OPEn: An Open-ended Physics Environment for Learning Without a Task

Chuang Gan[1,2], Abhishek Bhandwaldar[2], Antonio Torralba[1], Joshua B. Tenenbaum[1], Phillip Isola[1]

Abstract— **Humans have mental models that allow them to plan, experiment, and reason in the physical world. How should an intelligent agent go about learning such models? In this paper, we will study if models of the world learned in an open-ended physics environment, without any specific tasks, can be reused for downstream physics reasoning tasks. To this end, we build a benchmark Open-ended Physics Environment (OPEn) and also design several tasks to test learning representations in this environment explicitly. This setting reflects the conditions in which real agents (*i.e.* rolling robots) find themselves, where they may be placed in a new kind of environment and must adapt without any teacher to tell them how this environment works. This setting is challenging because it requires solving an exploration problem in addition to a model building and representation learning problem. We test several existing RL-based exploration methods on this benchmark and find that an agent using unsupervised contrastive learning for representation learning, and impact-driven learning for exploration, achieved the best results. However, all models still fall short in sample efficiency when transferring to the downstream tasks. We expect that OPEn will encourage the development of novel rolling robot agents that can build reusable mental models of the world that facilitate many tasks.**

I. Introduction

Reinforcement learning (RL) excels at specialist intelligence. When given a narrowly defined task, and a well-shaped reward function, training standard RL algorithms with a large amount of task-specific environment interactions can achieve impressive performance, yielding agents that beat humans at many video games [32] and board games [47]. Typically, however, the agent that masters one game, say Go, might have no idea how to make sense of a different game, such as Space Invaders. In contrast, human intelligence works differently. From infancy, humans have a mental model to infer the physical structure of the world, which is essential for making physical inferences about the world, predicting what will happen next, and planning actions.

Motivated by the age-old idea that model-building is critical to general intelligence, jointly learning a model and exploring the environment has received considerable recent attention [36], [20], [25], [46]. However, this challenge remains a mostly open research problem. Unlike past work that built models via either random exploration [20] or downstream task optimization [25], a more fundamental question is: how should an agent go about learning a useful model of the world when it is placed in an environment without an explicit task?

Although open-ended learning is a classic problem, most current RL environments are oriented toward learning to solve a specific task. For example, the clear goal in game environments like ALE [6] is to achieve a high score in the game. Although such environments can and have been used to study task-agnostic learning (e.g., [38]), doing so leaves the concern that the game task is so intimately baked into the environment that it inevitably shapes learning. More recently, research has extended toward environments that support *families* of tasks [54], such as random variations of physical parameters in robot simulations [16] or procedurally generated levels of a game [13], [41]. Nonetheless, these families are still typically focused on meta-task, such as “pick and place", “maze navigation," or “killing enemies in a platformer". We instead propose an environment that is designed from the ground up to be a “sandbox", where the implicit meta-task is simply to learn about how a physical world works.

To this end, we build an Open-ended Physics Environment (OPEn) to combine the idea of active learning, whereby a learner gets to pick the datapoints it trains on, and world modeling, where a learner attempts to model the dynamics of an environment. The design of this benchmark platform reflect two perspectives that we believed are essential for model building:

- **Active exploration.** Active exploration has the chance to be more efficient than passive learning since training data can be selected based on what would be most informative to the agent at any given moment. But more than this, passive learning is simply not an option in many settings. Passive learning relies on the agent being fed a data stream to learn from. Often this data stream is a highly instructive teaching signal, as is the case in so-called “unsupervised" learning from human-curated datasets like Imagenet [42]. In ecological settings, agents are not given Imagenet – they must explore the world to seek out useful data to learn from.
- **Physical interaction.** Interacting with an environment is a form of intervention that allows us to understand how their actions could affect the world and then build causal models of the world.

As shown in Figure 1, our environment includes a sandbox for learning, and a suite of evaluation metrics to probe what was learned. In the sandbox, the rolling robot agent is allowed to explore the world by interacting with objects. There are no pre-defined tasks or extrinsic rewards. The rolling robot must use its intrinsic motivations to actively explore the environment and build a model of its.

For evaluation, we propose an initial suite of tasks that may probe what was learned. On each task, the agent finetunes its policy using extrinsic rewards associated with that task.

