0.001$.

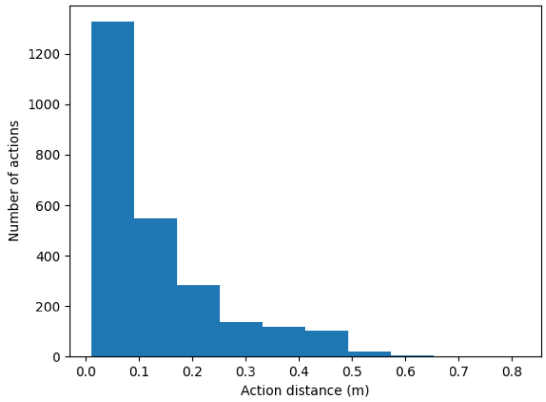


Fig. 6. REAL-LD in the REAL_IM condition: about 80% of the actions longer than 1 cm are shorter than 20 cm.

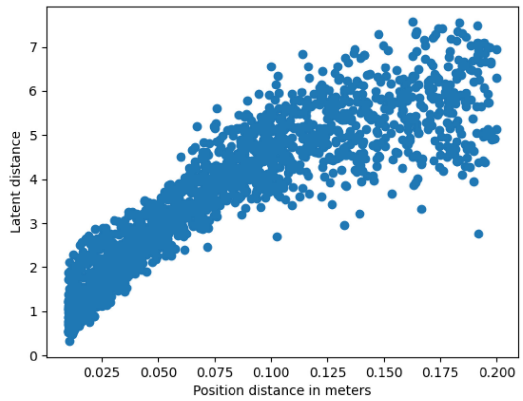


Fig. 7. REAL-LD in the REAL_IM condition: correlation between distances between couples of states, measured in both the real and the latent space, restricted to distances ranging between 1 and 20 cm. Spearman correlation $\rho = 0.94$, $p < 0.001$; Pearson correlation $r = 0.90$ $p < 0.001$.

object positions. The correlation is high (Spearman correlation $\rho = 0.75$, $p < 0.001$; Pearson correlation $r = 0.83$, $p < 0.001$), especially for short distances. About 80% of the actions longer than 1 cm (those mostly used for planning, see preceding analyses) are shorter than 20 cm (see Fig. 6): Fig. 7 shows that within that range the correlation is even higher (Spearman correlation $\rho = 0.94$, $p < 0.001$; Pearson

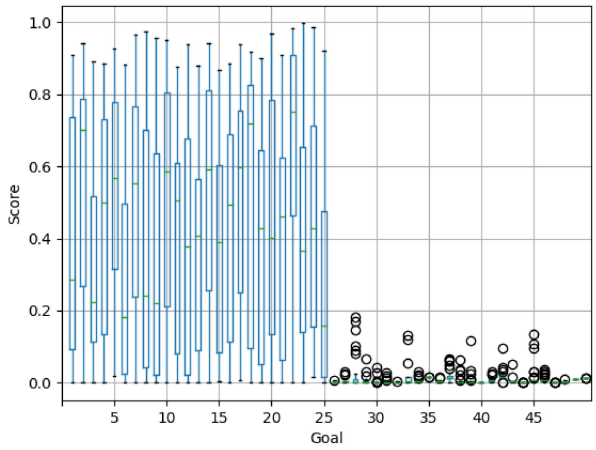


Fig. 8. REAL-LD in the REAL_IM condition: performance per goal. The first 25 goals are 2-D goals, while the rest are 2.5-D (15) and 3-D (10).

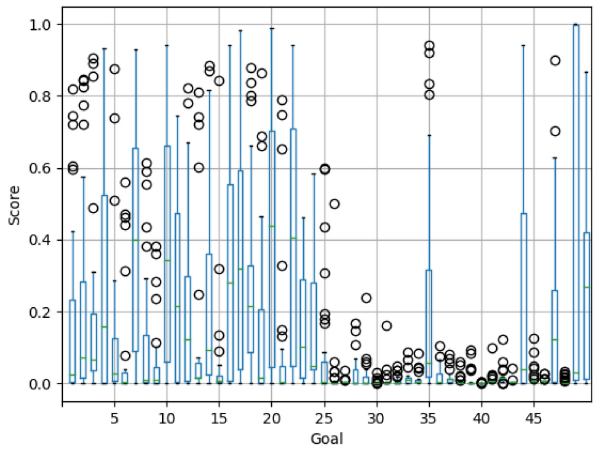


Fig. 9. REAL-LD on REAL_IJ condition: performance per goal. The first 25 goals are 2-D goals, while the rest are 2.5-D (15) and 3-D (10). The system manages to score also on the 2.5-D and 3-D goals as it can reach objects on the shelf and push them down on the table.

correlation $r = 0.90$ $p < 0.001$). However, REAL-LD is still limited by the use of the macro-action that can achieve only the first 25 goals but not the other goals involving an object located on the shelf (see Fig. 8), since the macro-action is restricted to pushing across the table surface.

C. REAL-LD: Test in the REAL_IJ Condition

REAL-LD was tested in the REAL_IJ condition to evaluate its ability to handle more general actions defined by an arbitrary sequence of joint via points. These general joint actions produce noncoordinated arm movements, which are however still able to hit the object and push it along the table; and they are able to reach the shelf as well. The REAL-LD system is capable of extracting from these noncoordinated actions the effective ones to reach its goals, with a performance of 0.139, which is lower than for the REAL_OM and REAL_IM conditions but still higher than the random system (0.038). Fig. 9 shows that while the REAL-LD performance is lower with respect to the previous condition with the macro-action, it can still move objects toward their goal. Moreover, the joint-based action can also reach objects located on the shelf, and

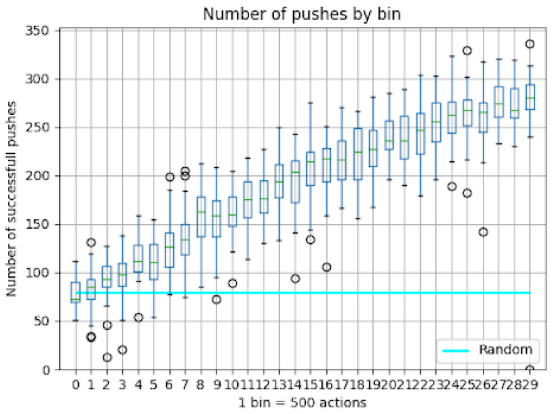


Fig. 10. REAL-ILD in the REAL_IM condition: progressive increase of object contacts/displacements per 500 actions (*y*-axis), achieved thanks to the IM change predictor used to select actions for exploration during the intrinsic phase (*x*-axis).

