BuilderBench: The Building Blocks of Intelligent Agents

Raj Ghugare[1], **Roger Creus Castanyer**[2,3], **Catherine Ji**[1], **Kathryn Wantlin**[1], **Jin Schofield**[1], **Karthik Narasimhan**[1,*], **Benjamin Eysenbach**[1,*]
rg9360@princeton.edu

[1]**Princeton University**
[2]**Mila – Quebec AI Institute**
[3]**Université de Montréal**

[*]Equal advising.

Accepted at the Workshop on Reinforcement Learning Beyond Rewards (RLBrew), RLC 2026.

Abstract

Today's AI models learn primarily through mimicry and refining, 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 by 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 training that centers open-ended exploration. BuilderBench requires agents to learn how to build any structure using blocks. BuilderBench is equipped with (1) a simulator of a robot interacting with various physical blocks, and (2) a task-suite with over 50 diverse target structures that are carefully curated to test an understanding of physics, mathematics, and long-horizon planning. Agents are provided with a target structure at the start, and can interact with the environment for multiple episodes to experiment and learn various skills for building the structure. Solving these tasks requires *embodied reasoning* in a way that is not reflected in words but rather in actions, experimenting with different strategies and piecing them together. Our experiments with multiple state-of-the-art frontier language model based agents and tabula rasa reinforcement learning algorithms show that these agents cannot solve any of the non-trivial tasks in the BuilderBench. Our analysis throws light on the lack of exploration abilities in these models.

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

Blog **Code**

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 diverse range 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, architecture) (Ying et al., 2025; Silver and Sutton, 2025). Making progress will require that agents learn not only from human experience, but also from their own, self-collected experience. Agents will

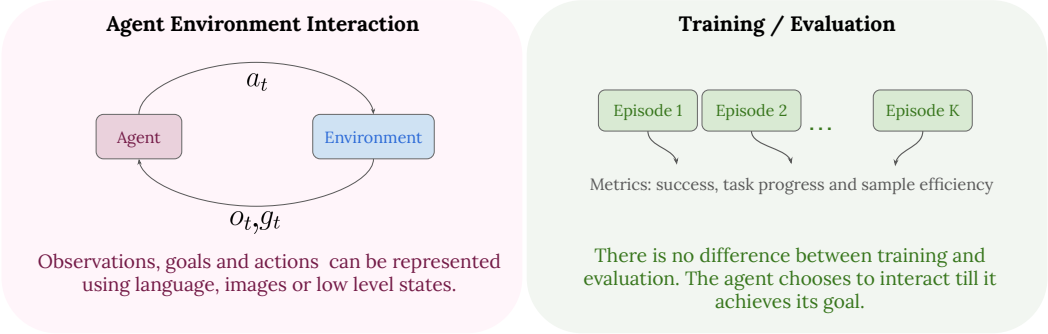


Figure 1: **The BuilderBench setup.** Vision, language, or tabula rasa agents can be trained / evaluated in BuilderBench. Agents can choose to interact with the simulated environment for multiple episodes to learn to solve the target task. The goal of the agent is to learn to solve *any* given task as efficiently as possible.

have to actively explore and run experiments to extract knowledge about the environment (Spelke and 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), such as maze navigation in ant-maze (Fu et al., 2021), Montezuma's Revenge (Bellemare et al., 2013), or the handful of tasks in the kitchen environment (Gupta et al., 2020). But 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 reasoning skills that agents trained on human text do acquire (Wei et al., 2022). We argue that the primary reason is that current interactive benchmarks offer limited learning opportunities. Existing benchmarks rarely allow agents to practice a spectrum of skills, ranging from exploration to prediction and 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 desire 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. 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 and Mix, 2014). In addition to being useful for early human cognitive development, block-building is a mathematically rich area[1] with a deep history in AI and planning (Gupta and Nau, 1992; Ahmad et al., 2019; Russell and 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

[1]In 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.

Table 1: Qualitative Comparison of BuilerBench with previous benchmarks. BuilderBench is a uniquely positioned robotics benchmark that contains an extensive reasoning task suite and a fast simulator with support for both frontier and tabula rasa RL agents.

Benchmark	**Extensive reasoning task-suite**	**Frontier model agents**	**RL agents**	**Hardware accelerated**	**Robotics**
BuilerBench (ours)	✓	✓	✓	✓	✓
ManiSkill (Mu et al., 2021)	✗	✗	✓	✓	✓
FurnitureBench (Heo et al., 2023)	✗	✗	✓	✗	✓
BabyAI (MCB et al., 2019)	✗	✗	✓	✗	✗
Kinetix (Matthews et al., 2025)	✗	✗	✓	✓	✓
Craftium (Malagón et al., 2025b)	✗	✓	✓	✗	✗
NetHack (Küttler et al., 2020)	✗	✓	✓	✗	✗

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 through exploration and trial-and-error learning. Our central hypothesis, which motivates the use of block-building is that *the space of skills and discoveries that an agent has to know to build all possible structures is so vast, that it is impossible to memorize them at design time.* An agent will always have to interact and make new discoveries to build any given target structure.

Contributions. To this end, we introduce the BuilderBench benchmark. BuilderBench is equipped with a simulated environment, consisting of a robot that can interact with blocks. The environment can output observations in the form of low level state information, images or language descriptions. Agents can control the robot using low level control actions or high level plans to control the robot. This enables many types of agents based on deep RL (Mnih et al., 2013; Haarnoja et al., 2018), language models (Brown et al., 2020; Wei et al., 2022), vision/video language models (Radford et al., 2021; Zhang et al., 2023) or vision language actions models (Brohan et al., 2023; Intelligence et al., 2025) to be evaluated in BuilderBench.

BuilderBench comes equipped with a set of over 50 challenging tasks (and growing) which span several orders of magnitude of complexity. Tasks require a wide variety of skills ranging from motor skills like locomotion, grasping and throwing to 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 discovering new tools for performing skills like temporary scaffolding or disassembling.

We evaluate some of the strongest frontier language model based agents available (at the time of writing) like GPT 5.2, Claude Opus 4.6 and Gemini 3 Flash. We find that these models do not manage to solve most of the hard tasks from the BuilderBench task suite. We provide a detailed qualitative analysis of the failure modes of these models.

Finally, we also 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.

Table 1 provides a qualitative comparison of BuilderBench with other related benchmarks. We provide a detailed comparison of BuilderBench with previous related works in Section 7. Code for the benchmark, all baselines, and experiments is included here. In the next section, we sketch out what BuilderBench is meant to evaluate and what could be the consequences of a satisfying solution.

2 What does BuilderBench measure?

BuilderBench measures the exploration efficiency of an agent in an open-ended environment. During training, the goal of an agent is to build a given target structure. The agent can choose to interact with the environment as long as it wants. During this time, it needs to explore various approaches, discover new skills and build the structure through trial and error. We hypothesize that agents which learn to build any unseen target structures efficiently will possess two different type of exploration abilities:

Exploration in actual interaction. Agents need to interact with the environment to gain information about the blocks and the physics. It needs to gain information about how its actions affect the environment. Moreover, it needs to make wide-ranging hypotheses and validate them by trying them out in the environment. Ultimately, learning via interaction and exploration in an open-ended environment[2] is a key missing piece in today's AI models. BuilderBench is one way to develop and measure the efficacy of such algorithms based on trial and error.

Exploration in thoughts. There are many tasks in BuilderBench which, despite knowing how the environment works (having a perfect model of the world), are difficult to solve programmatically. The agent needs to explore in the space of "thoughts" to come up with complex and new plans to attempt solving these tasks. The agent does not get any direct feedback on which plans are good, hence we say it has to explore in the space of "thoughts". If one were to anthropomorphize further, one would conclude that these tasks require *creative* and *out of the box thinking*.

There are many secondary skills that BuilderBench tasks can also be used to measure. **Tool discovery**: Many tasks require the agent to use blocks as tools to provide support or improve reachability. While the latest language models (at the time of writing) are trained to use tools, BuilderBench requires agents to discover them. **Long horizon generalization**: Most tasks can be broken down into sub-tasks whose solutions seem much simpler. Hence, agents need to be able to sequentially compose solutions to sub-tasks. **Robotics:** While agents can use high level planners that abstract away the details of control, better robotic systems could potentially be better at solving tasks which require finer control.

