PHYRE: A New Benchmark for Physical Reasoning

Anton Bakhtin **Laurens van der Maaten** **Justin Johnson**
Laura Gustafson **Ross Girshick**
Facebook AI Research
{yolo,lvdmaaten,jcjohns,lgustafson,rbg}@fb.com

Abstract

Understanding and reasoning about physics is an important ability of intelligent agents. We develop the PHYRE benchmark for physical reasoning that contains a set of simple classical mechanics puzzles in a 2D physical environment. The benchmark is designed to encourage the development of learning algorithms that are sample-efficient and generalize well across puzzles. We test several modern learning algorithms on PHYRE and find that these algorithms fall short in solving the puzzles efficiently. We expect that PHYRE will encourage the development of novel sample-efficient agents that learn efficient but useful models of physics. For code and to play PHYRE for yourself, please visit `https://player.phyre.ai`.

1 Introduction

Understanding and reasoning about physics is a hallmark of intelligence [9]. Humans can make sense of novel physical situations by reasoning about abstract concepts like gravity, mass, inertia, and friction. For this reason, testing the ability to solve novel physics puzzles has been used to measure the reasoning abilities of human children [7, 53] as well as non-human animals such as capuchin monkeys [52, 54], chimpanzees [40], crows [19, 48], finches [49], and rooks [5]. A key aspect of physical intelligence is *generalization*: after learning to solve a physics puzzle, an intelligent agent should be able to generalize that knowledge and quickly solve related tasks. Robust generalization may set humans apart from other species—prior research showed that four species of non-human primates can learn to solve novel physics puzzles, but struggle to generalize to related tasks [30].

We want to develop artificial systems that can reason and generalize about physics as well as people. However, we hypothesize that in the realm of physical reasoning, present-day machine learning methods will struggle to quickly solve new puzzles. We anticipate that more effective methods may involve fundamental improvements to sample-efficient learning and the ability to learn computationally efficient but useful models of physics.

Towards this goal, we have developed the PHYRE (PHYsical REasoning) benchmark. PHYRE provides a set of physics puzzles in a simulated 2D world. Each puzzle has a goal state (*e.g.*, *"make the green ball touch the blue wall"*) and an initial state in which the goal is not satisfied; see Figure 1. A puzzle can be solved by placing one or more new bodies in the environment such that when the physical simulation is run the goal is satisfied. An agent playing this game must solve previously unseen puzzles in as few attempts as possible. PHYRE was designed to satisfy three main goals:

- **Focus on physical reasoning:** Tasks are as simple as possible but still require nontrivial physical reasoning. Scenes are built only from balls and rectangular bars. Dynamics are deterministic, with only collision, gravity, and friction. Goals are symbolic, so natural language is not required.
- **Focus on generalization:** After training on one set of tasks, we should expect an effective agent to solve new, previously unseen puzzles. The benchmark is structured such that puzzles are split into training tasks and evaluation tasks, and involves two different degrees of generalization.
- **Focus on sample-efficiency:** Our evaluation rewards solving tasks with as few attempts as possible. Methods that master a task only after thousands of attempts will not perform well.

Preprint. Under review.

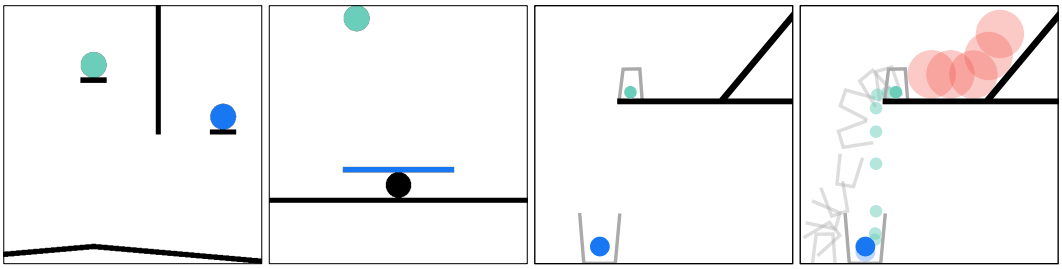


Figure 1: Three examples of PHYRE tasks (left) and one example solution (right). Black objects are static; objects with any other color are dynamic and subject to gravity. The tasks describe a terminal goal state that can be achieved by placing additional object(s) in the world and running the simulator. The task in the left-most pane requires placement of two balls to be solved, whereas the others can be solved with one ball. The right-most pane illustrates a solution (red ball) and the solution dynamics.

Figure 1 shows three examples of PHYRE tasks. Each task comprises static and dynamic objects in a 2D environment and a goal description. Upon looking at these examples, you, the reader, will likely form an intuitive hypothesis for how to solve each problem. If your first attempt were to fail, you would likely be able to use your observations of what happened to refine your attempt into a successful solution. PHYRE encourages the development of learning agents with similar abilities.

2 Related Work

PHYRE is related to prior work on intuitive physics, visual reasoning, and learning in computer games and simulated (robotics) environments. It was developed concurrently with the Tools game [1].

Intuitive physics. Foundational work in cognitive psychology suggests that people reason about the physical world using simplified intuitive theories [31–33]. Early computational instantiations of this framework used probabilistic models over physical simulations [3, 61], while more recent methods use feedforward neural networks trained to make pixelwise [11, 24, 34, 60] or qualitative [12, 26, 27, 34, 61] predictions about the future, sometimes in conjuction with simulation [18, 57, 58]. Many methods are evaluated on the constrained task of predicting whether a 3D stack of blocks will topple; some recent studies instead ask models to determine whether videos of more complex scenes are physically plausible [39, 41]. In contrast, PHYRE provides a suite of goal-driven tasks to test intuitive physical understanding: rather than evaluating intermediate tasks like future prediction or stability modeling, PHYRE requires agents to intervene in the scene to achieve a desired end goal.

Visual reasoning. Work on visual reasoning dates to SHRDLU [56] which probed scene understanding using natural language; more recent benchmarks require systems to answer natural-language questions about images [2, 20]. Recent methods use neural networks to extract sub-symbolic image representations, which are used in subsequent reasoning modules [16, 17, 21, 29, 37, 43]. Like PHYRE, these tasks require reasoning about the interactions of multiple objects in a scene; however unlike PHYRE they assume a static world, and do not require reasoning about world dynamics.

Learning in computer games. Computer games often involve complex 2D and 3D environments and require agents to possess some level of physical understanding [35, 46]. For instance, Atari games such as Pong involve precise positioning of a paddle based on observed ball dynamics [35]. The main difference between prior work in computer games and our work is that PHYRE requires the agent to learn a single model to solve a wide range of different tasks rather than a specialized model for each task. Moreover, in contrast to most work in computer games, PHYRE requires the agent to learn in a sample-efficient manner, penalizing agents that require many samples to learn.

Learning in simulated (robotics) environments. A range of prior work studies learning in simulated (robotics) environments for self-driving cars [10], humanoid robotics [50], or navigation tasks [44, 59]. In contrast to PHYRE, these prior studies focus on agents operating in realistic non-deterministic 3D environments in which the world is not fully observed, which hampers systematic study of reasoning capabilities of the agent. By contrast, PHYRE takes inspiration from CLEVR [20] and limits the complexity of the environment, facilitating more systematic analysis of reasoning abilities.

3 PHYRE Physical Reasoning Benchmark

The PHYRE environment is a two-dimensional world that simulates simple deterministic Newtonian physics. There is a constant downward gravitational force and a small amount of friction. All *bodies* are non-deformable and are either *static* or *dynamic*, distinguished by color. Static bodies remain at a fixed position and are not influenced by gravity or collision, while dynamic bodies move in response to these forces. All bodies come from a small *vocabulary*[1], varying in scale, location, and orientation.

A PHYRE *task* consists of an *initial world state* and a *goal*. The initial world state is a pre-defined configuration of bodies. The goal is a (subject, relation, object) triplet identifying a *relationship* between two bodies that the agent needs to achieve when the simulation terminates. At present all tasks use a single relation, `touching` for at least 3 seconds, which we found sufficient for developing a diverse set of tasks. The environment can be extended in the future to include additional relationships.

The agent aims to achieve the goal by taking a single *action*, placing one or more new dynamic bodies into the world. Bodies placed by the action may not extend beyond the world boundaries or intersect other bodies; such actions are rejected by the simulator as *invalid*. After the action is taken, the simulator runs until the goal is reached or until a time limit elapses, whichever happens first. The agent cannot perform additional actions while the simulator runs. Once the simulation is complete, the agent receives a binary *reward* indicating whether the goal was achieved, and gains access to observations of the intermediate world states produced by the simulator. If the goal was not achieved, the world resets to its initial state and the agent tries again, possibly informed by its prior attempts.