[1] Massachusetts Institute of Technology, [2] MIT-IBM Watson AI Lab, Project page: http://open.csail.mit.edu/

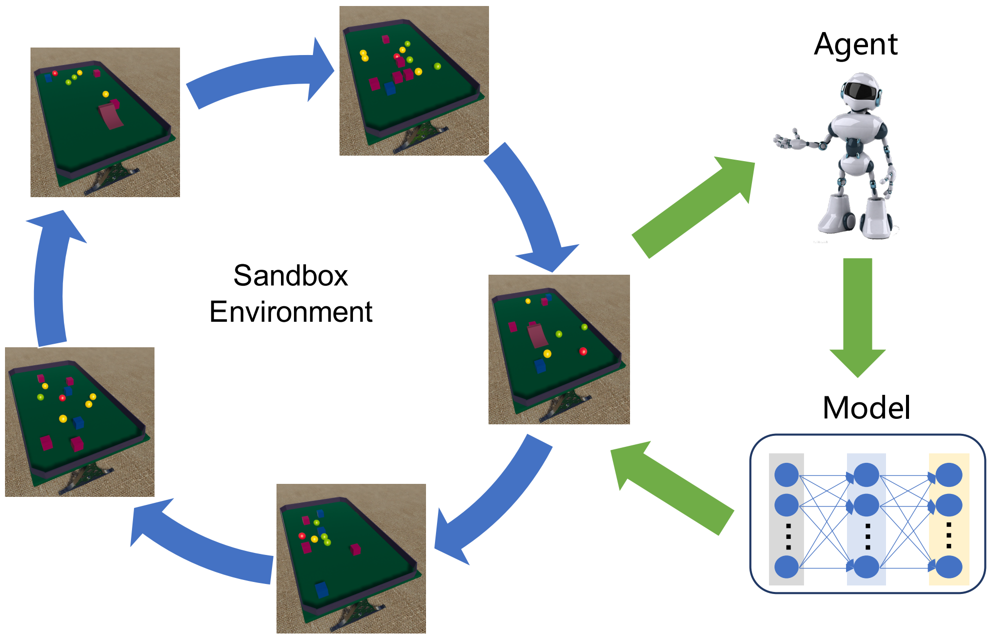

Fig. 1. Problem setup: a rolling robot agent is allowed to explore the sandbox-like 3D physical world by interacting with objects without any pre-defined tasks or extrinsic rewards. The green and blue loops indicate learning within an environment and swapping to a new environment, respectively.

The intent is not that these evaluation tasks become the learning targets themselves, but rather that they just reveal what was learned in the open-ended phase. As such, this suite should be expanded over time, so that users of the benchmark do not overfit to any particular set of evaluation tasks. Moreover, methods that use our benchmark should not be solely measured in terms of how well they do on the evaluation tasks, but also on whether or not they "cheated" by explicitly turning themselves toward the evaluation tasks. This may seem a delicate balance at first glance – how can we really tell that an algorithm was agnostic to the evaluation suite – but we note that this approach has lead to significant progress in unsupervised and self-supervised learning, where the goal is to develop *general-purpose* representations but the standard practice is, like ours, to evaluate by finetuning on a suite of *specific tasks*.

We test several existing RL-based agents on this benchmark by transferring their learned representations to the tasks in our evaluation suite and find that an agent using unsupervised contrastive representation learning for model building, and impact-driven exploration, achieved the best results. However, all models still fall short in sample efficiency when transferring their representations to the downstream tasks. To sum up, our work makes the following contributions:

- We introduce a new physical reasoning task, focusing on open-ended learning with active interactions and task-agnostic model-building.
- We build a 3D physical environment with fairly photo-realistic images and provide high-fidelity physical simulation to facilitate research on embodied intelligence in more challenging scenarios.
- We test several agents on this benchmark and find an agent that adopts impact-driven exploration, and contrastive unsupervised representations learning achieves the best results. All models fall short on sample efficiency.

Dataset	Realisic	3D	Interactive	Explicit task
Phyre [3]	×	×	✓	✓
IntPhy [40]	✓	✓	×	✓
CATER [19]	✓	✓	×	✓
CoPhy [4]	✓	✓	×	✓
CLEVRER [53]	✓	✓	×	✓
OPEn	✓	✓	✓	×

TABLE I
COMPARISON BETWEEN OPEN AND OTHER PHYSICS BENCHMARKS.

II. Related Work

Benchmarks for Physics Reasoning. Our work is in the domain of physical scene understanding [5], [1], [24], [23], [11], [18], [15]. Recently, several datasets have been developed for physics reasoning. Intphys[40] proposed a synthetic dataset for visual intuitive physics reasoning. CATER [19] introduced a video dataset for compositional temporal reasoning. COPHY [4] studied counterfactual physical dynamics prediction. CLEVRER [53] investigated casual reasoning in collision events and also grounding it to the language domain. Phyre [3] designed several puzzles to examine agents' ability to use tools. As summarized in Table I, our new benchmark goes beyond these existing datasets in two ways: 1) it is interactive, thereby supporting open-ended exploration (Phyre is also interactive, but 2D), 2) the learning phase is task-agnostic with no explicit goals, rewards, or supervision provided.

