

BUILDERBENCH – A BENCHMARK FOR GENERALIST AGENTS

Anonymous authors

Paper under double-blind review

ABSTRACT

Today’s AI models learn primarily through mimicry and sharpening, so it is not surprising that they struggle to solve problems beyond the limits set by existing data. To solve novel problems, agents should acquire skills for exploring and learning through experience. Finding a scalable learning mechanism for developing agents that learn through interaction remains a major open problem. In this work, we introduce BuilderBench, a benchmark to accelerate research into agent pre-training that centers open-ended exploration. BuilderBench requires agents to learn how to build any structure using blocks. BuilderBench is equipped with (1) a hardware accelerated simulator of a robotic agent interacting with various physical blocks, and (2) a task-suite with over 42 diverse target structures that are carefully curated to test an understanding of physics, mathematics, and long-horizon planning. During training, agents have to explore and learn general principles about the environment without any external supervision. During evaluation, agents have to build the unseen target structures from the task suite. Solving these tasks requires a sort of *embodied reasoning* that is not reflected in words but rather in actions, experimenting with different strategies and piecing them together. Our experiments show that many of these tasks challenge the current iteration of algorithms. Hence, we also provide a “training wheels” protocol, in which agents are trained and evaluated to build a single target structure from the task suite. Finally, we provide single-file implementations of six different algorithms as a reference point for researchers.

Can AI models build a world which today’s generative models can only dream of?

1 THE NEED FOR A NEW BENCHMARK

Today’s artificial intelligence (AI) models acquire knowledge by combing through massive collections of human-generated data, enabling them to generate a wide array of images and write a wide array of stories. While this recipe has been highly successful in domains like vision and language, where models can learn from expert human photographers and writers, it is less clear how to apply this recipe to application areas that humans understand poorly today (e.g., biology, chemistry) (Ying et al., 2025; Silver & Sutton, 2025). Making progress will require that agents learn not only from human experience, but also from their own, self-collected experience. Agents will have to actively explore and run experiments to extract knowledge about the environment (Spelke & Kinzler, 2007). Agents will then have to consolidate this knowledge and use it to quickly solve novel tasks. Despite many works recognizing the importance of **open-ended exploration** and **learning through experience** (Stanley, 2017; Team, 2023), most benchmarks for building foundation models today focus on learning solely from human data.

This is not for lack of trying. There is a long line of interaction and exploration benchmarks built by researchers in reinforcement learning (RL) (Ecoffet et al., 2021; Tang et al., 2017), control (Plappert et al., 2018), and developmental robotics (Oudeyer et al., 2007). For example, maze navigation in ant-maze (Fu et al., 2021) or montezuma’s revenge (Bellemare et al., 2013), or handful of tasks in the kitchen environment (Gupta et al., 2020). Other than a few exceptions like Minecraft (Guss et al., 2019), most widely used benchmarks only allow a handful of diverse behaviors (Rajeswar et al., 2023; Gupta et al., 2020; Fu et al., 2021; Tassa et al., 2018). Agents trained on even the most complex of these benchmarks (e.g., StarCraft (Vinyals et al., 2019), AI2Thor (Kolve et al., 2022), NetHack (Küttler et al., 2020)) do not seem to learn the same sort of common sense and

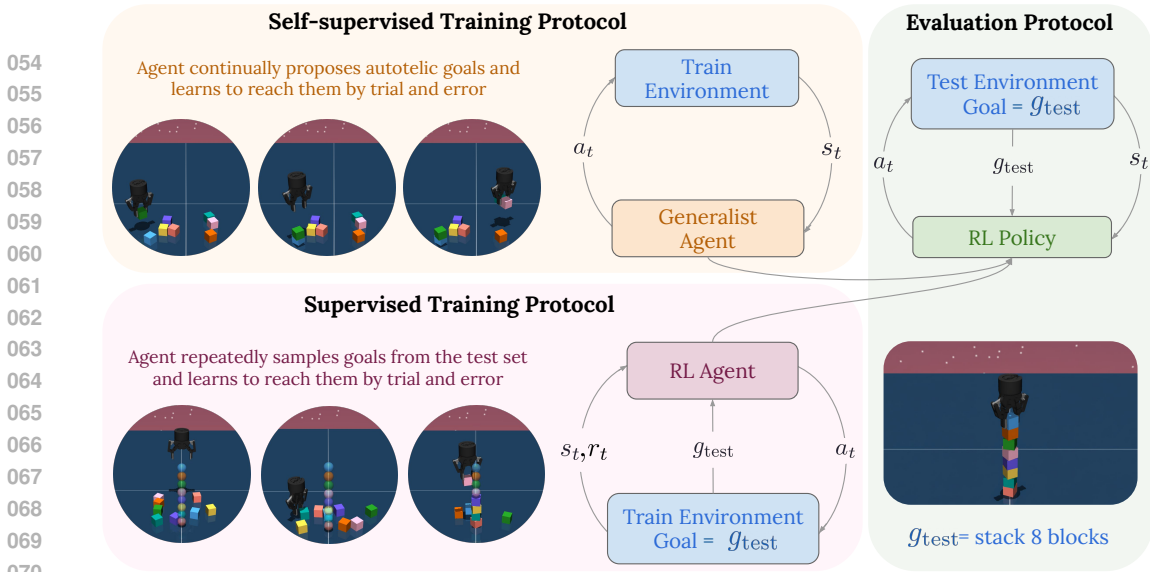


Figure 1: **The BuilderBench Benchmark.** (Top Left) Training consists of self-supervised exploration: agents can collect data to learn to reach various goals via trial and error. Agents are not given any information about the test-time goals or their distribution. (Right) During evaluation, the agent is given a goal and attempts to reach that goal by taking actions in the environment. (Bottom Left) For prototyping, we also include a “debug” mode where agents can learn to reach the test-time goals via trial and error.

reasoning skills that agents trained on human text do acquire (Wei et al., 2022). We argue that the key reason why, is that, there is not much that can be learned in the current generation of interactive benchmarks. Existing benchmarks rarely allow agents to practice skills ranging from exploration to prediction, from low-level control to high-level reasoning.

We envision a benchmark which enables an open-ended stream of interaction (Hughes et al., 2024; Sigaud et al., 2024), where training could only ever cover a tiny slice of all possible behaviors. In the same way that vision models today can paint pictures that go well beyond what is in their training data (e.g., an astronaut mowing the lawn), we envision embodied agentic systems that can solve tasks that go well beyond the tasks they have practiced solving before. Solving such a benchmark would require agents to have efficient exploration abilities. Moreover, it requires that exploration take into account an agent’s generalization capabilities, since it will be impossible to perform all possible behaviors (Hughes et al., 2024). Agents should, in effect, become scientists, performing micro experiments in the environment to discover the laws governing the environment. Once these physical laws have been found, they can be used to make wide-ranging generalizations about how the entire environment works, and how one should act within it. *Our paper constructs an environment where such exploration is possible. One central insight of our paper is to show that this can actually be done using a surprisingly simple setup: block-building.*

Why block-building? Blocks conceptually form an atomic unit, allowing agents to build diverse structures. Many children spend years playing with blocks. Research in child development highlights that block play builds spatial (Reifel, 1984; Wexler et al., 1998; Casey et al., 2008; Singer et al., 2006) and arithmetic skills (Verdine et al., 2014; Cheng & Mix, 2014). In addition to being useful for early human cognitive development, block-building is a mathematically rich area¹ with a deep history in AI and planning (Gupta & Nau, 1992; Ahmad et al., 2019; Russell & Norvig, 2010). Building stable structures with blocks requires long-horizon planning and complex reasoning capabilities. While research on reasoning and generalization capabilities has almost become synonymous with large language models in recent years (Touvron et al., 2023; DeepSeek-AI, 2025; OpenAI, 2024; Team, 2025), block-building allows us to study whether this sort of reasoning and generalization can emerge from the ground up, through exploration and trial-and-error learning.

¹In 2011, Paterson et al. (2007) was awarded the prestigious David P. Robbins Prize in mathematics for improving an upper bound on the maximum overhang using identical blocks.

To this end, we introduce the **BuilderBench** benchmark. BuilderBench is equipped with a fast simulator consisting of a robotic hand traversing in space and interacting with blocks, following Newtonian physics. This simple setup allows to design tasks which span several orders of magnitude of complexity. Tasks require motor skills like locomotion, grasping and throwing as well as higher-level skills such as logical reasoning (commutativity and associativity of pick and place ordering), geometrical reasoning (maximizing overhangs, packing problems) and intuitive physics (gravity, friction, toppling, balancing). Tasks also require reasoning about counterweights, buttresses, and temporary scaffolding. During training, agents must discover such skills through practice. During testing, agents must use those skills to build unseen structures (Figure 1). To succeed in building a large set of diverse structures, agents must learn general patterns of construction, rather than memorizing individual actions. Finally, one can easily scale the difficulty of tasks by increasing the number of blocks.

We summarize the main contributions of this paper below:

- We introduce BuilderBench, a benchmark that uses open-ended block-building to develop and evaluate agents for efficient exploration, reasoning and generalization abilities.
- The BuilderBench simulator is developed using MuJoCo (Todorov et al., 2012) and JAX (Bradbury et al., 2018). It is hardware accelerated and allows RL training between 10 to 100 times faster than purely CPU based open-ended benchmarks like Crafter (Hafner, 2022), Minecraft (Guss et al., 2019) or NetHack (Küttler et al., 2020) (see Appendix B for a speed test).
- We open-source BuilderBench, a task-suite of over 40 tasks to evaluate the performance of agents. Each task corresponds to a different block structure. Building each structure requires unique reasoning abilities.
- We open-source single-file implementations of four representative reinforcement learning (RL) algorithms and three self-supervised data-collection algorithms. Training runs are extremely fast (e.g., training a PPO agent to stack two blocks takes 30 minutes on a single GPU), reducing the barrier to entry for frontier RL research.

2 RELATED WORK.

AI benchmarks have driven progress in the field. Benchmarks such as MNIST (Deng, 2012), ImageNet (Russakovsky et al., 2015), Atari (Bellemare et al., 2013), Gym (Brockman et al., 2016), WMT (Chelba et al., 2014), SWE-bench (Jimenez et al., 2024), ARC-AGI (Chollet, 2019) have propelled research in deep learning, vision, RL and natural language processing. The aim of BuilderBench is to similarly propel research on RL agents. Below we discuss various aspects of this problem and prior attempts to tackle and benchmark them.

Reinforcement learning (RL) studies agents that learn through interaction. Standard RL benchmarks (Bellemare et al., 2013; Brockman et al., 2016; Tassa et al., 2018; Hafner, 2022; Küttler et al., 2020; Koyamada et al., 2023; Bonnet et al., 2024) have agents learn to maximize hand-designed rewards to solve a task of interest. These environments require agents to extract their own knowledge and novel solutions (e.g., endlessly bouncing the ball in breakout from DQN (Mnih et al., 2013) or the famous move 37 from AlphaGo (Silver et al., 2016)). However, these environments focus on solving a small range of tasks. As a result, RL agents typically possess narrow or poor generalization capabilities (Kirk et al., 2023). The type of generalization that is desired is not just towards perturbed observations or dynamics (Stone et al., 2021; Cobbe et al., 2020), but towards solving diverse unseen tasks (Ghosh et al., 2021).

Unsupervised RL is centered on devising objectives that let agents learn through trial and error without any rewards. Such methods usually try to learn generally useful skills (Gregor et al., 2016; Eysenbach et al., 2019) or collect exploratory data (Lee et al., 2020; Tang et al., 2017; Osband et al., 2016). But it is not clear how scalable these objectives are, mainly because the standard unsupervised RL benchmarks (Rajeswar et al., 2023; Fu et al., 2021; Tassa et al., 2018) contain only a handful of similar downstream tasks for evaluation. Hence, to properly evaluate generalization properties of agents, benchmarks need to have sufficiently complex and unseen test tasks.

Another set of methods that are closely related are ones which treat the problem of efficiently generalizing to unseen tasks as a learning problem itself. Meta-learning (Caruana, 1998; Finn et al., 2017; Schmidhuber, 1987) and few-shot learning (Vinyals et al., 2016; Snell et al., 2017) fall under this category. Initial progress was driven by benchmarks that arranged common supervised learning tasks episodically, testing how quickly models adapt to new tasks (Lake et al., 2015; Dhillon et al., 2020). Later work found that self-supervised pre-training on diverse datasets provided enough prior knowledge to directly solve most of the common supervised learning tasks (Radford et al., 2021; Brown et al., 2020; Devlin et al., 2018), blurring the boundary between memorizing prior knowledge and efficiently generalizing. We argue that open-ended domains and tasks are needed to disentangle the two. ARC-AGI (Chollet, 2019) uses the open-ended domain of discrete puzzles to measure a model’s ability to efficiently use its priors. ARC tests models on a set of novel puzzles that require on-the-fly composition of a minimal set of core principles (Chollet, 2019; Spelke & Kinzler, 2007). BuilderBench is similarly structured. Solving tasks from the BuilderBench task-suite not only requires a concrete set of priors (e.g., an understanding of Newtonian physics), but requires using these priors to build unseen structures on-the-fly. Unlike ARC-AGI, where priors are directly provided through examples of solved puzzles, in BuilderBench agents have to discover priors on their own through interaction.

