BUILDERBENCH – A BENCHMARK FOR GENERALIST AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Today's AI models learn primarily through mimicry and sharpening, so it is not surprising that they struggle to solve problems beyond the limits set by existing data. To solve novel problems, agents should acquire skills for exploring and learning through experience. Finding a scalable learning mechanism for developing agents that learn through interaction remains a major open problem. In this work, we introduce BuilderBench, a benchmark to accelerate research into agent pre-training that centers open-ended exploration. BuilderBench requires agents to learn how to build any structure using blocks. BuilderBench is equipped with (1) a hardware accelerated simulator of a robotic agent interacting with various physical blocks, and (2) a task-suite with over 50 diverse target structures that are carefully curated to test an understanding of physics, mathematics, and long-horizon planning. During training, agents have to explore and learn general principles about the environment without any external supervision. During evaluation, agents have to build the handmade and unseen target structures from the task suite. Solving these tasks requires a sort of *embodied reasoning*, that is not reflected in words, but rather in actions, experimenting with different strategies and piecing them together. Our experiments show that many of these tasks challenge the current iteration of algorithms. Hence, we also provide a "training wheels" protocol, in which agents are trained and evaluated to build a single target structure from the task suite. Finally, we provide clean implementations of seven different algorithms as a reference point for researchers.

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

1 THE NEED FOR A NEW BENCHMARK

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

This is not for lack of trying. There is a long line of interaction and exploration benchmarks built by researchers in reinforcement learning (RL) (Ecoffet et al., 2021; Tang et al., 2017), control (Plappert et al., 2018), and developmental robotics (Oudeyer et al., 2007). For example, agents that learn to navigate mazes like ant-maze (Fu et al., 2021), solve montezuma's revenge (Bellemare et al., 2013), or learn to perform the 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 learned on even the most complex of these benchmarks (e.g., StarCraft (Vinyals et al.,

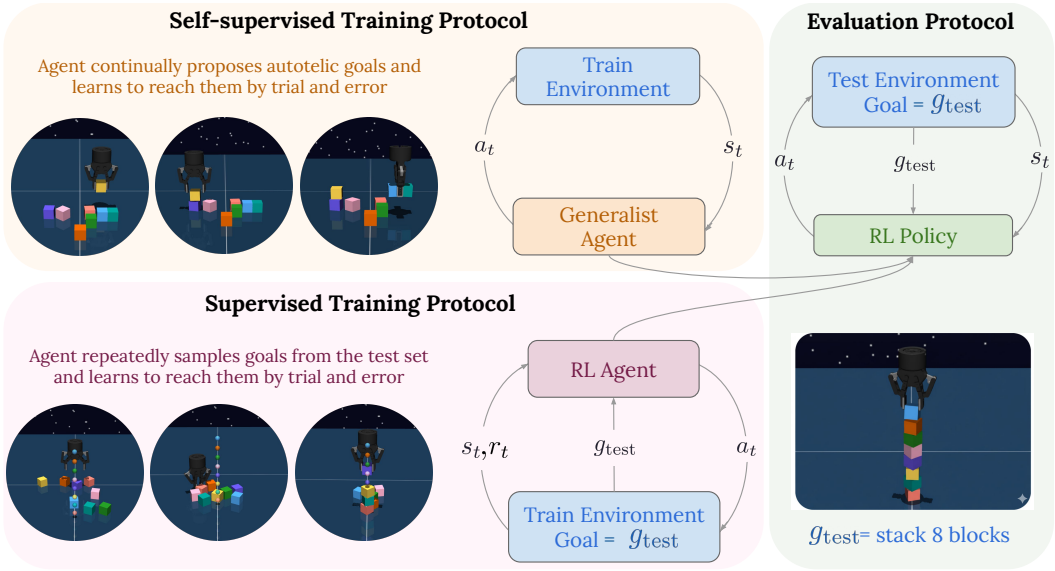


Figure 1: **The BuilderBench Benchmark.** *(Top Left)* Training consists of self-supervised exploration: agents can collect data to learn to reach various goals via trial and error. Agents are not given any information about the test-time goals or their distribution. *(Right)* During evaluation, the 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.

2019), AI2Thor (Kolve et al., 2022), NetHack (Küttler et al., 2020)) do not seem to learn the same sort of common sense and reasoning skills that agents trained on human text do acquire (Wei et al., 2022). We argue that the key reason why, is that, put simply, 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 to 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 one to build diverse open-ended 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 area[1] 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.