Open-ended Learning. Open-endedness is the problem of creating systems that develop ever-increasing abilities over time, rather than saturating when they manage to solve a narrowly defined task [28]. The evolution of life on Earth is the prototypical example of open-endedness in action. Human learning and culture also seem to exhibit open-endedness – we are ever striving toward greater understanding and control over our environment. Open-endedness has recently become a popular topic in the machine learning community, with the striking success of self-play algorithms like AlphaGo [48]. Many other multiagent environments have also demonstrated some level of open-ended, emergent behaviors (e.g., [49], [51], [2]). In the single-agent setting, research on open-ended learning has focused on the problem of intrinsic motivation. Rather than learning from a task or from external rewards, intrinsically motivated agents have to make up their own tasks and goals. Commonly, this is formalized as novelty search or curiosity (e.g., [43], [14], [36], [37], [21], [28], [29], [38], [9]). While most work in this area has focused on exploring a fixed environment, the idea has also been applied to procedurally generating a curriculum of environments [52], [10], [35]. Nonetheless, standard benchmarks for open-ended learning are lacking, which is what we attempt to address with this paper. In addition, there is little work that evaluates whether models and skills learned through open-ended exploration are useful for downstream tasks (an exception is [10]). We address this aspect via our suite of evaluation tasks.

Model-based RL Model-based RL has received considerable attention in recent years. Several works have shown that learning an internal model of the world can significantly improve data efficiency in RL [34], [12], [20], [33], [25]. For example, [34] uses video prediction to improve sample-efficiency on Atari games. [30] further predicts the reward of the environment to improve exploration. [20] proposed to train a generative model as a world model of the environment, and then use the learned features as inputs to train a simple policy to solve Vizdoom and 2D racing games. Most recently, several works have shown that contrastive learning [50] and data augmentation [26], [27] for representation learning can also be helpful for planning and control tasks. However, little work has been tested on open-ended exploration in a 3D physical world, which our work studies.

III. Environment

A. Platform

We build a 3D physical world on top of the TDW platform [17], which consists of a graphics engine, a physics engine, and an interaction API.

Graphic engine. TDW adopt Unity's game-engine technology to create a 3D virtual environment. The environment uses a combination of global illumination and high-resolution textures as well as the underlying game engine's capability to produce near-photo realistic rendering in real-time. We use directional lighting to simulate a sun-like light source and customize it by changing the angle and elevation of the source. We adopt a Unity camera object capable of giving multi-sensory data such as RGB images, segmentation masks, normal and depth maps. The camera object can be moved and rotated in 3D space.

3D Model. Our environment consists of a range of 3D objects, including primitives, shapes, and models with high-resolution colliders and materials. The model processing pipeline can generate object colliders using V-HACD[31], which gives fast and accurate collision behavior. It also allows users to import their custom models and customize their visual appearance, rigid body dynamics, object-to-object dynamics, and limited object-to-fluid dynamics. We use a 3D table model with the top surface bounded by re-scaled cube walls as the base of our scene. The object primitives in our scene consist of spheres, cubes, and ramps.

Physic simulation. We utilize Nvidia's Physx physics engine for simulating rigid body dynamics. We can set different physical parameters at an object level, which directly affects the object's interaction with the environment. We use a combination of force and drag in our scene, which is applied to the object's center of mass, to move it in a controlled manner. Our platform also supports pausing and stepping of the environment by a specific number of physics steps.

Interaction API. To support interaction with the environment, we provide a python API to send and receive commands and data from the Unity environment. The API is flexible enough to give users a fine-grain control over creating a scene, ranging from defining the scene layout to customizing an object's physical and visual properties. Therefore, the users could procedurally generate scene layouts based on their specified configuration. The API also helps control the simulation by allowing a series of commands to manipulate the object and advance physics in the scene. Users can also define the type of data the environment sends and includes visual information like images and segmentation masks, state data like object location, velocity, and interaction data like collisions. We integrate this API with the OpenAI gym [8] to facilitate training reinforcement learning algorithms.

B. Problem Setup

Scenario. Our scene is composed of a table with its top bounded by wall to prevent objects from falling off. The objects contained in the scene are heavy cubes (blue), light cubes (purple), three types of the sphere that act as an agent (red), goals with different rewards (yellow and green), and a ramp which can be used as a tool. The agent is capable of executing actions to move in one of the eight directions for a fixed distance and interacting with the objects in the scene by colliding with them.