In addition to exploration and generalization, the BuilderBench task-suite highlights how block-building can also be used to evaluate various types of reasoning abilities (see Section 5.1 for details). Many of these abilities are typically studied only in isolation. For e.g., intuitive physics is evaluated in Chow et al. (2025); Riochet et al. (2020), motor skills in James et al. (2019); Melnik et al. (2021), planning in Valmeekam et al. (2023), mathematical reasoning in Lewkowycz et al. (2022); Ahn et al. (2024). In recent years, reasoning is almost exclusively studied using language models pretrained on data. However, BuilderBench allows us to evaluate and visualize reasoning that is not grounded in language and not learned using human data.

The most similar benchmarks to BuilderBench are recent benchmarks like Kinetix (Matthews et al., 2025), XLand (Team et al., 2021; Nikulin et al., 2023), and Minecraft (Guss et al., 2019). Kinetix provides a diverse set of rigid body tasks, constrained to 2D, to test zero shot generalization of agents. Tasks in Kinetix are procedurally generated. Unlike BuilderBench, these tasks do not clearly test diverse logical and mathematical reasoning abilities. XLand provides a vast set of multi-agent video-game like tasks, but is closed source and not readily available for academic research. Minecraft is a popular open-ended game that revolves around building various artifacts with blocks that has been used to develop generally capable agents from scratch (Hafner et al., 2024; Ma et al., 2022; Zhao et al., 2024). While based on the similar block-building foundations and an appealing benchmark, we believe BuilderBench is better suited for academic research due to the much faster speed of its simulator and an extensive carefully curated task-suite. Finally, BuilderBench is fully open source, making all of its components flexible and easy to adapt. In Appendix C, we have added more discussion on qualitative comparison of BuilderBench with prior benchmarks.

3 BUILDERBENCH - A BENCHMARK FOR GENERALIST AGENTS

This paper proposes the BuilderBench benchmark, which comes with task-suite of 42 tasks, where each task is a target block structure carefully curated for evaluating unique abilities. BuilderBench comes with a fast simulator consisting of an agent interacting with a varying number of blocks. In the following sections, we will describe the environment (Section 4), the task-suite (Section 5) and the training and evaluation protocols (Section 6).

4 BUILDERBENCH ENVIRONMENT.

The environment can be formulated as a Markov decision process (MDP) (Sutton & Barto, 2018), with states $s_t \in \mathcal{S}$ and actions $a_t \in \mathbf{A}$ and transition dynamics $T(s_{t+1} | s_t, a_t)$ and a maximum episode length H . An additional context parameter n specifies the number of cube-shaped blocks in the environment. Each environment instance contains a single robot hand which can navigate in

3D space² and interact with the n cubes. All interactions approximate real physics simulated using MuJoCo (Todorov et al., 2012).

State space. The observations include information about the arm and the cubes. For the arm, we include its global position coordinates (\mathbb{R}^3), orientation quaternion (\mathbb{R}^4), linear velocity (\mathbb{R}^3), and the distance between its two fingers (\mathbb{R}). For every cube, we include the global position coordinates (\mathbb{R}^3), orientation quaternion (\mathbb{R}^4), linear and angular velocity (\mathbb{R}^6).

Action space. The agent can manipulate its environment using a 5 dimensional action space. The first three actions control its position actuators, enabling navigation along the standard basis vectors. The fourth action controls the agent’s yaw, enabling it to rotate about the global z-axis. The fifth action controls the finger actuators, which allows the agent to change its pinching width.

Task specification. Each task corresponds to a physically stable target structure built using cubes. To specify this structure, we provide a vector of target cube positions (\mathbb{R}^{3k}), where $k \leq n$ is the number of cubes in the target cube structure. This allows us to specify target structures that contain fewer cubes than the environment (see Figure 5 for an example).

As we will see in the next section, despite this seemingly simple setup, tasks can be arbitrarily complex and long-horizon. Qualitatively, we will see that solving tasks require multiple steps of high-level reasoning.

5 BUILDERBENCH TASK SUITE

The BuilderBench task suite contains a total of 42 tasks. In this section, we describe these tasks in detail and the design philosophy behind the task-suite. The task-suite is meant to be open-ended and address the challenges highlighted in Section 1. We start with a case-study of five different tasks from the BuilderBench task-suite, which is meant to showcase how each task requires the agent to unlock at least one distinct reasoning ability and compose various high-level skills sequentially. As described in Section 4, during evaluation, agents only have access to the positions of the masked cubes in the target structure. The complete list of tasks, along with visualizations and the capabilities required to solve them, is provided in Appendix E.

5.1 A CASE STUDY OF FIVE TASKS

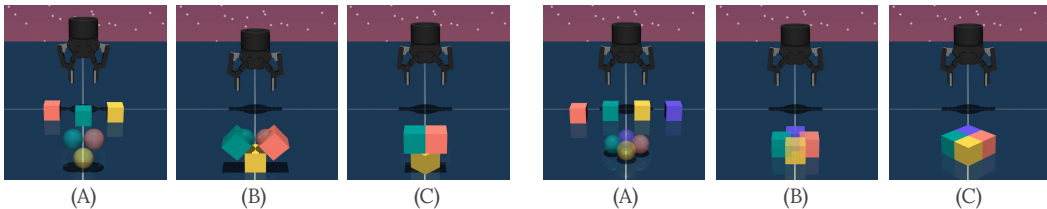


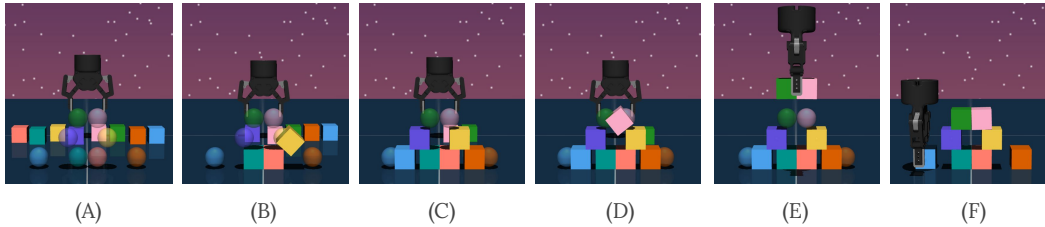
Figure 2: **T-Block** (Left) and **Four Cube Packing** (Right)

Example 1: T-Block. This task requires building a simple T shaped structure with one cube at the base, and two cubes on top (Figure 2). The second frame (B) shows what many people envision as the solution to this task. However, as show in the frame, this configuration isn’t stable. Solving this task requires the reasoning insight to rotate the bottom cube by about 45° . Since the diagonal of the cube’s top surface is longer than its edge length, the rotated base provides sufficient support for both top cubes, enabling a stable T-shaped structure (see third frame).

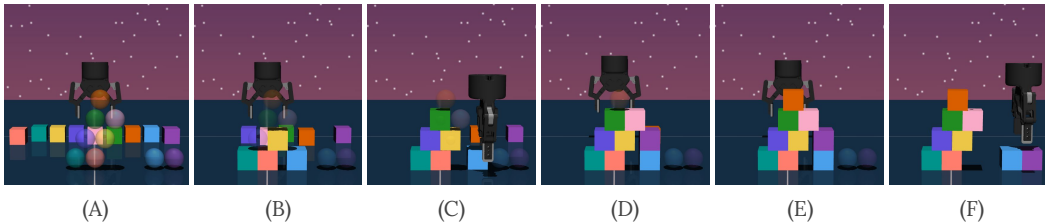
Example 2: Four Cube Packing. This task tests geometric reasoning and spatial packing. The target structure is an arrangement of four cube centers placed at some distance along the four cardinal directions on the floor (see (A) of Figure 2). The distance is chosen such that the placement is impossible with the default cube orientation: the cubes overlap (see (B)). This results in a packing problem of arranging the cubes such that its centers form the target structure. To solve this, the agent needs to rotate each cube by 45° before placing it, which ensures the centers align correctly

²We do not include the entire robot because inverse kinematics is a solvable and orthogonal problem. This also significantly increases the reach of the robotic arm. This robot can be conceptually thought of as a crane.

without collision (see (C)). Due to the two fingered morphology of the robot, this task cannot be solved using pick and place primitives, but would require nudging the final block in place.

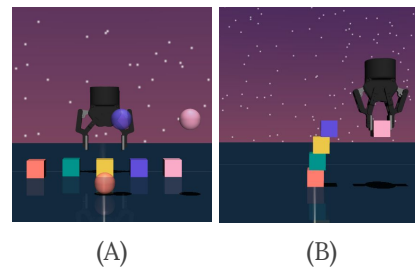
Figure 3: **Hexagonal Portal**

Example 3: Hexagonal Portal. This task requires constructing a hexagonal, portal-like structure using eight cubes and two extra cubes placed on the floor on either side. (Figure 3). A direct attempt to place the yellow or indigo cubes leads to toppling (see (B)). To stabilize them, the agent must first build two supporting scaffolds (see (C)). After the first two layers are in place, the pink and green cubes cannot be added independently since each would collapse inward (see (D)). Because no additional blocks remain, another scaffold cannot be used for support. Instead, the agent must discover a non-trivial maneuver – lifting and placing the pink and green cubes simultaneously (see (E)). Finally, the temporary orange and light-blue scaffolds must be carefully removed and placed in their desired position to complete the structure (see (F)). This task requires long-horizon planning and learning emergent skills like building scaffolds and learning to pick two cubes at once.

Figure 4: **Leaning Tower**

Example 4: Leaning Tower. The target is a leaning tower composed of seven blocks and two extra cubes placed on the floor (see (A) of Figure 4). Solving this task demands building two scaffolds and re-using the first one for the main tower. It also requires an understanding of the concept of counterweights for generating a stable overhang (an outward extension). The solution itself requires multiple steps of high level planning. After building the base, the yellow block in the second layer must be supported by a temporary scaffold (see blue cube in (B)). To stabilize the structure, the agent needs to add counterweights (the indigo and green cubes in (C)) and only then remove the scaffold (see (D)). To build the third and fourth layer, the agent has to build another set of scaffolds and counterweights. In particular, placing the pink block in the third layer requires a two-cube vertical scaffold (see (E)). Finally, the tower is completed by adding the counterweights (the green and orange cubes in (F)) and removing and repositioning the last scaffold (see (G)).

Example 5: Maximum Overhang Problem. In this task, the environment contains five cubes, but the task only specifies the target positions for three cubes (see (A) of Figure 5). But to put those three cubes in the target location, the agent will need to use all five blocks. To correctly place the green and the yellow cubes (whose target positions are not specified) in order to complete the task, the agent needs to solve the popular maximum overhang problem (see Paterson et al. (2007) for the solution). The main intuition is that at any level, the collective center of the mass of all the cubes above, should not be on the right of the level’s boundary. Without such a placement, the task is impossible to solve. The pink cube is specified to “distract” the agent from simply holding the indigo cube in place.

Figure 5: **Maximum overhang problem**

This case-study illustrates how a block-building setup with a handful of blocks can result in open-ended tasks that can be used to test high-level reasoning abilities. Agents which do not have access to these tasks have almost no chance of encountering them during training. For solving these tasks, we anticipate that agents would have to learn key knowledge priors via exploration (Spelke & Kinzler, 2007), and learn mechanisms to efficiently use them on the fly (Chollet, 2019). In the next section, we outline the general design principles that underlie the tasks in the BuilderBench task-suite.

5.2 DESIGN PHILOSOPHY BEHIND THE BUILDERBENCH TASK-SUITE.

The primary goal of the task-suite is to capture the main challenges in evaluating open-ended exploration and generalization (highlighted in Section 1) and provide a meaningful feedback signal for algorithmic research. To best support these goals, we followed the following design principles:

Solving different tasks should require distinct skills. For example, once an agent learns how to pick and place two blocks, extending this to three or more independent blocks does not qualitatively require an additional ability. We have designed tasks such that they demand a range of motor skills, including grasping, nudging, and throwing. Importantly, tasks also require logical reasoning skills, such as commutativity and associativity of blocks (pick and place ordering), induction (stacking n blocks vs stacking $n + 1$ blocks), geometry, and intuitive physics.