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

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 designing 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 (fig. 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 quite 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 towards developing increasingly *generalist agents*.[2] BuilderBench aims to use open-ended block building to develop and evaluate agents with 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), Procgen (Cobbe et al., 2020) or NetHack (Küttler et al., 2020)
- We open-source BuilderBench, a task-suite of over 50 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 (for e.g., training a PPO agent to stack two blocks takes 30 minutes on a single GPU), (1) making it easier for researchers to use and build from, and (2) reducing the barrier to entry for frontier RL research.

2 Related work.

AI benchmarks have driven progress in the field. Benchmarks such as MNIST (LeCun & Cortes, 2010), 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 generalist 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 (endlessly bouncing the ball in breakout from DQN (Mnih et al., 2013) or the famous move 37 from AlphaGo (Silver et al., 2016)), but only for solving a narrow 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 simply 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

[2]We will vaguely use the term generalist agents to mean RL policies that can solve diverse unseen tasks, which they haven't been explicitly trained to solve before.

of similar downstream tasks for evaluation. Hence, to properly evaluate generalization properties of agents, benchmarks need to have sufficiently complex and unseen test tasks.

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

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

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

3 BuilderBench - A benchmark for generalist agents

This paper proposes the BuilderBench benchmark, which comes with task-suite of over 50 tasks, where each task is a target block structure carefully curated for evaluating unique abilities. BuilderBench comes with a fast simulator consisting of an agent interacting with a varying number of blocks. In the following sections, we will describe the environment (section 4), the task-suite (section 5) and the training and evaluation protocols (section 6).

4 BuilderBench environment.

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

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

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

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

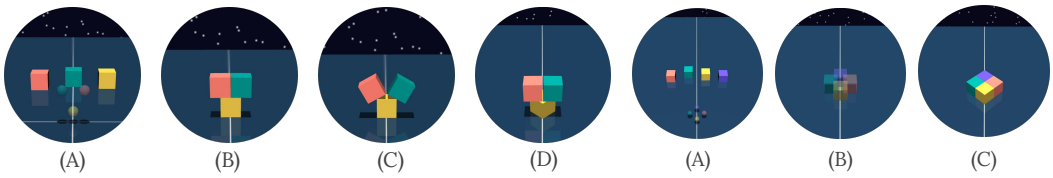
Task specification. Each task corresponds to a physically stable target structure built using cubes. To specify this structure, we provide a vector of target cube positions ($\mathbb{R}^{3k}$), where $k \leq n$ is the number of cubes in the target cube structure. This allows us to specify target structures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

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

5 BUILDERBENCH TASK SUITE

The BuilderBench task suite contains a total of 50 tasks. In this section, we describe these tasks in detail and the design philosophy behind the task-suite. The task-suite is meant to be open-ended and address the challenges highlighted in section 1. We start with a case-study of three different tasks from the BuilderBench task-suite, which is meant to showcase how each task requires the agent to unlock at least one distinct reasoning ability and compose various high-level skills sequentially. As described in section 4, during evaluation, agents only have access to the positions of the masked cubes in the target structure.

5.1 A CASE STUDY OF THE FOUR TASKS



(1) **T-block** (2) **Four cube packing**

Example 1: T block. This task requires building a simple T shaped structure with one cube at the base, and two cubes on top (fig. 21). The second frame (B) shows what many people envision as the solution to this task. However, this configuration isn't stable, as shown in the third frame. Solving this task requires the reasoning insight to rotate the bottom cube by about $45°$. Since the diagonal of the cube's top surface is longer than its edge length, the rotated base provides sufficient support for both top cubes, enabling a stable T-shaped structure (see fourth frame).

Example 2: Four cube packing. This task tests geometric reasoning and spatial packing. The target structure is an arrangement of four cube centers placed at some distance along the four cardinal directions on the floor (see (A) of fig. 22). The distance is chosen such that the placement is impossible with the default cube orientation: the cubes overlap (see (B)). This results in a packing problem of arranging the cubes such that its centers form the target structure. To solve this, the agent needs to rotate each cube by $45°$ before placing it, which ensures the centers align correctly without collision (see (C)). Due to the two fingered morphology of the robot, this task cannot be solved using pick and place primitives, but would require nudging the final block in place.

[3]We 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.