The full world state, comprising exact positions and orientations of bodies as well as their masses and velocities, is not revealed to agents since human observers cannot directly perceive such values from their environments. Instead, the agent receives coarser initial and intermediate world states as *observation images* which rasterize the world to a 256×256 grid. Each grid cell takes one of seven values specifying whether that location is a (1) dynamic goal object, (2) static goal subject, (3) dynamic goal subject, (4) static confounding body, (5) dynamic confounding body, (6) body placed by the agent, or (7) background. With only one relation, the colors in the initial observation encode the goal, eliminating the need for natural-language goal specification or grounding. Figure 1 shows three PHYRE tasks with goals written in natural language solely for the convenience of the reader.

Without any restrictions on the action space, for example on the body types, their properties, and the number of bodies that may be placed, the action space is large and complex. We therefore define two restricted action *tiers* for the current benchmark, which we describe next. After research progresses on these tiers, more complex ones may be added to the benchmark.

3.1 Benchmark Tiers

This work studies two benchmark tiers of increasing difficulty. A *tier* comprises a combination of: (1) a predefined set of all actions the agent is allowed perform and (2) a set of tasks that can be solved by at least one action from this action set. The two tiers we developed for this study are:

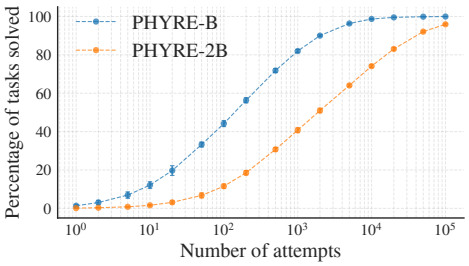
- **PHYRE-B.** Action set containing all valid locations and radii for a single ball (3D; continuous).
- **PHYRE-2B.** Action set containing all valid pairs of two balls (6D; continuous).

The two tiers each contain 25 task *templates*. A task template defines a set of related tasks that are generated by varying task template parameters (such as positions of initial world bodies). All tasks in the same template share a common goal, but have different initial world states. Each template defines 100 such tasks. Task templates are used to measure an agent's generalization ability in two settings. In the **within-template** setting, an agent trains on a subset of tasks in the template and is evaluated on the remaining tasks within that template. To measure **cross-template** generalization, test tasks are selected exclusively from templates that were not used for training. Our criteria for task design, additional analysis, and visualizations of tasks are provided in the supplement.

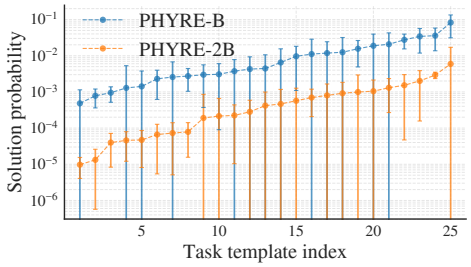
3.2 Learning Setting

Because the agent can only perform a single action to solve a PHYRE task, PHYRE is similar to a *contextual bandit* setting [23, 25]. PHYRE differs from traditional contextual bandit settings in two

[1]The current body vocabulary contains balls, bars, standing sticks, and jars.



(a) Percentage of tasks solved by a *random* agent (y-axis) as a function of the number of attempts (x-axis; log scale) for both PHYRE tiers.



(b) Probability (y-axis; log scale) that a *random* attempt solves each of the 25 task templates (x-axis) in a tier for both PHYRE tiers.

Figure 2: PHYRE complexity analysis. Values are averaged over 10 runs over all tasks in the tier; error bars indicate one standard deviation. Two-ball tasks are much harder to solve by chance than single ball tasks. Each tier contains a spectrum of task difficulty with respect to random guessing.

main ways: (1) it has an offline training phase that precedes the online learning testing phase and (2) the agent receives privileged information [51] in addition to the binary reward signal, *viz.*, it has access to observations of intermediate world states produced by the simulator on previous attempts.

In the **training phase**, the agent has access to the training tasks and unlimited access to the simulator. The agent does not have access to task solutions, but can use the simulator to train models that can solve tasks. Such models may include forward-prediction or action-prediction models.

In the **testing phase**, the agent receives test tasks that it needs to solve in as few *attempts* (queries to the simulator) as possible. After each attempt, the agent receives a binary reward and observations of intermediate world states. The agent can use this information to refine its action for the next attempt. Some actions may be invalid, *i.e.*, correspond to an object that overlaps with other objects. In such cases, we neither give the agent any reward nor count this attempt toward the query budget. The agent receives access to all test tasks at once, allowing it to choose the order in which it solves tasks.

Performance measure. We judge an agent's performance by how efficiently it solves tasks in the testing phase. We characterize efficiency in terms of the number of actions that were attempted to solve a given task; fewer attempts corresponds to greater efficiency. We formalize this intuition by recording the cumulative percentage of test tasks that were solved (the *success percentage*) as a function of the number of attempts taken per task. To compare the performance of agents on PHYRE, we plot this success-percentage curve. We also compute a performance measure, called **AUCCESS**, that aggregates the success percentages in the curve via a weighted average. To place more emphasis on solving tasks with fewer attempts, we consider the range of attempts $k \in \{1, \ldots, 100\}$ and use weights $w_k = \log(k+1) - \log(k)$, yielding AUCCESS $= \sum_k w_k \cdot s_k / \sum_k w_k$, where s_k is the success percentage at k attempts. The relative weight of the first 10 attempts in the AUCCESS measure is ~0.5: agents that need more than 10 attempts cannot get an AUCCESS score of more than 50%. This encourages the development of sample-efficient agents. AUCCESS is equivalent to the area under the success-percentage curve formed by replacing the discrete samples with a piecewise constant function and placing the number of attempts on a log scale.

3.3 Analysis

To assess the difficulty of the tasks in both PHYRE tiers, we measured what percentage of PHYRE tasks can be solved by an agent that randomly samples actions from the action space. Figure 2a shows the percentage of tasks (y-axis) that this random agent solves in at most k attempts (x-axis), averaged over 10 runs on all PHYRE tasks. The figure reveals that tasks vary greatly in difficulty level: a few tasks can be solved by a random agent in just a few attempts, whereas other tasks require thousands of attempts to be solved. The figure also shows that tasks in the PHYRE-B tier are, on average, harder than those in PHYRE-2B because the action space in that tier has more degrees of freedom.

We designed the PHYRE tasks such that, on average, it takes a random agent no more than 10,000 attempts to solve task in the PHYRE-B tier and no more than 100,000 attempts to solve a task in the PHYRE-2B tier. Figure 2b illustrates this by displaying the average probability that a random attempt

solves a task for each of the 25 task templates in both PHYRE tiers. In line with the previous analysis, the figure also shows that tasks in PHYRE-2B are substantially harder than those in PHYRE-B.

4 Experiments

We conduct experiments to obtain baseline results for within-template and cross-template generalization on the PHYRE benchmark. Experiments are performed separately on each tier. Code reproducing the results of our experiments is available from `https://phyre.ai`.

4.1 Baseline Agents

We experiment with five baseline agents that rank actions given an observation of the initial state (recall that the observation encodes the goal): (1) a random agent, (2) a non-parametric agent, (3) a deep Q-network [35], and (4-5) counterparts to (2) and (3) that update the agent online during testing.

Random agent (RAND). This agent does not perform any training and instead samples actions uniformly at random from the 3D or 6D (depending on the tier) action space at test time.

Non-parametric agent (MEM). At training time, this agent generates a set of R random actions and uses the simulator to check if each of these actions can solve each of the training tasks. For each action a, the agent computes p_a: the fraction of training tasks that the action solves. The agent then sorts the R actions by p_a (highest to lowest), and tries them in this order at test time. This agent is non-parametric because it uses a list of "memorized" actions at test time.

In the *cross-template setting*, the test tasks come from previously unseen task templates and this simple agent cannot relate them to tasks seen during training. It therefore uses the same action ranking for all tasks and ignores the observation of the initial state. In the *within-template setting*, each test task comes from a task template that was seen during training. In this case, we give the agent access to the task template id for each test task. The agent maintains a per-task-template ranking of the R actions. The same set of actions is shared across all templates; only the ranking changes. The set of actions attempted on each task may vary because invalid actions are ignored; see Section 3.2.

Non-parametric agent with online learning (MEM-O). This agent has the same training phase as the non-parametric agent, but continues to learn online at test time. Specifically, after finishing each test task (either successfully or unsuccessfully), the agent updates p_a based on the reward received for each action a in the subset of the actions it attempted. The updated ranking is used when the next task is attempted. Such online updates are beneficial, in particular, in the cross-template setting because they allow the agent to learn something about the tasks in the previously unseen templates. We use cross-validation to tune the relative weight of the update on each train-val fold (see Section 4.2).

