

000 BUILDERBENCH – A BENCHMARK FOR GENERALIST 001 AGENTS 002 003 004

005 **Anonymous authors**

006 Paper under double-blind review

007 008 ABSTRACT 009

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

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

032 1 THE NEED FOR A NEW BENCHMARK

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

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

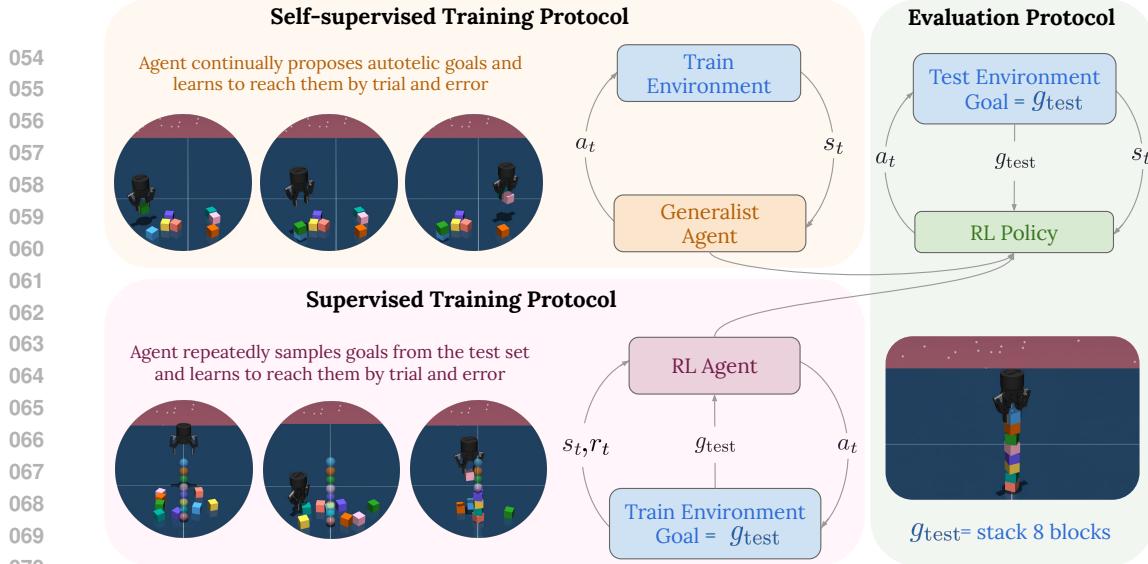


Figure 1: **The BuilderBench Benchmark.** (Top Left) Training consists of self-supervised exploration: agents can collect data to learn to reach various goals via trial and error. Agents are not given any information about the test-time goals or their distribution. (Right) During evaluation, the agent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

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

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

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

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

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

We summarize the main contributions of this paper below:

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

2 RELATED WORK.

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

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

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

162 Another set of methods that are closely related are ones which treat the problem of efficiently gen-
 163 eralizing to unseen tasks as a learning problem itself. Meta-learning (Caruana, 1998; Finn et al.,
 164 2017; Schmidhuber, 1987) and few-shot learning (Vinyals et al., 2016; Snell et al., 2017) fall under
 165 this category. Initial progress was driven by benchmarks that arranged common supervised learn-
 166 ing tasks episodically, testing how quickly models adapt to new tasks (Lake et al., 2015; Dhillon
 167 et al., 2020). Later work found that self-supervised pre-training on diverse datasets provided enough
 168 prior knowledge to directly solve most of the common supervised learning tasks (Radford et al.,
 169 2021; Brown et al., 2020; Devlin et al., 2018), blurring the boundary between memorizing prior
 170 knowledge and efficiently generalizing. We argue that open-ended domains and tasks are needed to
 171 disentangle the two. ARC-AGI (Chollet, 2019) uses the open-ended domain of discrete puzzles to
 172 measure a model’s ability to efficiently use its priors. ARC tests models on a set of novel puzzles
 173 that require on-the-fly composition of a minimal set of core principles (Chollet, 2019; Spelke & Kin-
 174 zler, 2007). BuilderBench is similarly structured. Solving tasks from the BuilderBench task-suite
 175 not only requires a concrete set of priors (e.g., an understanding of Newtonian physics), but requires
 176 using these priors to build unseen structures on-the-fly. Unlike ARC-AGI, where priors are directly
 177 provided through examples of solved puzzles, in BuilderBench agents have to discover priors on
 178 their own through interaction.

179 In addition to exploration and generalization, the BuilderBench task-suite highlights how block-
 180 building can also be used to evaluate various types of reasoning abilities (see Section 5.1 for details).
 181 Many of these abilities are typically studied only in isolation. For e.g., intuitive physics is evaluated
 182 in Chow et al. (2025); Riochet et al. (2020), motor skills in James et al. (2019); Melnik et al. (2021),
 183 planning in Valmeekam et al. (2023), mathematical reasoning in Lewkowycz et al. (2022); Ahn et al.
 184 (2024). In recent years, reasoning is almost exclusively studied using language models pretrained
 185 on data. However, BuilderBench allows us to evaluate and visualize reasoning that is not grounded
 186 in language and not learned using human data.

187 The most similar benchmarks to BuilderBench are recent benchmarks like Kinetix (Matthews et al.,
 188 2025), XLand (Team et al., 2021; Nikulin et al., 2023), and Minecraft (Guss et al., 2019). Kinetix
 189 provides a diverse set of rigid body tasks, constrained to 2D, to test zero shot generalization of
 190 agents. Tasks in Kinetix are procedurally generated. Unlike BuilderBench, these tasks do not
 191 clearly test diverse logical and mathematical reasoning abilities. XLand provides a vast set of multi-
 192 agent video-game like tasks, but is closed source and not readily available for academic research.
 193 Minecraft is a popular open-ended game that revolves around building various artifacts with blocks
 194 that has been used to develop generally capable agents from scratch (Hafner et al., 2024; Ma et al.,
 195 2022; Zhao et al., 2024). While based on the similar block-building foundations and an appealing
 196 benchmark, we believe BuilderBench is better suited for academic research due to the much faster
 197 speed of its simulator and an extensive carefully curated task-suite. Finally, BuilderBench is fully
 198 open source, making all of its components flexible and easy to adapt. In Appendix C, we have added
 199 more discussion on qualitative comparsion of BuilderBench with prior benchmarks.

200 3 BUILDERBENCH - A BENCHMARK FOR GENERALIST AGENTS

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

206 4 BUILDERBENCH ENVIRONMENT.

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

216 3D space² and interact with the n cubes. All interactions approximate real physics simulated using
 217 MuJoCo (Todorov et al., 2012).