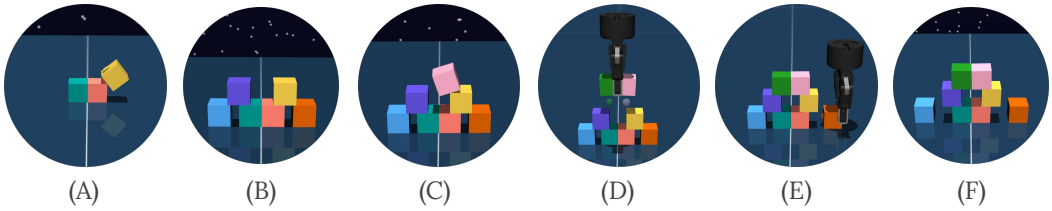


Figure 3: **Hexagonal Portal**

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

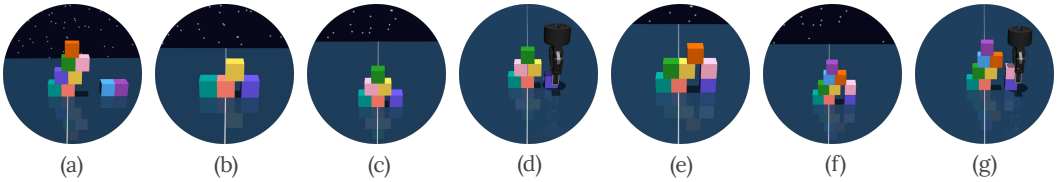


Figure 4: **Learning tower**

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

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

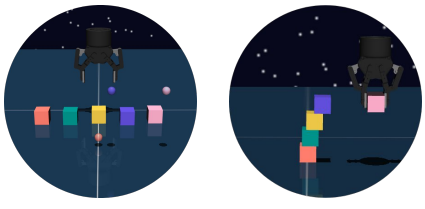


Figure 5: **Four cube packing**

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 open-ended and unseen tasks, we anticipate that agents would learn key knowledge priors via exploration and experimentation during training (Spelke & Kinzler, 2007), and learn mechanisms to efficiently use them for solving unseen tasks 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 building generalist agents (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.

The complete list of tasks, along with visualizations and the capabilities required to solve them, is provided in appendix C.

6 Training and Evaluation protocols.

In its current iteration, BuilderBench supports two training protocols, each serving a distinct purpose in research on generalist agents. Additional details are described 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 C. 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).

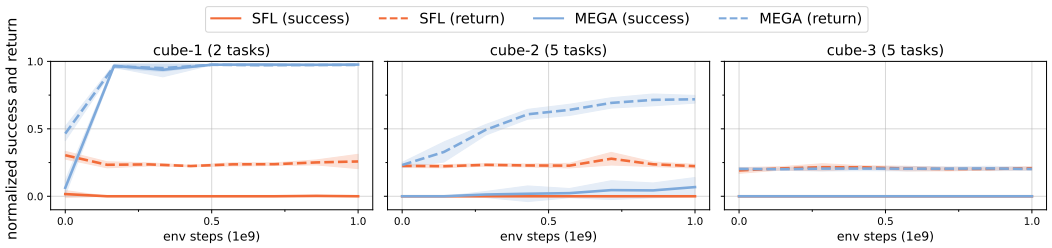


Figure 6: **Unsupervised evaluation on BuilderBench task-suite** We evaluate MEGA (Pitis et al., 2020) and SFL (Rutherford et al., 2024) on 12 simplest tasks from our task suite. The results show that directly using these algorithms out of the box only succeeds for the simplest tasks. Clearly, further research on old and new algorithms is required.

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. We also discuss various promising approaches to solve the benchmark and the scope of future research that can be facilitated by BuilderBench.

Multi-task self-supervised protocol. We implemented two algorithms, sampling for learnability (**SFL** (Rutherford et al., 2024)) and maximum entropy gain exploration (**MEGA** (Pitis et al., 2020)). Both algorithms 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). We train both algorithms in environments with one, two and three cubes separately. We test the learned policy on the respective tasks from the task-suite appendix C at various points during training and report its normalized episodic success and returns.

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

Single-task supervised protocol. For this protocol, we benchmark three representative RL algorithms, proximal policy optimization (**PPO**) (Schulman et al., 2017), soft actor critic (**SAC**) (Haarnoja et al., 2018), contrastive RL (**CRL**) (Eysenbach et al., 2022), and provide an implementation for random network distillation (**RND**) (Burda et al., 2019), and hindsight experience replay (**HER**) (Andrychowicz et al., 2017). The benchmarking results on 17 tasks are provided in fig. 7. All experiments use dense rewards.