What it would mean to solve BuilderBench with a satisfying solution: The most important and hopeful consequence would be that we would have algorithms which could efficiently learn "creative" solutions to unseen problems via interaction with the environment. BuilderBench can also accelerate research in other areas like discovering better interface systems between purely digital agents like LLMs and physical agent like robots. The AI agents might discover new architectural skills (similar to how chess players learned from alphazero Silver et al. (2017)).

Our central hypothesis, which motivated the use of block-building is that the space of skills and discoveries that an agent has to know to build all possible structures is so vast, that it is impossible to memorize them at design time. In the coming months or years, solutions which attempt to use such memorization might saturate the BuilderBench leaderboard. We are confident that we (or someone else) would be able to come up with new block-building tasks that such an approach will fail to learn to build.

3 BuilderBench

The BuilderBench benchmark is equipped with a simulator and a task-suite. The task-suite contains 51 tasks, where each task is a target block structure curated carefully for evaluating unique abilities. The simulator is built using MuJoCo (Todorov et al., 2012); it consists of a UR5e robot and a Robotiq 2F-85 parallel jaw gripper[3] interacting with a varying number of blocks. In the following sections, we will describe the environment (Section 3.1), the agent-environment interaction protocol (Section 3.1.1), and the task-suite (Section 3.2).

[2]Especially environments which are not purely lingual.

[3]The backbone code for controlling this robot is adapted from Park et al. (2025).

3.1 BuilderBench environment.

The environment can be formulated as a Markov decision process (MDP) (Sutton and Barto, 2018), with states $s_t \in \mathcal{S}$ and actions $a_t \in \mathbf{A}$ and transition dynamics $T(s_{t+1} \mid s_t, a_t)$ 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 that interacts with the n cubes. All interactions approximate real physics simulated using MuJoCo (Todorov et al., 2012). Additional details about the environment and various potential representations are provided in Section 8.

State space. The low level states include information about the robot and the cubes. This includes the position and velocity of the arm joints and end effector, the gripper opening, the current control inputs of the robot, the position and velocities of all cubes and the current timestep.

Observation space. The simulator exposes all the state information to the agent designer. The agent designer can select which information to provide to the agent and its representation. We provide a language wrapper for the environment which converts the relevant information in a language description of the scene. We also provide functions to render visuals of the scene to enable training vision based models.

Action space. The robot can be manipulated using a 5 dimensional low level control. These controls consists of delta position and delta yaw of the end effector and delta gripper strength. The language wrapper we provide extends this action space with various high level actions that operate for multiple time-steps such as pick-and-place, pick-and-hold, and end-effector-target. These high level actions allow language model agents to efficiently control the robot. Indeed, language models are able to solve most of the easy tasks from the BuilderBench task-suite using this interface. We do not restrict agent designers from adding more high level actions.

Task specification. Each task corresponds to 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 4 for an example).

3.1.1 Agent-environment interaction Setup

We aim to keep the agent environment interaction loop as general as possible to enable multiple types of agents (language based, vision based, proprioceptive control; zero shot, in-context learning, RL training). Figure 1 provides a visualization of the setup. At the start of the interaction, the agent is provided with the task specification. The agent gets an observation and has to output an action. The agent can choose to interact with the environment for a variable number of episodes (or until it learns to achieve the goal). The objective of the agent is to learn to build the target structure as fast as possible.

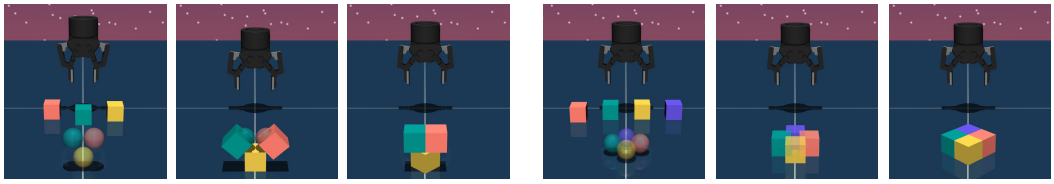
The next section will show how despite this seemingly simple setup, tasks can be arbitrarily complex and long-horizon. Qualitatively, we will see that solving tasks require multiple steps of high-level reasoning.

3.2 BuilderBench task suite

In this section, we describe some of the tasks from the BuilderBench task-suite in detail and the design philosophy behind the task-suite. The task-suite is meant to address the challenges highlighted in Section 1. We start with a case-study of four different tasks from the BuilderBench task-suite[4], 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. In Section 9, we outline the general design principles that underlie the tasks in the BuilderBench task-suite. The visualizations for all tasks is provided in Section 13.

[4]We have only visualized the arm of the robot in Section 3.2.1.

3.2.1 A case study of four tasks

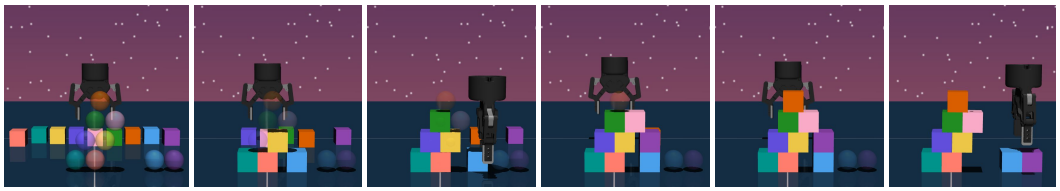


(A) (B) (C) (A) (B) (C)

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 shown 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 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.

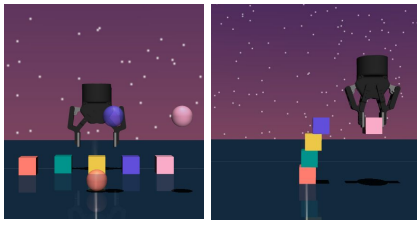


(A) (B) (C) (D) (E) (F)

Figure 3: **Leaning Tower**

Example 3: Leaning Tower. The target is a leaning tower composed of seven blocks and two extra cubes placed on the floor (see (A) of Figure 3). 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 indigo cube in (B)). To stabilize the structure, the agent needs to add counterweights (the pink 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 orange block in the third layer requires a two-cube vertical scaffold (see (E)). Finally, the tower is completed by adding the counterweights (the blue and purple cubes in (F)) and removing and repositioning the last scaffold (see (G)).

Example 4: 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 4). 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



(A) (B)

Figure 4: **Maximum overhang problem**

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.

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 and Kinzler, 2007), and learn mechanisms to efficiently use them on the fly (Chollet, 2019).

4 Experiments

The main goal of our experiments is to ascertain whether the strongest language models (at the time of writing) display abilities of exploration and learning from interaction to solve BuilderBench tasks. To achieve this, we evaluate three of the strongest available frontier models, GPT 5.2, Claude Opus 4.6, and Gemini 3 Flash (all models at highest levels of reasoning), on all BuilderBench tasks. Because all of these experiments cost money, we have only used one seed per run. We use two different algorithms that enable these models for interaction:

Chain of thought (Wei et al., 2023). Chain of thought (CoT) reasoning prompts the model to output and store a reasoning chain before outputting every action. All the models we evaluate utilize a significant number of thinking tokens before producing an output. CoT additionally forces the agent to explain its thinking in the output. This is shown to have a significant impact on the performance of language models (Wei et al., 2023). The CoT agent interacts with the environment for a single episode. Episode lengths have been set so that they provide enough time for exploration and adaptation. The agent has access to at most 16 previous time-steps of observations and actions and its previous reasoning while producing the current output.

Reflexion (Shinn et al., 2023). Reflexion agents are prompted to maintain a collection of running summaries of their previous experiences and attempts at solving a task. These summaries include running details of the agent's past mistakes, learnings and best plans of action, giving it arbitrary memory. This agent can interact with the environment for multiple episodes (three in our experiments) and use its past reflections to continually improve its decision-making. The agent has access to at most five previous time-steps of observations and actions, its running summary of the current episode, and summaries of all its previous episodes while producing an output.