Open-ended exploration. Our environment supports a sandbox mode that can procedurally generate puzzles with different objects placed in the scene. During open-ended exploration, we create several environments with distinctive, randomly generated configurations and select one at a time for exploration. The agent is not provided any explicit task or any extrinsic reward. The environment is non-episodic. In such an environment, the agent must learn to explore and probe different objects to learn reusable representations and models of the world. The learning algorithm is also free to switch between different environment on demand, allowing the study of learners that not only explore a single environment but

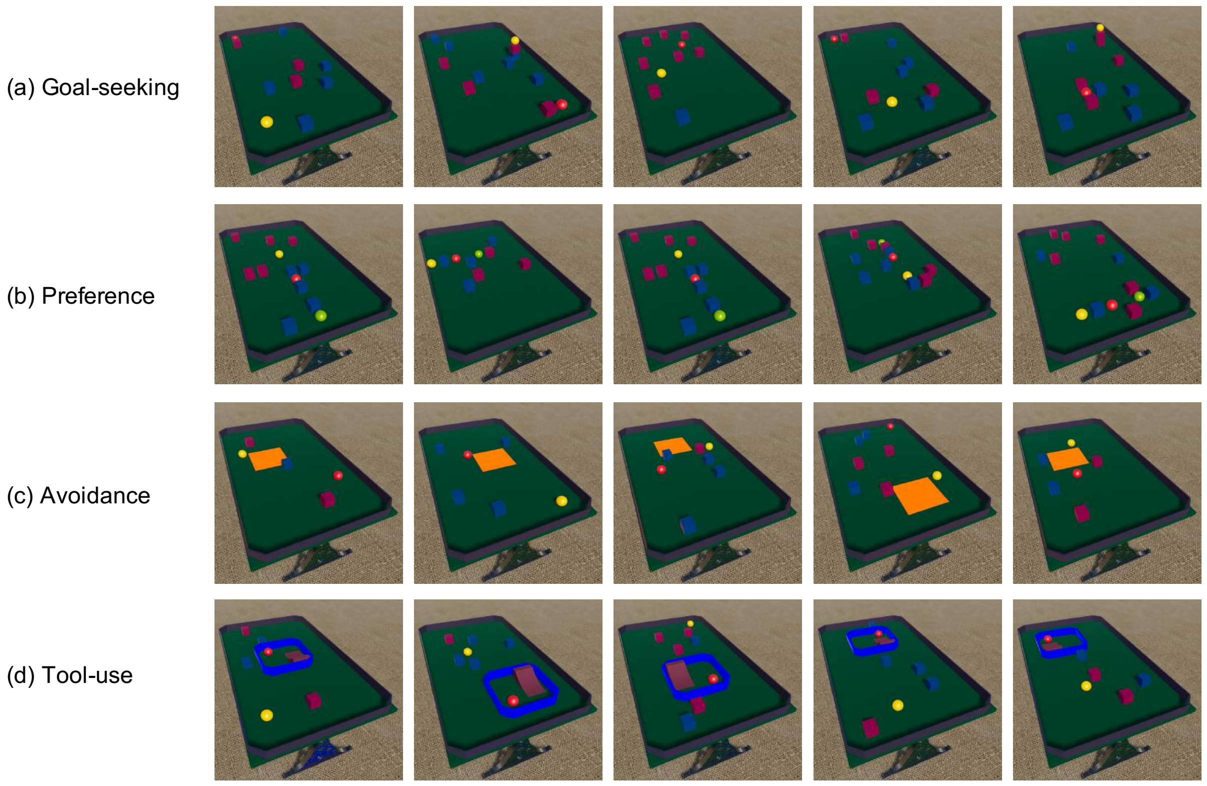

Fig. 2. Evaluation suite for 4 downstream physical reasoning tasks.

also can create curricula of multiple procedurally generated environments. In practice, in the methods we compare, we switch environments whenever the loss of the world model drops below a certain threshold. When this happens we switch to the environment with the highest loss among the current pool (a form of curiosity over environment configurations). Exploration terminates when no new environment gives the agent a loss higher than a threshold.

Evaluation suite. To evaluate what was learned by the model, we use a suite of intelligence tests inspired from [7], meant to probe the degree to which the agent has acquired general-purpose representations and skills.