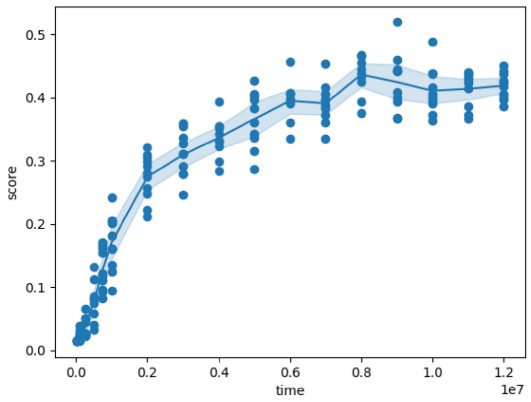


Fig. 11. REAL-LD in the REAL_IM condition: score in the extrinsic phase with different lengths of the intrinsic phase. The scores start to plateau after an intrinsic phase of about 8 million time steps, which leads to collect about 8000 actions.

this allows REAL-LD to partially accomplish some of the 2.5-D and 3-D goals (Fig. 9). In particular, the system did not find actions to put objects on the shelf but managed to find actions to push them down from the shelf. These tests thus shows that the REAL-X architecture can be used without predefined macro-actions. To further improve the performance of REAL-LD in the REAL_IJ condition, we would need the ability to discover more structured actions (out of the general via-point-based actions), but this would probably require a smarter exploration rather than the random-action sampling used by the REAL-LD system.

D. REAL-ILD: Test in the REAL_IM and REAL_IJ Condition

Fig. 10 shows the effectiveness of the change predictor during the intrinsic phase. Compared to the previous systems, where random action sampling led to hitting the object only one out of six actions, REAL-ILD gradually filters out the ineffective actions before executing them, so the chances to actually hit the object rise through the phase up to about 273 hits every 500 actions, so about half of the times. As an example, one of the simulations had 4269 actions pushing the object for more than 1 cm, compared to the 2435 in the REAL-LD system.

As shown in Table I, in the REAL_IM condition, the higher number of actions discovered allows REAL-ILD to achieve a higher performance than REAL-LD, 0.234 versus 0.222. In the REAL_IJ condition, these figures become 0.149 versus 0.139. This performance is only marginally higher so we investigated how the REAL-X performance scales with the number of actions. Fig. 11 shows the performance of REAL-LD on the REAL_IM condition with different lengths of the intrinsic phase. We can see that the performance of REAL-LD is not really limited by the number of experienced actions as it is already reaching a plateau with a length of the intrinsic phase of about 8ML steps (8000 actions). Most of the performance is indeed acquired during the first 2000 actions. These results suggest that to improve the performance the exploration should not simply provide "more actions" but rather focus on finding new actions most needed by the Planner. In the case of REAL_IJ this might for example mean finding actions that have a higher precision and actions able to bring the object to the shelf.

On the other hand, the current approach based on the change predictor can still be very useful where the performance is limited by the small number of actions collected during the intrinsic phase as it improved the speed of finding effective actions up to threefold. This can be especially relevant when using real robots, as the exploration can be very time consuming, so any improvement in speed is valuable even if it does not lead to a higher final performance.

E. REAL-LD With Multiple Objects

We also tested REAL-LD using multiple objects. With two objects, REAL-LD performance drops to 0.095. This is still higher than the performance of REAL-R, achieving 0.060, but it is drastically lower than when using only one object as the system is not equipped to deal with the complexity of multiple objects. In particular, multiple objects raise multiple challenges. First, the combinatorial explosion between the positions of multiple objects worsens the nonreset problem: the robot sees new conditions most of the times, so abstraction with the VAE is more challenging. Second, moving one object could also move another one, so making the outcomes appear less regular. We did not investigate which of these causes, or others, represented the most limiting factors.

IV. Discussion: REAL-X Challenges and Solutions

The tests performed highlight the different challenges posed by the REAL-X benchmark. These also reflect the current unsolved challenges of open-ended learning. We now discuss these challenges also considering the possible strategies that can be used to face them and that were in part explored here.

A. Exploration to Autonomously Discover Tasks/Goals and Learn Policies

An important challenge of open-ended learning is the performance of an exploration of the environment that allows the robots to gain knowledge on how to act in the world. A

major strategy followed in the literature is the autonomous construction of tasks, possibly requiring the achievement of "salient states" ("goals") in the environment [18], [42], [45]. The general solution followed here is based on the generation of movements in space (variants of already known actions) in order to experience new environment states. Then the new states are considered as "salient," and hence recorded as goals, if they satisfy two features: 1) *they involve the "*external*" environment:* to this purpose, here the robot brings the arm out of the camera sight and 2) *they are caused by the robot:* to this purpose, here the robot compares the world state before and after action performance. An additional feature that is often used, not considered here, is the fact that the salient state is different (or sufficiently different) from those already discovered [41], [47]. Few observations can be made on this strategy. In the REAL-X scenario, states do not change without the robot's action. This might be a future element to introduce into the scenario for additional realism and challenge. Facing this solution would require the robot to understand what is caused by its action and what by other agents or the environment dynamics [29], [38].

Another relevant aspect of exploration is the opportunity to identify and focus on some small regions of the exploration space where information gain is higher. In REAL-X, this region is represented by the work plane where objects lay. A possible strategy to do so, also followed here, is to use predictors capable of anticipating the effects of actions so as to perform in the environment those actions that are most promising [42], [45], [59].