Deep Q-network (DQN). As before, the DQN agent collects a set of observation-action-reward triplets by randomly sampling actions and running them through the simulator. The agent trains a deep network on the resulting data to predict the reward for an observation-action pair. Following [4], we train the network by minimizing the cross-entropy between the soft prediction and the observed reward. During training, we sample batches with an equal number of positive and negative triplets.

Our network comprises: (1) an *action encoder* that transforms the 3D or 6D (depending on the tier) action representation using a multi-layer perceptron with a single hidden layer; (2) an *observation encoder* that transforms the observation image into a hidden representation using a convolutional network (CNN); and (3) a *fusion module* that combines the action and observation representations and makes a prediction. Our action encoder is a MLP with a single hidden layer with 512 units and ReLU activations. Our observation encoder is a ResNet-18 [13]. For the fusion module, we follow [37] and use the action encoder to predict a bias and gain for each channel in the CNN. The output of the action encoder thus contains twice as many values as there are channels in the CNN at the fusion point. To expedite action ranking, we fuse both models before the last residual block of the CNN. We tried other fusion points but did not observe performance differences (see supplemental material).

The observation of the initial state is a 256×256 image with one of 7 colors at each pixel, which encodes properties of each body and the goal. We map this observation into a 7-channel image for input to the CNN; each colored pixel in the image yields a 7D one-hot vector. Following common practice, the network is trained end-to-end using stochastic gradient descent with the Adam optimizer [22]. We anneal the learning rate to 0 using a half cosine schedule without restarts [28].

Deep Q-network with online learning (DQN-O). Akin to MEM-O, this agent uses rewards from test tasks to perform online updates. After finishing a test task, the agent performs a number of gradient descent updates using examples obtained from that task. The updated model is then used for the next test task. The number of updates and corresponding learning rate are set via cross-validation.

Contextual bandits. While PHYRE is a contextual-bandit setting, we found that contextual bandits (CBs) do not work well on our complex observation and action space. Most CBs model the expected reward given the context and action using linear models [6, 8, 25], Gaussian processes [47], or deep neural networks [42]. Linear models do not yield useful context representations (which are observation images). Gaussian processes require a reasonable kernel function on the observation image space, which is difficult to define. Methods based on deep neural network seem more suitable. We tried to use the implementation from [42][2], but were unable to train the model once we replaced the shallow MLP used in [42] by a CNN that is better suited for image encoding. In addition, CBs generally assume a fixed (usually small) number of arms without a similarity metric between the arms, which is problematic for PHYRE tasks: when reasonably discretized, the number of arms in PHYRE-B is $\sim 10^6$. Moreover, without considering similarity between actions, agents try various non-working arms in the same region of the action space without diversifying them (see Section 4.4).

Policy learners. We faced similar issues with policy learners such as PPO [45] and A2C [36]. While we were able to factorize the action space over each dimension and use continuous action spaces, we were unable to train models that outperform our random baseline due to poor training stability.

4.2 Experimental Setup

We measure success percentage and AUCCESS on PHYRE using the learning setting of 3.2. To make results reproducible and allow fair comparisons between agents across studies, PHYRE provides:

- A fully deterministic environment: agents always produce the same result on a task.
- A process that deterministically splits the tasks into 10 *folds* containing a training, validation, and test set. As a result, agents are always compared on exactly the same task splits. Task splits are available for both tiers and both generalization settings (within-template and cross-template).

To avoid overfitting on test tasks, hyperparameter tuning is only to be performed based on the validation set: *we discourage tuning of hyperparameters based on test task performance.* For results on the test set, we use these tuned hyperparameter and train agents on the union of the training and validation sets. To compare agents, we use the non-parametric Wilcoxon signed-rank test for median difference [55] with one-sided null hypotheses and $p = 0.01$. We use this test as it does not have a normality assumption, is efficient with small sample sizes, and works with relative values on each fold instead of absolute values. To facilitate comparisons with our baselines, we provide our AUCCESS scores on all 10 folds in the supplementary material.

At test time, all agents (except the random agent) rank the same set of 10,000 actions on each task and propose the highest-scoring actions for that task as solution attempts. The MEM(-O) agents were trained on the same 10,000 actions. The DQN(-O) agents were trained on 100,000 actions per task.[3] All agents are permitted to make up to 100 attempts per task. This fact subtly implies that when computing the success percentage at $k < 100$ attempts, online agents will have learned from up to 100 (not k) attempts per task; this pragmatic choice makes the benchmark computationally tractable as otherwise online agents would need to be re-run for every value of $k \in \{1, \ldots, 100\}$.

4.3 Main Results

Figure 3 presents success-percentage curves for all five agents on both PHYRE tiers (-B and -2B) in both generalization settings (within-template and cross-template): the curves show the percentage of tasks solved as a function of the number of solution attempts per task, and are computed by averaging over all 10 folds in PHYRE. Table 1a presents the corresponding mean AUCCESS (and its standard deviation). The results are in line with the trends observed in Section 3.3: the within-template setting is much easier than the cross-template setting for all (non-random) agents. As forecasted, the two tiers also have different difficulty characteristics. In the cross-template setting, the best agent,

[2] https://github.com/tensorflow/models/tree/master/research/deep_contextual_bandits

[3] To simplify follow-up research, we will release the simulation results of these 100,000 actions per task.

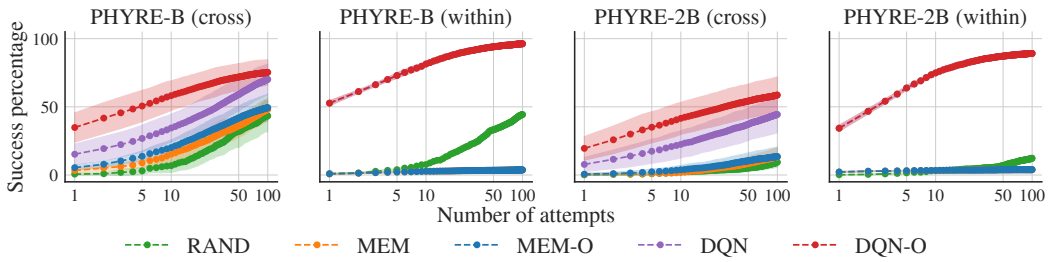


Figure 3: Percentage of solved tasks (success percentage) as a function of the number of attempts per task of five agents on PHYRE-{B, 2B} in the within-template and cross-template settings. Success percentages are averaged over all test tasks and 10 folds. Shaded regions show one standard deviation.

	PHYRE-B		PHYRE-2B	
	Cross	**Within**	**Cross**	**Within**
RAND	13.0±5.0	13.7±0.5	2.6±1.5	3.6±0.6
MEM	18.5±5.1	2.4±0.3	3.7±2.3	3.2±0.2
MEM-O	22.8±5.0	-	4.9±3.1	-
DQN	36.8±9.7	77.6±1.1*	23.2±9.1	67.8±1.5*
DQN-O	56.2±10.5*	-	39.6±11.1*	-

(a) Area under the success-percentage curve (AUCCESS) of five agents. Higher is better.

	PHYRE-B		PHYRE-2B	
	Cross	**Within**	**Cross**	**Within**
RAND	6.8±5.0	7.7±0.8	2.2±1.8	3.2±0.9
MEM	15.2±5.9	2.7±0.5	1.9±1.6	3.4±0.3
MEM-O	20.1±5.6	-	3.8±3.2	-
DQN	34.5±10.2	81.4±1.9	22.4±10.0	74.9±1.7
DQN-O	58.2±10.9	-	41.6±11.7	-

(b) Success percentage at $k = 10$ attempts of five agents. Higher is better.

Table 1: Comparison of the five agents on PHYRE-{B, 2B}. Mean and standard deviation on the 10 folds are reported. MEM-O and DQN-O perform best with no update in the within-template setting, making them equivalent to MEM and DQN in this case; thus, we omit their results. *Indicates an agent's AUCCESS is better than all others per the Wilcoxon one-sided test with $p = 0.01$.

DQN-O, is able to reach a reasonably high AUCCESS of 56.2% on PHYRE-B, but is at just 39.6% on PHYRE-2B. This small change in the action space substantially decreases agent success. Notably, agents that perform online learning (**⋆-O**) substantially outperform their offline counterparts.

In Table 1b, we present the percentage of tasks that were solved within 10 attempts by each agent. This low-attempt regime is emphasized by AUCCESS and a goal of the PHYRE benchmark is to encourage research that improves results in this regime. The results are in line with prior observations and illustrate that the PHYRE-2B cross-template setting presents a significant challenge for all agents.