Before moving to the results, we describe the language interface used by these agents for interaction.

Language interface. At the start, the agent is provided with an environment system prompt which describes the environment dynamics, semantics of observations and actions, and the expected action schema. See Figure 8 for the exact system prompt. At each timestep of the episode, the interface provides a language description of the scene in a neat tabular format. This description contains the current time, total time and the position and orientation of the end effector. It also contains cube wise positions, orientations, targets and success conditions (see Figure 9 for an example observation). Finally, the agent can control the robot using low level controls or by commanding a high level planner. The planner can be used to move the end effector any pose, pick a cube in different orientations and place or hold a cube in different poses (see Figure 10 for an example action).

The task suite contains tasks ranging from very easy to extremely hard. We divided the tasks into two categories (easy and hard) based on whether they are programmatically straightforward to solve.

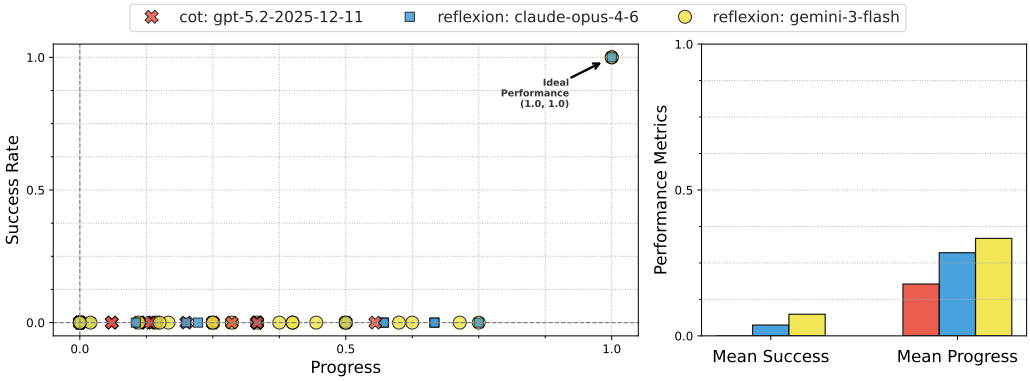


Figure 5: **Evaluating language model based agents on hard BuilderBench tasks.** Other than two exceptions, all models achieve zero success on all of the 27 hard tasks. The best performing agent, Reflexion with Gemini 3 flash, is able to solve two of the hard tasks from the task-suite. The progress metric is the fraction of blocks the agent manages to place correctly. While the Reflexion agent makes some progress, it does not show an affinity towards making the key discoveries that an agent has to make to solve these tasks.

Tasks which are straightforward compositions of pick and place primitives fall in the easy category.[5] We evaluate all agents on all the easy (23/51) and hard tasks (27/51). The visualizations for all tasks structures and their corresponding difficulty is provided in Section 13.

4.1 Results

Figure 5 shows the results of all agents on the hard tasks. Each point in the scatter plot denotes the performance of an agent solving a task. Success rate (y axis) is whether the agent is able to build the entire structure or not. Progress (y axis) is the fraction of blocks the agent manages to place correctly. Other than a single exception, all models achieve zero success on all of the 27 hard tasks. The results on easy tasks are shown in Figure 6.

Failure modes. These results suggest that agents are able to compress the observations and extract correct plans and actions for easy tasks. But solving difficult (unseen) tasks requires something beyond mere compression; agents have to show an affinity towards information gathering, make "out-of-the-box" hypotheses and validate them through interaction.

By analyzing the reasoning and reflection outputs, and the videos of agents, we observed three primary failure modes of these language model agents:

- **Exploration.** This is the key failure mode which recurs in most tasks where the obvious greedy solution doesn't work. Agents do not showcase hypothesis driven exploration (can I try balancing... over the other...?), nor do they generate playful hypotheses for information gathering (let me see what happens if...).
- **Planning.** Many times agents try strategies which are clearly going to fail (trying to place a block where there already is one). Such failure modes should be avoidable if agents have a decent world model of physics and use it to simulate plans.
- **Fine-grained control.** Agents mostly rely on high level primitives and rarely use skills such as nudging (although there are some exceptions). This was the most expected failure mode as these models are not trained to explicitly output low level controls.

We provide an in-depth analysis of failure modes, including details of agent outputs and videos of agent interaction in a separate blogpost. We visualize one such failure mode of the Reflexion agent

[5] Given access to a pick and place planner, stacking 10 blocks shouldn't be considered significantly harder than stacking two blocks.

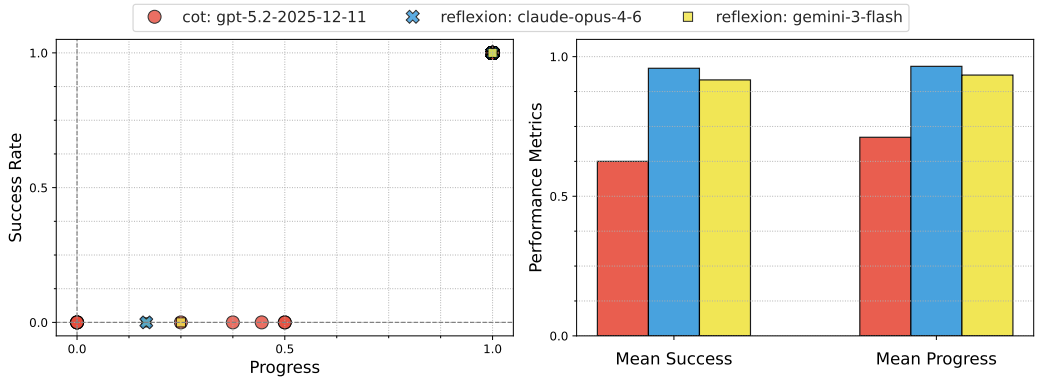
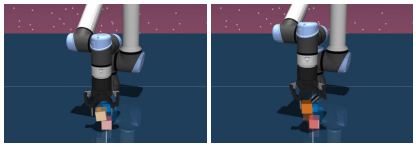


Figure 6: **Evaluating language model based agents on easy BuilderBench tasks.** The Reflexion agents (Shinn et al., 2023) are able to solve most easy tasks from the BuilderBench task-suite, achieving over 90% average success across all easy tasks. Most of these tasks are direct applications of the pick and place primitives. Although easy for language models, they still pose a challenge for tabula-rasa RL based agents (see Section 10).

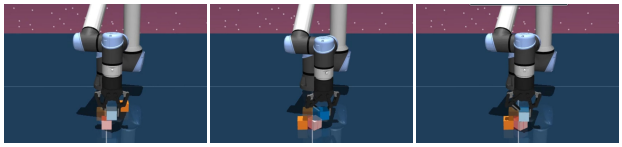
Trial 1: Tries to place the yellow block on the red one three times despite failing each time.



Root Cause Analysis:

1. …
2. **Stacking geometry concern:** The two cubes need each other for support, meaning the first cube placed may topple before the second is placed.

Trial 3: Gets stuck on making partial progress. Despite stumbling near the correct solution, concludes that its strategy is the best.



Key Outcome:
… physically unstable without external support. Since the gripper can only hold one cube at a time, maximum achievable was always 2/3.

If Facing This Again
… Use remaining time only if a creative 3/3 solution exists (e.g., wedging, angling — but unlikely).

Figure 7: **Reflexion agents based on claude opus 4.6 fails on the T-block task (Figure 2, left).** The text below the images show the intermediate snapshots of summaries maintained by the agent. The agent repeatedly tries the same greedy approach of directly stacking the yellow block at its target despite the structure being unstable. It then summarises that the structure is impossible to build and sticks to holding one of the top blocks in place.

trying to solve the T-block task in Figure 7. We see that the agent does not explore new strategies and defaults to a greedy straightforward approach despite failing previously.

Finally, we train seven different tabula-rasa RL algorithms to solve the BuilderBench tasks with at most four cubes. We find that as the number of cubes and the complexity of the tasks increase, no algorithm is able to achieve non-zero success. The experimental details and results are presented in Section 10 and Section 11.