B. Interesting Events That Occur Infrequently and Thus Cause Sparse Rewards

An important challenge of open-ended learning is the fact that events supporting learning might be rare, especially at the beginning of learning when the robot's motor repertoire is poor. This causes "sparse rewards," in case one uses RL, and in general little information for learning. REAL-X presents this challenge in at least two forms. First, the fact that touching objects based on random movements happens rarely. This challenge was already discussed in the previous point and the solution considered there leverages the fact that, although rare, the event sometimes still takes place. The second case involves events that in practical terms never happen with random movements (this belongs to a type of events that might be called *super-rare*). Consider the event of lifting objects on the plane. This event is extremely important as it might lead robots to learn to grasp objects thus opening the possibility to learn several other behaviors, for example in REAL-X to carry objects on the shelf. What could be the solutions adopted with super-rare events? Babies learn to grasp based on their tendency to keep the contact with objects for a prolonged time once they get in touch with them based on the *grasping reflex* [60], [61]. The capacity to have long interactions with objects would probably require close-loop policies and haptic sensors, solutions not explored here. Since hardwiring behaviors is not allowed in REAL-X (e.g., as done in the system presented in [62]), the attainment of effects similar to those of the grasping reflex might be possibly achieved with novelty or surprise intrinsic motivations used to mark the saliency of the hand–object interactions [63]: as an example, [46] used mutual information between the agent state (proprioception) and environment state (object positions) to achieve good results in a pick and place scenario.

C. Environment Reset

Learning (e.g., RL) is usually based on *trials* that allow the agent to have the *same experience multiple times* and thus to progressively accumulate knowledge. In open-ended learning, as reflected in the REAL-X benchmark, the experience is *not* divided in trials at the end of which the environment is *reset* to the same initial conditions. In this case, the robot can generate the trials autonomously to aid learning, as done in the REAL-X architecture, but the environment cannot be reset to a desired condition. This represents a great challenge as each trial involves different initial conditions (several strategies usable with constant initial conditions, see [23], [64], cannot be used). A possible strategy, used here, is to adopt solutions able to learn the different conditions as they are experienced. Another possible strategy, usable in some contexts, would be controllers able to self-recreate the initial conditions. The importance of the absence of trials has also been emphasized in a recent formalization of Autonomous Reinforcement Learning [65].

D. Learning Actions From Scratch

Another challenge of open-ended learning is to learn movements by controlling low-level elements as joints. In this respect, the results showed that control based on macro-actions allows a faster learning but does not allow learning some movements (e.g., with the macro-actions used here, reaching objects on the shelf). On the other side, joint-based control (a requirement of the REAL-X challenge) allows more flexibility but leads to a slow learning and (initially) irregular movements. A possible strategy to harvest the advantages of high and low-level control is suggested by the brain, which learns motor actions and then "habits" recalling them in correspondence to specific stimuli [66]. Hierarchical reinforcement learning [18] is a strategy imitating this solution and some systems use it in the context of open-ended learning [67].

E. Need to Learn From Scratch How to Represent Objects

Another important challenge of open-ended learning, reflected in the REAL-X scenario, is the need to represent objects based on low-level information such as image pixels. This problem is made even harder by the fact that the efficiency of this low-level perception strongly depends on action learning because an efficient exploration and motor behavior is important, as we have seen in the solutions adopted here, to gather information on objects. Here, information on objects was "compiled" into a neural network (variational autoencoder) at the end of the initial learning phase and then used to support the following planning phase (see also [22]). Object representations might however be built during the acquisition of information on objects if this can support action

learning (see [44], [45] where the VAE latent learning supports exploration).

F. Need to Generalize Acquired Knowledge to Varying Conditions

An important challenge of open-ended learning, reflected in the REAL-X benchmark, is the need to generalize knowledge related to perceptions, actions, and world models (supporting planning) to the ever changing environment conditions. A strategy often used to this purpose is to store the raw data and at the same time to use it to train neural networks capable of generalization. Then the knowledge encoded in one of the two forms is used depending on the needs (e.g., [51]). In particular, nonparametric models based on raw data can be useful in initial phases of learning to face bootstrapping conditions. Instead, neural networks, having higher generalization capabilities, can be used in later stages of learning when more data are available. Neural networks however also encounter the problem that open-ended learning produces nonstationary data and this causes catastrophic forgetting and interference. The machine learning field of "lifelong learning" offers solutions to these challenges [68].

G. Multiple Objects

Another problem of open-ended learning, and more in general of real-life scenarios, is that the robot has to deal with multiple objects at the same time. As shown in the study presented here, this poses different challenges for vision, for example for classifying and segmenting the different objects, due to the combinatorial explosion caused by multiple objects. Solutions to this problem might rely on attention mechanisms focusing on single objects [36], [69]. Another challenge is that multiple objects are often cluttered, and this requires dealing with partially visible objects [70]. Multiple clutter objects also pose problems to actions, which requires strategies for obstacle avoidance and iterative behaviors [71].

H. All Challenges Must Be Addressed Simultaneously

As mentioned multiple times, open-ended learning poses the problem that the robot faces all the challenges considered above at the same time, in particular those involving simultaneous learning problems. A possible strategy to address this problem is to use approaches that allow the reduction of the interdependencies between the different solutions. For example, some solutions presented here performed actions without relying much on a trained vision system. Another possibility is a suitable coordination between the different processes and strategies to use partially trained components. For example, some solutions presented here used a partially trained visual component to support action guidance.

V. Related Works

Recently, a number of papers have proposed systems and algorithms that seem promising for meeting the challenges posed by the REAL competition. Here, we review these systems, many of which have also been tested on robotic simulations or on real robots, whose ideas we believe can be integrated or compared with those presented in REAL-X.