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

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

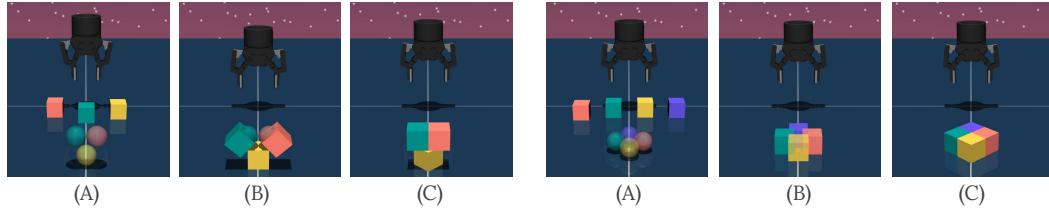
226 **Task specification.** Each task corresponds to a physically stable target structure built using cubes.
 227 To specify this structure, we provide a vector of target cube positions (\mathbb{R}^{3k}), where $k \leq n$ is the
 228 number of cubes in the target cube structure. This allows us to specify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

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

235 5 BUILDERBENCH TASK SUITE

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

245 5.1 A CASE STUDY OF FIVE TASKS



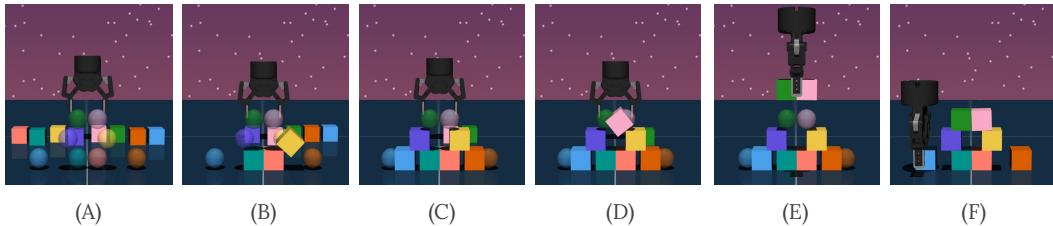
255 Figure 2: **T-Block** (Left) and **Four Cube Packing** (Right)

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

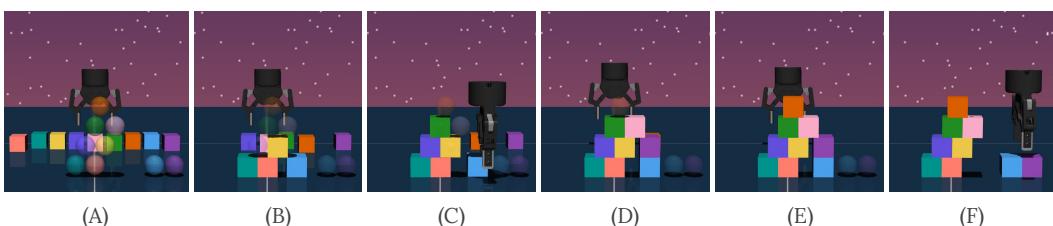
262 **Example 2: Four Cube Packing.** This task tests geometric reasoning and spatial packing. The
 263 target structure is an arrangement of four cube centers placed at some distance along the four
 264 cardinal directions on the floor (see (A) of Figure 2). The distance is chosen such that the placement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

268
 269 ²We do not include the entire robot because inverse kinematics is a solvable and orthogonal problem. This
 also significantly increases the reach of the robotic arm. This robot can be conceptually thought of as a crane.

270 without collision (see (C)). Due to the two fingered morphology of the robot, this task cannot be
 271 solved using pick and place primitives, but would require nudging the final block in place.
 272

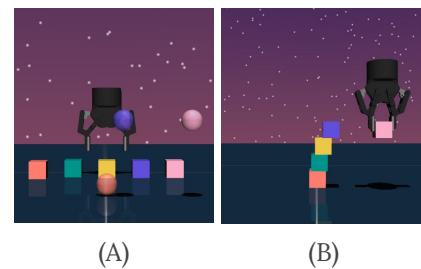
280 **Figure 3: Hexagonal Portal**

281 **Example 3: Hexagonal Portal.** This task requires constructing a hexagonal, portal-like structure
 282 using eight cubes and two extra cubes placed on the floor on either side. (Figure 3). A direct attempt
 283 to place the yellow or indigo cubes leads to toppling (see (B)). To stabilize them, the agent must
 284 first build two supporting scaffolds (see (C)). After the first two layers are in place, the pink and
 285 green cubes cannot be added independently since each would collapse inward (see (D)). Because
 286 no additional blocks remain, another scaffold cannot be used for support. Instead, the agent must
 287 discover a non-trivial maneuver – lifting and placing the pink and green cubes simultaneously (see
 288 (E)). Finally, the temporary orange and light-blue scaffolds must be carefully removed and placed
 289 in their desired position to complete the structure (see (F)). This task requires long-horizon planning
 290 and learning emergent skills like building scaffolds and learning to pick two cubes at once.
 291

299 **Figure 4: Leaning Tower**

300 **Example 4: Leaning Tower.** The target is a leaning tower composed of seven blocks and two extra
 301 cubes placed on the floor (see (A) of Figure 4). Solving this task demands building two scaffolds
 302 and re-using the first one for the main tower. It also requires an understanding of the concept of
 303 counterweights for generating a stable overhang (an outward extension). The solution itself requires
 304 multiple steps of high level planning. After building the base, the yellow block in the second layer
 305 must be supported by a temporary scaffold (see blue cube in (B)). To stabilize the structure, the
 306 agent needs to add counterweights (the indigo and green cubes in (C)) and only then remove the
 307 scaffold (see (D)). To build the third and fourth layer, the agent has to build another set of scaffolds
 308 and counterweights. In particular, placing the pink block in the third layer requires a two-cube
 309 vertical scaffold (see (E)). Finally, the tower is completed by adding the counterweights (the green
 310 and orange cubes in (F)) and removing and repositioning the last scaffold (see (G)).
 311

312 **Example 5: Maximum Overhang Problem.** In this
 313 task, the environment contains five cubes, but the task
 314 only specifies the target positions for three cubes (see (A)
 315 of Figure 5). But to put those three cubes in the target
 316 location, the agent will need to use all five blocks. To
 317 correctly place the green and the yellow cubes (whose target
 318 positions are not specified) in order to complete the task,
 319 the agent needs to solve the popular maximum overhang
 320 problem (see Paterson et al. (2007) for the solution). The
 321 main intuition is that at any level, the collective center
 322 of the mass of all the cubes above, should not be on the
 323 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.