4.4 Analysis

Here, we analyze the effect of: (1) the number of actions that are ranked by agents at test time and (2) the "aggressiveness" of agent updates on the performance of online agents. In the supplement, we also ablate the deep Q-network (DQN) design. For these experiments, agents are trained and evaluated on the train and validation splits, respectively, using the first three (out of 10) folds.

Number of actions ranked. Figure 4 shows the AUCCESS of the RAND, MEM, and DQN agents as a function of the number of actions that are ranked by the agents at test time. We also present an OPTIMAL ranking agent that performs oracle ranking of the action set. The performance of the OPTIMAL agent suggests that ranking is a reasonable strategy: it solves all tasks in PHYRE-B and 95% of tasks in PHYRE-2B by ranking fewer than 100,000 attempts. For non-oracle agents, DQN is a much better ranker than MEM. As expected, AUCCESS increases as more actions are ranked, but eventually plateaus and sometimes decreases beyond a certain number of attempts. This is due to a lack of diversity in the rankings produced by the agents, which do not have a model of similarity between actions and may suggest multiple similar attempts when sampling of actions is fine-grained.

Effect of online updates. Online agents use examples obtained during both the training and testing stages. Figure 5 analyzes the effect of re-weighting both types of examples on the performance of online agents (on three folds). The results show that the AUCCESS of MEM-O is fairly independent of the weight used. The AUCCESS of the DQN-O agent does vary as a function of how many updates were performed at test time: online updates even impede DQN-O in the within-template setting.

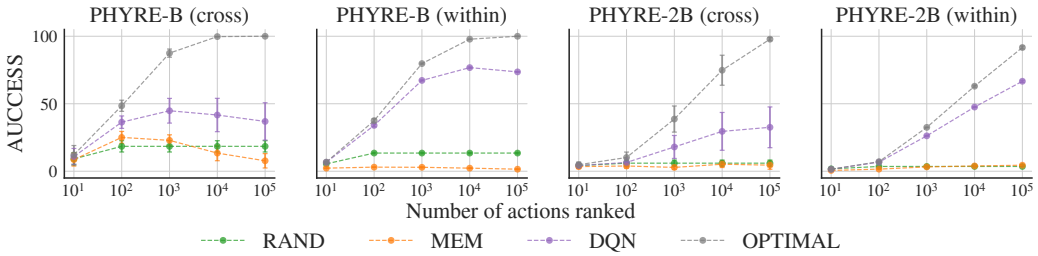


Figure 4: AUCCESS as a function of the number of actions being ranked by the agent for the RANDOM, MEM, and DQN agents and for an agent that is OPTIMAL in terms of scoring attempts.

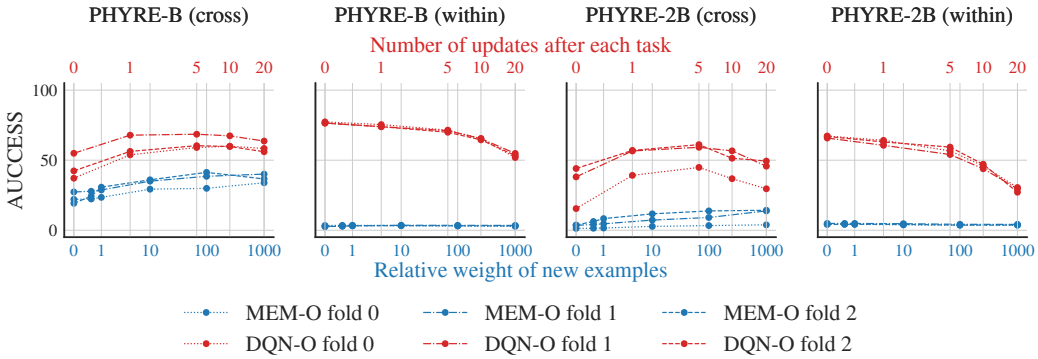


Figure 5: AUCCESS of MEM-O and DQN-O agents as the "aggressiveness" of the online update is varied during the testing phase. The left-most point in each plot is an offline version of the agent.

5 Discussion and Future Work

PHYRE aims to enable the development of physical reasoning algorithms with strong generalization properties mirroring those of humans [30]. Yet the baseline methods studied in this work are far from this goal, demonstrating limited generalization abilities. We foresee several areas for advancement:

- Agents should use intermediate observations from the simulator following an (unsuccessful) attempt to refine their next attempt. Our current failure to do so makes the agents sample-inefficient, as these observations contain rich information on the specific task that the agent is solving that should be used effectively for efficient problem-solving. Doing so requires *counterfactual reasoning*: agents need to reason about what would happen upon a particular change to a previous attempt.
- Agents should use a forward-prediction model that mimics the simulator by a learnable function [15]. Such a model can be integrated into a DQN by running attempts through it for a number of time steps, and using the resulting state predictions as additional inputs into the Q-network.
- Agents should explicitly diversify attempts when solving a task.
- Agents should use an active strategy at test time, *e.g.*, by starting with solving simple tasks.
- While each task is different from the others, they share the same underlying causal model (physics). Methods aimed at invariant causal prediction (ICP) [14, 38] may be well-suited for PHYRE.

Based on these observations, we expect to witness rapid progress on the PHYRE benchmark. To this point, we highlight that PHYRE is an extensible platform upon which more challenging puzzle tiers may be built. The two tiers provided in this initial benchmark are designed to be approachable, yet challenging. Future tiers may involve substantially larger and more complex action spaces.

We also foresee approaches that implement a simulator "internal" to the agent and then query it to brute-force a solution before submitting any attempts to the real simulator. Based on initial experiments, we expect that training a neural network to exactly mimic the simulator will be difficult. However, one might instead use hand-coded rules specific to PHYRE—in the extreme, one could simply call the real simulator inside the agent. We view such approaches as violating the spirit of the benchmark. We discourage this line of attack as well as in-between solutions that combine function approximation with extensive hand-coded inductive biases that are specific to PHYRE.

Acknowledgements

We thank Mayank Rana for his help with early versions of PHYRE, and Audrey Durand, Joelle Pineau, Arthur Szlam, Alessandro Lazaric, Devi Parikh, Dhruv Batra, and Tim Rocktäschel for helpful discussions.

References

[1] K. Allen, K. Smith, and J. Tenenbaum. The tools challenge: Rapid trial-and-error learning in physical problem solving. In *arXiv 1907.09620*, 2019.

[2] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh. VQA: Visual question answering. In *ICCV*, 2015.

[3] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum. Simulation as an engine of physical scene understanding. *PNAS*, 2013.

[4] M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 449–458. JMLR. org, 2017.

[5] C. D. Bird and N. J. Emery. Rooks use stones to raise the water level to reach a floating worm. *Current Biology*, 19(16):1410–1414, 2009.

[6] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In *NeurIPS*, 2011.

[7] L. G. Cheke, E. Loissel, and N. S. Clayton. How do children solve aesop's fable? *PloS one*, 7 (7):e40574, 2012.

[8] W. Chu, L. Li, L. Reyzin, and R. Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011.

[9] E. Davis. Physical reasoning. 2006.

[10] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. In *Proceedings of the Annual Conference on Robot Learning*, pages 1–16, 2017.

[11] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *NeurIPS*, 2016.

[12] O. Groth, F. B. Fuchs, I. Posner, and A. Vedaldi. Shapestacks: Learning vision-based physical intuition for generalised object stacking. In *ECCV*, 2018.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[14] C. Heinze-Deml, J. Peters, and N. Meinshausen. Invariant causal prediction for nonlinear models. *Journal of Causal Inference*, 6(2), 2018.

[15] M. Henaff, J. Zhao, and Y. LeCun. Prediction under uncertainty with error-encoding networks. In *arXiv 1711.04994*, 2017.

[16] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko. Learning to reason: End-to-end module networks for visual question answering. In *ICCV*, 2017.

[17] D. A. Hudson and C. D. Manning. Compositional attention networks for machine reasoning. In *ICLR*, 2018.

[18] M. Janner, S. Levine, W. T. Freeman, J. B. Tenenbaum, C. Finn, and J. Wu. Reasoning about physical interactions with object-oriented prediction and planning. In *ICLR*, 2019.

[19] S. A. Jelbert, A. H. Taylor, L. G. Cheke, N. S. Clayton, and R. D. Gray. Using the aesop's fable paradigm to investigate causal understanding of water displacement by new caledonian crows. *PloS one*, 9(3):e92895, 2014.

[20] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. Zitnick, and R. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017.

[21] J. Johnson, B. Hariharan, L. van der Maaten, J. Hoffman, L. Fei-Fei, C. Zitnick, and R. Girshick. Inferring and executing programs for visual reasoning. In *ICCV*, 2017.