5 Limitations and Future work

Despite our hypothesis about building blocks being an open-ended setup, the number of tasks in our benchmark is finite. This is because coming up with new non-trivial tasks is non-trivial. The scope of designing tasks could be drastically improved by adding blocks of new shapes or objects with different properties (for e.g., magnets). One interesting direction for future work would be to set up an adversarial game between a task designer and a task solver (OpenAI et al., 2021).

Another limitation is that methods which memorize the solutions to the tasks in BuilderBench (for e.g., solutions might leak into the training data of future language models) will saturate the scores, but still not provide a satisfying solution. We are confident that we (or someone else) would be able to come up with new block-building tasks that such an approach will fail to learn to build.

Finally, while we use the strongest available language models (at the time of writing) for our evaluation, the agents we evaluate are a tiny subset of all potential approaches. Exploring agents based on vision, control (vision language action models), recursive self improvement or RL fine tuning remain an interesting direction for future work.

6 Conclusion

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 via exploration and trial and error. Tasks in BuilderBench are designed to elicit long-horizon planning and reasoning abilities, many implicitly requiring agents to solve problems in physics and mathematics. We evaluated agents based on some of the strongest available language models (at the time of writing) like GPT 5.2, Claude Opus 4.6, and Gemini 3 Flash on BuilderBench tasks. Our experiments highlighted that these agents lacked the ability to explore, plan and perform fine grained control. We have open sourced all the code for our experiments. We hope that the research resulting from BuilderBench and our analysis will advance the development of agents that solve problems by interacting with the real world.

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, Cemlyn Neil Waters, Mohamed Ali Ali Mimouni, Ulrich Armel Mbou Sob, Ruan John de Kock, Siddarth Singh, Daniel Furelos-Blanco, Victor Le, Arnu 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.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2023. URL https://arxiv.org/abs/2212.06817.

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*, page 95–133. Kluwer Academic Publishers, USA, 1998. ISBN 0792380479.

Beth Casey, Nicole Andrews, Holly Schindle, Joanne Kersh, Alexandra Samper, and Juanita Copley. The development of spatial skills through interventions involving block building activities. *Cognition and Instruction - COGNITION INSTRUCT*, 26:269–309, 07 2008. doi: 10.1080/07370000802177177.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling, 2014. URL https://arxiv.org/abs/1312.3005.

Yi-Ling Cheng and Kelly S Mix. Spatial training improves children's mathematics ability. *Journal of cognition and development*, 15(1):2–11, 2014.

François Chollet. On the measure of intelligence, 2019. URL https://arxiv.org/abs/1911.01547.

Wei Chow, Jiageng Mao, Boyi Li, Daniel Seita, Vitor Campagnolo Guizilini, and Yue Wang. Physbench: Benchmarking and enhancing vision-language models for physical world understanding. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Q6a9W6kzv5.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning, 2020. URL https://arxiv.org/abs/1912.01588.

DeepSeek-AI. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. doi: 10.1109/MSP.2012.2211477.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. URL http://arxiv.org/abs/1810.04805. cite arxiv:1810.04805Comment: 13 pages.

Guneet S. Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification, 2020. URL https://arxiv.org/abs/1909.02729.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems, 2021. URL https://arxiv.org/abs/1901.10995.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJx63jRqFm.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=vGQiU5sqUe3.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017. URL https://arxiv.org/abs/1703.03400.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents, 2018. URL https://arxiv.org/abs/1705.06366.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2021. URL https://arxiv.org/abs/2004.07219.

Seyed Kamyar Seyed Ghasemipour, Daniel Freeman, Byron David, Shixiang Shane Gu, Satoshi Kataoka, and Igor Mordatch. Blocks assemble! learning to assemble with large-scale structured reinforcement learning, 2022. URL https://arxiv.org/abs/2203.13733.

Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P. Adams, and Sergey Levine. Why generalization in rl is difficult: epistemic pomdps and implicit partial observability. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *ArXiv*, abs/1611.07507, 2016. URL https://api.semanticscholar.org/CorpusID:2918187.

Leon Guertler, Bobby Cheng, Simon Yu, Bo Liu, Leshem Choshen, and Cheston Tan. Textarena, 2025. URL https://arxiv.org/abs/2504.11442.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1025–1037. PMLR, 30 Oct–01 Nov 2020. URL https://proceedings.mlr.press/v100/gupta20a.html.

Naresh Gupta and Dana S. Nau. On the complexity of blocks-world planning. *Artificial Intelligence*, 56(2):223–254, 1992. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(92)90028-V. URL https://www.sciencedirect.com/science/article/pii/000437029290028V.

William H. Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations, 2019. URL https://arxiv.org/abs/1907.13440.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Danijar Hafner. Benchmarking the spectrum of agent capabilities, 2022. URL https://arxiv.org/abs/2109.06780.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2024. URL https://arxiv.org/abs/2301.04104.

Minho Heo, Youngwoon Lee, Doohyun Lee, and Joseph J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation, 2023. URL https://arxiv.org/abs/2305.12821.

Edward Hughes, Michael D Dennis, Jack Parker-Holder, Feryal Behbahani, Aditi Mavalankar, Yuge Shi, Tom Schaul, and Tim Rocktäschel. Position: Open-endedness is essential for artificial superhuman intelligence. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 20597–20616. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/hughes24a.html.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025. URL https://arxiv.org/abs/2511.14759.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark and learning environment, 2019. URL https://arxiv.org/abs/1909.12271.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues?, 2024. URL https://arxiv.org/abs/2310.06770.

Leslie Pack Kaelbling. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence*, 1993. URL https://api.semanticscholar.org/CorpusID:5538688.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *J. Artif. Int. Res.*, 76, May 2023. ISSN 1076-9757. doi: 10.1613/jair.1.14174. URL https://doi.org/10.1613/jair.1.14174.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, Aniruddha Kembhavi, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai, 2022. URL https://arxiv.org/abs/1712.05474.

Sotetsu Koyamada, Shinri Okano, Soichiro Nishimori, Yu Murata, Keigo Habara, Haruka Kita, and Shin Ishii. Pgx: Hardware-accelerated parallel game simulators for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 45716–45743, 2023.

Heinrich Küttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment, 2020. URL https://arxiv.org/abs/2006.13760.

Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. doi: 10.1126/science.aab3050. URL https://www.science.org/doi/abs/10.1126/science.aab3050.

Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching, 2020. URL https://arxiv.org/abs/1906.05274.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=IFXTZERXdM7.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *ArXiv*, abs/2210.00030, 2022. URL https://api.semanticscholar.org/CorpusID:252683397.

Mikel Malagón, Josu Ceberio, and Jose A. Lozano. Craftium: Bridging flexibility and efficiency for rich 3d single- and multi-agent environments, 2025a. URL https://arxiv.org/abs/2407.03969.

Mikel Malagón, Josu Ceberio, and Jose A. Lozano. Craftium: Bridging flexibility and efficiency for rich 3d single- and multi-agent environments, 2025b. URL https://arxiv.org/abs/2407.03969.

Michael Matthews, Michael Beukman, Chris Lu, and Jakob Nicolaus Foerster. Kinetix: Investigating the training of general agents through open-ended physics-based control tasks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=zCxGCdzreM.

MCB, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning, 2019. URL https://arxiv.org/abs/1810.08272.

Andrew Melnik, Luca Lach, Matthias Plappert, Timo Korthals, Robert Haschke, and Helge J. Ritter. Using tactile sensing to improve the sample efficiency and performance of deep deterministic policy gradients for simulated in-hand manipulation tasks. *Frontiers in Robotics and AI*, 8, 2021. URL https://api.semanticscholar.org/CorpusID:235663648.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013. URL https://arxiv.org/abs/1312.5602.

Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations, 2021. URL https://arxiv.org/abs/2107.14483.

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample-efficient continuous control, 2024. URL https://arxiv.org/abs/2405.16158.