856 **Figure 5: Maximum overhang problem**

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

5.2 DESIGN PHILOSOPHY BEHIND THE BUILDERBENCH TASK-SUITE.

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

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

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

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

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

6 TRAINING AND EVALUATION PROTOCOLS.

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

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

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

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

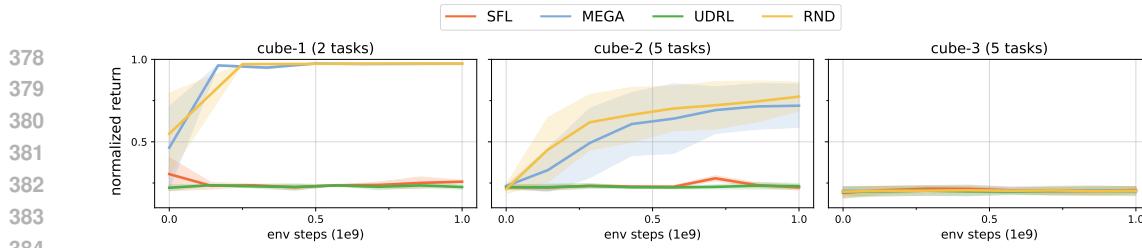


Figure 6: **Self-supervised evaluation on BuilderBench task-suite.** We evaluate MEGA (Pitis et al., 2020) and SFL (Rutherford et al., 2024) on 12 of the lowest complexity (yet still difficult) tasks from our task suite. The results show that directly using these algorithms out of the box only succeeds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

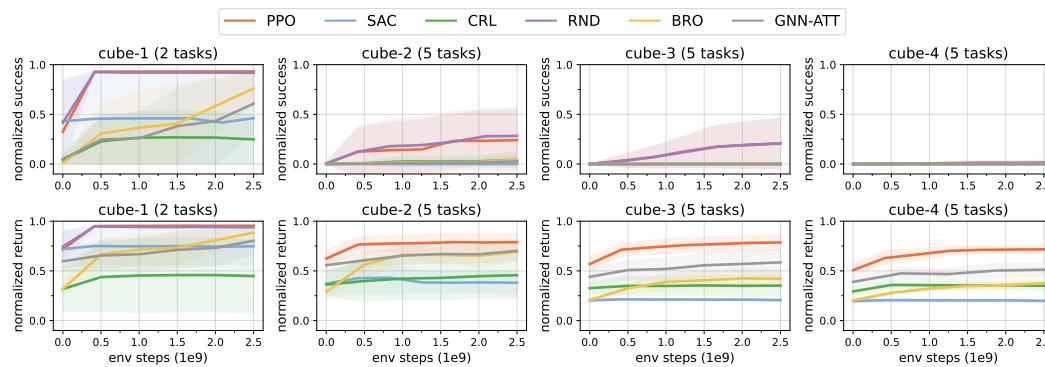


Figure 7: **Training on the test goals.** Training on the test goals improves both the returns and success achieved by the best agents. However as the number of cubes and the complexity of the tasks increase, current algorithms are not able to achieve a non zero success.

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

7 BENCHMARKING AND DISCUSSION.

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

Multi-task self-supervised protocol. 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 Appendix E at various points during training. We report normalized episodic success and returns in Figure 6.

432 As seen in Figure 6, both algorithms achieve trivial performance on tasks with three cubes. MEGA
 433 is able to complete both tasks with one cube, and shows improvement on tasks with two cubes.
 434 While these results indicate that the tested algorithms are not directly scalable to complex tasks,
 435 it primarily underscores the inherent difficulty of the task setup itself. We believe that research in
 436 developing new algorithms (or revisiting old ones) is required to solve these tasks.
 437

438 **Single-task supervised protocol.** For this protocol, we benchmark six RL algorithms, proximal
 439 policy optimization (PPO) (Schulman et al., 2017), soft actor critic (SAC) (Haarnoja et al.,
 440 2018), contrastive RL (CRL) (Eysenbach et al., 2022), random network distillation (RND) (Burda
 441 et al., 2019), bigger-regularized-optimistic (BRO) (Nauman et al., 2024) and graph-attention-
 442 network (GNN-ATT) (Ghasemipour et al., 2022). The benchmarking results on 17 tasks are pro-
 443 vided in Figure 7. All experiments use dense rewards.

444 7.1 EVALUATING LARGE LANGUAGE MODELS

446 It has been shown that scaling pretraining and inference-time compute can significantly
 447 enhance the reasoning abilities of language models (Kaplan et al., 2020). To test
 448 whether the latest proprietary models can solve tasks from our task-suite, we evaluated
 449 ChatGPT-5³ and Gemini 2.5 Pro (Team, 2025) on all five tasks discussed in Section 5.1.
 450 Each model was provided with a descriptive prompt about the environment and the
 451 task. The goal of the model was to provide
 452 a high-level open-loop plan in language,
 453 such that following this plan would stably
 454 build the target structure. A simple exam-
 455 ple task with a correct solution was also
 456 included in the prompt (see Appendix D
 457 for the exact prompts and solutions). Figure
 458 8 shows that both models, despite us-
 459 ing inference-time compute, are not able
 460 to provide the correct high-level plan to solve any of the tasks. While this is not meant to be an
 461 extensive evaluation of current models’ abilities, it highlights how solving our tasks requires non-
 462 obvious steps of reasoning that are beyond what current models can achieve through scaling alone.
 463

464 Figure 8: Evaluating language models on Builder-
 465 Bench.

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

466 8 LIMITATIONS AND CONCLUSION

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

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

483
 484
 485 ³<https://openai.com/index/introducing-gpt-5/>

486 9 REPRODUCIBILITY STATEMENT

488 All experiments in this paper are completely reproducible. We have attached our code for the simu-
 489 lator, the task-suite and the implementation of all algorithms as a part of the supplemental materials.
 490 Additionally, BuilderBench is based on MuJoCo (Todorov et al., 2012) and Jax (Bradbury et al.,
 491 2018) both of which are open-sourced libraries. For experiments using proprietary language mod-
 492 els, we have provided the exact models Section 7.1 and the prompts Appendix D which were used
 493 for experiments.

494 495 REFERENCES

497 Faseeh Ahmad, Esra Erdem, and Volkan Patoglu. A formal framework for robot construction prob-
 498 lems: A hybrid planning approach, 2019. URL <https://arxiv.org/abs/1903.00745>.

499 Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models
 500 for mathematical reasoning: Progresses and challenges, 2024. URL <https://arxiv.org/abs/2402.00157>.

503 Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob
 504 McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay, 2018.
 505 URL <https://arxiv.org/abs/1707.01495>.

506 Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning envi-
 507 ronment: an evaluation platform for general agents. *J. Artif. Int. Res.*, 47(1):253–279, May 2013.
 508 ISSN 1076-9757.

510 Clément Bonnet, Daniel Luo, Donal John Byrne, Shikha Surana, Sasha Abramowitz, Paul Duck-
 511 worth, Vincent Coyette, Laurence Illing Midgley, Elshadai Tegegn, Tristan Kalloniatis, Omayma
 512 Mahjoub, Matthew Macfarlane, Andries Petrus Smit, Nathan Grinsztajn, Raphael Boige, Cem-
 513 lyn Neil Waters, Mohamed Ali Ali Mimouni, Ulrich Armel Mbou Sob, Ruan John de Kock,
 514 Siddarth Singh, Daniel Furelos-Blanco, Victor Le, Arnu Pretorius, and Alexandre Laterre. Ju-
 515 manji: a diverse suite of scalable reinforcement learning environments in JAX. In *The Twelfth*
 516 *International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=C4CxQmp9wc>.

517 James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal
 518 Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao
 519 Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL <http://github.com/jax-ml/jax>.

522 Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and
 523 Wojciech Zaremba. Openai gym, 2016. URL <https://arxiv.org/abs/1606.01540>.

525 Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhari-
 526 wal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal,
 527 Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M.
 528 Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin,
 529 Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford,
 530 Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of the*
 531 *34th International Conference on Neural Information Processing Systems*, NIPS ’20, Red Hook,
 532 NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

533 Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network
 534 distillation, 2018. URL <https://arxiv.org/abs/1810.12894>.