[22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] J. Langford and T. Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. In *NeurIPS*, 2008.

[24] A. Lerer, S. Gross, and R. Fergus. Learning physical intuition of block towers by example. In *ICML*, 2016.

[25] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[26] W. Li, A. Leonardis, and M. Fritz. To fall or not to fall: A visual approach to physical stability prediction. In *arXiv:1604.00066*, 2016.

[27] W. Li, A. Leonardis, and M. Fritz. Visual stability prediction and its application to manipulation. In *ICRA*, 2017.

[28] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[29] J. Mao, C. Gan, P. Kohli, J. Tenenbaum, and J. Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *ICLR*, 2019.

[30] G. Martin-Ordas, J. Call, and F. Colmenares. Tubes, tables and traps: great apes solve two functionally equivalent trap tasks but show no evidence of transfer across tasks. *Animal cognition*, 11(3):423–430, 2008.

[31] M. McCloskey and D. Kohl. Naive physics: The curvilinear impetus principle and its role in interactions with moving objects. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 9(1):146, 1983.

[32] M. McCloskey, A. Caramazza, and B. Green. Curvilinear motion in the absence of external forces: Naive beliefs about the motion of objects. *Science*, 210(4474):1139–1141, 1980.

[33] M. McCloskey, A. Washburn, and L. Felch. Intuitive physics: the straight-down belief and its origin. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 9(4):636, 1983.

[34] M. Mirza, A. Courville, and Y. Bengio. Generalizable features from unsupervised learning. In *ICLR*, 2017.

[35] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[36] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

[37] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

[38] J. Peters, J. Mooij, D. Janzing, and B. Schölkopf. Causal discovery with continuous additive noise models. *Journal of Machine Learning Research*, 15:2009–2053, 2014.

[39] L. Piloto, A. Weinstein, A. Ahuja, M. Mirza, G. Wayne, D. Amos, C.-c. Hung, and M. Botvinick. Probing physics knowledge using tools from developmental psychology. *arXiv preprint arXiv:1804.01128*, 2018.

[40] D. Povinelli. *Folk physics for apes: The chimpanzee's theory of how the world works*. 2000.

[41] R. Riochet, M. Y. Castro, M. Bernard, A. Lerer, R. Fergus, V. Izard, and E. Dupoux. Intphys: A framework and benchmark for visual intuitive physics reasoning. In *arXiv 1803.07616*, 2018.

[42] C. Riquelme, G. Tucker, and J. Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127*, 2018.

[43] A. Santoro, D. Raposo, D. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems*, pages 4967–4976, 2017.

[44] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A platform for embodied ai research. *arXiv preprint arXiv:1904.01201*, 2019.

[45] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. In *arXiv 1707.06347*, 2017.

[46] M. Smith. Running the table: An AI for computer billiards. In *American Association for Artificial Intelligence*, 2006.

[47] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

[48] A. H. Taylor, G. R. Hunt, F. S. Medina, and R. D. Gray. Do new caledonian crows solve physical problems through causal reasoning? *Proceedings of the Royal Society B: Biological Sciences*, 276(1655):247–254, 2008.

[49] I. Teschke and S. Tebbich. Physical cognition and tool-use: performance of darwin's finches in the two-trap tube task. *Animal Cognition*, 14(4):555–563, 2011.

[50] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 2012.

[51] V. Vapnik and A. Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5-6):544–557, 2009.

[52] E. Visalberghi and L. Limongelli. Lack of comprehension of cause effect relations in tool-using capuchin monkeys (cebus apella). *Journal of Comparative Psychology*, 108(1):15, 1994.

[53] E. Visalberghi and L. Limongelli. Acting and understanding: Tool use revisited through the minds of capuchin monkeys. *Reaching into thought: The minds of the great apes*, pages 57–79, 1996.

[54] E. Visalberghi and L. Trinca. Tool use in capuchin monkeys: Distinguishing between performing and understanding. *Primates*, 30(4):511–521, 1989.

[55] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83, 1945.

[56] T. Winograd. Procedures as a representation for data in a computer program for understanding natural language. Technical report, MIT AI, 1971.

[57] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *NeurIPS*, 2015.

[58] J. Wu, E. Lu, P. Kohli, B. Freeman, and J. Tenenbaum. Learning to see physics via visual de-animation. In *NeurIPS*, 2017.

[59] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian. Building generalizable agents with a realistic and rich 3D environment. In *arXiv:1801.02209*, 2018.

[60] T. Xue, J. Wu, K. Bouman, and B. Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *NeurIPS*, 2016.

[61] R. Zhang, J. Wu, C. Zhang, W. Freeman, and J. Tenenbaum. A comparative evaluation of approximate probabilistic simulation and deep neural networks as accounts of human physical scene understanding. In *CogSci*, 2016.

A Ablation Study of Deep Q-Network (DQN)

Figure 6 shows the effect on AUCCESS of six modifications to our DQN agent. The modifications encompass changes to the architecture of the action encoder (*Act1024* and *Act1024×2*), the fusion mechanism (*FuseGlobal*, *FuseFirst*, and *FuseAll*), and the balancing of training batches (*NoBalancing*); see the figure caption for full details. We make four main observations:

- Class-balancing training batches is critical to the DQN agent's performance, particularly on the PHYRE-2B tier where only 0.3% of randomly chosen actions yield a positive example.
- Early fusion of action information into the ResNet-18 observation encoder does not help. Early fusion is also inefficient for action ranking: it prohibits caching of the observation encoder's output.
- Our default fusion method uses channel-wise bias and gain modulation immediately before the ResNet-18 `conv5` stage; applying this fusion the final globally pooled features, instead, substantially deteriorates AUCCESS.
- Larger action encoders can improve performance, but the gains are not consistent across settings.

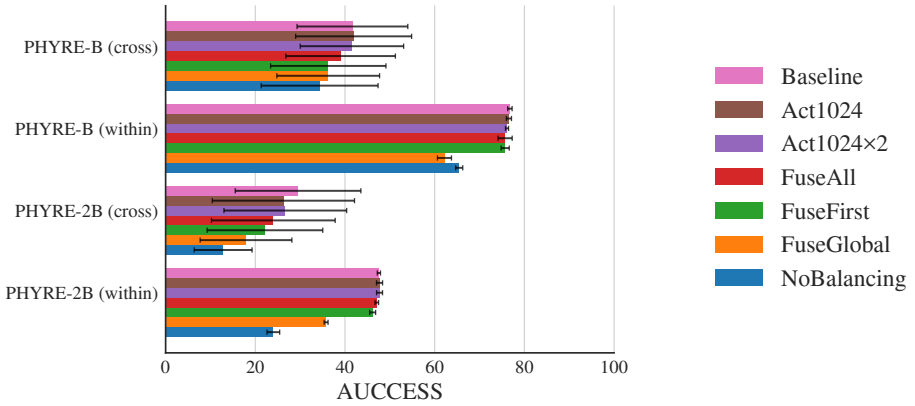


Figure 6: Mean AUCCESS on PHYRE-{B, 2B} of six DQN variants of the *Baseline* in the main text. Error bars show one standard deviation. *FuseFirst*, *FuseAll*, and *FuseGlobal* DQN agents perform fusion of observation and action features in alternative locations via channel-wise bias and gain modulation (akin to [37]): *Baseline* fuses with the input to the ResNet-18 `conv5` stage; *FuseFirst* fuses with the input to the `conv2` stage; *FuseAll* fuses with the inputs to each stage from `conv2` to `conv5`; and *FuseGlobal* fuses with the globally max-pooled output of the `conv5` stage. *Act1024* and *Act1024×2* DQN agents use *Baseline* fusion but larger action encoder networks with one or two hidden layers of 1024 units, respectively. The *NoBalancing* agents trains the *Baseline* DQN without balancing the positive and negative examples in the batches. We refer the reader to our code release on `https://phyre.ai` for full details.

B PHYRE Tasks

As discussed in the main paper, the current PHYRE benchmark provides two task *tiers*: PHYRE-B tasks can be solved by placing a single ball in the initial world, whereas PHYRE-2B tasks require placement of two balls in the initial scene. Each tier provides 25 *task templates*, and each task template contains 100 *tasks* that are similar in design but that have a different initial configuration of bodies in the world. Figure 7 shows an example task from each of the 25 task templates in the PHYRE-B tier, and Figure 8 shows an example task for each of the 25 task templates in the PHYRE-2B tier.

Stable solutions. When designing the PHYRE tasks, we made sure that each task has a *stable solution*. We define a stable solution to be an action that: (1) solves the task and (2) still solves the task if the action is slightly perturbed. The perturbations we consider are translations by 0.5 pixels along each axis (8 shifts in total).