7.1 Evaluating Large Language Models

It has been shown that scaling pretraining and inference-time compute can significantly enhance the reasoning abilities of language models (Kaplan et al., 2020). To test whether

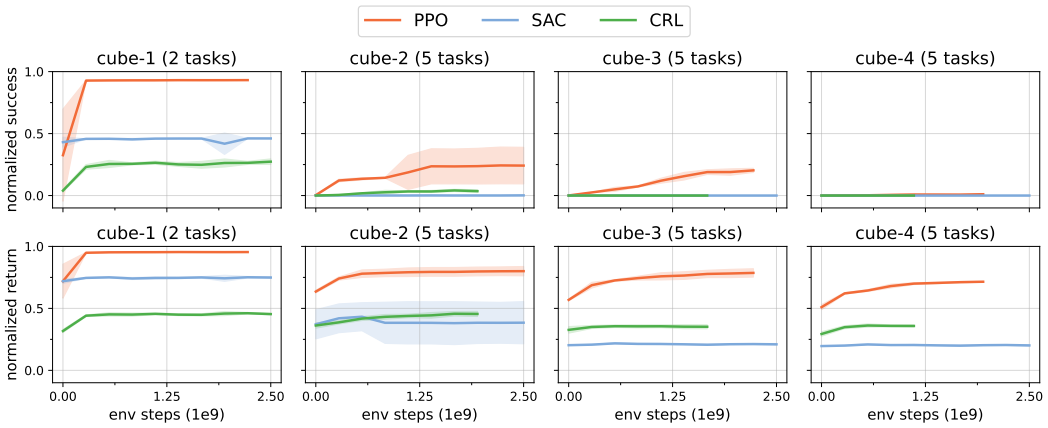


Figure 7: **Training on the test goals** While training on the test goals improves both the returns and success achieved by the best agents, as the number of cubes and the complexity of the tasks increase, current algorithms are not able to achieve a non zero success. We believe hypothesis driven exploration is key to solve the more complex tasks and most RL algorithms do not have an explicit mechanism for this. See appendix C for a description of all tasks.

the latest proprietary models can solve tasks from our task-suite, we evaluated ChatGPT-5 Thinking[4] and Gemini 2.5 Pro (Team, 2025) on all five tasks discussed in section 5.1. Each model was provided with a descriptive prompt about the environment and the task. The goal of the model was to provide a high-level open-loop plan in language, such that following this plan would stably build the target structure. A simple example task with a correct solution was also included in the prompt (see appendix B for the exact prompts and solutions). Figure 8 shows that both models, despite using inference-time compute, are not able to provide the correct high-level plan to solve any of the tasks. While this is not meant to be an extensive evaluation of current models' abilities, it highlights how solving our tasks requires non-obvious steps of reasoning that are beyond what current models can achieve through scaling alone.

Figure 8: **Evaluating language models on BuilderBench.**

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

8 Limitations and Conclusion

Currently, the BuilderBench task-suite is limited to tasks with blocks of the same shape and size. Adding different shapes is straightforward in MuJoCo (Todorov et al., 2012). In future versions, we aim to continually add different types of blocks and complex tasks. A key advantage of block-building is its scalability; one could come up with structures that are arbitrarily diverse and require large numbers of reasoning steps. Another limitation is that we have not provided implementations for all approaches for open-ended exploration which exist in the literature (see discussion in section 2). This is outside the scope of the paper. The main aim of the paper is to present an effective benchmark to accelerate research on scalable and generalizable learning through open-ended exploration.

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

[4]https://openai.com/index/introducing-gpt-5/

9 REPRODUCIBILITY STATEMENT

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

REFERENCES

Faseeh Ahmad, Esra Erdem, and Volkan Patoglu. A formal framework for robot construction problems: A hybrid planning approach, 2019. URL https://arxiv.org/abs/1903.00745.

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

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf.

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

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

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

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

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

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1lJJnR5Ym.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

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

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

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

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

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

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. 2013. URL http://arxiv.org/abs/1312.5602. cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Jurgen Schmidhuber. Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Diploma thesis, Technische Universitat Munchen, Germany, 14 May 1987. URL http://www.idsia.ch/~juergen/diploma.html.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

A Environment details

A.1 Episode lengths.

The episode length depends on the number of cubes present in the environment (N). For supervised tasks, the episode length is $100 + 50 * N$ and for self-supervised tasks, the episode length is $500 * N$.

A.2 Reward functions.