535 Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network
 536 distillation. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=H11JJnR5Ym>.

539 Rich Caruana. *Multitask learning*, pp. 95–133. Kluwer Academic Publishers, USA, 1998. ISBN
 0792380479.

540 Beth Casey, Nicole Andrews, Holly Schindle, Joanne Kersh, Alexandra Samper, and Juanita
 541 Copley. The development of spatial skills through interventions involving block building ac-
 542 tivities. *Cognition and Instruction - COGNITION INSTRUCT*, 26:269–309, 07 2008. doi:
 543 10.1080/07370000802177177.

544 Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Philipp Koehn, and Tony
 545 Robinson. One billion word benchmark for measuring progress in statistical language modeling,
 546 2014. URL <https://arxiv.org/abs/1312.3005>.

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

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

550 Wei Chow, Jiageng Mao, Boyi Li, Daniel Seita, Vitor Campagnolo Guizilini, and Yue Wang. Phys-
 551 bench: Benchmarking and enhancing vision-language models for physical world under-
 552 standing. In *The Thirteenth International Conference on Learning Representations*, 2025. URL
 553 <https://openreview.net/forum?id=Q6a9W6kzv5>.

554 Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural genera-
 555 tion to benchmark reinforcement learning, 2020. URL <https://arxiv.org/abs/1912.01588>.

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

557 Li Deng. The mnist database of handwritten digit images for machine learning research [best of
 558 the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. doi: 10.1109/MSP.2012.
 559 2211477.

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

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

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

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

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

573 Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learn-
 574 ing as goal-conditioned reinforcement learning. In Alice H. Oh, Alekh Agarwal, Danielle Bel-
 575 grave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.
 576 URL <https://openreview.net/forum?id=vGQiU5sqUe3>.

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

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

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

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

597

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

602 Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic con-
 603 trol. *ArXiv*, abs/1611.07507, 2016. URL <https://api.semanticscholar.org/CorpusID:2918187>.

604

605 Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy
 606 learning: Solving long-horizon tasks via imitation and reinforcement learning. In Leslie Pack
 607 Kaelbling, Danica Kragic, and Komei Sugiura (eds.), *Proceedings of the Conference on Robot
 608 Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1025–1037. PMLR,
 609 30 Oct–01 Nov 2020. URL <https://proceedings.mlr.press/v100/gupta20a.html>.

610

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

612

613 William H. Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela
 614 Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations,
 615 2019. URL <https://arxiv.org/abs/1907.13440>.

616

617 Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy
 618 maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and An-
 619 dreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*,
 620 volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 10–15 Jul
 621 2018. URL <https://proceedings.mlr.press/v80/haarnoja18b.html>.

622

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

624

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

627

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

630

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

637

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

640

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

644

645 Leslie Pack Kaelbling. Learning to achieve goals. In *International Joint Conference on Artificial In-
 646 telligence*, 1993. URL <https://api.semanticscholar.org/CorpusID:5538688>.

648 Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child,
 649 Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language
 650 models, 2020. URL <https://arxiv.org/abs/2001.08361>.

651 Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot gener-
 652 alisation in deep reinforcement learning. *J. Artif. Int. Res.*, 76, May 2023. ISSN 1076-9757. doi:
 653 10.1613/jair.1.14174. URL <https://doi.org/10.1613/jair.1.14174>.

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

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

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

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

665 Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhut-
 666 dinov. Efficient exploration via state marginal matching, 2020. URL <https://arxiv.org/abs/1906.05274>.

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

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

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

678 Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson,
 679 Samuel Coward, and Jakob Foerster. Craftax: a lightning-fast benchmark for open-ended re-
 680 inforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*,
 681 ICML’24. JMLR.org, 2024.

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

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

688 MCB, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal,
 689 Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable rein-
 690 forcement learning environments for goal-oriented tasks, 2023. URL <https://arxiv.org/abs/2306.13831>.

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

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

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

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

715 Alexander Nikulin, Vladislav Kurenkov, Ilya Zisman, Viacheslav Sinii, Artem Agarkov, and Sergey
 716 Kolesnikov. XLand-minigrid: Scalable meta-reinforcement learning environments in JAX.
 717 In *Intrinsically-Motivated and Open-Ended Learning Workshop @NeurIPS2023*, 2023. URL
 718 <https://openreview.net/forum?id=xALDC4aHGz>.

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

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

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

726 Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V. Hafner. Intrinsic motivation systems for au-
 727 tonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–
 728 286, 2007. doi: 10.1109/TEVC.2006.890271.

729 Mike Paterson, Yuval Peres, Mikkel Thorup, Peter Winkler, and Uri Zwick. Maximum overhang,
 730 2007. URL <https://arxiv.org/abs/0707.0093>.

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

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

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

742 Sai Rajeswar, Pietro Mazzaglia, Tim Verbelen, Alexandre Piché, Bart Dhoedt, Aaron Courville, and
 743 Alexandre Lacoste. Mastering the unsupervised reinforcement learning benchmark from pixels.
 744 In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org,
 745 2023.

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

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

759

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

763

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

766

767 Alexander Rutherford, Michael Beukman, Timon Willi, Bruno Lacerda, Nick Hawes, and
 768 Jakob Nicolaus Foerster. No regrets: Investigating and improving regret approximations for cur-
 769 riculum discovery. In *The Thirty-eighth Annual Conference on Neural Information Processing
 770 Systems*, 2024. URL <https://openreview.net/forum?id=iEeiZ1Tbts>.

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

773 Jürgen Schmidhuber. Evolutionary principles in self-referential learning, or on learning how to learn:
 774 The meta-meta-. hook, 1987. URL [https://api.semanticscholar.org/CorpusID:
 775 264351059](https://api.semanticscholar.org/CorpusID:264351059).

776

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

779 Olivier Sigaud, Gianluca Baldassarre, Cedric Colas, Stephane Doncieux, Richard Duro, Pierre-Yves
 780 Oudeyer, Nicolas Perrin-Gilbert, and Vieri Giuliano Santucci. A definition of open-ended learn-
 781 ing problems for goal-conditioned agents, 2024. URL [https://arxiv.org/abs/2311.
 782 00344](https://arxiv.org/abs/2311.00344).

783

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

785 David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche,
 786 Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman,
 787 Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine
 788 Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go
 789 with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016. doi:
 790 10.1038/nature16961.

791 Dorothy G. Singer, Roberta Michnick Golinkoff, and Kathy Hirsh-Pasek. *Play = Learning: How
 792 Play Motivates and Enhances Children's Cognitive and Social-Emotional Growth*. Oxford
 793 University Press, 09 2006. ISBN 9780195304381. doi: 10.1093/acprof:oso/9780195304381.
 794 001.0001. URL [https://doi.org/10.1093/acprof:oso/9780195304381.001.
 795 0001](https://doi.org/10.1093/acprof:oso/9780195304381.001.0001).

796

797 Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning.
 798 In *Proceedings of the 31st International Conference on Neural Information Processing Sys-
 799 tems*, NIPS'17, pp. 4080–4090, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN
 800 9781510860964.