- *Goal-seeking*. This category tests the agent's basic ability to understand goals and perform action towards them. As shown in figure 2 (a), there is one yellow sphere and several obstacle cubes on a table. The agent is required to hit the yellow sphere as fast as possible.
- *Preferences*. This category tests an agent's ability to choose the most rewarding course of action. As shown in figure 2 (b), the agent is presented with two spheres (i.e., yellow and green) and several obstacle cubes in the scene. Hitting the green sphere will earn more reward than hitting the yellow sphere. The agent is required to earn as much reward as possible with a limited interaction budget.
- *Avoidance*. This category identifies an agent's ability to detect and avoid negative stimuli. As shown in figure 2 (c), the goal is still to hit the yellow sphere, but there is an orange region on the table. The agent will immediately die if it enters this region. To achieve success in this task, the agent needs to understand the goal and the cost.
- *Tool-use*. This category tests the agent's ability on a more challenging task: using tools to achieve goals. As shown in figure 2 (d), the agent is trapped in a fenced region and has to use the ramp to get out and hit the yellow sphere. The agent is required to complete the task as fast as possible.

IV. Experiments

A. Compared Methods

We implement seven agents that use intrinsic motivation for exploration and model learning, and transfer their learned representations to the tasks in our evaluation suite.

ICM. Intrinsic Curiosity Module (ICM) [38] is a curiosity-driven exploration strategy. It adopts the prediction errors of a forward and inverse dynamics model as an intrinsic reward, which encourage the agent to take actions that improve its ability to predict the consequence of its actions.

RND. Random Network Distillation (RND) [9] is a counts-based exploration algorithm. An agent learns a model that predicts the feature representations of its current state extracted from a fixed randomly initialized network. The agent is encouraged to visit more unseen states and improve its coverage.

ICM + RIDE. Rewarding Impact-driven exploration (RIDE) [39] defines a novel intrinsic reward that encourages an agent to take actions that can change the representation of the environment state. Instead of directly using the prediction error of the dynamics model as an intrinsic reward, they propose an impact-driven bonus measured by the distances between the current and next state in latent feature space.

Approach	Goal-seeking	Preferences	Avoidance	Tool-use
PPO (From scratch)	$0.509_{\pm 0.009}$	$0.351_{\pm 0.02}$	$0.185_{\pm 0.01}$	$0.113_{\pm 0.08}$
ICM	$0.508_{\pm 0.016}$	$0.344_{\pm 0.01}$	$0.176_{\pm 0.023}$	$0.118_{\pm 0.013}$
ICM + RIDE	$0.486_{\pm 0.005}$	$0.367_{\pm 0.011}$	$0.178_{\pm 0.019}$	$0.120_{\pm 0.012}$
RND	$0.470_{\pm 0.01}$	$0.266_{\pm 0.02}$	$0.168_{\pm 0.01}$	$0.106_{\pm 0.01}$
RND + RIDE	$0.509_{\pm 0.014}$	$0.372_{\pm 0.0129}$	$0.179_{\pm 0.018}$	$0.116_{\pm 0.008}$
CURL + Random	$0.149_{\pm 0.02}$	$0.007_{\pm 0.05}$	$0.103_{\pm 0.01}$	$0.045_{\pm 0.01}$
CURL+ RIDE	$\mathbf{0.532}_{\pm 0.006}$	$\mathbf{0.386}_{\pm 0.04}$	$\mathbf{0.191}_{\pm 0.02}$	$\mathbf{0.125}_{\pm 0.02}$
Human	$0.584_{\pm 0.008}$	$0.588_{\pm 0.015}$	$0.453_{\pm 0.025}$	$0.363_{\pm 0.023}$

TABLE II
A-SUCCESS SCORES OF DIFFERENT METHOD ON DIFFERENT TASKS.

Larger state changes lead to higher rewards. We use ICM for model building and impact-driven rewards for exploration.
RND + RIDE. We use RND for model building and RIDE for exploration.
CURL + RIDE. CURL [50] adopts contrastive unsupervised representations learning [22] for model-building and shows promising results on off-policy control on top of the extracted features for many plannning and control tasks. We adapt this baseline to our problem. In particular, we use CURL for model-building and RIDE for exploration.
CURL + Random. We use data collected from a random policy as input to CURL.
Plain PPO. PPO [44] is a SOTA policy gradient method for model-free RL. We train it from scratch.