Most tasks should be solvable by humans. To ensure that solving the tasks is theoretically possible, we manually solved most tasks using the same action space as the agent. We also provide scripts that allow researchers to explore the environments and attempt to solve tasks themselves.

Tasks should range from very easy to extremely hard. This is an important feature of BuilderBench, meant to provide breadcrumbs of feedback to go from current algorithms capable of solving only the simplest tasks and agents that can build anything.

Tasks should include some whose solutions are unknown even to the authors. One aim of BuilderBench is also to see if artificial agents can come up with solutions to problems whose solutions are unknown. Hence, we have included a small minority of tasks which we were not able to solve.

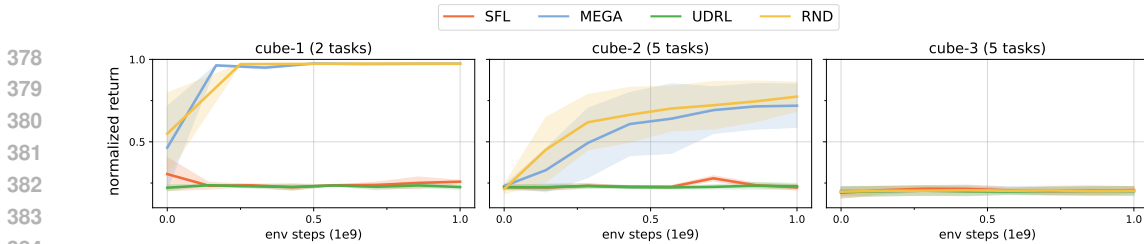
6 TRAINING AND EVALUATION PROTOCOLS.

To evaluate open-ended exploration and generalization, we design the multi-task self-supervised protocol. Because this protocol contains various challenges, we also provide a debug single-task supervised protocol meant to provide additional feedback for researchers. We describe both protocols below and provide additional details in Appendix A.

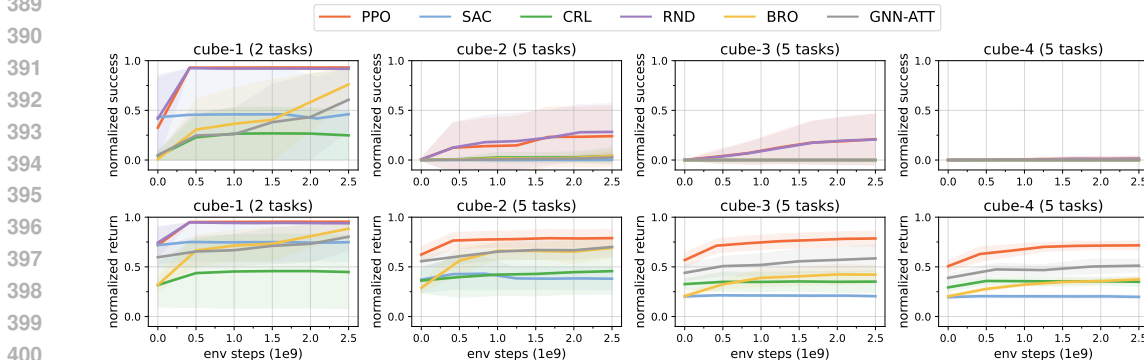
Multi-task self-supervised protocol. The agent interacts with the environment, but does not receive any task specification during training. The agent’s goal is to explore its environment to acquire general knowledge and skills that might help it to solve future tasks. The agent has to learn a task conditioned policy (Kaelbling, 1993), which can take as input a state (\mathbb{R}^{11+13n}) as well as a task specification (\mathbb{R}^{3k}). Each environment has a number of hand-designed tasks associated with it Appendix E. The agent is evaluated by running its task-conditioned policy on these tasks and measuring the reward obtained by it.

During training, it is highly unlikely that the agents will have seen these hand-designed tasks. Hence, to solve this protocol, agents will have to learn general reusable skills and concepts through purely self-supervised interaction. Many of these tasks are very difficult and unsolvable by the initial algorithms we tried. To provide additional feedback for algorithmic development, we also provide a simpler “training-wheels” protocol.

Single-task supervised protocol. In this standard RL protocol, the agents interact with a single environment to solve a single task from the task-suite. Each environment comes with a reward r_t . For each task, we currently support two types of reward functions – dense vs sparse, and permutation variant vs invariant to the cube order. By default, the rewards used are dense and permutation invariant to the cube order. Exact details of the reward functions are provided in Appendix A.2. The agent’s objective is to learn a policy that maximizes the expected sum of rewards (Sutton & Barto, 2018).



378
379
380
381
382
383
384
385 **Figure 6: Self-supervised evaluation on BuilderBench task-suite.** We evaluate MEGA (Pitis et al., 2020)
386 and SFL (Rutherford et al., 2024) on 12 of the lowest complexity (yet still difficult) tasks from our task suite.
387 The results show that directly using these algorithms out of the box only succeeds for the simplest tasks.
388



391
392
393
394
395
396
397
398
399
400
401 **Figure 7: Training on the test goals.** Training on the test goals improves both the returns and success
402 achieved by the best agents. However as the number of cubes and the complexity of the tasks increase, current
403 algorithms are not able to achieve a non zero success.
404

405 Although this setup does not directly evaluate generalization, it makes the problem of building gen-
406 eral agents much more approachable. For instance, researchers could study various design choices
407 or estimate whether an architecture is even capable of representing the solution to a complex task.
408 Finally, because of the diversity of tasks, spanning a wide range of difficulties and reward formu-
409 lations, this protocol is itself a useful benchmark for many RL fields such as goal conditioned,
410 hierarchical and multi-task RL.
411

412 **7 BENCHMARKING AND DISCUSSION.**
413
414

415 In our experiments, we present benchmarking results for existing algorithms suited for the self-
416 supervised as well as the supervised protocol. All experimental results are reported across three
417 seeds. We also attempt to solve tasks using large language models. In Appendix B, we plot the
418 training speed of various environments in BuilderBench.
419

420 **Multi-task self-supervised protocol.** We implemented four algorithms, sampling for learnabil-
421 ity (SFL (Rutherford et al., 2024)), maximum entropy gain exploration (MEGA (Pitis et al., 2020)),
422 upside down RL (UDRL) (Schmidhuber, 2020) and random network distillation (RND) (Burda
423 et al., 2018). SFL and MEGA sample autotelic goals from previously visited states, for the agent
424 to learn to reach them. SFL is an unsupervised environment design (Dennis et al., 2020) algorithm,
425 which samples goals with the highest learnability (variance of success). MEGA is an unsupervised
426 goal sampling (Florensa et al., 2018; OpenAI et al., 2021) algorithm, which samples goals inversely
427 proportional to their visitation density. Both algorithms are implemented using proximal policy op-
428 timization (PPO) (Schulman et al., 2017). UDRL and RND are self-supervised algorithms. UDRL
429 learns to reach previously explored goals using hindsight relabelling (Andrychowicz et al., 2018)
430 and RND explores the environment using an intrinsic reward bonus. Both of these algorithms sam-
431 ple data collection goals using MEGA. All algorithms are trained in environments with one, two and
three cubes and the learned policies are tested on the respective tasks from the task-suite Appendix E
at various points during training. We report normalized episodic success and returns in Figure 6.

As seen in Figure 6, both algorithms achieve trivial performance on tasks with three cubes. MEGA is able to complete both tasks with one cube, and shows improvement on tasks with two cubes. While these results indicate that the tested algorithms are not directly scalable to complex tasks, it primarily underscores the inherent difficulty of the task setup itself. We believe that research in developing new algorithms (or revisiting old ones) is required to solve these tasks.

Single-task supervised protocol. For this protocol, we benchmark six RL algorithms, proximal policy optimization (PPO) (Schulman et al., 2017), soft actor critic (SAC) (Haarnoja et al., 2018), contrastive RL (CRL) (Eysenbach et al., 2022), random network distillation (RND) (Burda et al., 2019), bigger-regularized-optimistic (BRO) (Nauman et al., 2024) and graph-attention-network (GNN-ATT) (Ghasemipour et al., 2022). The benchmarking results on 17 tasks are provided in Figure 7. All experiments use dense rewards.

7.1 EVALUATING LARGE LANGUAGE MODELS

It has been shown that scaling pretraining and inference-time compute can significantly enhance the reasoning abilities of language models (Kaplan et al., 2020). To test whether the latest proprietary models can solve tasks from our task-suite, we evaluated ChatGPT-5³ and Gemini 2.5 Pro (Team, 2025) on all five tasks discussed in Section 5.1. Each model was provided with a descriptive prompt about the environment and the task. The goal of the model was to provide a high-level open-loop plan in language, such that following this plan would stably build the target structure. A simple example task with a correct solution was also included in the prompt (see Appendix D for the exact prompts and solutions). Figure 8 shows that both models, despite using inference-time compute, are not able to provide the correct high-level plan to solve any of the tasks. While this is not meant to be an extensive evaluation of current models’ abilities, it highlights how solving our tasks requires non-obvious steps of reasoning that are beyond what current models can achieve through scaling alone.

Figure 8: Evaluating language models on BuilderBench.

Task Name	ChatGPT-5	Gemini 2.5 Pro
T block	✗	✗
Four cube packing	✗	✗
Hexagonal Portal	✗	✗
Leaning tower	✗	✗
Maximum Overhang	✗	✗

8 LIMITATIONS AND CONCLUSION

Although BuilderBench contains tasks that require diverse and open-ended skills, it does not focus on evaluating other problems like stochasticity, partial observability, or multi-agent learning. However, we expect extending BuilderBench to incorporate these settings should be easy. Another limitation is that we have not provided implementations for all approaches for open-ended exploration which exist in the literature (see discussion in Section 2). This is outside the scope of the paper. The main aim of the paper is to present an effective benchmark to accelerate research on scalable and generalizable learning through open-ended exploration.

Developing agents that can learn through open-ended exploration and generalize across diverse tasks remains an open problem in AI. Current AI models are pretrained on human generated data. As a result, they largely lack the ability to explore and learn through interaction. We have designed BuilderBench, to accelerate research towards agents that learn to explore in an open-ended environment and generalize to diverse tasks. Tasks in BuilderBench are designed to elicit long-horizon planning and reasoning abilities, many implicitly requiring agents to solve problems in physics and mathematics. BuilderBench provides a common framework for studying problems like open-ended exploration, generalization and embodied reasoning. We expect that the resulting research will advance the development of agents that solve problems by interacting with the real world.

³<https://openai.com/index/introducing-gpt-5/>

9 REPRODUCIBILITY STATEMENT

All experiments in this paper are completely reproducible. We have attached our code for the simulator, the task-suite and the implementation of all algorithms as a part of the supplemental materials. Additionally, BuilderBench is based on MuJoCo (Todorov et al., 2012) and Jax (Bradbury et al., 2018) both of which are open-sourced libraries. For experiments using proprietary language models, we have provided the exact models Section 7.1 and the prompts Appendix D which were used for experiments.

REFERENCES

- Faseeh Ahmad, Esra Erdem, and Volkan Patoglu. A formal framework for robot construction problems: A hybrid planning approach, 2019. URL <https://arxiv.org/abs/1903.00745>.
- Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges, 2024. URL <https://arxiv.org/abs/2402.00157>.
- Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay, 2018. URL <https://arxiv.org/abs/1707.01495>.
- Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: an evaluation platform for general agents. *J. Artif. Int. Res.*, 47(1):253–279, May 2013. ISSN 1076-9757.
- Clément Bonnet, Daniel Luo, Donal John Byrne, Shikha Surana, Sasha Abramowitz, Paul Duckworth, Vincent Coyette, Laurence Illing Midgley, Elshadai Tegegn, Tristan Kalloniatis, Omayma Mahjoub, Matthew Macfarlane, Andries Petrus Smit, Nathan Grinsztajn, Raphael Boige, Cemylyn Neil Waters, Mohamed Ali Ali Mimouni, Ulrich Armel Mbou Sob, Ruan John de Kock, Siddharth Singh, Daniel Furelos-Blanco, Victor Le, Arnau Pretorius, and Alexandre Laterre. Jumanji: a diverse suite of scalable reinforcement learning environments in JAX. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=C4CxQmp9wc>.
- James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL <http://github.com/jax-ml/jax>.
- Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. URL <https://arxiv.org/abs/1606.01540>.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20*, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.
- Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation, 2018. URL <https://arxiv.org/abs/1810.12894>.
- Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=H1lJJnR5Ym>.
- Rich Caruana. *Multitask learning*, pp. 95–133. Kluwer Academic Publishers, USA, 1998. ISBN 0792380479.