Some of these works have focused on exploration. For example, [38], [39], and [40] train a policy to explore unknown environments using only images by providing a reward based on prediction errors. In particular, the policy is rewarded if it leads the agent to places where a predictor gives a high error [38], [39], or where an ensemble of predictors do not agree [40] thus showing uncertainty of information. The exploration reward can be combined with an extrinsic task reward to directly train a desired task policy, or it could be used alone to first explore the environment and then use the data obtained with the exploration to later train a task oriented policy with other methods. Sekar et al. [41] further augmented the exploration strategy by adding planning: instead of computing the prediction error retrospectively, they add a planning step where the consequences of the current policy from the current starting state are imagined using a world model, and then the policy is optimised to seek "novel states" before executing it in the environment to explore (similar to our use of a change predictor to prune actions before executing them). To learn a world model and optimize the policy using imagined trajectories, they rely on the work of [72] that shows the advantages of learning long-horizon behaviors based on purely latent variable-based imagination. In contrast to the previous works, the system presented in [41] uses continuous actions and learns a world model that is then used to train the agent without additional data for any task for which the reward function is available. The agent uses the world model and the reward function to run a policy in its "imagination" (i.e., running imaginary trials with the world model) and adapt it to the task specified by the reward function. The trained policy can be then run on the external world without additional training. However, this requires that the task assigned can be specified with a reward function and that this reward function is provided to the agent. This is not the case in the REAL benchmark, where only a goal state is provided in the form of an image. An additional component would thus be needed to run the system from [41] in REAL, in particular to create a suitable reward function that enables the policy to be trained to reach the target image. In a subsequent work, [42], this limitation is lifted as an "achiever" goal-conditioned policy is concurrently trained, so that after training, the system can be readily used to try and achieve any goal-image state.

Another work [44] also focused on the exploration problem, but in this case the images are first processed through a VAE, and then the exploration is done by drawing from the VAE latent space using biased-weights to encourage exploration of low probability areas. Similarly, the system proposed in [45] used the VAE latent space but added a learned reachability network to ensure that exploration was done on the frontier of already explored space. The same approach of using the latent space of a VAE has been applied earlier in [43]. While [43] did not use biased-weights it showed a full application of the concept on a robotic scenario: starting with a first set of images to train the VAE (either given or obtained by random exploration) the robot automatically samples new images from the VAE and then learns to achieve them, thus setting goals in an

autonomous way. The data collected while trying to achieve those goals can then be further fed back into the VAE, allowing the creation of further new goals.

While the above works base exploration on the "novelty" of states, another work [23] instead progressively trains an agent by using a network to predict states of "intermediate difficulty," that is, states that the agent finds possible but hard to reach. This latter work thus grounds its exploration on "competence" instead of "novelty" [47]. However, the system from [23] assumes a continuous goal-space representation but does not use images but rather positions and velocities. This approach also requires to evaluate the feasibility of each goal by trying to reach multiple times, which can be very time consuming. Furthermore, the system assumes the reset of the environment to its starting condition after every episode, which assumes an external control on the environment to aid the training that is not available in REAL.

Other relevant works have focused on learning suitable representations to enable planning in an unknown environment directly based on images [48], [49], [50], [51], [52]. The system in [48] uses a mutual information constraint between observations and latent states to train the latent space representations, while at the same time optimizing for the rewards by jointly training a reward predictor. The authors present the results of experiments that show how the learned representations are robust to distractors in the images that do not alter the dynamics of the task. While the focus on mutual information is interesting and could be incorporated in our abstraction module (i.e., in the training objective of the VAE to improve the latent representation), the reward prediction part has no direct application in our intrinsic phase scenario. In a similar vein, the system proposed in [52] uses a mutual information plus an empowerment objective to improve the training of an image encoder so that it focuses on encoding action-relevant features and not on general image reconstruction. However, [52] also includes in the objective function an external reward which is not present in our autonomous setting. A mutual information constraint is also used in the system proposed in [46], which learns to optimize a reward based on the mutual information between the agent's state (e.g., proprioception) and the environment state. The approach used, called MUSIC, is tested successfully in the FetchPush, FetchSlide, and FetchPickAndPlace OpenAI Gym scenarios, which are similar in spirit to the REAL scenario. The algorithm, as it is presented, seems to require a reset of the agent to an initial state at the end of each episode and it also seems directed to learn a general useful single policy, to be later tuned for a task with an external reward instead of a goal-conditioned policy. The authors however also combine MUSIC with earlier unsupervised learning methods DIAYN [73] and DISCERN [74], which are able to learn multiple policies or a goal-conditioned policy, respectively. The authors of MUSIC show that DIAYN and DISCERN alone do not achieve good performances in these environments alone, but they can be combined with MUSIC to do so. In those scenarios, however, the agent did not use a camera but directly accessed the true object position. The system proposed in [50] uses a distributional planning network to learn a latent space over which to plan actions to reach a goal. While the trained distributional planning network they obtain could be directly used as a controller, they then actually train a soft-actor–critic controller using rewards derived from the distances in the latent space. Their tests show how the distributional planning network creates a latent space with a distance metric that is more successful and meaningful than the metric learned by an inverse model, a VAE, or by simply calculating the distance in the pixel space. While their simulations and real world experiments use a separate SAC system trained for the task, the number of samples used is comparable to our approach (e.g., 20 000 10-frame videos of random interactions; or 28 h of capture in one of the robotic experiments, which correspond to about 20 million time steps in REAL).

The system proposed in [49] is based on a sparse graphical memory (SGM): a new data structure that stores observations and feasible transitions in a sparse memory. This is similar to our usage of triplets as a graph on which to do planning, but their approach focuses on how to keep the nodes of the graph limited (merging observations) while keeping at the same time the consistency of the graph (e.g., do not merge two states that cannot have similar successors). The system also assumes to have access to a short-horizon parametric controller that is capable of accomplishing the task when the starting and goal states are nearby, that is, the optimal action sequence is short. Similarly to [51] a low-level controller is thus needed to execute the plans. The system proposed in this latter work, [51], learns a latent space from images with a Causal InfoGAN or a Context Conditional CIGAN to then make a plan with A*. This plan however does not contain actions, but only a sequence of images (reconstructed from the latent variables) of the states that the system has to achieve to reach its goal. This visual plan is then fed to a "visual tracker," a short-horizon controller that tries to reduce the differences between the current input image and the image of the current step in the visual plan. Both the inverse model of the visual tracker and the latent space are learned by a data set that can be constructed by the robot itself with random exploration, with a number of samples comparable to our system (i.e., about 12 000 observations in one of their robotic scenarios). Unfortunately, no code is available to compare their approach to ours.