B. Experiment Setup

Implementation Details. In all experiments, the agent only takes visual observations as input. The input of the network is an image of 84×84 size. We set frame rate skip as S=4 for all the tasks. To train our RL agent, we use PPO [45] based on the PyTorch implementation. We use a three-layer convolutional network to encode the image observations. During the open-ended exploration phase, we share the weights of the policy and model encoders. We find this strategy is important for a successful transfer to the downstream tasks. During the evaluation phase, we use the weights of the pre-trained encoder as initialization and then fine-tuned the network to learn a task-specific policy with extrinsic rewards provided by the task. For open-ended exploration, we set the initial learning rate as 1e-3, the environment switching threshold as 0.001, and the maximum interaction budget as 2 million. For fine-tuning, we set learning rate as 2.5e-4. All evaluation tasks use 1 million interactions with extrinsic rewards for policy learning.
Reward function. We use different reward functions to train RL policies for each evaluation task. In the goal-seeking and tool-use tasks, we reward the agent with 1 if it hits the yellow ball, and otherwise 0. In the avoidance task, we only reward the agent with 1 if it hits the yellow sphere without entering the danger region. For the preference task, the agent receives a reward of 0.8 if it hits only the green ball, a reward of 0.2 if it hits only the yellow ball, and a reward of 1 if it hits both of them.
Evaluation metrics. We test all the agents on the same set of 100 randomly generated puzzles for each task. To quantify the performance of different agents, we adopt two metrics: mean return and A-Success score [3] (i.e., mean returns weighted by number of actions taken).

- *Mean return* is defined as average returns over all 100 puzzles, given a task and budget of actions taken. For tasks of goal-seeking, avoidance and tool-use, the return is either 0 or 1. For preference task, the return for one episode could be 0, 0.2, 0.8 or 1.
- *A-Success* [3] is a metric that jointly considers the mean returns and the number of actions taken. We use a weighted average of the returns under different action budgets to reflect how efficiently an agent solves a given task during the testing phase. Following [2], we formulate it as

$$\texttt{A-Success} = \sum_{i}^{N} \frac{\alpha_i * s_i}{\sum_{j=1}^{N} \alpha_j} \ , \qquad (1)$$
$$\alpha_i = \log(i+1) - \log(i) \ , \ i \in \{1, 2, ..., N\}$$

where s_i is the mean return (that is the average return across all the testing puzzles for each task) using i actions, α_i is a weight for s_i, and N denotes the maximum of action steps we consider. In this paper, we set $N = 100$ for task 1, 2, 3, and $N = 200$ for task 4. Since the weight α_i will decrease as the number of actions i increases, solving a given task with fewer actions will lead to a higher A-success score.

C. Overall Performance

For all the methods, we experiment with three different random seeds. Table II summarizes the results measured by the A-success score for the different agents. From this table, we find that learning models without a task indeed can help downstream tasks, if we can design the algorithm appropriately. For example, CURL+RIDE outperform the baseline plain PPO for all the four downstream tasks.

We also plot the mean return curves for all seven agents on four downstream tasks in Figure 3 by averaging 3 runs. These curves show the mean return as a function of the number of

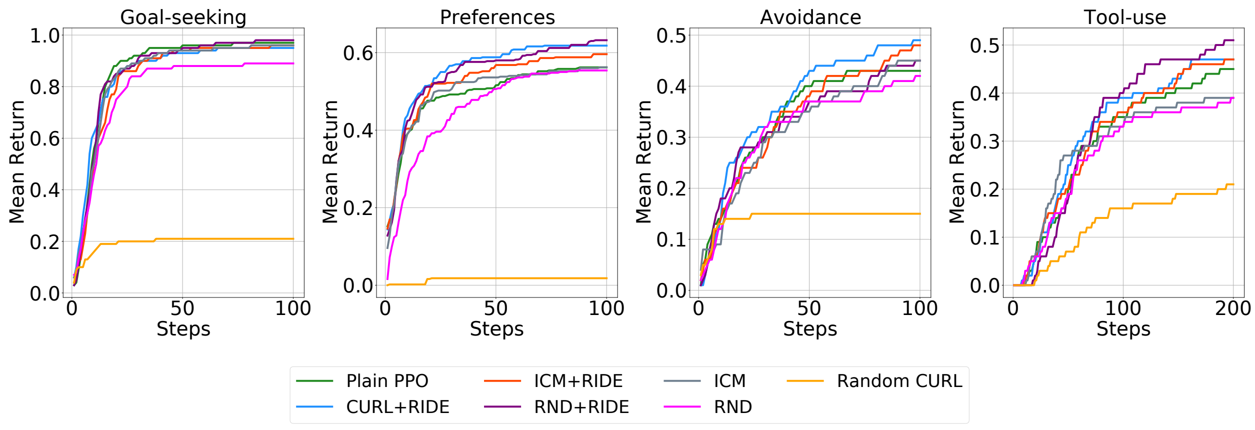

Fig. 3. Mean return of different models as a function of number of interactions with the environment during a single test episode.