- 540 Beth Casey, Nicole Andrews, Holly Schindle, Joanne Kersh, Alexandra Samper, and Juanita
541 Copley. The development of spatial skills through interventions involving block building ac-
542 tivities. *Cognition and Instruction - COGNITION INSTRUCT*, 26:269–309, 07 2008. doi:
543 10.1080/07370000802177177.
- 544 Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony
545 Robinson. One billion word benchmark for measuring progress in statistical language modeling,
546 2014. URL <https://arxiv.org/abs/1312.3005>.
- 547 Yi-Ling Cheng and Kelly S Mix. Spatial training improves children’s mathematics ability. *Journal*
548 *of cognition and development*, 15(1):2–11, 2014.
- 550 François Chollet. On the measure of intelligence, 2019. URL [https://arxiv.org/abs/](https://arxiv.org/abs/1911.01547)
551 [1911.01547](https://arxiv.org/abs/1911.01547).
- 552 Wei Chow, Jiageng Mao, Boyi Li, Daniel Seita, Vitor Campagnolo Guizilini, and Yue Wang. Phys-
553 bench: Benchmarking and enhancing vision-language models for physical world understand-
554 ing. In *The Thirteenth International Conference on Learning Representations*, 2025. URL
555 <https://openreview.net/forum?id=Q6a9W6kzv5>.
- 556 Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural genera-
557 tion to benchmark reinforcement learning, 2020. URL [https://arxiv.org/abs/1912.](https://arxiv.org/abs/1912.01588)
558 [01588](https://arxiv.org/abs/1912.01588).
- 559 DeepSeek-AI. Deepseek-v3 technical report, 2025. URL [https://arxiv.org/abs/2412.](https://arxiv.org/abs/2412.19437)
560 [19437](https://arxiv.org/abs/2412.19437).
- 561 Li Deng. The mnist database of handwritten digit images for machine learning research [best of
562 the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. doi: 10.1109/MSP.2012.
563 2211477.
- 564 Michael Dennis, Natasha Jaques, Eugene Vinitzky, Alexandre Bayen, Stuart Russell, Andrew Critch,
565 and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment
566 design. In *Proceedings of the 34th International Conference on Neural Information Processing*
567 *Systems, NIPS ’20*, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.
- 568 Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep
569 bidirectional transformers for language understanding, 2018. URL [http://arxiv.org/](http://arxiv.org/abs/1810.04805)
570 [abs/1810.04805](http://arxiv.org/abs/1810.04805). cite arxiv:1810.04805Comment: 13 pages.
- 571 Guneet S. Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for
572 few-shot image classification, 2020. URL <https://arxiv.org/abs/1909.02729>.
- 573 Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Go-explore:
574 a new approach for hard-exploration problems, 2021. URL [https://arxiv.org/abs/](https://arxiv.org/abs/1901.10995)
575 [1901.10995](https://arxiv.org/abs/1901.10995).
- 576 Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need:
577 Learning skills without a reward function. In *International Conference on Learning Representa-*
578 *tions*, 2019. URL <https://openreview.net/forum?id=SJx63jRqFm>.
- 579 Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learn-
580 ing as goal-conditioned reinforcement learning. In Alice H. Oh, Alekh Agarwal, Danielle Bel-
581 grave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.
582 URL <https://openreview.net/forum?id=vGQiu5sqUe3>.
- 583 Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation
584 of deep networks, 2017. URL <https://arxiv.org/abs/1703.03400>.
- 585 Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for
586 reinforcement learning agents, 2018. URL <https://arxiv.org/abs/1705.06366>.
- 587 Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep
588 data-driven reinforcement learning, 2021. URL <https://arxiv.org/abs/2004.07219>.

- 594 Seyed Kamyar Seyed Ghasemipour, Daniel Freeman, Byron David, Shixiang Shane Gu, Satoshi
595 Kataoka, and Igor Mordatch. Blocks assemble! learning to assemble with large-scale structured
596 reinforcement learning, 2022. URL <https://arxiv.org/abs/2203.13733>.
- 597 Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P. Adams, and Sergey Levine. Why
598 generalization in rl is difficult: epistemic pomdps and implicit partial observability. In *Proceed-*
599 *ings of the 35th International Conference on Neural Information Processing Systems, NIPS '21*,
600 Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.
- 602 Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic con-
603 trol. *ArXiv*, abs/1611.07507, 2016. URL [https://api.semanticscholar.org/](https://api.semanticscholar.org/CorpusID:2918187)
604 [CorpusID:2918187](https://api.semanticscholar.org/CorpusID:2918187).
- 605 Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy
606 learning: Solving long-horizon tasks via imitation and reinforcement learning. In Leslie Pack
607 Kaelbling, Danica Kragic, and Komei Sugiura (eds.), *Proceedings of the Conference on Robot*
608 *Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1025–1037. PMLR,
609 30 Oct–01 Nov 2020. URL [https://proceedings.mlr.press/v100/gupta20a.](https://proceedings.mlr.press/v100/gupta20a.html)
610 [html](https://proceedings.mlr.press/v100/gupta20a.html).
- 611 Naresh Gupta and Dana S. Nau. On the complexity of blocks-world planning. *Ar-*
612 *tificial Intelligence*, 56(2):223–254, 1992. ISSN 0004-3702. doi: [https://doi.org/10.](https://doi.org/10.1016/0004-3702(92)90028-V)
613 [1016/0004-3702\(92\)90028-V](https://doi.org/10.1016/0004-3702(92)90028-V). URL [https://www.sciencedirect.com/science/](https://www.sciencedirect.com/science/article/pii/000437029290028V)
614 [article/pii/000437029290028V](https://www.sciencedirect.com/science/article/pii/000437029290028V).
- 616 William H. Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela
617 Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations,
618 2019. URL <https://arxiv.org/abs/1907.13440>.
- 619 Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy
620 maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and An-
621 dreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*,
622 volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 10–15 Jul
623 2018. URL <https://proceedings.mlr.press/v80/haarnoja18b.html>.
- 624 Danijar Hafner. Benchmarking the spectrum of agent capabilities, 2022. URL [https://arxiv.](https://arxiv.org/abs/2109.06780)
625 [org/abs/2109.06780](https://arxiv.org/abs/2109.06780).
- 627 Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains
628 through world models, 2024. URL <https://arxiv.org/abs/2301.04104>.
- 629 Minh Heo, Youngwoon Lee, Doohyun Lee, and Joseph J. Lim. Furniturebench: Reproducible
630 real-world benchmark for long-horizon complex manipulation, 2023. URL [https://arxiv.](https://arxiv.org/abs/2305.12821)
631 [org/abs/2305.12821](https://arxiv.org/abs/2305.12821).
- 633 Edward Hughes, Michael D Dennis, Jack Parker-Holder, Feryal Behbahani, Aditi Mavalankar, Yuge
634 Shi, Tom Schaul, and Tim Rocktäschel. Position: Open-endedness is essential for artificial super-
635 human intelligence. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria
636 Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International*
637 *Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*,
638 pp. 20597–20616. PMLR, 21–27 Jul 2024. URL [https://proceedings.mlr.press/](https://proceedings.mlr.press/v235/hughes24a.html)
639 [v235/hughes24a.html](https://proceedings.mlr.press/v235/hughes24a.html).
- 640 Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot
641 learning benchmark and learning environment, 2019. URL [https://arxiv.org/abs/](https://arxiv.org/abs/1909.12271)
642 [1909.12271](https://arxiv.org/abs/1909.12271).
- 643 Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik
644 Narasimhan. Swe-bench: Can language models resolve real-world github issues?, 2024. URL
645 <https://arxiv.org/abs/2310.06770>.
- 646 Leslie Pack Kaelbling. Learning to achieve goals. In *International Joint Conference on Artificial In-*
647 *telligence*, 1993. URL <https://api.semanticscholar.org/CorpusID:5538688>.

- 648 Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child,
649 Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language
650 models, 2020. URL <https://arxiv.org/abs/2001.08361>.
- 651 Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot gener-
652 alisation in deep reinforcement learning. *J. Artif. Int. Res.*, 76, May 2023. ISSN 1076-9757. doi:
653 10.1613/jair.1.14174. URL <https://doi.org/10.1613/jair.1.14174>.
- 654 Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt
655 Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, Aniruddha Kembhavi, Abhinav Gupta, and Ali
656 Farhadi. Ai2-thor: An interactive 3d environment for visual ai, 2022. URL <https://arxiv.org/abs/1712.05474>.
- 657 Sotetsu Koyamada, Shinri Okano, Soichiro Nishimori, Yu Murata, Keigo Habara, Haruka Kita, and
658 Shin Ishii. Pgx: Hardware-accelerated parallel game simulators for reinforcement learning. In
659 *Advances in Neural Information Processing Systems*, volume 36, pp. 45716–45743, 2023.
- 660 Heinrich Küttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatichi, Edward
661 Grefenstette, and Tim Rocktäschel. The nethack learning environment, 2020. URL <https://arxiv.org/abs/2006.13760>.
- 662 Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning
663 through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. doi: 10.1126/
664 science.aab3050. URL <https://www.science.org/doi/abs/10.1126/science.aab3050>.
- 665 Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhut-
666 dinov. Efficient exploration via state marginal matching, 2020. URL <https://arxiv.org/abs/1906.05274>.
- 667 Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski,
668 Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo,
669 Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reason-
670 ing problems with language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave,
671 and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL
672 <https://openreview.net/forum?id=IFXTZERXdm7>.
- 673 Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and
674 Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-
675 training. *ArXiv*, abs/2210.00030, 2022. URL <https://api.semanticscholar.org/CorpusID:252683397>.
- 676 Mikel Malagón, Josu Ceberio, and Jose A. Lozano. Craftium: Bridging flexibility and efficiency
677 for rich 3d single- and multi-agent environments, 2025. URL <https://arxiv.org/abs/2407.03969>.
- 678 Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson,
679 Samuel Coward, and Jakob Foerster. Craftax: a lightning-fast benchmark for open-ended re-
680 inforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*,
681 ICML 24. JMLR.org, 2024.
- 682 Michael Matthews, Michael Beukman, Chris Lu, and Jakob Nicolaus Foerster. Kinetix: Inves-
683 tigating the training of general agents through open-ended physics-based control tasks. In
684 *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=zCxGCdzreM>.
- 685 MCB, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen,
686 and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language
687 learning, 2019. URL <https://arxiv.org/abs/1810.08272>.
- 688 MCB, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal,
689 Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable rein-
690 forcement learning environments for goal-oriented tasks, 2023. URL <https://arxiv.org/abs/2306.13831>.

- 702 Andrew Melnik, Luca Lach, Matthias Plappert, Timo Korthals, Robert Haschke, and Helge J. Ritter.
703 Using tactile sensing to improve the sample efficiency and performance of deep deterministic
704 policy gradients for simulated in-hand manipulation tasks. *Frontiers in Robotics and AI*, 8, 2021.
705 URL <https://api.semanticscholar.org/CorpusID:235663648>.
706
- 707 Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan
708 Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013. URL
709 <https://arxiv.org/abs/1312.5602>.
- 710 Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhi-
711 wei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale
712 demonstrations, 2021. URL <https://arxiv.org/abs/2107.14483>.
713
- 714 Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Big-
715 ger, regularized, optimistic: scaling for compute and sample-efficient continuous control, 2024.
716 URL <https://arxiv.org/abs/2405.16158>.
- 717 Alexander Nikulin, Vladislav Kurenkov, Ilya Zisman, Viacheslav Sinii, Artem Agarkov, and Sergey
718 Kolesnikov. XLand-minigrid: Scalable meta-reinforcement learning environments in JAX.
719 In *Intrinsically-Motivated and Open-Ended Learning Workshop @NeurIPS2023*, 2023. URL
720 <https://openreview.net/forum?id=xALDC4aHGz>.
721
- 722 OpenAI. Openai o1 system card, 2024. URL <https://arxiv.org/abs/2412.16720>.
723
- 724 OpenAI OpenAI, Matthias Plappert, Raul Sampedro, Tao Xu, Ilge Akkaya, Vineet Kosaraju, Pe-
725 ter Welinder, Ruben D’Sa, Arthur Petron, Henrique P. d. O. Pinto, Alex Paino, Hyeonwoo Noh,
726 Lilian Weng, Qiming Yuan, Casey Chu, and Wojciech Zaremba. Asymmetric self-play for au-
727 tomatic goal discovery in robotic manipulation, 2021. URL <https://arxiv.org/abs/2101.04882>.
728
- 729 Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via
730 bootstrapped dqn, 2016. URL <https://arxiv.org/abs/1602.04621>.
731
- 732 Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V. Hafner. Intrinsic motivation systems for au-
733 tonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–
734 286, 2007. doi: 10.1109/TEVC.2006.890271.
- 735 Mike Paterson, Yuval Peres, Mikkel Thorup, Peter Winkler, and Uri Zwick. Maximum overhang,
736 2007. URL <https://arxiv.org/abs/0707.0093>.
737
- 738 Silviu Pitis, Harris Chan, Stephen Zhao, Bradly Stadie, and Jimmy Ba. Maximum entropy gain
739 exploration for long horizon multi-goal reinforcement learning. In *Proceedings of the 37th Inter-
740 national Conference on Machine Learning, ICML’20*. JMLR.org, 2020.
- 741 Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell,
742 Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech
743 Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request
744 for research, 2018. URL <https://arxiv.org/abs/1802.09464>.
745
- 746 Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agar-
747 wal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya
748 Sutskever. Learning transferable visual models from natural language supervision, 2021. URL
749 <https://arxiv.org/abs/2103.00020>.
- 750 Sai Rajeswar, Pietro Mazzaglia, Tim Verbelen, Alexandre Piché, Bart Dhoedt, Aaron Courville, and
751 Alexandre Lacoste. Mastering the unsupervised reinforcement learning benchmark from pixels.
752 In *Proceedings of the 40th International Conference on Machine Learning, ICML’23*. JMLR.org,
753 2023.
754
- 755 Stuart Reifel. Block construction: Children’s developmental landmarks in representation of space.
Young children, 1984.