OpenAI. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

OpenAI OpenAI, Matthias Plappert, Raul Sampedro, Tao Xu, Ilge Akkaya, Vineet Kosaraju, Peter Welinder, Ruben D'Sa, Arthur Petron, Henrique P. d. O. Pinto, Alex Paino, Hyeonwoo Noh, Lilian Weng, Qiming Yuan, Casey Chu, and Wojciech Zaremba. Asymmetric self-play for automatic goal discovery in robotic manipulation, 2021. URL https://arxiv.org/abs/2101.04882.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn, 2016. URL https://arxiv.org/abs/1602.04621.

Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–286, 2007. doi: 10.1109/TEVC.2006.890271.

Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wolczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. Balrog: Benchmarking agentic llm and vlm reasoning on games, 2025. URL https://arxiv.org/abs/2411.13543.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl, 2025. URL https://arxiv.org/abs/2410.20092.

Mike Paterson, Yuval Peres, Mikkel Thorup, Peter Winkler, and Uri Zwick. Maximum overhang, 2007. URL https://arxiv.org/abs/0707.0093.

Silviu Pitis, Harris Chan, Stephen Zhao, Bradly Stadie, and Jimmy Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research, 2018. URL https://arxiv.org/abs/1802.09464.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Sai Rajeswar, Pietro Mazzaglia, Tim Verbelen, Alexandre Piché, Bart Dhoedt, Aaron Courville, and Alexandre Lacoste. Mastering the unsupervised reinforcement learning benchmark from pixels. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Stuart Reifel. Block construction: Children's developmental landmarks in representation of space. *Young children*, 1984.

Ronan Riochet, Mario Ynocente Castro, Mathieu Bernard, Adam Lerer, Rob Fergus, Véronique Izard, and Emmanuel Dupoux. Intphys: A framework and benchmark for visual intuitive physics reasoning, 2020. URL https://arxiv.org/abs/1803.07616.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015. URL https://arxiv.org/abs/1409.0575.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition, 2010.

Alexander Rutherford, Michael Beukman, Timon Willi, Bruno Lacerda, Nick Hawes, and Jakob Nicolaus Foerster. No regrets: Investigating and improving regret approximations for curriculum discovery. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=iEeiZlTbts.

Juergen Schmidhuber. Reinforcement learning upside down: Don't predict rewards – just map them to actions, 2020. URL https://arxiv.org/abs/1912.02875.

Jürgen Schmidhuber. Evolutionary principles in self-referential learning, or on learning how to learn: The meta-meta-. hook, 1987. URL https://api.semanticscholar.org/CorpusID:264351059.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Olivier Sigaud, Gianluca Baldassarre, Cedric Colas, Stephane Doncieux, Richard Duro, Pierre-Yves Oudeyer, Nicolas Perrin-Gilbert, and Vieri Giuliano Santucci. A definition of open-ended learning problems for goal-conditioned agents, 2024. URL https://arxiv.org/abs/2311.00344.

David Silver and Richard S Sutton. Welcome to the era of experience. *Google AI*, 2025.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016. doi: 10.1038/nature16961.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017. URL https://arxiv.org/abs/1712.01815.

Dorothy G. Singer, Roberta Michnick Golinkoff, and Kathy Hirsh-Pasek. *Play = Learning: How Play Motivates and Enhances Children's Cognitive and Social-Emotional Growth*. Oxford University Press, 09 2006. ISBN 9780195304381. doi: 10.1093/acprof:oso/9780195304381.001.0001. URL https://doi.org/10.1093/acprof:oso/9780195304381.001.0001.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4080–4090, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Elizabeth S. Spelke and Katherine D. Kinzler. Core knowledge. *Developmental Science*, 10(1): 89–96, 2007. doi: 10.1111/j.1467-7687.2007.00569.x.

Kenneth Stanley. Open-endedness: The last grand challenge you've never heard of. https://www.uber.com/blog/research/open-endedness-the-last-grand-challenge-youve-never-heard-of/, December 2017. Uber Engineering Blog.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards, 2025. URL https://arxiv.org/abs/2505.24760.

Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. The distracting control suite – a challenging benchmark for reinforcement learning from pixels, 2021. URL https://arxiv.org/abs/2101.02722.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Exploration: A study of count-based exploration for deep reinforcement learning, 2017. URL https://arxiv.org/abs/1611.04717.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite, 2018. URL https://arxiv.org/abs/1801.00690.

Adaptive Agent Team. Human-timescale adaptation in an open-ended task space, 2023. URL https://arxiv.org/abs/2301.07608.

Gemini Team. Gemini: A family of highly capable multimodal models, 2025. URL https://arxiv.org/abs/2312.11805.

Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, Nat McAleese, Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Roberta Raileanu, Steph Hughes-Fitt, Valentin Dalibard, and Wojciech Marian Czarnecki. Open-ended learning leads to generally capable agents, 2021. URL https://arxiv.org/abs/2107.12808.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change, 2023. URL https://arxiv.org/abs/2206.10498.

Brian N. Verdine, Roberta M. Golinkoff, Kathryn Hirsh-Pasek, Nora S. Newcombe, Andrew T. Filipowicz, and Alicia Chang. Deconstructing building blocks: Preschoolers' spatial assembly performance relates to early mathematical skills. *Child Development*, 85(3):1062–1076, 2014. ISSN 00093920, 14678624. URL http://www.jstor.org/stable/24031910.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 3637–3645, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Joseph Dudzik, Junyoung Chung, David Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, L. Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Caglar Gulcehre, Ziyun Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575:350 – 354, 2019. URL https://api.semanticscholar.org/CorpusID:204972004.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=yzkSU5zdwD. Survey Certification.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

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

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

Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding, 2023. URL https://arxiv.org/abs/2306.02858.

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

Supplementary Materials

The following content was not necessarily subject to peer review.

7 Related Works

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 agents that learn through trial and error. 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). Recently, such environments have also been adapted to evaluate language model or vision model based agents Stojanovski et al. (2025); Paglieri et al. (2025); Guertler et al. (2025). But ultimately the adaptations face the same problem. 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 learning to solve 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 permit sufficiently complex and open ended interaction.

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 and 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 3.2.1 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.

There exists other similar benchmarks which aim to capture open-ended interaction like Kinetix (Matthews et al., 2025), XLand (Team et al., 2021), 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 generalist agents from scratch (Hafner et al., 2024; Ma et al., 2022; Zhao et al., 2024; Malagón et al., 2025a). While based on the similar block-building foundations and an appealing benchmark, we believe BuilderBench is better suited for academic research due to its simplicity, 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.

8 Environment Details

Figure 8 shows the system prompt provided to language model agents, which describes the environment and the action schema. Figure 9 and Figure 10 provide an example of a language based observation and a language based action.

The entire essential information about the scene can be conveyed by representing the state of the robot (joint angles and velocities, end effector poses and velocities) and the state of the blocks (poses and velocities). This information can be represented and compressed in language using the provided language wrapper. Users can also render multiple images of the scene via various cameras using our code.

9 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[6].

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 easy tasks to 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 structures which even we were not able to build.

[6]Please checkout our code for the script to teleoperate the robot.

10 Reinforcement learning from scratch

This section provides details on our RL experiments. These experiments are conducted with a simpler setup as compared to our language model experiments. In the simulator, instead of the entire robot, we only include the robot gripper which can move around in 3-D space. We do not include the entire robot to make the simulator faster and parallelizable with jax's jit compilation. Moreover, inverse kinematics is a solvable and orthogonal problem.

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).

Reward function. These experiments use dense rewards which are permutation invariant to the cube order. Cubes are said to form the target structure if the distance between each cube and its corresponding target is less than two 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 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).

Results. The benchmarking results on tasks with atmost 4 cubes are provided in Figure 11. As the number of cubes and the complexity of the tasks increase, no RL algorithm is able to achieve a non zero success. Issues like sample inefficiency and exploration are the main bottleneck for RL algorithms. We leave a more detailed study of failure modes and improvements for tabula rasa RL algorithms for future work.

11 Self-supervised RL pretraining

To evaluate exploration based RL pretraining, we propose a different protocol. The agent first 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 (Section 13). 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.