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

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

B Evaluating language models: Prompts and Solutions

B.1 T block

User

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

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

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

Task:
N = 1
K = 1
Start location = [0.05, 0.0, 0.02]

Target location = [0.1, 0.0, 0.02]

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

Here is the task:
N = 3
K = 3
Start location = [[0.05, -0.08, 0.02], [0.05, 0. , 0.02], [0.05, 0.08, 0.02]]
Target location = [[0.1, 0.02, 0.06], [0.1 , -0.02, 0.06], [0.1 , 0. , 0.02]]

ChatGPT-5 Thinking

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

Gemini 2.5 Pro

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

B.2 FOUR CUBE PACKING

User

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

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

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

Task:
N = 1
K = 1
Start location = [0.05, 0.0, 0.02]
Target location = [0.1, 0.0, 0.02]

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

Here is the task:
N = 4
K = 4
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]]
Target location = [[0.1, 0.02828427, 0.02], [0.1, -0.02828427, 0.02], [0.12828427, 0.0, 0.02], [0.07171573, 0.0, 0.02]]

ChatGPT-5 Thinking

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, 0.12, 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 [0.07171573, 0.0, 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.

Gemini 2.5 Pro

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

B.3 HEXAGONAL PORTAL

User

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

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

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

Task:
N = 1
K = 1
Start location = [0.05, 0.0, 0.02]
Target location = [0.1, 0.0, 0.02]

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

Here is the task:
N = 8
K = 8
Start location = [[0.05, -0.24, 0.02], [0.05, -0.18, 0.02], [0.05, -0.12, 0.02], [0.05, -0.04, 0.02], [0.05, 0.04, 0.02], [0.05, 0.12, 0.02], [0.05, 0.18, 0.02], [0.05, 0.24, 0.02]]
Target location = [[0.1, 0.02, 0.02], [0.1, -0.02, 0.02], [0.1, 0.04, 0.06], [0.1, -0.04, 0.06], [0.1, 0.02, 0.1], [0.1, -0.02, 0.1], [0.1, 0.1, 0.02], [0.1, -0.1, 0.02]]

ChatGPT-5 Thinking

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

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

Gemini 2.5 Pro

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

B.4 LEARNING TOWER

User

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

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

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

Task:
N = 1
K = 1
Start location = [0.05, 0.0, 0.02]
Target location = [0.1, 0.0, 0.02]

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

Here is the task:
N = 9
K = 9
Start location = [[0.05, -0.3, 0.02], [0.05, -0.24, 0.02], [0.05, -0.16, 0.02], [0.05, -0.08, 0.02], [0.05, 0.0, 0.02], [0.05, 0.08, 0.02], [0.05, 0.16, 0.02], [0.05, 0.24, 0.02], [0.05, 0.3, 0.02]]
Target location = [[0.1, 0.0, 0.02], [0.1, -0.04, 0.02], [0.1, 0.02, 0.06], [0.1, -0.02, 0.06], [0.1, 0.04, 0.1], [0.1, 0.0, 0.1], [0.1, 0.01, 0.14], [0.1, 0.12, 0.02], [0.1, 0.16, 0.02]]

ChatGPT-5 Thinking

Pick up the cube from [0.05, -0.3, 0.02]. Place the cube at [0.1, -0.04, 0.02] with a yaw of 0 radians.
Pick up the cube from [0.05, -0.24, 0.02]. Place the cube at [0.1, 0.0, 0.02] with a yaw of 0 radians.
Pick up the cube from [0.05, 0.24, 0.02]. Place the cube at [0.1, 0.12, 0.02] with a yaw of 0 radians.
Pick up the cube from [0.05, 0.3, 0.02]. Place the cube at [0.1, 0.16, 0.02] with a yaw of 0 radians.
Pick up the cube from [0.05, -0.16, 0.02]. Place the cube at [0.1, -0.02, 0.06] with a yaw of

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

Gemini 2.5 Pro

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

B.5 MAXIMUM OVERHANG

User

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

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

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

Task:
N = 1
K = 1
Start location = [0.05, 0.0, 0.02]
Target location = [0.1, 0.0, 0.02]

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

Here is the task:
N = 5
K = 3
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], [0.05, 0.16, 0.02]]
Target location = [[0.1, 0.0, 0.02], [0.1, 0.031, 0.14], [0.1, 0.16, 0.14]]

ChatGPT-5 Thinking

Pick up the cube from [0.05, 0.0, 0.02]. Place the cube at [0.1, 0.0, 0.02] with a yaw of 0 radians.
Pick up the cube from [0.05, -0.08, 0.02]. Place the cube at [0.1, 0.031, 0.02] with a yaw of 0 radians. (bottom of the pillar)
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 0 radians. (middle of the pillar)
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 0 radians. (top of the pillar)
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 0 radians. (target at z=0.14 realized)