- 756 Ronan Riochet, Mario Ynocente Castro, Mathieu Bernard, Adam Lerer, Rob Fergus, Véronique
757 Izard, and Emmanuel Dupoux. Intphys: A framework and benchmark for visual intuitive physics
758 reasoning, 2020. URL <https://arxiv.org/abs/1803.07616>.
- 759
760 Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng
761 Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-
762 Fei. Imagenet large scale visual recognition challenge, 2015. URL <https://arxiv.org/abs/1409.0575>.
- 763
764 Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3
765 edition, 2010.
- 766
767 Alexander Rutherford, Michael Beukman, Timon Willi, Bruno Lacerda, Nick Hawes, and
768 Jakob Nicolaus Foerster. No regrets: Investigating and improving regret approximations for cur-
769 riculum discovery. In *The Thirty-eighth Annual Conference on Neural Information Processing*
770 *Systems*, 2024. URL <https://openreview.net/forum?id=iEeiZlTbts>.
- 771
772 Juergen Schmidhuber. Reinforcement learning upside down: Don’t predict rewards – just map them
773 to actions, 2020. URL <https://arxiv.org/abs/1912.02875>.
- 774
775 Jürgen Schmidhuber. Evolutionary principles in self-referential learning, or on learning how to learn:
776 The meta-meta-. hook, 1987. URL <https://api.semanticscholar.org/CorpusID:264351059>.
- 777
778 John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy
779 optimization algorithms, 2017. URL <https://arxiv.org/abs/1707.06347>.
- 780
781 Olivier Sigaud, Gianluca Baldassarre, Cedric Colas, Stephane Doncieux, Richard Duro, Pierre-Yves
782 Oudeyer, Nicolas Perrin-Gilbert, and Vieri Giuliano Santucci. A definition of open-ended learn-
783 ing problems for goal-conditioned agents, 2024. URL <https://arxiv.org/abs/2311.00344>.
- 784
785 David Silver and Richard S Sutton. Welcome to the era of experience. *Google AI*, 2025.
- 786
787 David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche,
788 Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman,
789 Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine
790 Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go
791 with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016. doi:
792 10.1038/nature16961.
- 793
794 Dorothy G. Singer, Roberta Michnick Golinkoff, and Kathy Hirsh-Pasek. *Play = Learning: How*
795 *Play Motivates and Enhances Children’s Cognitive and Social-Emotional Growth*. Oxford
796 University Press, 09 2006. ISBN 9780195304381. doi: 10.1093/acprof:oso/9780195304381.
797 001.0001. URL <https://doi.org/10.1093/acprof:oso/9780195304381.001.0001>.
- 798
799 Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning.
800 In *Proceedings of the 31st International Conference on Neural Information Processing Sys-*
801 *tems, NIPS’17*, pp. 4080–4090, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN
802 9781510860964.
- 803
804 Elizabeth S. Spelke and Katherine D. Kinzler. Core knowledge. *Developmental Science*, 10(1):
805 89–96, 2007. doi: 10.1111/j.1467-7687.2007.00569.x.
- 806
807 Kenneth Stanley. Open-endedness: The last grand challenge you’ve
808 never heard of. [https://www.uber.com/blog/research/
809 open-endedness-the-last-grand-challenge-youve-never-heard-of/](https://www.uber.com/blog/research/open-endedness-the-last-grand-challenge-youve-never-heard-of/),
December 2017. Uber Engineering Blog.
- 810
811 Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. The distracting control
812 suite – a challenging benchmark for reinforcement learning from pixels, 2021. URL <https://arxiv.org/abs/2101.02722>.

- 810 Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press,
811 second edition, 2018. URL <http://incompleteideas.net/book/the-book-2nd.html>.
812
- 813 Haoran Tang, Rein Houthoofd, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman,
814 Filip De Turck, and Pieter Abbeel. Exploration: A study of count-based exploration for deep
815 reinforcement learning, 2017. URL <https://arxiv.org/abs/1611.04717>.
816
- 817 Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Bud-
818 den, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Ried-
819 miller. Deepmind control suite, 2018. URL <https://arxiv.org/abs/1801.00690>.
820
- 821 Adaptive Agent Team. Human-timescale adaptation in an open-ended task space, 2023. URL
822 <https://arxiv.org/abs/2301.07608>.
- 823 Gemini Team. Gemini: A family of highly capable multimodal models, 2025. URL <https://arxiv.org/abs/2312.11805>.
824
- 825 Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob
826 Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, Nat McAleese,
827 Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Roberta Raileanu, Steph Hughes-
828 Fitt, Valentin Dalibard, and Wojciech Marian Czarnecki. Open-ended learning leads to generally
829 capable agents, 2021. URL <https://arxiv.org/abs/2107.12808>.
830
- 831 Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control.
832 In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033.
833 IEEE, 2012. doi: 10.1109/IROS.2012.6386109.
- 834 Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée
835 Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Ar-
836 mand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation
837 language models, 2023. URL <https://arxiv.org/abs/2302.13971>.
838
- 839 Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kamb-
840 hampati. Planbench: An extensible benchmark for evaluating large language models on planning
841 and reasoning about change, 2023. URL <https://arxiv.org/abs/2206.10498>.
- 842 Brian N. Verdine, Roberta M. Golinkoff, Kathryn Hirsh-Pasek, Nora S. Newcombe, Andrew T.
843 Filipowicz, and Alicia Chang. Deconstructing building blocks: Preschoolers’ spatial assembly
844 performance relates to early mathematical skills. *Child Development*, 85(3):1062–1076, 2014.
845 ISSN 00093920, 14678624. URL <http://www.jstor.org/stable/24031910>.
846
- 847 Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Match-
848 ing networks for one shot learning. In *Proceedings of the 30th International Conference on Neural
849 Information Processing Systems, NIPS’16*, pp. 3637–3645, Red Hook, NY, USA, 2016. Curran
850 Associates Inc. ISBN 9781510838819.
- 851 Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Joseph Dudzik,
852 Junyoung Chung, David Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan
853 Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, L. Sifre, Trevor Cai, John P. Agapiou, Max
854 Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David
855 Budden, Yury Sulsky, James Molloy, Tom Le Paine, Caglar Gulcehre, Ziyun Wang, Tobias Pfaff,
856 Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom
857 Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver.
858 Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575:350 – 354,
859 2019. URL <https://api.semanticscholar.org/CorpusID:204972004>.
- 860 Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani
861 Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto,
862 Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large lan-
863 guage models. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL
<https://openreview.net/forum?id=yzkSU5zdwD>. Survey Certification.

864 Mark Wexler, Stephen M Kosslyn, and Alain Berthoz. Motor processes in mental rotation. *Cogni-*
865 *tion*, 68(1):77–94, 1998.

866

867 Lance Ying, Katherine M. Collins, Prafull Sharma, Cedric Colas, Kaiya Ivy Zhao, Adrian Weller,
868 Zenna Tavares, Phillip Isola, Samuel J. Gershman, Jacob D. Andreas, Thomas L. Griffiths, Fran-
869 cois Chollet, Kelsey R. Allen, and Joshua B. Tenenbaum. Assessing adaptive world models in
870 machines with novel games, 2025. URL <https://arxiv.org/abs/2507.12821>.

871 Zhonghan Zhao, Wenhao Chai, Xuan Wang, Li Boyi, Shengyu Hao, Shidong Cao, Tian Ye, and
872 Gaoang Wang. See and think: Embodied agent in virtual environment, 2024. URL <https://arxiv.org/abs/2311.15209>.

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

Outline of Appendices. In Appendix A, we include details about the tasks including reward functions, episode length. In Appendix B, we add speed tests for two algorithms trained using the BuilderBench simulator. In Appendix D, we provide the exact prompts and the answers provided by LLMs for experiments in Section 7.1. Lastly, in Appendix E, we provide a visualization of all tasks in the BuilderBench task-suite.

A ENVIRONMENT DETAILS

A.1 EPISODE LENGTHS

The episode length depends on the number of cubes present in the environment (N). For supervised tasks, the episode length is $100 + 100 \times N$ and for self-supervised tasks, the episode length is $500 \times N$.

A.2 REWARD FUNCTIONS

There are two types of reward functions provided in the benchmark, sparse and dense. The sparse reward is equal to -1 for all timesteps where the cubes do not form the target structure and 0 if the cubes form the target structure. Cubes are said to form the target structure if the distance between each cube and its corresponding target is less than 2 centimeters. At each timestep, every cube is assigned a specific target position from the target structure. This assignment is calculated such that the total sum of distances between the cubes and their assigned targets is minimized. This is a convex optimization problem and can be solved efficiently with GPUs using the hungarian algorithm implemented in jax [Bradbury et al. \(2018\)](#). The sparse rewards are calculated by checking whether the best assigned distances are all less than 2 centimeters. The dense rewards are calculated by applying $1 - \tanh(x)$ to the best assigned distances and summing them over all cubes. As distances tend to zero, the reward tends to N (number of cubes in the environment).

The permutation sensitive variants of both the sparse and dense rewards are calculated similarly, but without solving the assignment problem. Each cube is already assigned a specific target position by the task, and this assignment is used to estimate both the dense and the sparse rewards.

B TRAINING SPEED.

In Figure 9, we plot the average environment steps per second for PPO (Figure 7) and MEGA (Figure 6). The PPO experiments were run on a single A100 GPU with 12 CPU threads and the MEGA experiments were run on a single A6000 GPU with 12 CPU threads. The speed of the training depends on many factors like number of parallel environments, the update to data ratio, type of GPU and the number of CPU threads. The speed also depends on the environment step-time, number of cubes and the type of interactions in the environments (interactions with more active contacts typically take longer time to execute). An example of this can be seen in the right plot in Figure 9. For the self-supervised protocol, the environment step-time is set to be small for better physics approximation. MEGA with one block learns to pick the block up and move it around which we found to be inversely correlated with speed. The speeds could be scaled with more CPU threads, parallel environments, and faster algorithms for physics approximation.

Although not precise, comparing these speeds with fig. 2 from [Matthews et al. \(2024\)](#) shows that BuilderBench is between $10^2 - 10^3$ times faster than Minecraft ([Guss et al., 2019](#)), 2 - 20 faster than Nethack and Procgen ([Küttler et al., 2020](#); [Cobbe et al., 2020](#)), and 10 - 10^2 times faster than Crafter ([Hafner, 2022](#)).

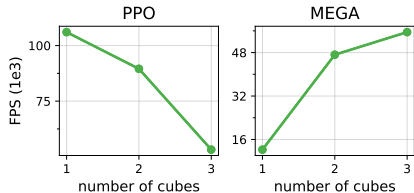


Figure 9: **Environment steps per second during PPO and MEGA training runs**

Table 1: Qualitative Comparison of BuilerBench with previous benchmarks

Benchmark	Extensive reasoning task-suite	Hardware accelerated	Primary Focus
BuilerBench (ours)	✓	✓	1. Self-supervised exploration 2. Zero shot reasoning 3. Learning from scratch
ManiSkill (Mu et al., 2021)	✗	✓	1. Robot learning 2. Fast simulation and training for manipulation tasks.
FurnitureBench (Heo et al., 2023)	✗	✗	1. Long-horizon assembly 2. Imitation learning and Offline RL
BabyAI (MCB et al., 2019)	✗	✗	1. Human in the loop learning 2. Language grounded reasoning
Kinetix (Matthews et al., 2025)	✗	✓	1. Zero shot reasoning 2. Unsupervised environment design
Craftium (Malagón et al., 2025)	✗	✗	1. LLM based agents 2. Code based environment generation
MiniGrid (MCB et al., 2023)	✗	✗	1. Exploration 2. Curriculum Learning

C QUALITATIVE COMPARISON WITH PRIOR WORK.