We implemented four algorithms, sampling for learnability (**SFL** (Rutherford et al., 2024)), maximum entropy gain exploration (**MEGA** (Pitis et al., 2020)), upside down RL (**UDRL**) (Schmidhuber, 2020) and random network distillation (**RND**) (Burda et al., 2018). SFL and MEGA sample autotelic goals from previously visited states, for the agent to learn to reach them. SFL is an unsupervised environment design (Dennis et al., 2020) algorithm, which samples goals with the highest learnability (variance of success). MEGA is an unsupervised goal sampling (Florensa et al., 2018; OpenAI et al., 2021) algorithm, which samples goals inversely proportional to their visitation density. Both algorithms are implemented using proximal policy optimization (PPO) (Schulman et al., 2017).UDRL and RND are self-supervised algorithms. UDRL learns to reach previously explored goals using hindsight relabelling (Andrychowicz et al., 2018) and RND explores the environment using an intrinsic reward bonus. Both of these algorithms sample 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 (Section 13) at various points during training. We report normalized episodic success and returns in Figure 12.

Results. As seen in Figure 12, 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.

12 Compute Usage

The LLM based experiments in fig. 6 and fig. 5 did not require any GPUs. Some runs from these figures were replicated on a relatively low end personal laptop with AMD Ryzen 5 4600H CPU. The experiments in fig. 11 and fig. 12 were performaned using A100 and H100 GPUs. Each run can be performed on a single A100 or H100 GPU with atleast 40 GB of VRAM.

Environment System Prompt

```
You are an agent who must control a UR5e robot arm with a Robotiq 2F-85 parallel jaw
gripper in a simulated environment with cube-shaped blocks.

Environment Overview

1. Simulation is implemented using MuJoCo and approximates Newtonian physics.
2. All positions are in meters; all rotations are in radians. All coordinates are in the
global frame, with z=0 as the ground surface.
3. Each cube has an edge length of 0.04 meters.
4. The gripper's maximum opening is 0.085 meters.
5. The environment provides these observations:
- Current timestep and the total number of timesteps in the episode.
- End-effector position and yaw.
- Potential target position for the end-effector.
- Positions and yaws of all cubes.
- Target locations for some cubes.
- Success condition for cubes with targets.
6. You have to output an action (conforming to the defined Action Schema below) at each
step with the goal of achieving the success condition for ALL cubes that have an assigned
target. Success condition: all cubes are at their respective targets and remain there
stably. Move the end effector at its target position (if specified) upon task completion.

Action Schema

Action Types
1. "pick_and_place": Executes a Pick -> Lift -> Place -> Retreat plan using low-level
controls (no collision avoidance).
- "cube_id": int -- ID of the object to grasp
- "grasp_yaw": int -- 0 or 1 (perpendicular axes for grasping a cube)
- "pos": [x, y, z] -- Target position to place the cube
- "yaw": float -- Cube placement rotation (radians)
- Note: After placing, the arm retreats to [0.3, 0.0, 0.25].

2. "pick_and_hold": Executes a Pick -> Lift -> Hold plan (no collision avoidance).
- "cube_id": int -- ID of the object to grasp
- "grasp_yaw": int -- 0 or 1
- "pos": [x, y, z] -- Target hold position
- "yaw": float -- Target hold rotation (radians)
- Note: The arm holds the cube in the specified pose.

3. "eef_target": Uses a PD controller to move the end-effector to a specified target
position and yaw (no collision avoidance).
- "pos": [x, y, z] -- Target end-effector position (meters)
- "yaw": float -- Target end-effector yaw (radians)
- "gripper": float -- 0.0 (open) to 1.0 (closed)

4. "low_level": Applies delta end-effector control for a few timesteps (fine-grained
control; plan sequences to achieve high-level tasks).
- "action": [delta_x, delta_y, delta_z, delta_yaw, delta_gripper_strength]

Output Format
Always output actions as a single, valid JSON object with the specified key/value
structure.

Examples
- Pick and Place:
"type": "pick_and_place", "cube_id": 0, "grasp_yaw": 0, "pos": [0.5, -0.2, 0.02],
"yaw": 1.57

- Pick and Hold:
"type": "pick_and_hold", "cube_id": 0, "grasp_yaw": 0, "pos": [0.5, -0.2, 0.2], "yaw":
0.0

- End-Effector Target:
"type": "eef_target", "pos": [0.45, 0.1, 0.3], "yaw": 1.57, "gripper": 1.0

- Low Level:
"type": "low_level", "action": [0.3, 0.0, 0.1, 0.0, 1.0]
```

Figure 8: Environment system prompt provided to language model agents.

Example Language Observation

```
Time: 2.88 / 18.00

End Effector State
- End Effector: pos=[0.323, 0.004, 0.228], yaw=-0.032, target=[0.300, 0.000, 0.250]
- Gripper: 0.000

Cube State
- Cube 0: pos=[0.452, -0.001, 0.020], yaw=0.009, target=[0.450, 0.000, 0.020],
success=True
- Cube 1: pos=[0.300, 0.000, 0.020], yaw=0.000, target=[0.450, 0.000, 0.060],
success=False
- Cube 2: pos=[0.300, 0.080, 0.020], yaw=0.000, target=[0.450, 0.000, 0.100],
success=False
```

Figure 9: Example language observation provided by the language interface of BuilderBench for a cube stacking task. The observation contains the current episode time, the total episode time, the state of the end effector and the state and success conditions for each cube.

Example Language Action

```
"'type': 'pick_and_place', 'cube_id': 1, 'grasp_yaw': 0, 'pos': [0.45, 0.0, 0.06],
'yaw': 0.0"
```

Figure 10: Example of an action output by an LLM agent in the desired JSON format. The agent commands the environment to pick cube 1 and place in a particular pose.

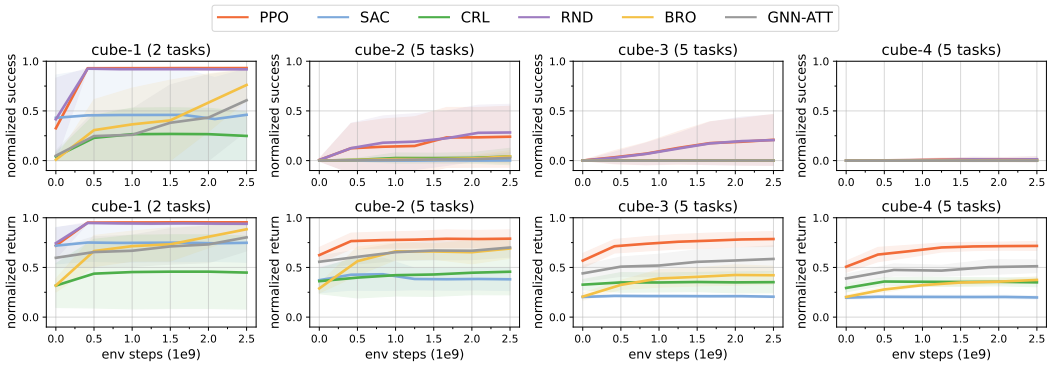


Figure 11: **RL from scratch.** As the number of cubes and the complexity of the tasks increase, no RL algorithm is able to achieve a non zero success.