801 Elizabeth S. Spelke and Katherine D. Kinzler. Core knowledge. *Developmental Science*, 10(1):
 802 89–96, 2007. doi: 10.1111/j.1467-7687.2007.00569.x.

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

807

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

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

812

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

816

817 Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Bud-
 818 den, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Ried-
 819 miller. Deepmind control suite, 2018. URL <https://arxiv.org/abs/1801.00690>.

820

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

823

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

825

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

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

834

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

839

840 Karthik Valmecikam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kamb-
 841 hampati. Planbench: An extensible benchmark for evaluating large language models on planning
 842 and reasoning about change, 2023. URL <https://arxiv.org/abs/2206.10498>.

843

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

848

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

853

854 Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Joseph Dudzik,
 855 Junyoung Chung, David Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan
 856 Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, L. Sifre, Trevor Cai, John P. Agapiou, Max
 857 Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David
 858 Budden, Yury Sulsky, James Molloy, Tom Le Paine, Caglar Gulcehre, Ziyun Wang, Tobias Pfaff,
 859 Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom
 860 Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver.
 861 Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575:350 – 354,
 862 2019. URL <https://api.semanticscholar.org/CorpusID:204972004>.

863

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

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

866

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

871

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

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

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

924 A ENVIRONMENT DETAILS

926 A.1 EPISODE LENGTHS

928 The episode length depends on the number of cubes present in the environment (N). For supervised
 929 tasks, the episode length is $100 + 100 \times N$ and for self-supervised tasks, the episode length is
 930 $500 \times N$.

932 A.2 REWARD FUNCTIONS

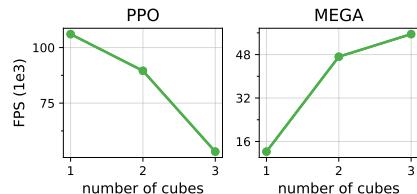
934 There are two types of reward functions provided in the benchmark, sparse and dense. The sparse
 935 reward is equal to -1 for all timesteps where the cubes do not form the target structure and 0 if the
 936 cubes form the target structure. Cubes are said to form the target structure if the distance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

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

948 B TRAINING SPEED.

950 In Figure 9, we plot the average environment steps per
 951 second for PPO (Figure 7) and MEGA (Figure 6). The
 952 PPO experiments were run on a single A100 GPU with
 953 12 CPU threads and the MEGA experiments were run on
 954 a single A6000 GPU with 12 CPU threads. The speed
 955 of the training depends on many factors like number of
 956 parallel environments, the update to data ratio, type of
 957 GPU and the number of CPU threads. The speed also
 958 depends on the environment step-time, number of cubes
 959 and the type of interactions in the environments (interac-
 960 tions with more active contacts typically take longer time
 961 to execute). An example of this can be seen in the right
 962 plot in Figure 9. For the self-supervised protocol, the envi-
 963 ronment step-time is set to be small for
 964 better physics approximation. MEGA with one block learns to pick the block up and move it around
 965 which we found to be inversely correlated with speed. The speeds could be scaled with more CPU
 966 threads, parallel environments, and faster algorithms for physics approximation.

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



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

Table 1: Qualitative Comparison of BuilerBench with previous benchmarks

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

C QUALITATIVE COMPARISON WITH PRIOR WORK.

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

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

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

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

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

1026 **D EVALUATING LANGUAGE MODELS: PROMPTS AND SOLUTIONS**
10271028
1029 **D.1 T BLOCK**
10301031 **User**
1032

1033 You are an agent that controls a two-finger robotic gripper (Robotiq 2F-85) which can
1034 navigate a constrained 3D space using position actuators for controlling x,y,z directions
1035 and yaw. The robot hand has two fingers with maximum width equal to 0.085 meters. The
1036 environment consists of a variable number of cubes of size 0.04 meters. The environment
1037 is implemented in MuJoCo and approximates Newtonian physics. All length position
1038 coordinates have units in meters and the yaw will be measured in radians.

1039 Problem description - You will be given the positions and the orientations of all the N cubes
1040 in the environment. You will also be given the target center positions of K cubes, where K
1041 can be less or equal to N. The target center positions correspond to a target structure that
1042 you need to build such that it always remains stable. The structures which are in unstable
1043 equilibria, that would fall apart after applying minimal perturbation, are not considered
1044 stable. Your goal is to provide high level language instructions for the gripper to follow
1045 such that it can build the stable target structure. All steps should be physically possible
1046 and implementing those steps one after the other should stably result in building the target
1047 structure. Only provide a list of an open loop plan consisting of simple instructions, for
1048 example, pick cube 1, place cube 1 at 0.1,0,0.02 with yaw 0, rotate cube 2 by pi/3 radians,
1049 nudge the cube at [0.1,0,0.02] to [0.12,0,0.02], etc. You can think before providing the final
1050 answer but please separate the final answer by saying “Final answer:” before providing the
1051 ordered plan of high level instructions.

1052 The following is an example of a simple task followed by its solution:
1053

1054 Task:
1055 N = 1
1056 K = 1

1057 Start location = [0.05, 0.0, 0.02]
1058 Target location = [0.1, 0.0, 0.02]

1059 Final answer:

1060 1) Pick up the cube from [0.05, 0.0, 0.02].
1061 2) Place the cube at [0.1, 0.0, 0.02] with a yaw of 0 radians.

1063 Here is the task:
1064 N = 3
1065 K = 3

1066 Start location = [[0.05, -0.08, 0.02], [0.05, 0. , 0.02], [0.05, 0.08, 0.02]]
1067 Target location = [[0.1, 0.02, 0.06], [0.1 , -0.02, 0.06], [0.1 , 0. , 0.02]]

1070 **ChatGPT-5**
1071

1072 Pick up the cube from [0.05, 0.0, 0.02].
1073 Place the cube at [0.1, 0.0, 0.02] with a yaw of 0 radians.
1074 Pick up the cube from [0.05, -0.08, 0.02].
1075 Place the cube at [0.1, -0.02, 0.06] with a yaw of 0 radians.
1076 Pick up the cube from [0.05, 0.08, 0.02].
1077 Place the cube at [0.1, 0.02, 0.06] with a yaw of 0 radians.