Also with a focus on learning useful representation of visual planning with images, [53] proposes a system called MBOLD where both a forward dynamics model and a distance model are jointly learned. The forward dynamics learns to predict subsequent images given a starting image and a sequence of actions, while the distance model uses Q-learning to learn a distance function between images. This distance function is trained by using an indicator goal-dependant reward function which equals to 1 when a goal image is reached. As the Q-learning scores images further away with temporally discount rewards, the final trained values are equivalent to scoring images based on how much they are distant from the goal image not in terms of visual features but in terms of "functional" distance, that is, how many timesteps it would take to reach the goal image from there. After the learning phase, done on a database of random interactions with the environment, the authors use model-predictive control to achieve any goal in the environment; in practice, random trajectories are

evaluated using the forward dynamics model and the distance function, and iteratively improved until a trajectory is selected for execution; trajectories are also replanned at every timestep so that inaccurate predictions do not sum up over time. In a similar way, [75] also trains offline from a database of visual interactions to learn Q-values that are later used to drive goal-conditioned policy. However, in their experiment they either use this training as a support for later task-specific training (with task rewards) or in the general case, they do learn a goal-reaching policy (without rewards) but they start from a database of interactions with the environment executed by other RL algorithms.

Mechanisms from all these works might be integrated into the REAL-X architecture components to solve different aspects of the REAL competition, for example to achieve better exploration or to learn better representations. Among these works, only [42], [43], [49], [50], [51], and [53] seem to offer complete solutions that could be applied to solve the REAL benchmark from the intrinsic phase to the extrinsic phase. However, there is a relevant technical challenge that is encountered when readapting their software to interface it with the benchmark software, which we encountered while trying to apply the code of [43] to REAL. The software of these systems is indeed organized as commonly done for reinforcement learning systems rather than as it would be required by the REAL open-ended learning benchmark. In particular, the software requires that the environment variable is passed to the system program which then initializes it and controls it whereas the REAL benchmark requires the system to be passed to the REAL environment (i.e., the REAL evaluation function). The REAL evaluation function also requires that: 1) the system is organized in terms of an initialization function and a 1-step function that can be called to pass the last observation to the system and get its next action; 2) the system can run in an intrinsic motivation mode where the environment is not reset and the system is not given any guidance in terms of reward function, goals, etc., and 3) the system can be run in an extrinsic motivation mode where it can receive goals to pursue within a given time. When available, the software of most of the systems discussed above present this problem and so it is technically cumbersome to adapt their code to test them, or parts of them, with the REAL benchmark software.

This work focused on illustrating and exploring all the several challenges posed by the REAL open-ended learning benchmark, and to develop a general architecture that can be used to implement different systems to face those challenges. Here, we have shown and compared multiple instances of these systems. In future work, we plan to use the REAL benchmark to further develop these solutions and systematically compare and integrate them with the systems and mechanisms reviewed above.

VI. Conclusion

Several years of research within the developmental robotics community have uncovered important mechanisms for supporting open-ended learning [76]. However, this far the community has not managed to propose a benchmark on open-ended learning, allowing quantitative comparison of competing approaches. The REAL competition closes this gap by formulating an open-ended learning benchmark that allows a rigorous measure of the quality of the knowledge that an autonomous learning agent is able to acquire during free interaction with its environment. This measure leverages an extrinsic phase, where an agent is requested to solve a number of tasks sampled from an environment of interest, based on the knowledge that it has autonomously acquired in a previous intrinsic phase involving a long autonomous learning experience in the same environment. The particular open-ended learning benchmark proposed here involves a camera-arm-gripper robot engaged in manipulating a number of objects on a table and a shelf. The benchmark is extremely challenging as it requires the solution of a number of problems such as exploring to get in contact with the objects, learning to perceive them, self-generating goals, acquiring the skills to accomplish these goals, etc. In addition, the robot has to face all these challenges at the same time, finding itself in a condition similar to the one of newborn infants.

Here, we have presented several systems implementing a blueprint robot architecture, REAL-X, that can be used to face different versions of the REAL benchmark where initial simplifications are progressively removed. The architecture abilities involve the capacities: to autonomously process images to extract relevant information on objects; to explore the environment and progressively focus on relevant action outcomes; to dynamically abstract over state representations in order to support action planning. The performance of the different implementations of the architecture was well above chance level when some simplifications were still kept, and still above chance level in the most challenging conditions. This suggests that the REAL benchmark might remain a challenge for years to come. We are aware that the REAL-X systems represent only a first step toward a full solution of the benchmark. Future work might consider longer intrinsic phases to allow the emergence of more sophisticated behaviors, for example for manipulating the objects beyond only pushing them (e.g., for hitting, turning, or grasping them); the introduction of other mechanisms to support a further focusing of the robot's learning processes; and the introduction of other abstraction or attentional processes allowing the robot to cope with multiple objects. Notwithstanding these open problems, the REAL benchmark and the REAL-X architectures represent valuable tools to study the fascinating challenges posed by truly open-ended learning.

Acknowledgment

The authors thank Francesco Mannella for developing the REAL environment, and Simone Asci for developing and testing earlier versions of the change predictor. They also thank Hewlett Packard Enterprise for providing the hardware to run several of the simulations whose results are reported here.

References

[1] D. E. Berlyne, *Conflict, Arousal, and Curiosity*. New York, NY, USA: McGraw-Hill Book Company, 1960.

[2] G. Baldassarre, "What are intrinsic motivations? a biological perspective," in *Proc. Int. Conf. Develop. Learn. Epigenet. Robot.*, Frankfurt am Main, Germany, 2011, pp. 1–8.