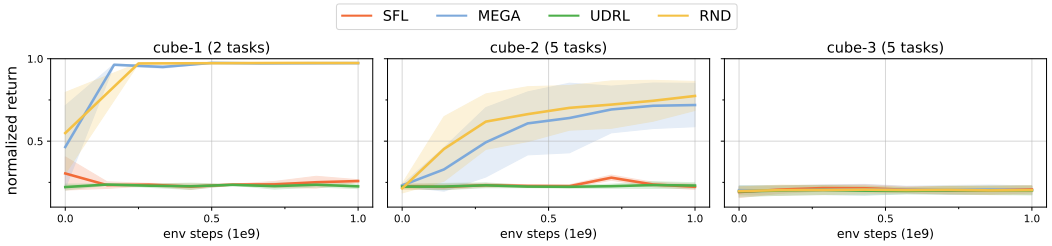
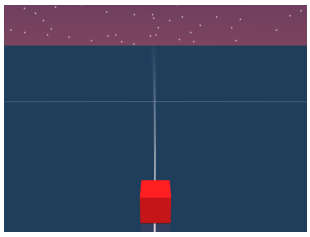
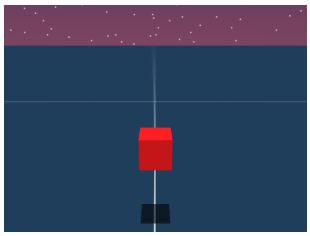
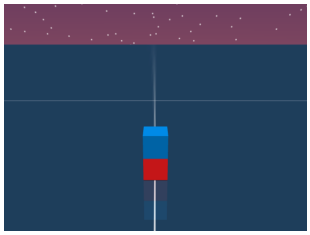
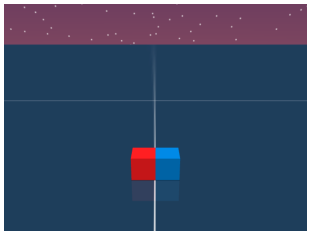
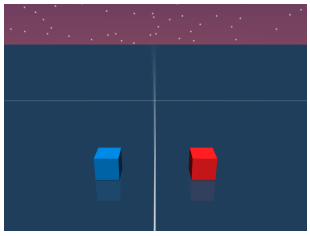
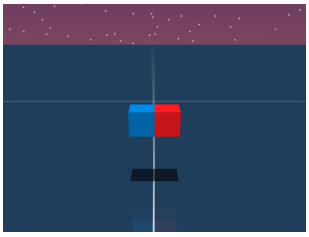
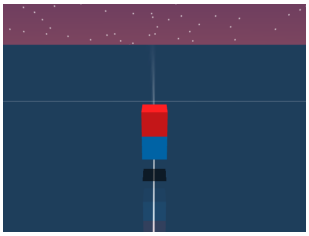
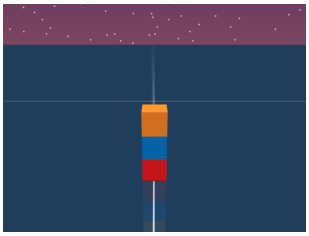
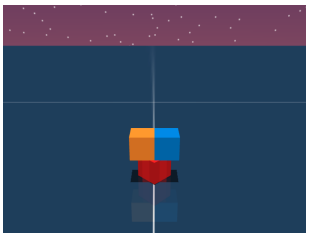
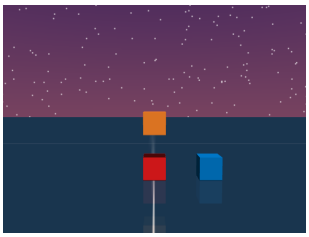
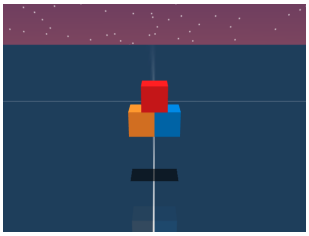


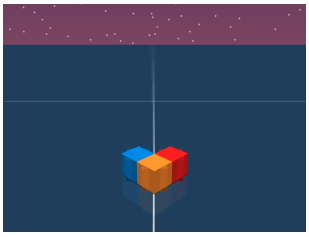
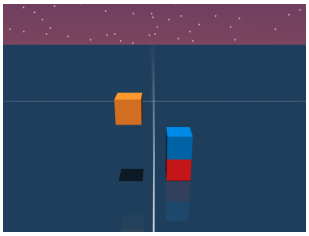
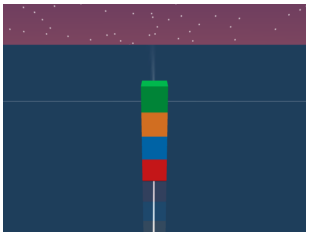
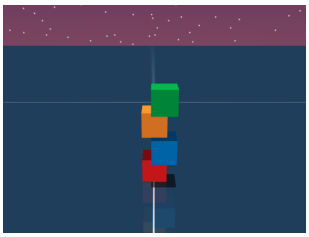
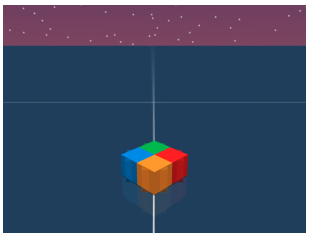
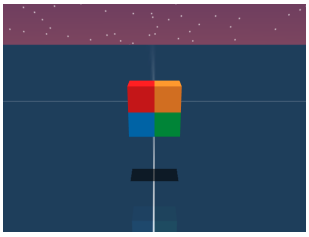
Figure 12: **Self-supervised evaluation on BuilderBench task-suite.** We evaluate MEGA (Pitis et al., 2020) and SFL (Rutherford et al., 2024) on 12 of the lowest complexity (yet still difficult) tasks from our task suite. The results show that directly using these algorithms out of the box only succeeds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

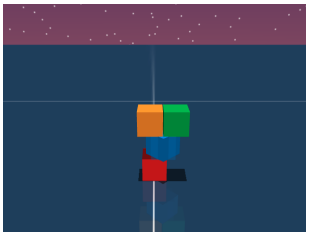
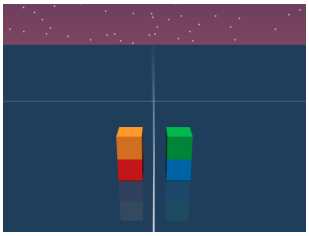
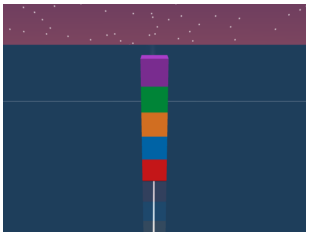
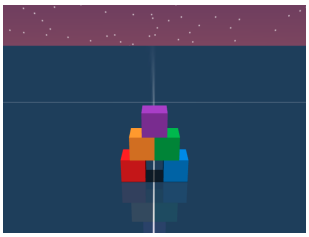
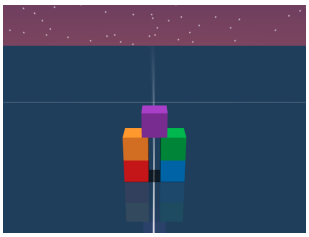
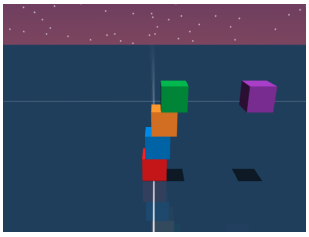
13 BuilderBench Task Suite

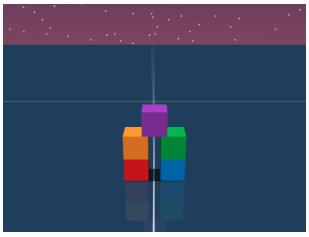
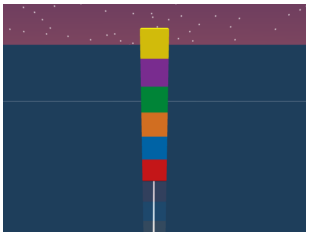
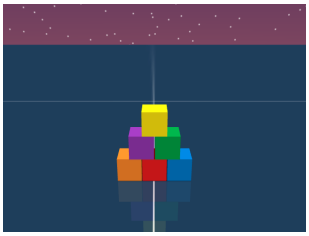
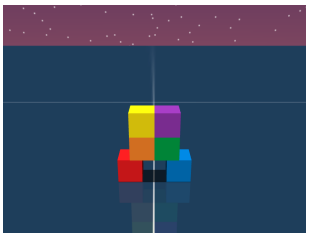
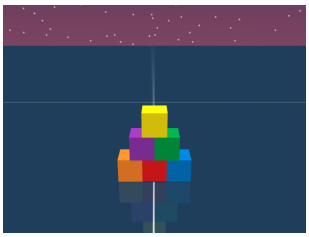
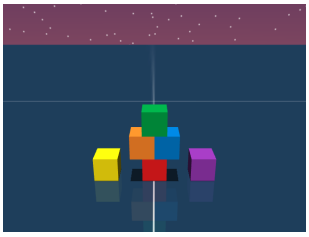
Section 13 provides the visualizations for the target structures of each tasks along with its assigned difficulty level.