Gemini 2.5 Pro

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

C TASKS AND CHALLENGES IN BUILDERBENCH

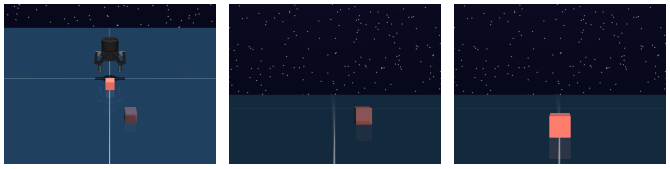
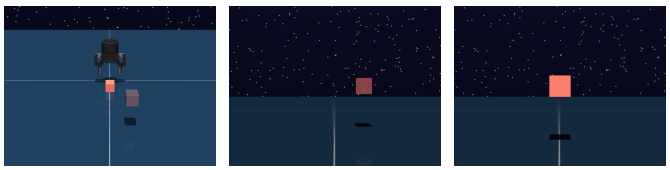
Task name	Task visualization	Abilities
cube-1-task1		1. Pick and Place
cube-1-task2		1. Pick and hold

Table 1: **Tasks with 1 cube**.

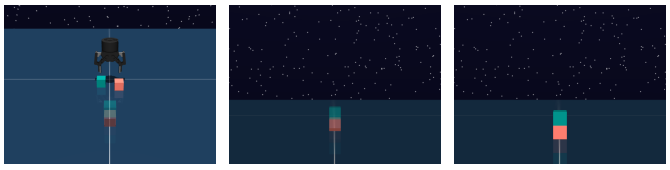
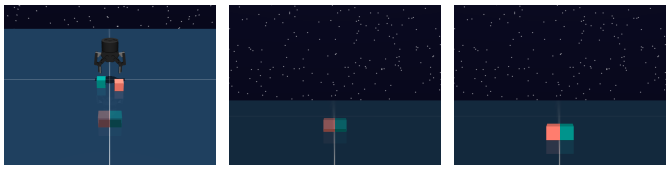
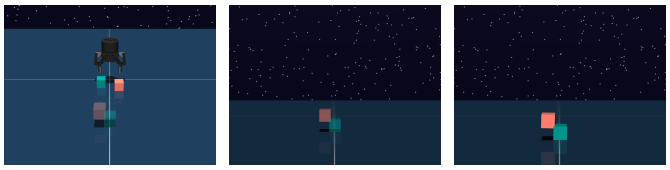
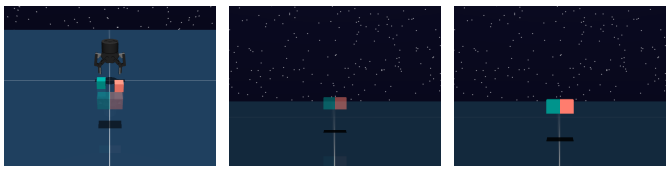
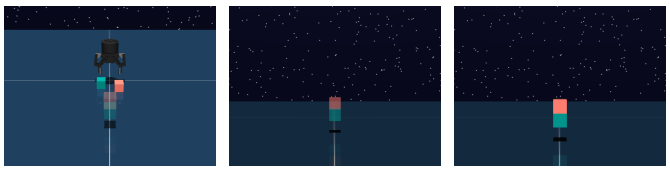
Task name	Task visualization	Abilities
cube-2-task1		1. Motor skills
cube-2-task2		1. Sequential-logic 2. Motor skills
cube-2-task3		1. Solving a packing problem 2. Sequential-logic 2. Motor skills
cube-2-task4		1. Two cube lift maneuver 2. Sequential-logic 3. Motor skills
cube-2-task5		1. Two cube lift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

Table 2: **Tasks with 2 cube**.