Task solvability. Because the current benchmark contains $(25 + 25) \times 100 = 5,000$ tasks, it is cumbersome to manually find stable solutions for each task. Moreover, it is not possible to do

brute-force search over all possible actions because the action space is continuous. Therefore, we used the following stochastic approach to evaluate whether or not a task is solvable. Let a denote an action and τ a task. We define the random variable $\texttt{stably\_solves}(a, \tau)$ to be 1 if action a is a stable solution for task τ and 0 otherwise. The random variable $\texttt{valid}(a, \tau)$ is 1 iff action a is a valid action for task τ. We define the *solvability level* of task τ to be: $s(\tau) = P(\texttt{stably\_solves}(a, \tau) = 1 | \texttt{valid}(a, \tau) = 1)$. To determine whether task τ is solvable, we would ideally seek to reject the hypothesis $s(\tau) = 0$.

Exact testing of this hypothesis is, however, infeasible, and so we resort to a proxy that uses a small constant p_0, randomly selected actions, and a binomial statistical test to reject at least one of the hypotheses: $s(\tau) \leq p_0$ or $s(\tau) \geq 2p_0$. We sample random actions until we can reject one of the two hypotheses. If the $s(\tau) \leq p_0$ hypothesis is rejected, we define the task to be *solvable*. Alternatively, we define the task *unsolvable* if the $s(\tau) \geq 2p_0$ hypothesis is rejected. In the unlikely event that both hypotheses are rejected we categorize the task as solvable.

It is possible to show that this algorithm requires no more than $\frac{1}{32p_0}$ action samples to reject at least one of the hypotheses with p-value 0.05. In practice, the value of p_0 was chosen to match our intuitive sense of task solvability: for PHYRE-B, we set $p_0 = 10^{-5}$; for PHYRE-2B, we set $p_0 = 10^{-6}$.

Tier requirements. We used the definition of task solvability to check the correctness of the implementation of a task template. We also used task solvability to guide the selection of tasks within a template, *e.g.*, the task creator may impose the constraint that a template only contains tasks with two-ball solutions and no single-ball solutions and enforce this constraint automatically.

We designed the task templates in both tiers to meet the following criteria: (1) all tasks in a tier to be solvable according to the definition of task solvability described above using samples from the action space corresponding to that tier and (2) less than 50% of the tasks in a PHYRE-2B task template can be solvable using a single ball. Hence, the task templates in PHYRE-2B are strictly harder to solve than those in PHYRE-B.

Solution diversity. The task templates are designed such that solving a task instance within a template should not be trivial for an agent that knows how to solve other tasks in the template. For example, a task template should not have a single "master solution" that solves (nearly) all tasks in the template. At the same time, it is nearly impossible to prevent that multiple tasks in the same template share solutions because these tasks share the same design (see Figure 9).

To measure the *solution diversity* of a task template, we count the number of tasks within the template that each action can solve. Since the action space is continuous we cannot check every action. Instead, we randomly sample 10^6 actions to estimate solution diversity. We plot the results, for each task template, as histograms in Figure 10 and 11. Each histogram shows the number of actions (y-axis) that can each solve a particular number of tasks (x-axis) within the template. We are interested to see if one or more actions are able to solve a large fraction of the tasks within a template, which will appear as bars (of any height) on the right side of the x-axis. The figures show that in general tasks in the PHYRE-2B tier require more diverse solutions to be solved than those in the PHYRE-B tier.

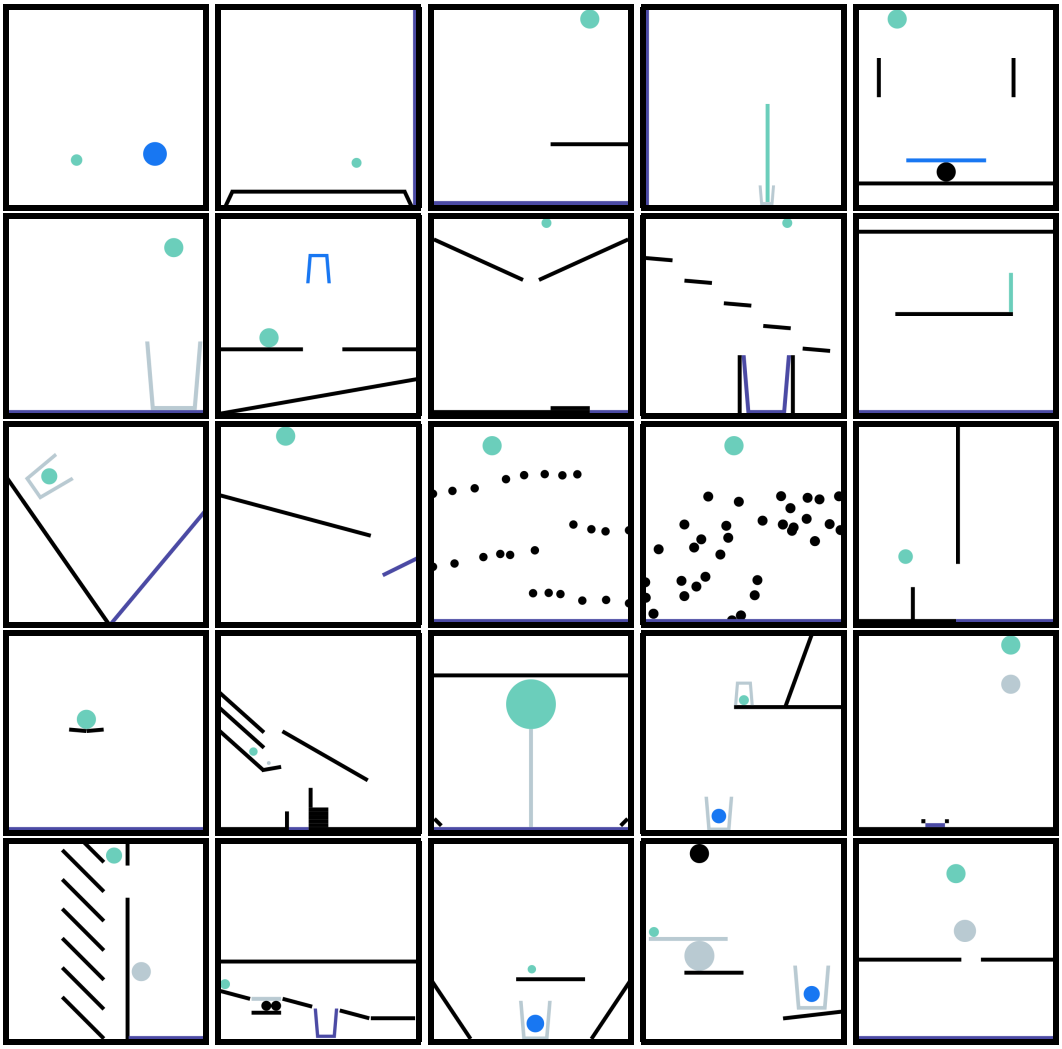


Figure 7: The 25 task templates in the PHYRE-B tier. In each task the goal is to make the (dynamic) green body touch the (static) purple body or the (dynamic) blue body; black bodies are static and gray bodies are dynamic. Each of the PHYRE-B task templates gives rise to 100 tasks, each of which can be solved by adding a single dynamic ball to the scene.