In this section, we provide a qualitative comparison of BuilderBench with prior work. Table 1 compares the primary focus of BuilderBench with various benchmarks, *highlighting the need for a new benchmark that focuses on learning to reason via exploration and trial and error from scratch.*

We argue that the components of BuilderBench are necessary to facilitate research on this problem:

Tasks which demonstrably require reasoning. We have provided an extensive task suite which require skills such as logical reasoning (commutativity and associativity of pick and place ordering), geometrical reasoning (maximizing overhangs, packing problems) and intuitive physics (gravity, friction, toppling, balancing). Tasks also require reasoning about counterweights, buttresses, and temporary scaffolding. Designing such tasks is not trivial. In comparison, benchmarks like ManiSkill (Mu et al., 2021) focus on simpler tasks which require pick and place primitives.

Scaling interaction speed. Training agents to solve such complex problems will presumably require large amounts of interaction data. While there are complex environments like MineCraft (Guss et al., 2019; Malagón et al., 2025) that are also centered around building blocks, the simulators for these environments are about $10^2 - 10^3$ times slower than BuilderBench.

Focus on training from trial and error. BuilderBench focuses on agents that learn purely from trial and error whereas benchmarks like FurnitureBench (Heo et al., 2023) focus on solve long-horizon assembly using demonstrations and offline data.

D EVALUATING LANGUAGE MODELS: PROMPTS AND SOLUTIONS

D.1 T BLOCK

User

You are an agent that controls a two-finger robotic gripper (Robotiq 2F-85) which can navigate a constrained 3D space using position actuators for controlling x,y,z directions and yaw. The robot hand has two fingers with maximum width equal to 0.085 meters. The environment consists of a variable number of cubes of size 0.04 meters. The environment is implemented in MuJoCo and approximates Newtonian physics. All length position coordinates have units in meters and the yaw will be measured in radians.

Problem description - You will be given the positions and the orientations of all the N cubes in the environment. You will also be given the target center positions of K cubes, where K can be less or equal to N. The target center positions correspond to a target structure that you need to build such that it always remains stable. The structures which are in unstable equilibria, that would fall apart after applying minimal perturbation, are not considered stable. Your goal is to provide high level language instructions for the gripper to follow such that it can build the stable target structure. All steps should be physically possible and implementing those steps one after the other should stably result in building the target structure. Only provide a list of an open loop plan consisting of simple instructions, for example, pick cube 1, place cube 1 at 0.1,0,0.02 with yaw 0, rotate cube 2 by pi/3 radians, nudge the cube at [0.1,0,0.02] to [0.12,0,0.02], etc. You can think before providing the final answer but please separate the final answer by saying "Final answer:" before providing the ordered plan of high level instructions.

The following is an example of a simple task followed by its solution:

Task:

N = 1

K = 1

Start location = [0.05, 0.0, 0.02]

Target location = [0.1, 0.0, 0.02]

Final answer:

1) Pick up the cube from [0.05, 0.0, 0.02].

2) Place the cube at [0.1, 0.0, 0.02] with a yaw of 0 radians.

Here is the task:

N = 3

K = 3

Start location = [[0.05, -0.08, 0.02], [0.05, 0. , 0.02], [0.05, 0.08, 0.02]]

Target location = [[0.1, 0.02, 0.06], [0.1 , -0.02, 0.06], [0.1 , 0. , 0.02]]

ChatGPT-5

Pick up the cube from [0.05, 0.0, 0.02].

Place the cube at [0.1, 0.0, 0.02] with a yaw of 0 radians.

Pick up the cube from [0.05, -0.08, 0.02].

Place the cube at [0.1, -0.02, 0.06] with a yaw of 0 radians.

Pick up the cube from [0.05, 0.08, 0.02].

Place the cube at [0.1, 0.02, 0.06] with a yaw of 0 radians.

1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133

Gemini 2.5 Pro

Pick up the cube from [0.05, 0.0, 0.02].
Place the cube at [0.1, 0.0, 0.02] with a yaw of 0 radians.
Pick up the cube from [0.05, -0.08, 0.02].
Place the cube at [0.1, -0.02, 0.06] with a yaw of 0 radians.
Pick up the cube from [0.05, 0.08, 0.02].
Place the cube at [0.1, 0.02, 0.06] with a yaw of 0 radians.

1134 D.2 FOUR CUBE PACKING
1135

1136 **User**

1137
1138 You are an agent that controls a two-finger robotic gripper (Robotiq 2F-85) which can
1139 navigate a constrained 3D space using position actuators for controlling x,y,z directions
1140 and yaw. The robot hand has two fingers with maximum width equal to 0.085 meters. The
1141 environment consists of a variable number of cubes of size 0.04 meters. The environment
1142 is implemented in MuJoCo and approximates Newtonian physics. All length position
1143 coordinates have units in meters and the yaw will be measured in radians.

1144 Problem description - You will be given the positions and the orientations of all the N cubes
1145 in the environment. You will also be given the target center positions of K cubes, where K
1146 can be less or equal to N. The target center positions correspond to a target structure that
1147 you need to build such that it always remains stable. The structures which are in unstable
1148 equilibria, that would fall apart after applying minimal perturbation, are not considered
1149 stable. Your goal is to provide high level language instructions for the gripper to follow
1150 such that it can build the stable target structure. All steps should be physically possible
1151 and implementing those steps one after the other should stably result in building the target
1152 structure. Only provide a list of an open loop plan consisting of simple instructions, for
1153 example, pick cube 1, place cube 1 at 0.1,0,0.02 with yaw 0, rotate cube 2 by pi/3 radians,
1154 nudge the cube at [0.1,0,0.02] to [0.12,0,0.02], etc. You can think before providing the final
1155 answer but please separate the final answer by saying "Final answer:" before providing the
1156 ordered plan of high level instructions.

1157 The following is an example of a simple task followed by its solution:

1158 Task:

1159 N = 1

1160 K = 1

1161 Start location = [0.05, 0.0, 0.02]

1162 Target location = [0.1, 0.0, 0.02]

1163 Final answer:

1164 1) Pick up the cube from [0.05, 0.0, 0.02].

1165 2) Place the cube at [0.1, 0.0, 0.02] with a yaw of 0 radians.

1166 Here is the task:

1167 N = 4

1168 K = 4

1169 Start location = [[0.05, -0.12, 0.02], [0.05, -0.04, 0.02], [0.05, 0.04, 0.02], [0.05, 0.12, 0.02]

1170]
1171 Target location = [[0.1, 0.02828427, 0.02], [0.1, -0.02828427, 0.02], [0.12828427, 0.0,
1172 0.02], [0.07171573, 0.0, 0.02]]

1173 **ChatGPT-5**

1174 Pick up the cube from [0.05, -0.12, 0.02].

1175 Place the cube at [0.1, -0.02828427, 0.02] with a yaw of 0 radians.

1176 Pick up the cube from [0.05, 0.12, 0.02].

1177 Place the cube at [0.1, 0.02828427, 0.02] with a yaw of 0 radians.

1178 Pick up the cube from [0.05, -0.04, 0.02].

1179 Place the cube at [0.07171573, 0.0, 0.02] with a yaw of 0 radians.

1180 Pick up the cube from [0.05, 0.04, 0.02].

1181 Place the cube at [0.12828427, 0.0, 0.02] with a yaw of 0 radians.

1182
1183
1184
1185
1186
1187

1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241

Gemini 2.5 Pro

Pick up the cube from [0.05, -0.12, 0.02].
Place the cube at [0.1, 0.02828427, 0.02] with a yaw of 0 radians.
Pick up the cube from [0.05, -0.04, 0.02].
Place the cube at [0.1, -0.02828427, 0.02] with a yaw of 0 radians.
Pick up the cube from [0.05, 0.04, 0.02].
Place the cube at [0.12828427, 0.0, 0.02] with a yaw of 0 radians.
Pick up the cube from [0.05, 0.12, 0.02].
Place the cube at [0.07171573, 0.0, 0.02] with a yaw of 0 radians.

D.3 HEXAGONAL PORTAL

User

You are an agent that controls a two-finger robotic gripper (Robotiq 2F-85) which can navigate a constrained 3D space using position actuators for controlling x,y,z directions and yaw. The robot hand has two fingers with maximum width equal to 0.085 meters. The environment consists of a variable number of cubes of size 0.04 meters. The environment is implemented in MuJoCo and approximates Newtonian physics. All length position coordinates have units in meters and the yaw will be measured in radians.

Problem description - You will be given the positions and the orientations of all the N cubes in the environment. You will also be given the target center positions of K cubes, where K can be less or equal to N . The target center positions correspond to a target structure that you need to build such that it always remains stable. The structures which are in unstable equilibria, that would fall apart after applying minimal perturbation, are not considered stable. Your goal is to provide high level language instructions for the gripper to follow such that it can build the stable target structure. All steps should be physically possible and implementing those steps one after the other should stably result in building the target structure. Only provide a list of an open loop plan consisting of simple instructions, for example, pick cube 1, place cube 1 at $[0.1,0,0.02]$ with yaw 0, rotate cube 2 by $\pi/3$ radians, nudge the cube at $[0.1,0,0.02]$ to $[0.12,0,0.02]$, etc. You can think before providing the final answer but please separate the final answer by saying "Final answer:" before providing the ordered plan of high level instructions.

The following is an example of a simple task followed by its solution:

Task:

$N = 1$

$K = 1$

Start location = $[0.05, 0.0, 0.02]$

Target location = $[0.1, 0.0, 0.02]$

Final answer:

1) Pick up the cube from $[0.05, 0.0, 0.02]$.

2) Place the cube at $[0.1, 0.0, 0.02]$ with a yaw of 0 radians.

Here is the task:

$N = 8$

$K = 8$

Start location = [$[0.05, -0.24, 0.02]$, $[0.05, -0.18, 0.02]$, $[0.05, -0.12, 0.02]$, $[0.05, -0.04, 0.02]$, $[0.05, 0.04, 0.02]$, $[0.05, 0.12, 0.02]$, $[0.05, 0.18, 0.02]$, $[0.05, 0.24, 0.02]$]

Target location = [$[0.1, 0.02, 0.02]$, $[0.1, -0.02, 0.02]$, $[0.1, 0.04, 0.06]$, $[0.1, -0.04, 0.06]$, $[0.1, 0.02, 0.1]$, $[0.1, -0.02, 0.1]$, $[0.1, 0.1, 0.02]$, $[0.1, -0.1, 0.02]$]

ChatGPT-5

Pick up the cube from $[0.05, 0.18, 0.02]$. Place the cube at $[0.1, 0.02, 0.02]$ with a yaw of 0 radians.

Pick up the cube from $[0.05, 0.24, 0.02]$. Place the cube at $[0.1, 0.1, 0.02]$ with a yaw of 0 radians.

Pick up the cube from $[0.05, 0.12, 0.02]$. Place the cube at $[0.1, 0.04, 0.06]$ with a yaw of 0 radians.

Pick up the cube from $[0.05, 0.04, 0.02]$. Place the cube at $[0.1, 0.02, 0.1]$ with a yaw of 0 radians.

Pick up the cube from $[0.05, -0.18, 0.02]$. Place the cube at $[0.1, -0.02, 0.02]$ with a yaw of 0 radians.

1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349

Pick up the cube from [0.05, -0.24, 0.02]. Place the cube at [0.1, -0.1, 0.02] with a yaw of 0 radians.
Pick up the cube from [0.05, -0.12, 0.02]. Place the cube at [0.1, -0.04, 0.06] with a yaw of 0 radians.
Pick up the cube from [0.05, -0.04, 0.02]. Place the cube at [0.1, -0.02, 0.1] with a yaw of 0 radians.