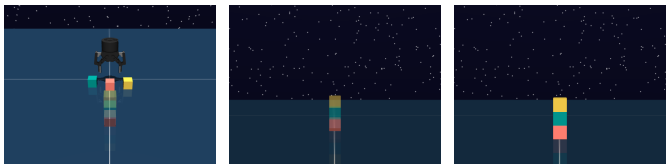
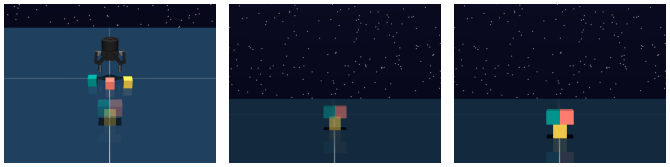
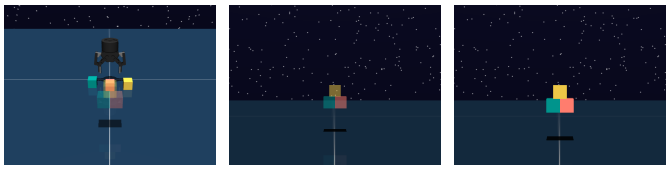
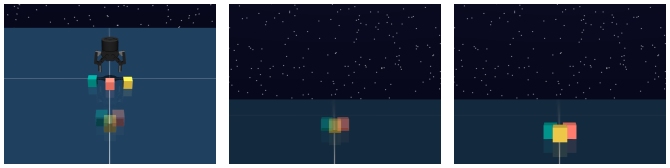
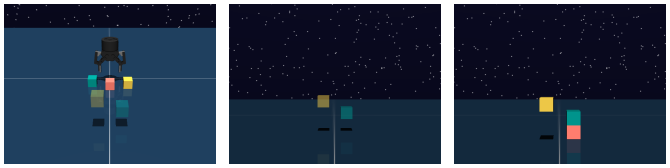
Task name	Task visualization	Abilities
cube-3-task1		1. Sequential-logic 2. Motor skills
cube-3-task2		1. Stable equilibria 2. Toppling 2. Motor skills
cube-3-task3		1. Three cube lift maneuver 2. Sequential-logic 2. Motor skills
cube-3-task4		1. Solving a packing problem 2. Sequential-logic 3. Motor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

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

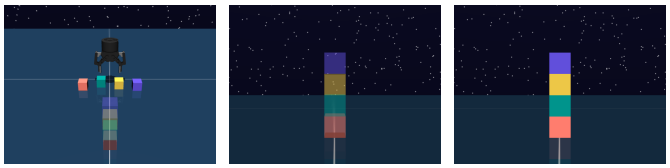
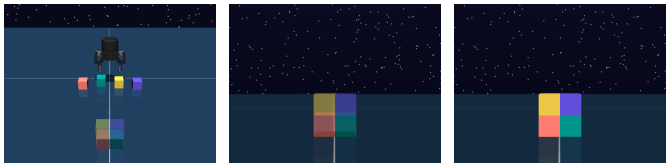
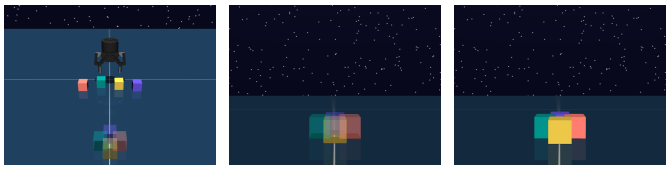
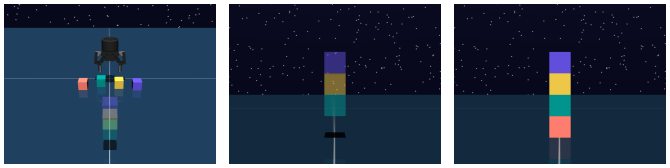
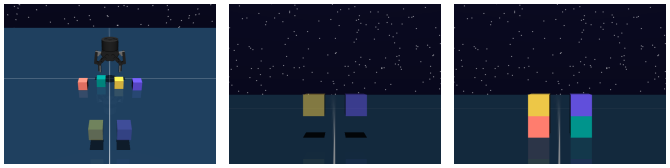
Task name	Task visualization	Abilities
cube-4-task1		1. Sequential-logic 2. Motor skills
cube-4-task2		1. Sequential-logic 2. Motor skills
cube-4-task3		1. Solve a packing problem 2. Sequential-logic 2. Motor skills
cube-4-task4		1. Building support structures 2. Sequential-logic 3. Motor skills
cube-4-task5		1. Building support structures 2. Sequential-logic 3. Motor skills