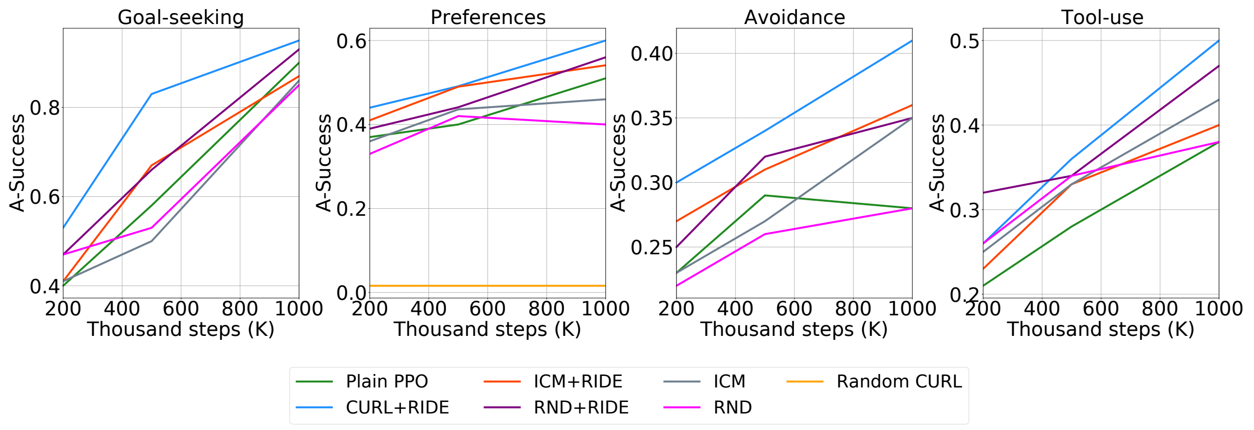

Fig. 4. A-success score of different models using 200K, 500K, and 1M interaction steps over the course of fine-tuning.

actions performed for each given task. We calculate the return by averaging over all 100 testing puzzles in OPEn. From these curves, we can see that the design of OPEn contains a diverse set of tasks of various difficulty levels. For instance, it is relatively easy to solve the goal-seeking task within 100 interactions. However, in the challenging tool use task, which involves complex cognitive behaviors, i.e. using the ramp to go out of the fenced region, the mean return at 200 interactions is still not very high.

Human Evaluation. Two authors tested their performance on same set of 100 puzzles for each task. Their average score is summarized at the bottom of Table II. We find that when using 1 million interactions for fine-tuning, the best model can achieve human-level performance on the basic goal-seeking task. However, for the other tasks, the performances of all RL models are significantly worse than the performance of the authors. These results indicate that current RL algorithms still struggle on physics reasoning tasks that require advanced cognitive skills.

D. Results Analysis

Are active explorations important for model-building? In Table II, we find that an agent (CURL+Random), which takes random actions to collect data, does not learn an effective representation for downstream tasks. The CURL+Random agent is even significantly worse when transferring the learned representation to the downstream tasks than the plain PPO model that is trained from scratch on the downstream tasks. These results provide validation for our claim that active exploration plays a vital role in learning models and representations that could be reused.

Sample efficiency for model-adaptation. We also examine the sample efficiency for model adaptation for different models. In figure 4, we plot the curve of A-Success score as a function of the number of fine-tuning interactions with the environment. In particular, we evaluate different models trained with 200K, 500K, and 1 million interaction steps with the environment. We can find that most of the models do not learn meaningful skills using 200K step interactions. These results indicate that the representation learned with current model-building approaches are not highly sample efficient.

V. Conclusions and Discussions

We have introduced a new benchmark for open-ended learning in a physics environment. We examined several RL-based exploration methods and found a model learned through impact-driven exploration and unsupervised contrastive representation learning transfers better to the downstream tasks compared to alternatives. However, we also found that no agents could achieve human-level generalization and fast adaptation. We hope this benchmark can enable the development of model-building in open environments, and make progress toward agents that can learn general-purpose world knowledge that can be used for many tasks.

Acknowledgement: This work is in part supported by ONR MURI N00014-16-1-2007, the Center for Brain, Minds, and Machines (CBMM, NSF STC award CCF-1231216), and IBM Research.

References

[1] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in neural information processing systems*, pages 5074–5082, 2016.

[2] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019.

[3] Anton Bakhtin, Laurens van der Maaten, Justin Johnson, Laura Gustafson, and Ross Girshick. Phyre: A new benchmark for physical reasoning. In *NIPS*, pages 5083–5094, 2019.

[4] Fabien Baradel, Natalia Neverova, Julien Mille, Greg Mori, and Christian Wolf. Cophy: Counterfactual learning of physical dynamics. *ICLR*, 202-.

[5] Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332, 2013.

[6] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[7] Benjamin Beyret, José Hernández-Orallo, Lucy Cheke, Marta Halina, Murray Shanahan, and Matthew Crosby. The animal-ai environment: Training and testing animal-like artificial cognition. *arXiv preprint arXiv:1909.07483*, 2019.