[3] J. Gottlieb, P.-Y. Oudeyer, M. Lopes, and A. Baranes, "Information-seeking, curiosity, and attention: Computational and neural mechanisms," *Trends Cogn. Sci.*, vol. 17, no. 11, pp. 585–593, 2013.

[4] G. Baldassarre and M. Mirolli, *Intrinsically Motivated Learning in Natural and Artificial Systems*. Berlin, Germany: Springer, 2013.

[5] S. Doncieux et al., "Open-ended learning: A conceptual framework based on representational redescription," *Front. Neurorobot.*, vol. 12, p. 59, Sep. 2018.

[6] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini, "Developmental robotics: A survey," *Connection Sci.*, vol. 15, no. 4, pp. 151–190, 2003.

[7] A. Cangelosi and M. Schlesinger, *Developmental Robotics: From Babies to Robots*. Cambridge, MA, USA: MIT Press, 2015.

[8] A. G. Barto, S. Singh, and N. Chentanez, "Intrinsically motivated learning of hierarchical collections of skills," in *Proc. 3rd Int. Conf. Develop. Learn.*, 2004, pp. 112–119.

[9] M. Schembri, M. Mirolli, and G. Baldassarre, "Evolving childhood's length and learning parameters in an intrinsically motivated reinforcement learning robot," in *Proc. 7th Int. Conf. Epigenet. Robot. (EpiRob)*, 2007, pp. 141–148.

[10] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner, "Intrinsic motivation systems for autonomous mental development," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 265–286, Apr. 2007.

[11] J. Schmidhuber, "Formal theory of creativity, fun, and intrinsic motivation (1990–2010)," *IEEE Trans. Auton. Mental Develop.*, vol. 2, no. 3, pp. 230–247, Sep. 2010.

[12] Y. Zhao, C. A. Rothkopf, J. Triesch, and B. E. Shi, "A unified model of the joint development of disparity selectivity and vergence control," in *Proc. IEEE Int. Conf. Develop. Learn. Epigenet. Robot. (ICDL)*, 2012, pp. 1–6.

[13] V. G. Santucci, G. Baldassarre, and M. Mirolli, "Cumulative learning through intrinsic reinforcements," in *Evolution, Complexity and Artificial Life*. Berlin, Germany: Springer, 2014, pp. 107–122.

[14] D. Tanneberg, J. Peters, and E. Rueckert, "Intrinsic motivation and mental replay enable efficient online adaptation in stochastic recurrent networks," *Neural Netw.*, vol. 109, pp. 67–80, Jan. 2019.

[15] S. Eckmann, L. Klimmasch, B. E. Shi, and J. Triesch, "Active efficient coding explains the development of binocular vision and its failure in amblyopia," *Proc. Nat. Acad. Sci.*, vol. 117, no. 11, pp. 6156–6162, 2020.

[16] F. Bellas, R. J. Duro, A. Faiña, and D. Souto, "Multilevel darwinist brain (MDB): Artificial evolution in a cognitive architecture for real robots," *IEEE Trans. Auton. Mental Develop.*, vol. 2, no. 4, pp. 340–354, Dec. 2010.

[17] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[18] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete Event Dyn. Syst.*, vol. 13, nos. 1–2, pp. 341–379, 2003.

[19] V. G. Santucci, G. Baldassarre, and M. Mirolli, "Intrinsic motivation signals for driving the acquisition of multiple tasks: A simulated robotic study," in *Proc. 12th Int. Conf. Cogn. Model. (ICCM)*, 2013, pp. 1–6.

[20] V. G. Santucci, G. Baldassarre, and M. Mirolli, "Autonomous selection of the 'what' and the 'how' of learning: An intrinsically motivated system tested with a two armed robot," in *Proc. Joint IEEE Int. Conf. Develop. Learn. Epigenet. Robot.*, Genoa, Italy, 2014, pp. 434–439.

[21] S. Forestier, R. Portelas, Y. Mollard, and P.-Y. Oudeyer, "Intrinsically motivated goal exploration processes with automatic curriculum learning," 2017, *arXiv:1708.02190*.

[22] A. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, "Visual reinforcement learning with imagined goals," in *Proc. 2nd Lifelong Learn. Reinforcement Learn. Approach Workshop (LLRLA FAIM)*, Stockholm, Sweden, Jul. 2018, pp. 9209–9220.

[23] D. Held, X. Geng, C. Florensa, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," 2017, *arXiv:1705.06366*.

[24] L. Meeden and D. Blank, "Developing grounded goals through instant replay learning," in *Proc. 7th Joint IEEE Int. Conf. Develop. Learn. Epigenet. Robot.*, 2017, pp. 196–201.

[25] M. Rolf and M. Asada, "Autonomous development of goals: From generic rewards to goal and self detection," in *Proc. Joint IEEE Int. Conf. Develop. Learn. Epigenet. Robot.*, Genoa, Italy, 2014, pp. 187–194.

[26] V. G. Santucci, G. Baldassarre, and M. Mirolli, "GRAIL: A goal-discovering robotic architecture for intrinsically-motivated learning," *IEEE Trans. Cogn. Devel. Syst.*, vol. 8, no. 3, pp. 214–231, Sep. 2016.

[27] K. Seepanomwan, V. G. Santucci, and G. Baldassarre, "Intrinsically motivated discovered outcomes boost user's goals achievement in a humanoid robot," in *Proc. Joint IEEE Int. Conf. Develop. Learn. Epigenet. Robot.*, Lisbon, Portugal, 2017, pp. 178–183.

[28] S. Kim, A. Coninx, and S. Doncieux, "From exploration to control: Learning object manipulation skills through novelty search and local adaptation," *Robot. Auton. Syst.*, vol. 136, Feb. 2021, Art. no. 103710.

[29] V. Sperati and G. Baldassarre, "Bio-inspired model learning visual goals and attention skills through contingencies and intrinsic motivations," *IEEE Trans. Cogn. Devel. Syst.*, vol. 10, no. 2, pp. 326–344, Jun. 2018.