1080
1081
1082 **Gemini 2.5 Pro**
1083 Pick up the cube from [0.05, 0.0, 0.02].
1084 Place the cube at [0.1, 0.0, 0.02] with a yaw of 0 radians.
1085 Pick up the cube from [0.05, -0.08, 0.02].
1086 Place the cube at [0.1, -0.02, 0.06] with a yaw of 0 radians.
1087 Pick up the cube from [0.05, 0.08, 0.02].
1088 Place the cube at [0.1, 0.02, 0.06] with a yaw of 0 radians.
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133

1134 D.2 FOUR CUBE PACKING
11351136 **User**
11371138 You are an agent that controls a two-finger robotic gripper (Robotiq 2F-85) which can
1139 navigate a constrained 3D space using position actuators for controlling x,y,z directions
1140 and yaw. The robot hand has two fingers with maximum width equal to 0.085 meters. The
1141 environment consists of a variable number of cubes of size 0.04 meters. The environment
1142 is implemented in MuJoCo and approximates Newtonian physics. All length position
1143 coordinates have units in meters and the yaw will be measured in radians.
11441145 Problem description - You will be given the positions and the orientations of all the N cubes
1146 in the environment. You will also be given the target center positions of K cubes, where K
1147 can be less or equal to N. The target center positions correspond to a target structure that
1148 you need to build such that it always remains stable. The structures which are in unstable
1149 equilibria, that would fall apart after applying minimal perturbation, are not considered
1150 stable. Your goal is to provide high level language instructions for the gripper to follow
1151 such that it can build the stable target structure. All steps should be physically possible
1152 and implementing those steps one after the other should stably result in building the target
1153 structure. Only provide a list of an open loop plan consisting of simple instructions, for
1154 example, pick cube 1, place cube 1 at 0.1,0,0.02 with yaw 0, rotate cube 2 by pi/3 radians,
1155 nudge the cube at [0.1,0,0.02] to [0.12,0,0.02], etc. You can think before providing the final
1156 answer but please separate the final answer by saying “Final answer:” before providing the
1157 ordered plan of high level instructions.
11581159 The following is an example of a simple task followed by its solution:
11601161 Task:
1162 N = 1
1163 K = 1
1164 Start location = [0.05, 0.0, 0.02]
1165 Target location = [0.1, 0.0, 0.02]1166 Final answer:
1167 1) Pick up the cube from [0.05, 0.0, 0.02].
1168 2) Place the cube at [0.1, 0.0, 0.02] with a yaw of 0 radians.
11691170 Here is the task:
1171 N = 4
1172 K = 4
1173 Start location = [[0.05, -0.12, 0.02], [0.05, -0.04, 0.02], [0.05, 0.04, 0.02], [0.05, 0.12, 0.02]
1174]
1175 Target location = [[0.1, 0.02828427, 0.02], [0.1, -0.02828427, 0.02], [0.12828427, 0.0,
1176 0.02], [0.07171573, 0.0, 0.02]]
11771178 **ChatGPT-5**1179 Pick up the cube from [0.05, -0.12, 0.02].
1180 Place the cube at [0.1, -0.02828427, 0.02] with a yaw of 0 radians.
1181 Pick up the cube from [0.05, 0.12, 0.02].
1182 Place the cube at [0.1, 0.02828427, 0.02] with a yaw of 0 radians.
1183 Pick up the cube from [0.05, -0.04, 0.02].
1184 Place the cube at [0.07171573, 0.0, 0.02] with a yaw of 0 radians.
1185 Pick up the cube from [0.05, 0.04, 0.02].
1186 Place the cube at [0.12828427, 0.0, 0.02] with a yaw of 0 radians.
1187

1188
1189

Gemini 2.5 Pro

1190
1191
1192
1193
1194
1195
1196
1197

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

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

1242 D.3 HEXAGONAL PORTAL
12431244 User
12451246 You are an agent that controls a two-finger robotic gripper (Robotiq 2F-85) which can
1247 navigate a constrained 3D space using position actuators for controlling x,y,z directions
1248 and yaw. The robot hand has two fingers with maximum width equal to 0.085 meters. The
1249 environment consists of a variable number of cubes of size 0.04 meters. The environment
1250 is implemented in MuJoCo and approximates Newtonian physics. All length position
1251 coordinates have units in meters and the yaw will be measured in radians.1252 Problem description - You will be given the positions and the orientations of all the N cubes
1253 in the environment. You will also be given the target center positions of K cubes, where K
1254 can be less or equal to N. The target center positions correspond to a target structure that
1255 you need to build such that it always remains stable. The structures which are in unstable
1256 equilibria, that would fall apart after applying minimal perturbation, are not considered
1257 stable. Your goal is to provide high level language instructions for the gripper to follow
1258 such that it can build the stable target structure. All steps should be physically possible
1259 and implementing those steps one after the other should stably result in building the target
1260 structure. Only provide a list of an open loop plan consisting of simple instructions, for
1261 example, pick cube 1, place cube 1 at 0.1,0,0.02 with yaw 0, rotate cube 2 by pi/3 radians,
1262 nudge the cube at [0.1,0,0.02] to [0.12,0,0.02], etc. You can think before providing the final
1263 answer but please separate the final answer by saying "Final answer:" before providing the
1264 ordered plan of high level instructions.1265 The following is an example of a simple task followed by its solution:
12661267 Task:
1268 N = 1
1269 K = 1
1270 Start location = [0.05, 0.0, 0.02]
1271 Target location = [0.1, 0.0, 0.02]1272
1273 Final answer:
1274 1) Pick up the cube from [0.05, 0.0, 0.02].
1275 2) Place the cube at [0.1, 0.0, 0.02] with a yaw of 0 radians.1276 Here is the task:
1277 N = 8
1278 K = 8
1279 Start location = [[0.05, -0.24, 0.02], [0.05, -0.18, 0.02], [0.05, -0.12, 0.02], [0.05, -0.04,
1280 0.02], [0.05, 0.04, 0.02], [0.05, 0.12, 0.02], [0.05, 0.18, 0.02], [0.05, 0.24, 0.02]]
1281 Target location = [[0.1, 0.02, 0.02], [0.1, -0.02, 0.02], [0.1, 0.04, 0.06], [0.1, -0.04, 0.06],
1282 [0.1, 0.02, 0.1], [0.1, -0.02, 0.1], [0.1, 0.1, 0.02], [0.1, -0.1, 0.02]]1283
1284 ChatGPT-51285 Pick up the cube from [0.05, 0.18, 0.02]. Place the cube at [0.1, 0.02, 0.02] with a yaw of 0
1286 radians.
1287 Pick up the cube from [0.05, 0.24, 0.02]. Place the cube at [0.1, 0.1, 0.02] with a yaw of 0
1288 radians.
1289 Pick up the cube from [0.05, 0.12, 0.02]. Place the cube at [0.1, 0.04, 0.06] with a yaw of 0
1290 radians.
1291 Pick up the cube from [0.05, 0.04, 0.02]. Place the cube at [0.1, 0.02, 0.1] with a yaw of 0
1292 radians.
1293 Pick up the cube from [0.05, -0.18, 0.02]. Place the cube at [0.1, -0.02, 0.02] with a yaw of 0
1294 radians.
1295

1296
 1297 Pick up the cube from [0.05, -0.24, 0.02]. Place the cube at [0.1, -0.1, 0.02] with a yaw of 0
 1298 radians.
 1299 Pick up the cube from [0.05, -0.12, 0.02]. Place the cube at [0.1, -0.04, 0.06] with a yaw of
 1300 0 radians.
 1301 Pick up the cube from [0.05, -0.04, 0.02]. Place the cube at [0.1, -0.02, 0.1] with a yaw of 0
 1302 radians.