[8] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[9] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *ICLR*, 2018.

[10] Emilio Cartoni, Francesco Mannella, Vieri Giuliano Santucci, Jochen Triesch, Elmar Rueckert, and Gianluca Baldassarre. Robot open-ended autonomous learning competition. 2020.

[11] Zhenfang Chen, Jiayuan Mao, Jiajun Wu, Kwan-Yee Kenneth Wong, Joshua B Tenenbaum, and Chuang Gan. Grounding physical concepts of objects and events through dynamic visual reasoning. *arXiv preprint arXiv:2103.16564*, 2021.

[12] Silvia Chiappa, Sébastien Racaniere, Daan Wierstra, and Shakir Mohamed. Recurrent environment simulators. *arXiv preprint arXiv:1704.02254*, 2017.

[13] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588*, 2019.

[14] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

[15] Kiana Ehsani, Shubham Tulsiani, Saurabh Gupta, Ali Farhadi, and Abhinav Gupta. Use the force, luke! learning to predict physical forces by simulating effects. *CVPR*, 2020.

[16] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[17] Chuang Gan, Jeremy Schwartz, Seth Alter, Martin Schrimpf, James Traer, Julian De Freitas, Jonas Kubilius, Abhishek Bhandwaldar, Nick Haber, Megumi Sano, et al. Threedworld: A platform for interactive multi-modal physical simulation. *NeurIPS*, 2021.

[18] Chuang Gan, Siyuan Zhou, Jeremy Schwartz, Seth Alter, Abhishek Bhandwaldar, Dan Gutfreund, Daniel LK Yamins, James J DiCarlo, Josh McDermott, Antonio Torralba, et al. The threedworld transport challenge: A visually guided task-and-motion planning benchmark for physically realistic embodied ai. *arXiv preprint arXiv:2103.14025*, 2021.

[19] Rohit Girdhar and Deva Ramanan. Cater: A diagnostic dataset for compositional actions and temporal reasoning. *ICLR*, 2020.

[20] David Ha and Jürgen Schmidhuber. World models. *NIPS*, 2018.

[21] Nick Haber, Damian Mrowca, Stephanie Wang, Li F Fei-Fei, and Daniel L Yamins. Learning to play with intrinsically-motivated, self-aware agents. In *Advances in Neural Information Processing Systems*, pages 8388–8399, 2018.

[22] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *CVPR*, 2020.

[23] Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B Tenenbaum, and Chuang Gan. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. *ICLR*, 2021.

[24] Carlo Innamorati, Bryan Russell, Danny M Kaufman, and Niloy J Mitra. Neural re-simulation for generating bounces in single images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8719–8728, 2019.

[25] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

[26] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[27] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020.

[28] Joel Lehman and Kenneth O Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336, 2008.

[29] Joel Lehman and Kenneth O Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011.

[30] Felix Leibfried, Nate Kushman, and Katja Hofmann. A deep learning approach for joint video frame and reward prediction in atari games. *arXiv preprint arXiv:1611.07078*, 2016.

[31] Khaled Mamou and Faouzi Ghorbel. A simple and efficient approach for 3d mesh approximate convex decomposition. In *ICIP*, pages 3501–3504, 2009.

[32] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[33] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *ICRA*, pages 7559–7566, 2018.

[34] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in neural information processing systems*, pages 2863–2871, 2015.

[35] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand. *arXiv preprint*, 2019.

[36] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

[37] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

[38] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

[39] Roberta Raileanu and Tim Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. *ICLR*, 2020.

[40] Ronan Riochet, Mario Ynocente Castro, Mathieu Bernard, Adam Lerer, Rob Fergus, Véronique Izard, and Emmanuel Dupoux. Intphys: A framework and benchmark for visual intuitive physics reasoning. *arXiv preprint arXiv:1803.07616*, 2018.

[41] Sebastian Risi and Julian Togelius. Procedural content generation: from automatically generating game levels to increasing generality in machine learning. *arXiv preprint arXiv:1911.13071*, 2019.

[42] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla,

Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[43] Jürgen Schmidhuber. Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science*, 18(2):173–187, 2006.

[44] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[45] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[46] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *ICML*, 2020.

[47] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[48] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

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

[50] Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

[51] Joseph Suarez, Yilun Du, Phillip Isola, and Igor Mordatch. Neural mmo: A massively multiagent game environment for training and evaluating intelligent agents. 2019.

[52] Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019.

[53] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. Clevrer: Collision events for video representation and reasoning. *ICLR*, 2020.

[54] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100, 2020.