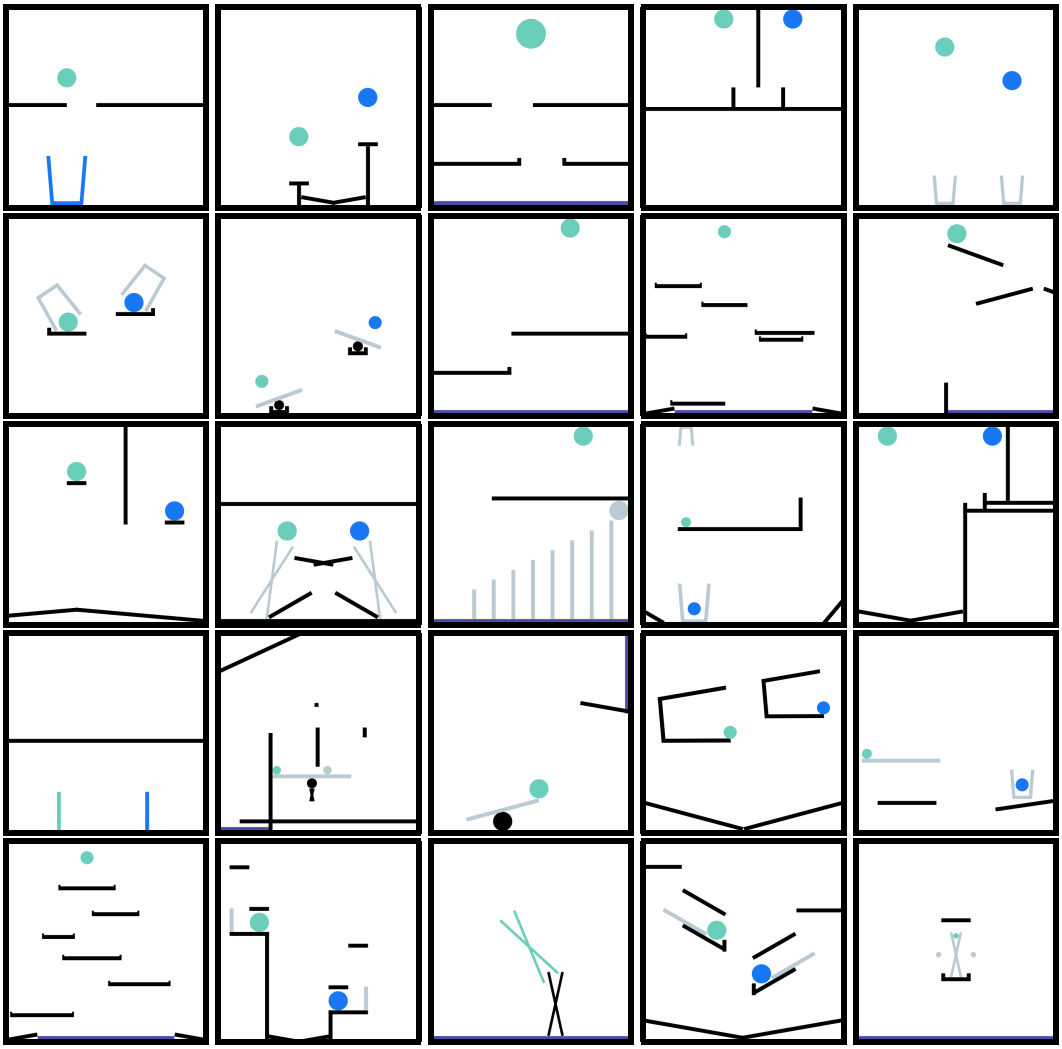


Figure 8: The 25 task templates in the PHYRE-2B tier. In each task the goal is to make the (dynamic) green body touch the (static) purple body or the (dynamic) blue body; black bodies are static and gray bodies are dynamic. Each of the PHYRE-2B task templates gives rise to 100 related tasks, all of which can be solved by adding two dynamic balls to the scene.

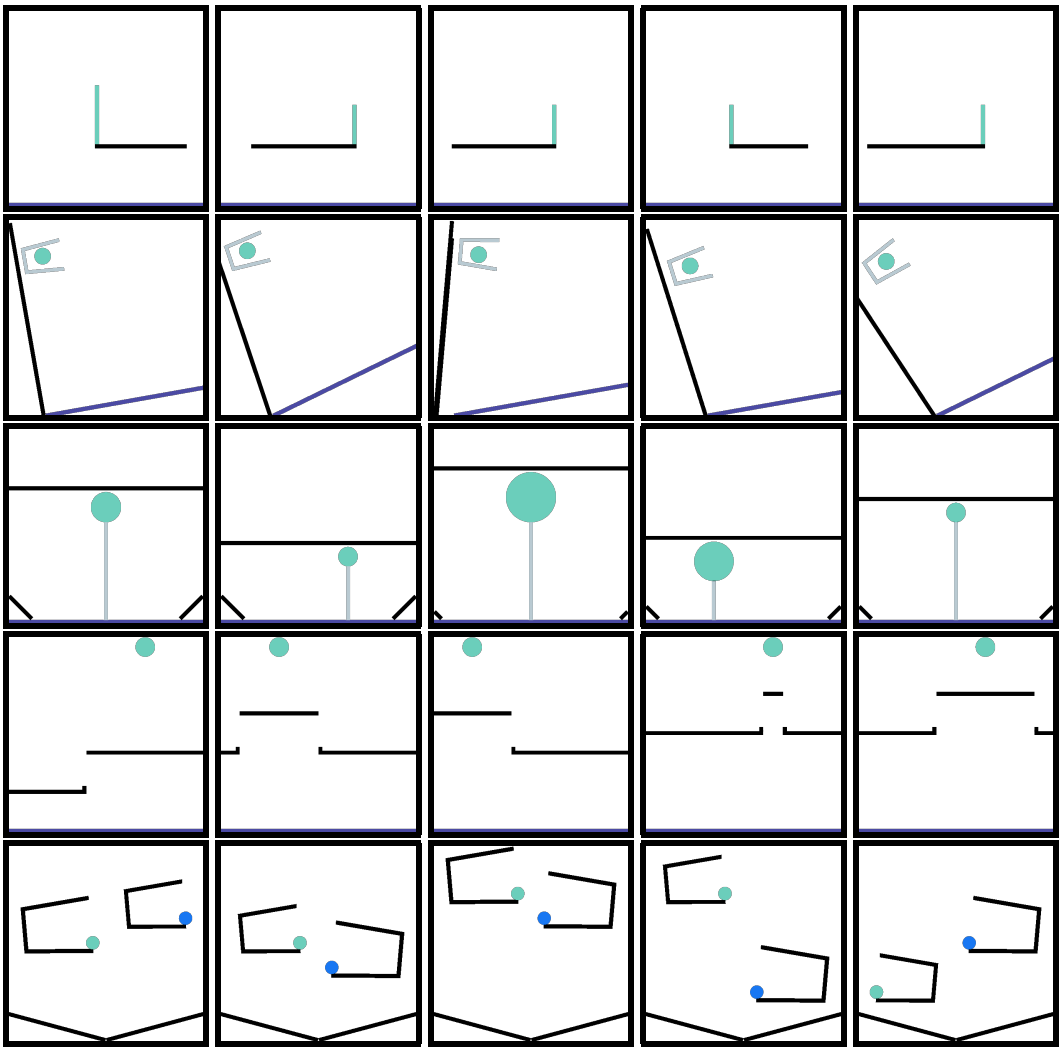


Figure 9: Each row shows five example tasks from the same task template. The size, initial position, and orientation of bodies vary within a template, so each task requires its own solution; however all tasks within a template share similar physical intuition and high-level strategy.