1303
 1304 **Gemini 2.5 Pro**

1305 Pick up the cube from [0.05, -0.24, 0.02].
 1306 Place the cube at [0.1, 0.02, 0.02] with a yaw of 0 radians.
 1307 Pick up the cube from [0.05, -0.18, 0.02].
 1308 Place the cube at [0.1, -0.02, 0.02] with a yaw of 0 radians.
 1309 Pick up the cube from [0.05, -0.12, 0.02].
 1310 Place the cube at [0.1, 0.1, 0.02] with a yaw of 0 radians.
 1311 Pick up the cube from [0.05, -0.04, 0.02].
 1312 Place the cube at [0.1, -0.1, 0.02] with a yaw of 0 radians.
 1313 Pick up the cube from [0.05, 0.04, 0.02].
 1314 Place the cube at [0.1, 0.04, 0.06] with a yaw of 0 radians.
 1315 Pick up the cube from [0.05, 0.12, 0.02].
 1316 Place the cube at [0.1, -0.04, 0.06] with a yaw of 0 radians.
 1317 Pick up the cube from [0.05, 0.18, 0.02].
 1318 Place the cube at [0.1, 0.02, 0.1] with a yaw of 0 radians.
 1319 Pick up the cube from [0.05, 0.24, 0.02].
 1320 Place the cube at [0.1, -0.02, 0.1] with a yaw of 0 radians.

1321
 1322
 1323
 1324
 1325
 1326
 1327
 1328
 1329
 1330
 1331
 1332
 1333
 1334
 1335
 1336
 1337
 1338
 1339
 1340
 1341
 1342
 1343
 1344
 1345
 1346
 1347
 1348
 1349

1350
1351

D.4 LEANING TOWER

1352
1353

1354

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

1359

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

1372

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

1373

Task:

1374

N = 1

1375

K = 1

1376

Start location = [0.05, 0.0, 0.02]

1377

Target location = [0.1, 0.0, 0.02]

1378

Final answer:

1379

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

1380

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

1381

Here is the task:

1382

N = 9

1383

K = 9

1384

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

1385

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

1386

1387

1388

1389

1390

1391

1392

1393

1394

1395

1396

1397

1398

1399

1400

1401

1402

1403

1404

1405

1406

1407

1408

1409

1410

1411

1412

1413

1414

1415

1416

1417

1418

1419

1420

1421

1422

1423

1424

1425

1426

1427

1428

1429

1430

1431

1432

1433

1434

1435

1436

1437

1438

1439

1440

1441

1442

1443

1444

1445

1446

1447

1448

1449

1450

1451

1452

1453

1454

1455

1456

1457

1458

1459

1460

1461

1462

1463

1464

1465

1466

1467

1468

1469

1470

1471

1472

1473

1474

1475

1476

1477

1478

1479

1480

1481

1482

1483

1484

1485

1486

1487

1488

1489

1490

1491

1492

1493

1494

1495

1496

1497

1498

1499

1500

1501

1502

1503

1504

1505

1506

1507

1508

1509

1510

1511

1512

1513

1514

1515

1516

1517

1518

1519

1520

1521

1522

1523

1524

1525

1526

1527

1528

1529

1530

1531

1532

1533

1534

1535

1536

1537

1538

1539

1540

1541

1542

1543

1544

1545

1546

1547

1548

1549

1550

1551

1552

1553

1554

1555

1556

1557

1558

1559

1560

1561

1562

1563

1564

1565

1566

1567

1568

1569

1570

1571

1572

1573

1574

1575

1576

1577

1578

1579

1580

1581

1582

1583

1584

1585

1586

1587

1588

1589

1590

1591

1592

1593

1594

1595

1596

1597

1598

1599

1600

1601

1602

1603

1604

1605

1606

1607

1608

1609

1610

1611

1612

1613

1614

1615

1616

1617

1618

1619

1620

1621

1404
 1405 Pick up the cube from [0.05, -0.08, 0.02]. Place the cube at [0.1, 0.02, 0.06] with a yaw of 0
 1406 radians.
 1407 Pick up the cube from [0.05, 0.0, 0.02]. Place the cube at [0.1, 0.0, 0.10] with a yaw of 0
 1408 radians.
 1409 Pick up the cube from [0.05, 0.08, 0.02]. Place the cube at [0.1, 0.04, 0.10] with a yaw of 0
 1410 radians.
 1411 Pick up the cube from [0.05, 0.16, 0.02]. Place the cube at [0.1, 0.01, 0.14] with a yaw of 0
 1412 radians.
 1413
 1414

Gemini 2.5 Pro

1415 Pick up the cube from [0.05, -0.3, 0.02] and place it at [0.1, -0.04, 0.02] with a yaw of 0
 1416 radians.
 1417 Pick up the cube from [0.05, -0.24, 0.02] and place it at [0.1, 0.0, 0.02] with a yaw of 0
 1418 radians.
 1419 Pick up the cube from [0.05, -0.16, 0.02] and place it at [0.1, 0.12, 0.02] with a yaw of 0
 1420 radians.
 1421 Pick up the cube from [0.05, -0.08, 0.02] and place it at [0.1, 0.16, 0.02] with a yaw of 0
 1422 radians.
 1423 Pick up the cube from [0.05, 0.0, 0.02] and place it at [0.1, -0.02, 0.06] with a yaw of 0
 1424 radians.
 1425 Pick up the cube from [0.05, 0.08, 0.02] and place it at [0.1, 0.02, 0.06] with a yaw of 0
 1426 radians.
 1427 Pick up the cube from [0.05, 0.16, 0.02] and place it at [0.1, 0.0, 0.1] with a yaw of 0
 1428 radians.
 1429 Pick up the cube from [0.05, 0.24, 0.02] and place it at [0.1, 0.04, 0.1] with a yaw of 0
 1430 radians.
 1431 Pick up the cube from [0.05, 0.3, 0.02] and place it at [0.1, 0.01, 0.14] with a yaw of 0
 1432 radians.
 1433
 1434
 1435
 1436
 1437
 1438
 1439
 1440
 1441
 1442
 1443
 1444
 1445
 1446
 1447
 1448
 1449
 1450
 1451
 1452
 1453
 1454
 1455
 1456
 1457

1458 D.5 MAXIMUM OVERHANG

1459

1460 **User**

1461

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

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

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

1482

1483 Task:

1484 N = 1

1485 K = 1

1486 Start location = [0.05, 0.0, 0.02]

1487 Target location = [0.1, 0.0, 0.02]

1488

1489 Final answer:

1490 1) Pick up the cube from [0.05, 0.0, 0.02].
 1491 2) Place the cube at [0.1, 0.0, 0.02] with a yaw of 0 radians.