Gemini 2.5 Pro

Pick up the cube from [0.05, -0.24, 0.02].
Place the cube at [0.1, 0.02, 0.02] with a yaw of 0 radians.
Pick up the cube from [0.05, -0.18, 0.02].
Place the cube at [0.1, -0.02, 0.02] with a yaw of 0 radians.
Pick up the cube from [0.05, -0.12, 0.02].
Place the cube at [0.1, 0.1, 0.02] with a yaw of 0 radians.
Pick up the cube from [0.05, -0.04, 0.02].
Place the cube at [0.1, -0.1, 0.02] with a yaw of 0 radians.
Pick up the cube from [0.05, 0.04, 0.02].
Place the cube at [0.1, 0.04, 0.06] with a yaw of 0 radians.
Pick up the cube from [0.05, 0.12, 0.02].
Place the cube at [0.1, -0.04, 0.06] with a yaw of 0 radians.
Pick up the cube from [0.05, 0.18, 0.02].
Place the cube at [0.1, 0.02, 0.1] with a yaw of 0 radians.
Pick up the cube from [0.05, 0.24, 0.02].
Place the cube at [0.1, -0.02, 0.1] with a yaw of 0 radians.

1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403

D.4 LEANING TOWER

User

You are an agent that controls a two-finger robotic gripper (Robotiq 2F-85) which can navigate a constrained 3D space using position actuators for controlling x,y,z directions and yaw. The robot hand has two fingers with maximum width equal to 0.085 meters. The environment consists of a variable number of cubes of size 0.04 meters. The environment is implemented in MuJoCo and approximates Newtonian physics. All length position coordinates have units in meters and the yaw will be measured in radians.

Problem description - You will be given the positions and the orientations of all the N cubes in the environment. You will also be given the target center positions of K cubes, where K can be less or equal to N. The target center positions correspond to a target structure that you need to build such that it always remains stable. The structures which are in unstable equilibria, that would fall apart after applying minimal perturbation, are not considered stable. Your goal is to provide high level language instructions for the gripper to follow such that it can build the stable target structure. All steps should be physically possible and implementing those steps one after the other should stably result in building the target structure. Only provide a list of an open loop plan consisting of simple instructions, for example, pick cube 1, place cube 1 at 0.1,0,0.02 with yaw 0, rotate cube 2 by pi/3 radians, nudge the cube at [0.1,0,0.02] to [0.12,0,0.02], etc. You can think before providing the final answer but please separate the final answer by saying "Final answer:" before providing the ordered plan of high level instructions.

The following is an example of a simple task followed by its solution:

Task:

N = 1

K = 1

Start location = [0.05, 0.0, 0.02]

Target location = [0.1, 0.0, 0.02]

Final answer:

1) Pick up the cube from [0.05, 0.0, 0.02].

2) Place the cube at [0.1, 0.0, 0.02] with a yaw of 0 radians.

Here is the task:

N = 9

K = 9

Start location = [[0.05, -0.3, 0.02], [0.05, -0.24, 0.02], [0.05, -0.16, 0.02], [0.05, -0.08, 0.02], [0.05, 0.0, 0.02], [0.05, 0.08, 0.02], [0.05, 0.16, 0.02], [0.05, 0.24, 0.02], [0.05, 0.3, 0.02]]

Target location = [[0.1, 0.0, 0.02], [0.1, -0.04, 0.02], [0.1, 0.02, 0.06], [0.1, -0.02, 0.06], [0.1, 0.04, 0.1], [0.1, 0.0, 0.1], [0.1, 0.01, 0.14], [0.1, 0.12, 0.02], [0.1, 0.16, 0.02]]

ChatGPT-5

Pick up the cube from [0.05, -0.3, 0.02]. Place the cube at [0.1, -0.04, 0.02] with a yaw of 0 radians.

Pick up the cube from [0.05, -0.24, 0.02]. Place the cube at [0.1, 0.0, 0.02] with a yaw of 0 radians.

Pick up the cube from [0.05, 0.24, 0.02]. Place the cube at [0.1, 0.12, 0.02] with a yaw of 0 radians.

Pick up the cube from [0.05, 0.3, 0.02]. Place the cube at [0.1, 0.16, 0.02] with a yaw of 0 radians.

Pick up the cube from [0.05, -0.16, 0.02]. Place the cube at [0.1, -0.02, 0.06] with a yaw of 0 radians.

1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457

Pick up the cube from [0.05, -0.08, 0.02]. Place the cube at [0.1, 0.02, 0.06] with a yaw of 0 radians.
Pick up the cube from [0.05, 0.0, 0.02]. Place the cube at [0.1, 0.0, 0.10] with a yaw of 0 radians.
Pick up the cube from [0.05, 0.08, 0.02]. Place the cube at [0.1, 0.04, 0.10] with a yaw of 0 radians.
Pick up the cube from [0.05, 0.16, 0.02]. Place the cube at [0.1, 0.01, 0.14] with a yaw of 0 radians.

Gemini 2.5 Pro

Pick up the cube from [0.05, -0.3, 0.02] and place it at [0.1, -0.04, 0.02] with a yaw of 0 radians.
Pick up the cube from [0.05, -0.24, 0.02] and place it at [0.1, 0.0, 0.02] with a yaw of 0 radians.
Pick up the cube from [0.05, -0.16, 0.02] and place it at [0.1, 0.12, 0.02] with a yaw of 0 radians.
Pick up the cube from [0.05, -0.08, 0.02] and place it at [0.1, 0.16, 0.02] with a yaw of 0 radians.
Pick up the cube from [0.05, 0.0, 0.02] and place it at [0.1, -0.02, 0.06] with a yaw of 0 radians.
Pick up the cube from [0.05, 0.08, 0.02] and place it at [0.1, 0.02, 0.06] with a yaw of 0 radians.
Pick up the cube from [0.05, 0.16, 0.02] and place it at [0.1, 0.0, 0.1] with a yaw of 0 radians.
Pick up the cube from [0.05, 0.24, 0.02] and place it at [0.1, 0.04, 0.1] with a yaw of 0 radians.
Pick up the cube from [0.05, 0.3, 0.02] and place it at [0.1, 0.01, 0.14] with a yaw of 0 radians.

1458 D.5 MAXIMUM OVERHANG
 1459
 1460
 1461

User

1462 You are an agent that controls a two-finger robotic gripper (Robotiq 2F-85) which can
 1463 navigate a constrained 3D space using position actuators for controlling x,y,z directions
 1464 and yaw. The robot hand has two fingers with maximum width equal to 0.085 meters. The
 1465 environment consists of a variable number of cubes of size 0.04 meters. The environment
 1466 is implemented in MuJoCo and approximates Newtonian physics. All length position
 1467 coordinates have units in meters and the yaw will be measured in radians.

1468 Problem description - You will be given the positions and the orientations of all the N cubes
 1469 in the environment. You will also be given the target center positions of K cubes, where K
 1470 can be less or equal to N. The target center positions correspond to a target structure that
 1471 you need to build such that it always remains stable. The structures which are in unstable
 1472 equilibria, that would fall apart after applying minimal perturbation, are not considered
 1473 stable. Your goal is to provide high level language instructions for the gripper to follow
 1474 such that it can build the stable target structure. All steps should be physically possible
 1475 and implementing those steps one after the other should stably result in building the target
 1476 structure. Only provide a list of an open loop plan consisting of simple instructions, for
 1477 example, pick cube 1, place cube 1 at 0.1,0,0.02 with yaw 0, rotate cube 2 by pi/3 radians,
 1478 nudge the cube at [0.1,0,0.02] to [0.12,0,0.02], etc. You can think before providing the final
 1479 answer but please separate the final answer by saying "Final answer:" before providing the
 1480 ordered plan of high level instructions.

1481 The following is an example of a simple task followed by its solution:
 1482

1483 Task:

1484 N = 1

1485 K = 1

1486 Start location = [0.05, 0.0, 0.02]

1487 Target location = [0.1, 0.0, 0.02]

1488 Final answer:

1489 1) Pick up the cube from [0.05, 0.0, 0.02].

1490 2) Place the cube at [0.1, 0.0, 0.02] with a yaw of 0 radians.
 1491

1492 Here is the task:

1493 N = 5

1494 K = 3

1495 Start location = [[0.05, -0.16, 0.02], [0.05, -0.08, 0.02], [0.05, 0.0, 0.02], [0.05, 0.08, 0.02],
 1496 [0.05, 0.16, 0.02]]

1497 Target location = [[0.1, 0.0, 0.02], [0.1, 0.031, 0.14], [0.1, 0.16, 0.14]]
 1498
 1499

ChatGPT-5

1500 Pick up the cube from [0.05, 0.0, 0.02]. Place the cube at [0.1, 0.0, 0.02] with a yaw of 0
 1501 radians.
 1502

1503 Pick up the cube from [0.05, -0.08, 0.02]. Place the cube at [0.1, 0.031, 0.02] with a yaw of
 1504 0 radians. (bottom of the pillar)

1505 Pick up the cube from [0.05, 0.08, 0.02]. Place the cube at [0.1, 0.031, 0.06] with a yaw of
 1506 0 radians. (middle of the pillar)

1507 Pick up the cube from [0.05, -0.16, 0.02]. Place the cube at [0.1, 0.031, 0.10] with a yaw of
 1508 0 radians. (top of the pillar)

1509 Pick up the cube from [0.05, 0.16, 0.02]. Place the cube at [0.1, 0.031, 0.14] with a yaw of
 1510 0 radians. (target at z=0.14 realized)
 1511

1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565

Gemini 2.5 Pro

It is not possible to generate a plan for this task. The target structure is physically unstable.

E TASKS AND CHALLENGES IN BUILDERBENCH

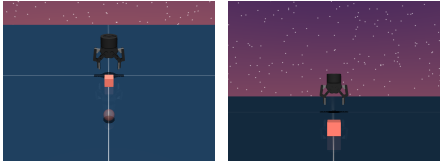

Task name	Task visualization	Abilities
cube-1-task1		1. Pick and Place
cube-1-task2		1. Pick and hold

Table 2: Tasks with 1 cube.

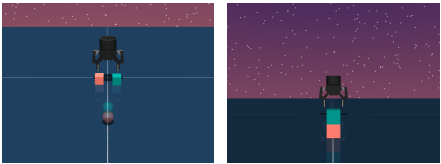
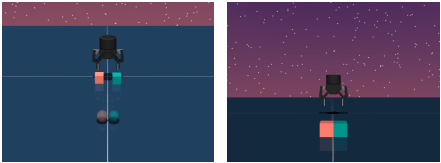



Task name	Task visualization	Abilities
cube-2-task1		1. Motor skills
cube-2-task2		1. Sequential-logic 2. Motor skills
cube-2-task3		1. Solving a packing problem 2. Sequential-logic 3. Motor skills
cube-2-task4		1. Two cube lift maneuver 2. Sequential-logic 3. Motor skills
cube-2-task5		1. Two cube lift maneuver 2. Sequential-logic 3. Motor skills

Table 3: Tasks with 2 cubes.

1620
 1621
 1622
 1623
 1624
 1625
 1626
 1627
 1628
 1629
 1630
 1631
 1632
 1633
 1634
 1635
 1636
 1637
 1638
 1639
 1640
 1641
 1642
 1643
 1644
 1645
 1646
 1647
 1648
 1649
 1650
 1651
 1652
 1653
 1654
 1655
 1656
 1657
 1658
 1659
 1660
 1661
 1662
 1663
 1664
 1665
 1666
 1667
 1668
 1669
 1670
 1671
 1672
 1673

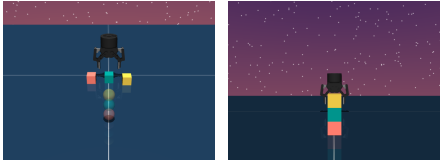
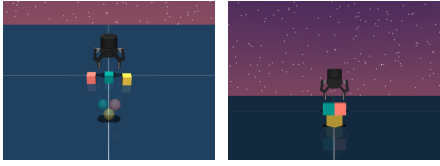



Task name	Task visualization	Abilities
cube-3-task1		<ol style="list-style-type: none"> 1. Sequential-logic 2. Motor skills
cube-3-task2		<ol style="list-style-type: none"> 1. Stable equilibria 2. Toppling 2. Motor skills
cube-3-task3		<ol style="list-style-type: none"> 1. Three cube lift maneuver 2. Sequential-logic 2. Motor skills
cube-3-task4		<ol style="list-style-type: none"> 1. Solving a packing problem 2. Sequential-logic 3. Motor skills
cube-3-task5		<ol style="list-style-type: none"> 1. Building support structures 2. Sequential-logic 3. Motor skills

Table 4: Tasks with 3 cubes.