[30] M. C. Machado, M. G. Bellemare, and M. Bowling, "A Laplacian framework for option discovery in reinforcement learning," 2017, *arXiv:1703.00956*.

[31] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1471–1479.

[32] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," 2018, *arXiv:1802.06070*.

[33] J. Achiam, H. Edwards, D. Amodei, and P. Abbeel, "Variational option discovery algorithms," 2018, *arXiv:1807.10299*.

[34] E. Cartoni, F. Mannella, V. G. Santucci, J. Triesch, E. Rueckert, and G. Baldassarre, "REAL-2019: Robot open-ended autonomous learning competition," in *Proc. Int. Conf. Mach. Learn. Res.*, 2020, pp. 142–152.

[35] G. Baldassarre, D. Montella, V. G. Santucci, and E. Cartoni, "REAL 2021—Robot open-ended autonomous learning: A competition and benchmark," in *Proc. IEEE Int. Conf. Develop. Learn. (ICDL)*, 2021, pp. 1–8.

[36] G. Baldassarre, W. Lord, G. Granato, and V. G. Santucci, "An embodied agent learning affordances with intrinsic motivations and solving extrinsic tasks with attention and one-step planning," *Front. Neurorobot.*, vol. 13, p. 45, Jul. 2019.

[37] W. James, *Principles of Psychology*. New York, NY, USA: Pantianos Classics, 1980.

[38] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," May 2017, *arXiv:1705.05363*.

[39] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros, "Large-scale study of curiosity-driven learning," 2018, *arXiv:1808.04355*.

[40] D. Pathak, D. Gandhi, and A. Gupta, "Self-supervised exploration via disagreement," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, Jun. 2019, pp. 8887–8896.

[41] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak, "Planning to explore via self-supervised world models," 2020, *arXiv:2005.05960*.

[42] R. Mendonca, O. Rybkin, K. Daniilidis, D. Hafner, and D. Pathak, "Discovering and achieving goals via world models," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds. Red Hook, NY, USA: Curran, 2021. [Online]. Available: https://openreview.net/forum?id=6vWuYzkp8d

[43] A. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, "Visual reinforcement learning with imagined goals," 2018, *arXiv:1807.04742*.

[44] V. H. Pong, M. Dalal, S. Lin, A. Nair, S. Bahl, and S. Levine, "Skew-fit: State-covering self-supervised reinforcement learning," 2019, *arXiv:1903.03698*.

[45] H. Bharadhwaj, A. Garg, and F. Shkurti, "LEAF: Latent exploration along the frontier," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2021, pp. 677–684.

[46] R. Zhao, Y. Gao, P. Abbeel, V. Tresp, and W. Xu, "Mutual information state intrinsic control," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–16. [Online]. Available: https://openreview.net/forum?id=OthEq8I5v1

[47] V. G. Santucci, G. Baldassarre, and M. Mirolli, "Which is the best intrinsic motivation signal for learning multiple skills?" *Front. Neurorobot.*, vol. 7, p. 22, Nov. 2013.

[48] Y. Ding, I. Clavera, and P. Abbeel, "Mutual information maximization for robust plannable representations," May 2020, *arXiv:2005.08114*.

[49] M. Laskin, S. Emmons, A. Jain, T. Kurutach, P. Abbeel, and D. Pathak, "Sparse graphical memory for robust planning," 2019, *arXiv:2003.06417*.

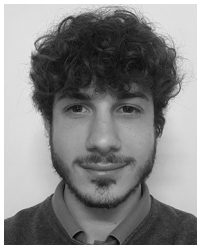
[50] T. Yu, G. Shevchuk, D. Sadigh, and C. Finn, "Unsupervised visuomotor control through distributional planning networks," 2019, *arXiv:1902.05542*.

[51] A. Wang, T. Kurutach, K. Liu, P. Abbeel, and A. Tamar, "Learning robotic manipulation through visual planning and acting," May 2019, *arXiv:1905.04411*.