1492

1493 Here is the task:

1494 N = 5

1495 K = 3

1496 Start location = [[0.05, -0.16, 0.02], [0.05, -0.08, 0.02], [0.05, 0.0, 0.02], [0.05, 0.08, 0.02],
 1497 [0.05, 0.16, 0.02]]

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

1499

1500

ChatGPT-5

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

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

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

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

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

1511

1512

1513

1514

Gemini 2.5 Pro

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

1515

1516

1517

1518

1519

1520

1521

1522

1523

1524

1525

1526

1527

1528

1529

1530

1531

1532

1533

1534

1535

1536

1537

1538

1539

1540

1541

1542

1543

1544

1545

1546

1547

1548

1549

1550

1551

1552

1553

1554

1555

1556

1557

1558

1559

1560

1561

1562

1563

1564

1565

1566 E TASKS AND CHALLENGES IN BUILDERBENCH
1567

1569 Task name	1570 Task visualization	1571 Abilities
1572 cube-1-task1		1. Pick and Place
1573 cube-1-task2		1. Pick and hold

1576 Table 2: Tasks with 1 cube.
1577
1578
1579
1580
1581

1585 Task name	1586 Task visualization	1587 Abilities
1588 cube-2-task1		1. Motor skills
1589 cube-2-task2		1. Sequential-logic 2. Motor skills
1590 cube-2-task3		1. Solving a packing problem 2. Sequential-logic 3. Motor skills
1591 cube-2-task4		1. Two cube lift maneuver 2. Sequential-logic 3. Motor skills
1592 cube-2-task5		1. Two cube lift maneuver 2. Sequential-logic 3. Motor skills

1597 Table 3: Tasks with 2 cubes.
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619

1620

1621

1622

1623

1624

1625

1626

1627

1628

1629

1630

1631

1632

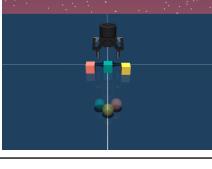
Task name	Task visualization	Abilities
cube-3-task1	 	1. Sequential-logic 2. Motor skills
cube-3-task2	 	1. Stable equilibria 2. Toppling 2. Motor skills
cube-3-task3	 	1. Three cube lift maneuver 2. Sequential-logic 2. Motor skills
cube-3-task4	 	1. Solving a packing problem 2. Sequential-logic 3. Motor skills
cube-3-task5	 	1. Building support structures 2. Sequential-logic 3. Motor skills

Table 4: **Tasks with 3 cubes.**

1662

1663

1664

1665

1666

1667

1668

1669

1670

1671

1672

1673

1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685

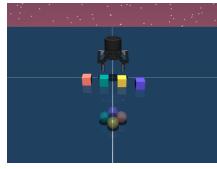
1686	Task name	Task visualization	Abilities	
1687	cube-4-task1			1. Sequential-logic 2. Motor skills
1688	cube-4-task2			1. Sequential-logic 2. Motor skills
1689	cube-4-task3			1. Solve a packing problem 2. Sequential-logic 2. Motor skills
1690	cube-4-task4			1. Unconstrained Maximum overhang 2. Sequential-logic 3. Motor skills
1691	cube-4-task5			1. Building support structures 2. Sequential-logic 3. Motor skills

Table 5: **Tasks with 4 cubes.**

1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727

1728

1729

1730

1731

1732

1733

1734

1735

1736

1737

1738

1739

1740

1741

1742

1743

1744

1745

1746

1747

1748

1749

1750

1751

1752

1753

1754

1755

1756

1757

1758

1759

1760

1761

1762

1763

1764

1765

1766

1767

1768

1769

1770

1771

1772

1773

1774

1775

1776

1777

1778

1779

1780

1781

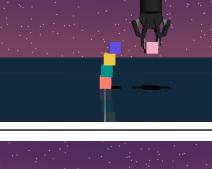
Task name	Task visualization	Abilities
cube-5-task1	 	1. Sequential-logic 2. Motor skills
cube-5-task2	 	1. Sequential-logic 2. Motor skills
cube-5-task3	 	1. Sequential-logic 2. Motor skills
cube-5-task4	 	1. Maximum overhang 2. Sequential-logic 3. Motor skills
cube-5-task5	 	1. Building support structures 2. Sequential-logic 3. Motor skills

Table 6: **Tasks with 5 cubes.**

1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793

Task name	Task visualization	Abilities
cube-6-task1	 	1. Sequential-logic 2. Motor skills
cube-6-task2	 	1. Sequential-logic 2. Motor skills
cube-6-task3	 	1. Two cube lift 2. Sequential-logic 2. Motor skills
cube-6-task4	 	1. Building support structures 2. Sequential-logic 3. Motor skills
cube-6-task5	 	1. Temporary bi-scaffolding 2. Sequential-logic 3. Motor skills

Table 7: **Tasks with 6 cubes.**

1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835

1836

1837

1838

1839

1840

1841

1842

1843

1844

1845

1846

1847

1848

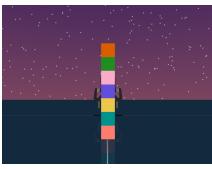
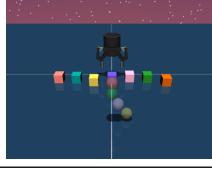
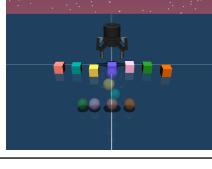
Task name	Task visualization	Abilities
cube-7-task1	 	<ul style="list-style-type: none"> 1. Sequential-logic 2. Motor skills
cube-7-task2	 	<ul style="list-style-type: none"> 1. Building a support structure 2. Sequential-logic 2. Motor skills
cube-7-task3	 	<ul style="list-style-type: none"> 1. Building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

Table 8: **Tasks with 7 cubes.**

1878

1879

1880

1881

1882

1883

1884

1885

1886

1887

1888

1889

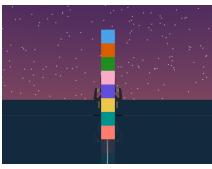
1890	Task name	Task visualization	Abilities	
1891	cube-8-task1			1. Sequential-logic 2. Motor skills
1892	cube-8-task2			1. Building a support structure 2. Sequential-logic 2. Motor skills
1893	cube-8-task3			1. Temporary bi-scaffolding 2. Sequential-logic 2. Motor skills
1894	cube-8-task4			1. Temporary bi-scaffolding 2. Sequential-logic 3. Motor skills
1895	cube-8-task5			1. Three cube lift 2. Sequential-logic 3. Motor skills

Table 9: **Tasks with 8 cubes.**

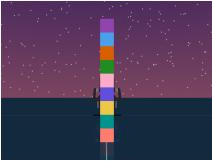
1944				
1945				
1946				
1947				
1948				
1949				
1950				
1951				
1952				
1953				
1954				
1955				
1956	Task name	Task visualization	Abilities	
1957				
1958				
1959	cube-9-task1			1. Sequential-logic 2. Motor skills
1960				
1961				
1962				
1963				
1964				
1965	cube-9-task2			1. Sequential-logic 2. Motor skills
1966				
1967				
1968				
1969				
1970				
1971	cube-9-task3			1. Temporary mono-scaffolding 2. Mono-scaffold reuse 3. Sequential-logic 4. Motor skills
1972				
1973				
1974				
1975				
1976	cube-9-task4			1. Temporary tri-scaffolding 2. Bi-scaffold reuse 3. Sequential-logic 4. Motor skills
1977				
1978				
1979				
1980				
1981				
1982	cube-9-task5			1. Building a support structure 2. Sequential-logic 3. Motor skills
1983				
1984				
1985				

Table 10: **Tasks with 9 cubes.**