1674
 1675
 1676
 1677
 1678
 1679
 1680
 1681
 1682
 1683
 1684
 1685
 1686
 1687
 1688
 1689
 1690
 1691
 1692
 1693
 1694
 1695
 1696
 1697
 1698
 1699
 1700
 1701
 1702
 1703
 1704
 1705
 1706
 1707
 1708
 1709
 1710
 1711
 1712
 1713
 1714
 1715
 1716
 1717
 1718
 1719
 1720
 1721
 1722
 1723
 1724
 1725
 1726
 1727

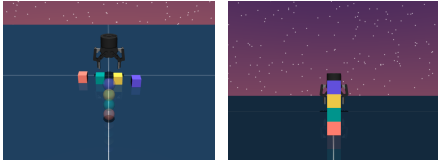
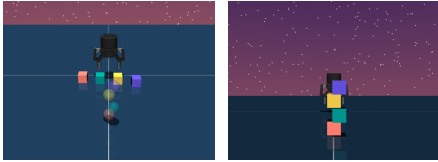
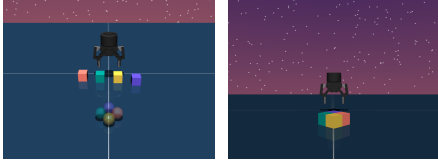
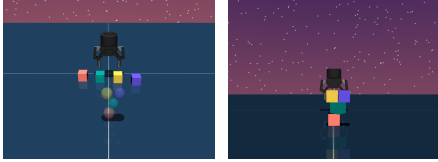

Task name	Task visualization	Abilities
cube-4-task1		<ol style="list-style-type: none"> 1. Sequential-logic 2. Motor skills
cube-4-task2		<ol style="list-style-type: none"> 1. Sequential-logic 2. Motor skills
cube-4-task3		<ol style="list-style-type: none"> 1. Solve a packing problem 2. Sequential-logic 2. Motor skills
cube-4-task4		<ol style="list-style-type: none"> 1. Unconstrained Maximum overhang 2. Sequential-logic 3. Motor skills
cube-4-task5		<ol style="list-style-type: none"> 1. Building support structures 2. Sequential-logic 3. Motor skills

Table 5: Tasks with 4 cubes.

1728
 1729
 1730
 1731
 1732
 1733
 1734
 1735
 1736
 1737
 1738
 1739
 1740
 1741
 1742
 1743
 1744
 1745
 1746
 1747
 1748
 1749
 1750
 1751
 1752
 1753
 1754
 1755
 1756
 1757
 1758
 1759
 1760
 1761
 1762
 1763
 1764
 1765
 1766
 1767
 1768
 1769
 1770
 1771
 1772
 1773
 1774
 1775
 1776
 1777
 1778
 1779
 1780
 1781

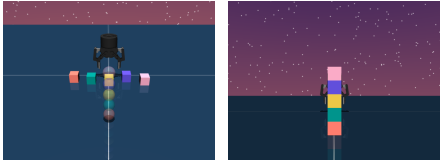
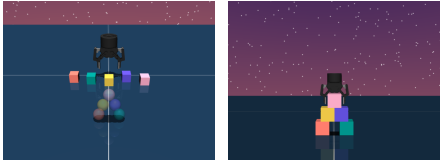

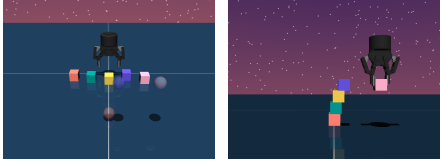

Task name	Task visualization	Abilities
cube-5-task1		<ol style="list-style-type: none"> 1. Sequential-logic 2. Motor skills
cube-5-task2		<ol style="list-style-type: none"> 1. Sequential-logic 2. Motor skills
cube-5-task3		<ol style="list-style-type: none"> 1. Sequential-logic 2. Motor skills
cube-5-task4		<ol style="list-style-type: none"> 1. Maximum overhang 2. Sequential-logic 3. Motor skills
cube-5-task5		<ol style="list-style-type: none"> 1. Building support structures 2. Sequential-logic 3. Motor skills

Table 6: Tasks with 5 cubes.

1782
 1783
 1784
 1785
 1786
 1787
 1788
 1789
 1790
 1791
 1792
 1793
 1794
 1795
 1796
 1797
 1798
 1799
 1800
 1801
 1802
 1803
 1804
 1805
 1806
 1807
 1808
 1809
 1810
 1811
 1812
 1813
 1814
 1815
 1816
 1817
 1818
 1819
 1820
 1821
 1822
 1823
 1824
 1825
 1826
 1827
 1828
 1829
 1830
 1831
 1832
 1833
 1834
 1835

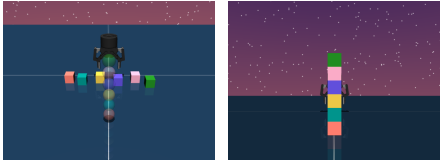
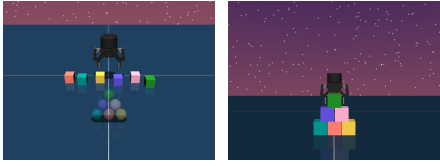
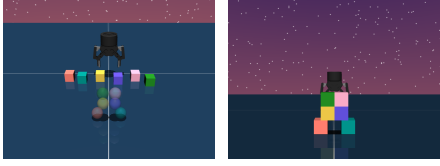
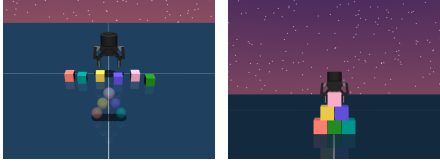

Task name	Task visualization	Abilities
cube-6-task1		<ol style="list-style-type: none"> 1. Sequential-logic 2. Motor skills
cube-6-task2		<ol style="list-style-type: none"> 1. Sequential-logic 2. Motor skills
cube-6-task3		<ol style="list-style-type: none"> 1. Two cube lift 2. Sequential-logic 2. Motor skills
cube-6-task4		<ol style="list-style-type: none"> 1. Building support structures 2. Sequential-logic 3. Motor skills
cube-6-task5		<ol style="list-style-type: none"> 1. Temporary bi-scaffolding 2. Sequential-logic 3. Motor skills

Table 7: Tasks with 6 cubes.

1836
 1837
 1838
 1839
 1840
 1841
 1842
 1843
 1844
 1845
 1846
 1847
 1848
 1849
 1850
 1851
 1852
 1853
 1854
 1855
 1856
 1857
 1858
 1859
 1860
 1861
 1862
 1863
 1864
 1865
 1866
 1867
 1868
 1869
 1870
 1871
 1872
 1873
 1874
 1875
 1876
 1877
 1878
 1879
 1880
 1881
 1882
 1883
 1884
 1885
 1886
 1887
 1888
 1889

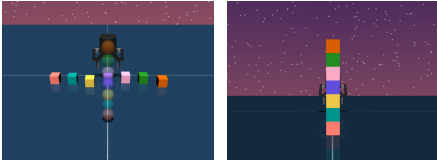
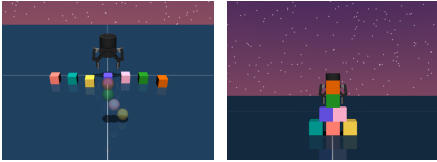
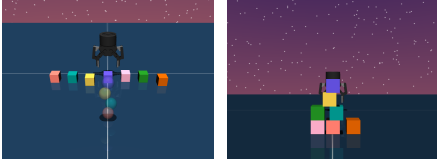
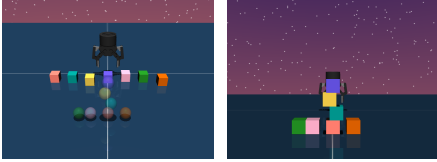
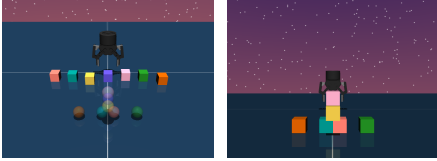
Task name	Task visualization	Abilities
cube-7-task1		<ol style="list-style-type: none"> 1. Sequential-logic 2. Motor skills
cube-7-task2		<ol style="list-style-type: none"> 1. Building a support structure 2. Sequential-logic 2. Motor skills
cube-7-task3		<ol style="list-style-type: none"> 1. Building a support structure 2. Sequential-logic 2. Motor skills
cube-7-task4		<ol style="list-style-type: none"> 1. Temporary bi-scaffolding 2. Sequential-logic 3. Motor skills
cube-7-task5		<ol style="list-style-type: none"> 1. Temporary bi-scaffolding 2. Sequential-logic 3. Motor skills

Table 8: Tasks with 7 cubes.

1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943

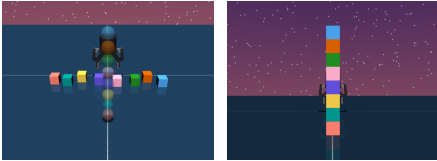
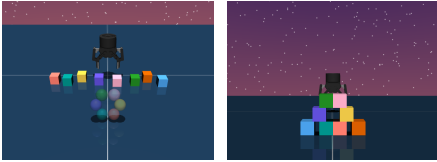
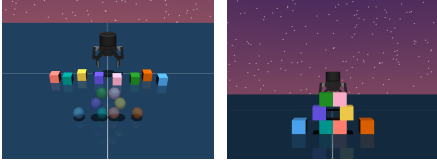
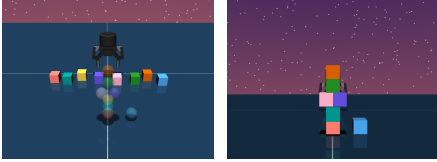
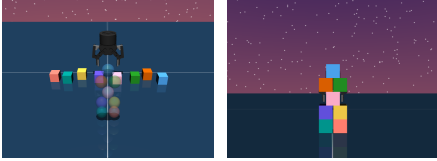
Task name	Task visualization	Abilities
cube-8-task1		<ol style="list-style-type: none"> 1. Sequential-logic 2. Motor skills
cube-8-task2		<ol style="list-style-type: none"> 1. Building a support structure 2. Sequential-logic 2. Motor skills
cube-8-task3		<ol style="list-style-type: none"> 1. Temporary bi-scaffolding 2. Sequential-logic 2. Motor skills
cube-8-task4		<ol style="list-style-type: none"> 1. Temporary bi-scaffolding 2. Sequential-logic 3. Motor skills
cube-8-task5		<ol style="list-style-type: none"> 1. Three cube lift 2. Sequential-logic 3. Motor skills

Table 9: Tasks with 8 cubes.

1944
 1945
 1946
 1947
 1948
 1949
 1950
 1951
 1952
 1953
 1954
 1955
 1956
 1957
 1958
 1959
 1960
 1961
 1962
 1963
 1964
 1965
 1966
 1967
 1968
 1969
 1970
 1971
 1972
 1973
 1974
 1975
 1976
 1977
 1978
 1979
 1980
 1981
 1982
 1983
 1984
 1985
 1986
 1987
 1988
 1989
 1990
 1991
 1992
 1993
 1994
 1995
 1996
 1997

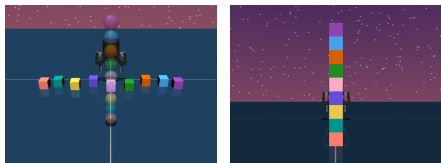
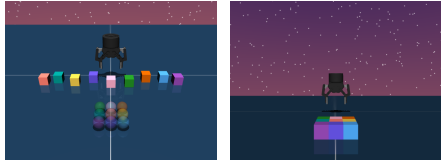
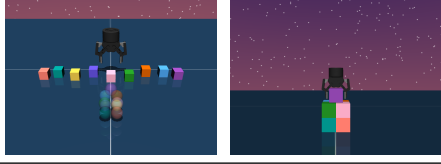

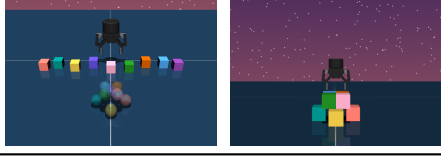
Task name	Task visualization	Abilities
cube-9-task1		<ol style="list-style-type: none"> 1. Sequential-logic 2. Motor skills
cube-9-task2		<ol style="list-style-type: none"> 1. Sequential-logic 2. Motor skills
cube-9-task3		<ol style="list-style-type: none"> 1. Temporary mono-scaffolding 2. Mono-scaffold reuse 3. Sequential-logic 4. Motor skills
cube-9-task4		<ol style="list-style-type: none"> 1. Temporary tri-scaffolding 2. Bi-scaffold reuse 3. Sequential-logic 4. Motor skills
cube-9-task5		<ol style="list-style-type: none"> 1. Building a support structure 2. Sequential-logic 3. Motor skills

Table 10: Tasks with 9 cubes.