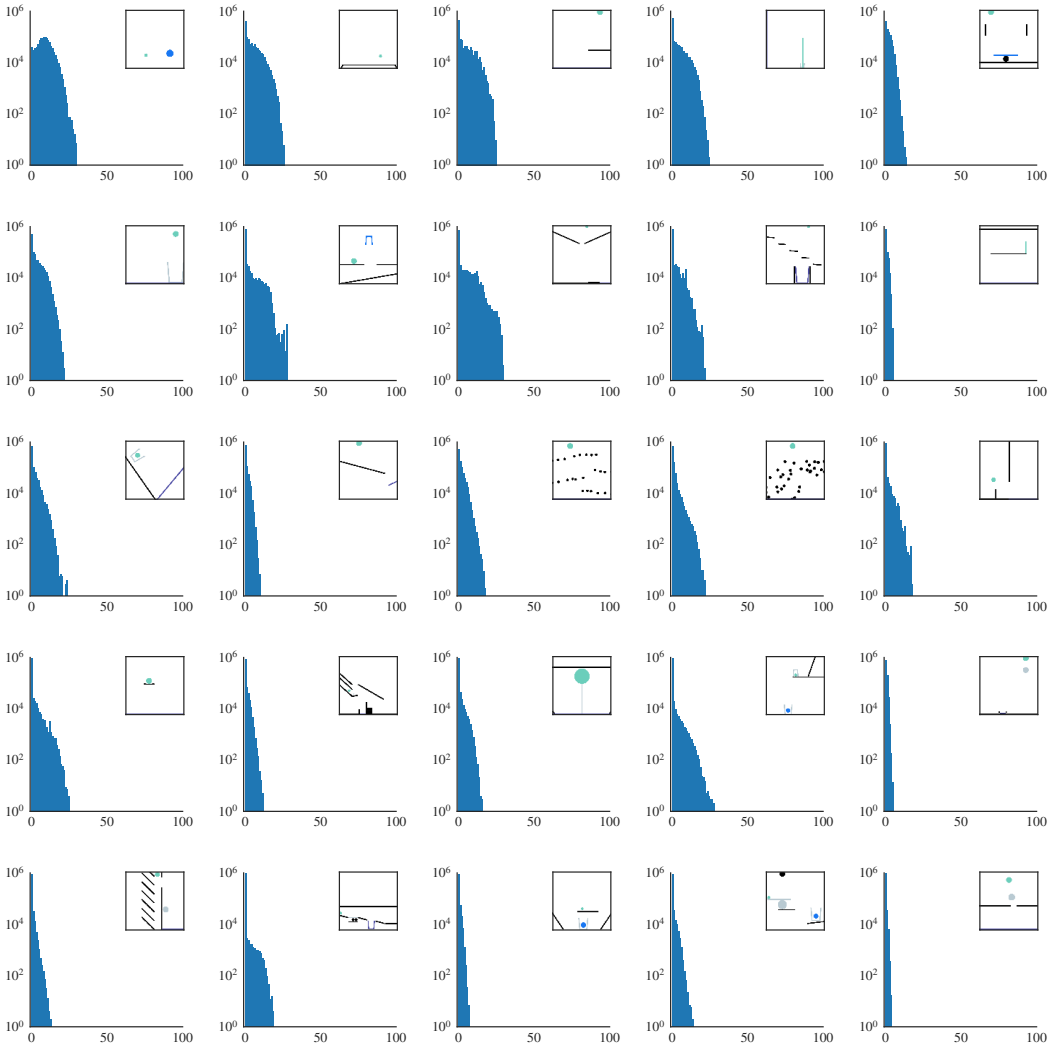


Figure 10: Analysis of the *solution diversity* of the task templates in the PHYRE-B tier. Histograms show the number of actions (y-axis) that solve a certain number of tasks in the template (x-axis).

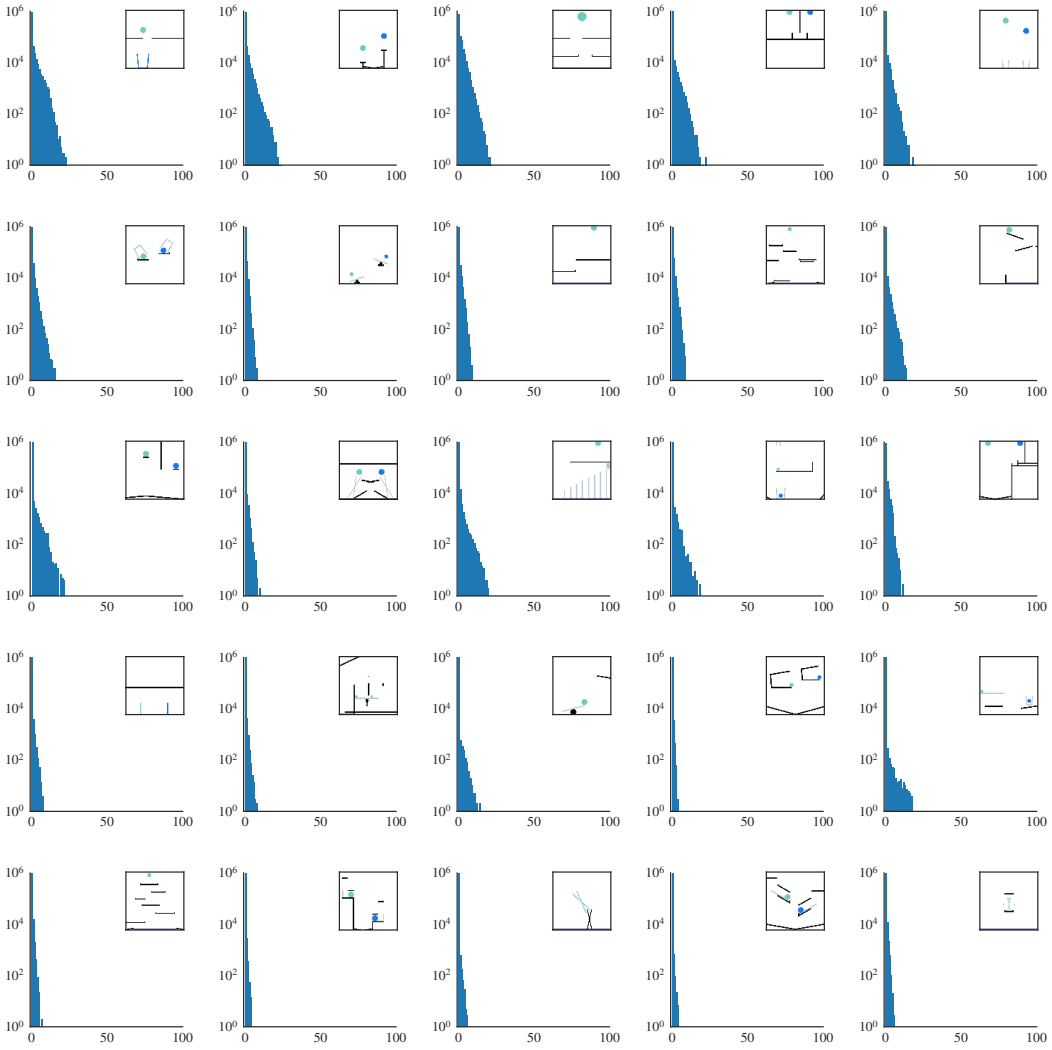


Figure 11: Analysis of the *solution diversity* of the task templates in the PHYRE-2B tier. Histograms show the number of actions (y-axis) that solve a certain number of tasks in the template (x-axis).

C Comparing Agents

To determine if one agent outperforms another agent, we use the one-sided Wilcoxon test as implemented in the `scipy.stats` Python package.[4] To enable future work to compare with our baselines, we provide AUCCESS scores for all folds and evaluation settings in Table 2.

Setting	Fold / Agent	0	1	2	3	4	5	6	7	8	9
2B (cross)	RAND	0.0517	0.0212	0.0099	0.0442	0.0038	0.0356	0.0178	0.0177	0.0264	0.0275
	MEM	0.0728	0.0289	0.0135	0.0783	0.0090	0.0463	0.0186	0.0387	0.0376	0.0274
	MEM-O	0.0967	0.0371	0.0164	0.0933	0.0094	0.0815	0.0242	0.0451	0.0535	0.0345
	DQN	0.3818	0.1944	0.1072	0.3051	0.0732	0.2703	0.2388	0.2216	0.2528	0.2730
	DQN-O	0.5149	0.2682	0.2596	0.5298	0.2809	0.5313	0.4828	0.3330	0.3581	0.3987
2B (within)	RAND	0.0271	0.0367	0.0428	0.0301	0.0394	0.0452	0.0336	0.0287	0.0380	0.0335
	MEM	0.0325	0.0336	0.0315	0.0371	0.0304	0.0314	0.0282	0.0320	0.0330	0.0347
	MEM-O	0.0325	0.0336	0.0315	0.0371	0.0304	0.0314	0.0282	0.0320	0.0330	0.0347
	DQN	0.6447	0.6829	0.6747	0.6763	0.6999	0.6700	0.6879	0.6704	0.6877	0.6824
	DQN-O	0.6447	0.6829	0.6747	0.6763	0.6999	0.6700	0.6879	0.6704	0.6877	0.6824
B (cross)	RAND	0.1178	0.1242	0.1818	0.1242	0.0381	0.2250	0.1173	0.1329	0.0894	0.1460
	MEM	0.2059	0.1656	0.2004	0.2263	0.1159	0.2488	0.1416	0.2467	0.1055	0.1881
	MEM-O	0.2578	0.2551	0.2443	0.2552	0.2327	0.2508	0.1469	0.2801	0.1281	0.2273
	DQN	0.4369	0.3096	0.4305	0.4391	0.2277	0.4440	0.3453	0.3920	0.1898	0.4646
	DQN-O	0.6859	0.4867	0.6671	0.5995	0.4916	0.6560	0.5100	0.6573	0.3733	0.4884
B (within)	RAND	0.1344	0.1401	0.1379	0.1380	0.1275	0.1334	0.1395	0.1430	0.1336	0.1433
	MEM	0.0198	0.0258	0.0230	0.0269	0.0223	0.0286	0.0237	0.0214	0.0223	0.0288
	MEM-O	0.0198	0.0258	0.0230	0.0269	0.0223	0.0286	0.0237	0.0214	0.0223	0.0288
	DQN	0.7682	0.7972	0.7822	0.7586	0.7703	0.7842	0.7801	0.7734	0.7804	0.7687
	DQN-O	0.7682	0.7972	0.7822	0.7586	0.7703	0.7842	0.7801	0.7734	0.7804	0.7687

Table 2: AUCCESS scores (on a 0.0 to 1.0 scale) of our five agents in both generalization settings, for each of our 10 folds.

[4] Specifically, we call `scipy.stats.wilcoxon(A, B, zero_method='wilcox', correction=False, alternative='greater')` to test if the AUCCESS vector `A` outperforms AUCCESS vector `B`, where `A` and `B` are component-wise paired with one component for each fold.