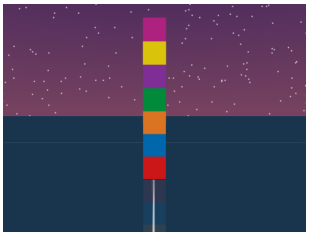
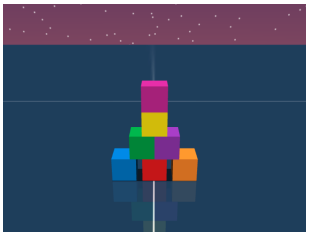
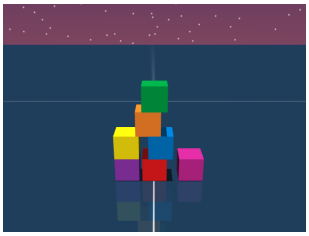
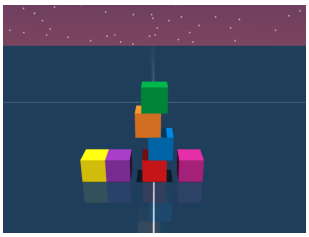
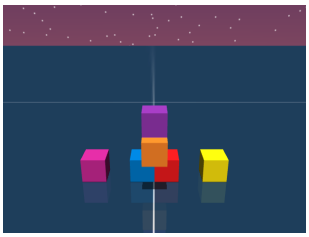
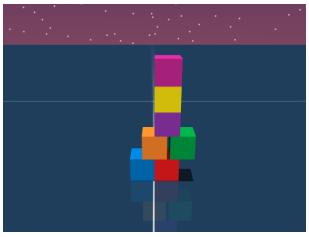
Task Name	Image	Difficulty
cube-1-task-1		Easy
cube-1-task-2		Easy
cube-2-task-1		Easy
cube-2-task-2		Easy
cube-2-task-3		Easy

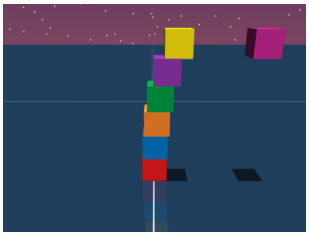
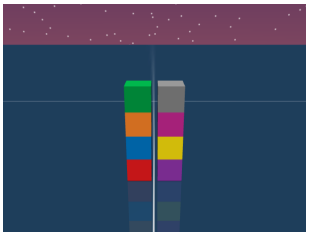
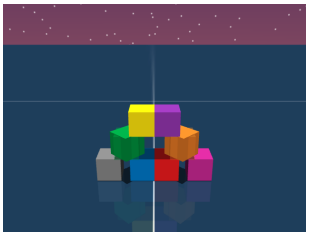
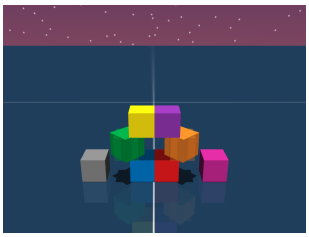
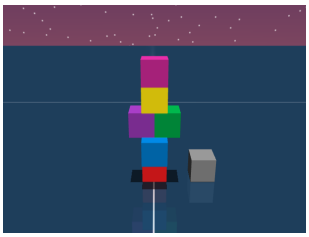
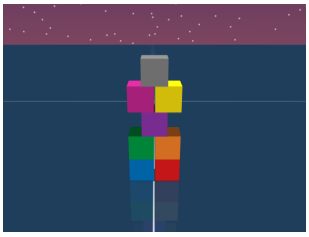
Task Name	Image	Difficulty
cube-2-task-4		Hard
cube-2-task-5		Hard
cube-3-task-1		Easy
cube-3-task-2		Hard
cube-3-task-3		Easy
cube-3-task-4		Hard

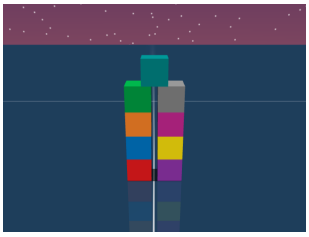
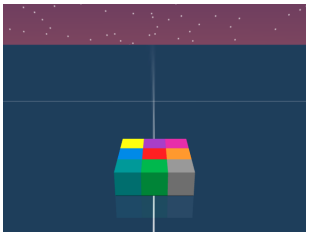
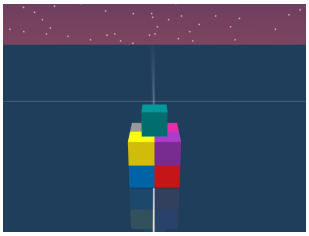
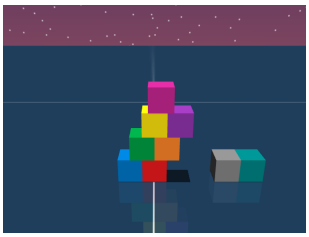
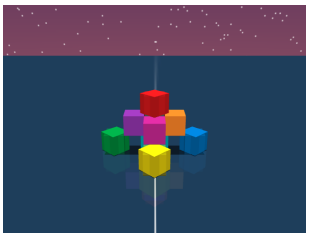
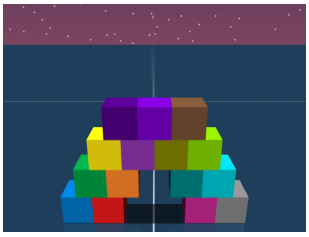
Task Name	Image	Difficulty
cube-3-task-5		Hard
cube-3-task-6		Easy
cube-4-task-1		Easy
cube-4-task-2		Easy
cube-4-task-3		Hard
cube-4-task-4		Hard

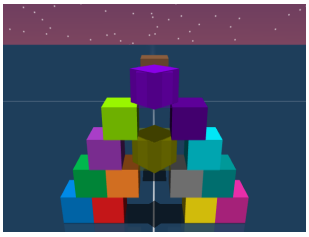
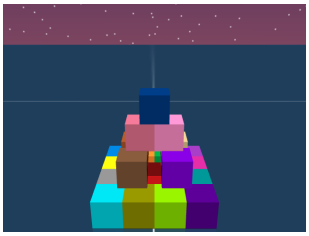
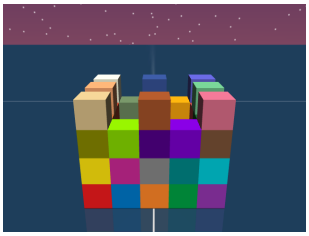
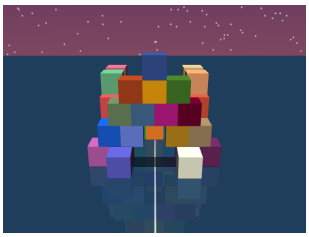
Task Name	Image	Difficulty
cube-4-task-5		Hard
cube-4-task-6		Easy
cube-5-task-1		Easy
cube-5-task-2		Easy
cube-5-task-3		Easy
cube-5-task-4		Hard

Task Name	Image	Difficulty
cube-5-task-5		Easy
cube-6-task-1		Easy
cube-6-task-2		Easy
cube-6-task-3		Hard
cube-6-task-4		Easy
cube-6-task-5		Hard

Task Name	Image	Difficulty
cube-7-task-1		Easy
cube-7-task-2		Easy
cube-7-task-3		Easy
cube-7-task-4		Hard
cube-7-task-5		Hard
cube-7-task-6		Hard

Task Name	Image	Difficulty
cube-7-task-7		Hard
cube-8-task-1		Easy
cube-8-task-2		Easy
cube-8-task-3		Hard
cube-8-task-4		Hard
cube-8-task-5		Hard

Task Name	Image	Difficulty
cube-9-task-1		Easy
cube-9-task-2		Hard
cube-9-task-3		Hard
cube-9-task-4		Hard
cube-10-task-1		Hard
cube-15-task-1		Hard

Task Name	Image	Difficulty
cube-15-task-2		Hard
cube-20-task-1		Hard
cube-50-task-1		Hard
cube-50-task-2		Hard