[52] H. Bharadhwaj, M. Babaeizadeh, D. Erhan, and S. Levine, "Information prioritization through empowerment in visual model-based RL," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–22. [Online]. Available: https://openreview.net/forum?id=DfUjyyRW90
[53] S. Tian et al., "Model-based visual planning with self-supervised functional distances," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–19. [Online]. Available: https://openreview.net/forum?id=UcoXdfrORC
[54] M. D. Verme, B. C. da Silva, and G. Baldassarre, "Optimal options for multi-task reinforcement learning under time constraints," 2020, *arXiv:2001.01620*.
[55] "Robogym." OpenAI. 2020. [Online]. Available: https://github.com/openai/robogym
[56] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "RLBench: The robot learning benchmark & learning environment," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3019–3026, Apr. 2020.
[57] T. Yu et al., "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," 2019, *arXiv:1910.10897*.
[58] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, *arXiv:1312.6114v10*.
[59] L. Keller, D. Tanneberg, S. Stark, and J. Peters, "Model-based quality-diversity search for efficient robot learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2020, pp. 9675–9680.
[60] C. Lantz, K. Melén, and H. Forssberg, "Early infant grasping involves radial fingers," *Develop. Med. Child Neurol.*, vol. 38, no. 8, pp. 668–674, 1996.
[61] D. Caligiore and G. Baldassarre, "The development of reaching and grasping: Towards an integrated framework based on a critical review of computational and robotic models," in *Reach-to-Grasp Behaviour: Brain, Behaviour, and Modelling Across the Life Span*, D. Corbetta and M. Santello, Eds. New York, NY, USA: Taylor Francis Group, 2019, ch. 13, pp. 319–348.
[62] P. Savastano and S. Nolfi, "A robotic model of reaching and grasping development," *IEEE Trans. Auton. Mental Develop.*, vol. 5, no. 4, pp. 326–336, Dec. 2013.
[63] A. Barto, M. Mirolli, and G. Baldassarre, "Novelty or surprise?" *Front. Psychol. Cogn. Sci.*, vol. 4, p. 907, Dec. 2013.
[64] E. Cartoni and G. Baldassarre, "Autonomous discovery of the goal space to learn a parameterized skill," 2018, *arXiv:1805.07547*.
[65] A. Sharma et al., "Autonomous reinforcement learning: Formalism and benchmarking," in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–17. [Online]. Available: https://openreview.net/forum?id=nkaba3ND7B5
[66] F. Mannella and G. Baldassarre, "Selection of cortical dynamics for motor behaviour by the basal ganglia," *Biol. Cybern.*, vol. 109, pp. 575–595, Nov. 2015.
[67] A. Romero, G. Baldassarre, R. J. Duro, and V. G. Santucci, "Analysing autonomous open-ended learning of skills with different interdependent subgoals in robots," in *Proc. 20th Int. Conf. Adv. Robot. (ICAR)*, Ljubljana, Slovenia, Dec. 2021, pp. 646–651.
[68] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Netw.*, vol. 113, pp. 54–71, May 2019.
[69] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.
[70] B. Santra and D. P. Mukherjee, "A comprehensive survey on computer vision based approaches for automatic identification of products in retail store," *Image Vis. Comput.*, vol. 86, pp. 45–63, Jun. 2019.
[71] D. Kalashnikov et al., "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Proc. Int. Conf. Robot Learn.*, 2018, pp. 651–673.
[72] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," Dec. 2019, *arXiv:1912.01603*.
[73] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–22. [Online]. Available: https://openreview.net/forum?id=SJx63jRqFm
[74] D. Warde-Farley, T. V. de Wiele, T. Kulkarni, C. Ionescu, S. Hansen, and V. Mnih, "Unsupervised control through non-parametric discriminative rewards," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–17. [Online]. Available: https://openreview.net/forum?id=r1eVMnA9K7
[75] Y. Chebotar et al., "Actionable models: Unsupervised offline reinforcement learning of robotic skills," in *Proc. ICML*, 2021, pp. 1518–1528. [Online]. Available: http://proceedings.mlr.press/v139/chebotar21a.html
[76] A. Cangelosi and M. Schlesinger, *Developmental Robotics: From Babies to Robots*. Boston, MA, USA: MIT Press, 2015.

Emilio Cartoni received the Ph.D. degree in behavioral neuroscience from the Sapienza University of Rome, Rome, Italy.

He is a Researcher with the Institute of Cognitive Sciences and Technologies, Italian National Research Council, Rome. He contributed to writing the winning proposal for the EU FET-OPEN project: "GOAL-Robots—Goal-based Open-ended Autonomous Learning Robots" from 2016 to 2020, the European Space Agency project "IMPACT" from 2018 to 2019, and the EU FET Innovation Launchpad project "GROW" from 2021 to 2023 which he coordinated. His research expertise and interests involve artificial intelligence, autonomous robotics, exploration in open-ended learning, autonomous robot learning based on self-generated goals, and computational models of animal learning (from neural structures to behavior).



Davide Montella received the B.A. degree in computer science from the Sapienza University of Rome, Rome, Italy, in 2019.

He collaborated with the Institute of Cognitive Sciences and Technologies, Italian National Research Council, Rome, from 2019 to 2021, where he contributed to the development of robotic architectures based on intrinsically motivated open-ended learning. He is a science and technology enthusiast and believes that these can be used to improve everyone's quality of life. His life purpose is to develop technologies that contribute to the betterment of the world.

Jochen Triesch (Member, IEEE) received the Diploma and Ph.D. degrees in physics from the University of Bochum, Bochum, Germany, in 1994 and 1999, respectively.

After two years as a Postdoctoral Fellow with the Computer Science Department, University of Rochester, Rochester, NY, USA, he joined the Faculty of the Cognitive Science Department, University of California at San Diego, La Jolla, CA, USA, in 2001, as an Assistant Professor. He also holds professorships with the Department of Physics and the Department of Computer Science and Mathematics, Goethe University, Frankfurt am Main, Germany. He obtained a visiting professorship with the Université Clermont Auvergne, Clermont–Ferrand, France, in 2019. He has been the Johanna Quandt Research Professor for Theoretical Life Sciences with Frankfurt Institute for Advanced Studies (FIAS), Frankfurt am Main, since 2007. His research interests span computational neuroscience, machine learning, and developmental robotics.

Dr. Triesch received the Marie Curie Excellence Center Award of the European Union in 2006. He became a Fellow of FIAS in 2005.

Gianluca Baldassarre received the B.A. and M.A. degrees in economics and the Diploma degree in the specialization course "cognitive psychology and neural networks" from the Sapienza University of Rome, Rome, Italy, in 1998 and 1999, respectively, and the Ph.D. degree in computer science from the University of Essex, Colchester, U.K., in 2003.

He was later a Postdoctoral Fellow with the Italian Institute of Cognitive Sciences and Technologies, National Research Council (ISTC-CNR), Rome. Since 2006, he has been a Researcher, currently a Director of Research with ISTC-CNR where he founded and is the Coordinator of the Research Group "Laboratory of Embodied Natural and Artificial Intelligence." He was the Principal Investigator for the EU project "ICEA–Integrating Cognition Emotion and Autonomy" from 2006 to 2009, the Coordinator of the EU Integrated Project "IM-CLeVeR–Intrinsically Motivated Cumulative Learning Versatile Robots" from 2009 to 2013, and the Coordinator of the EU FET-OPEN Project "GOALRobots–Goal-Based Open-Ended Autonomous Learning Robots" from 2016 to 2021. His research interests are on open-ended learning of sensorimotor skills, driven by extrinsic and intrinsic motivations, and goal-directed and habitual behavior, in animals, humans, and robots. He investigates these topics through machine learning and bioinspired autonomous robotics systems, and with computational models of brain and behavior.

Open Access funding provided by 'Consiglio Nazionale delle Ricerche-CARI-CARE-ITALY' within the CRUI CARE Agreement