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

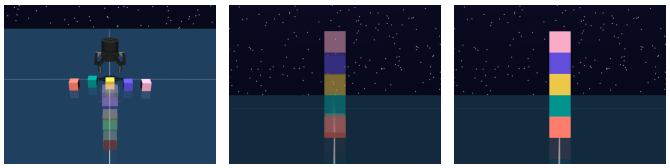
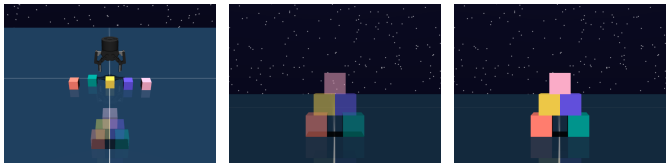
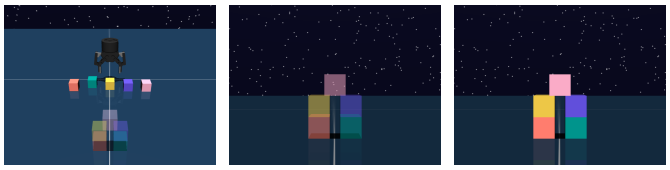
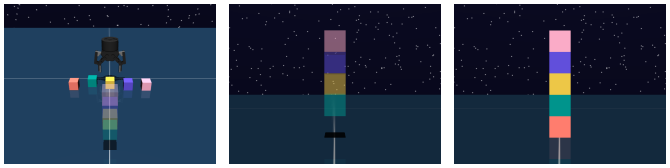
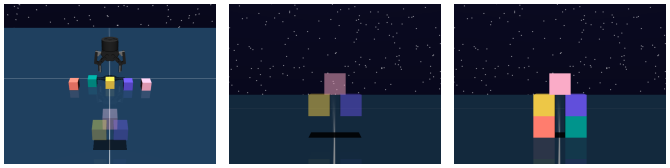
Task name	Task visualization	Abilities
cube-5-task1		1. Sequential-logic 2. Motor skills
cube-5-task2		1. Sequential-logic 2. Motor skills
cube-5-task3		1. Sequential-logic 2. Motor skills
cube-5-task4		1. Building support structures 2. Sequential-logic 3. Motor skills
cube-5-task5		1. Building support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

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

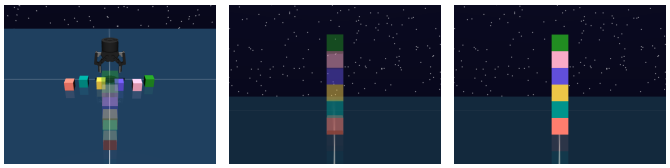
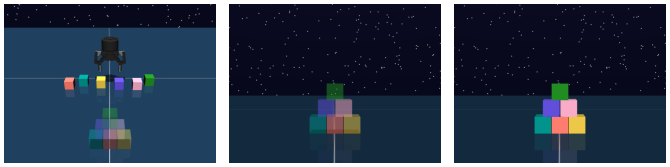
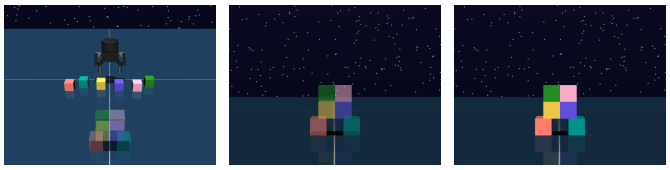
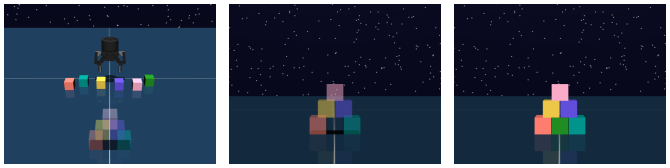
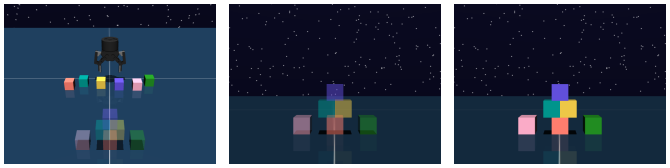
Task name	Task visualization	Abilities
cube-6-task1		1. Sequential-logic 2. Motor skills
cube-6-task2		1. Sequential-logic 2. Motor skills
cube-6-task3		1. Two cube lift 2. Sequential-logic 2. Motor skills
cube-6-task4		1. Building support structures 2. Sequential-logic 3. Motor skills
cube-6-task5	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

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

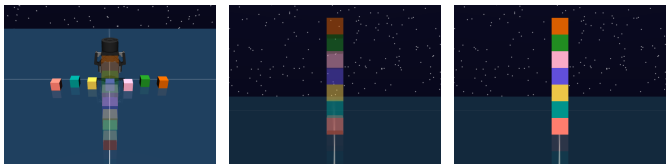
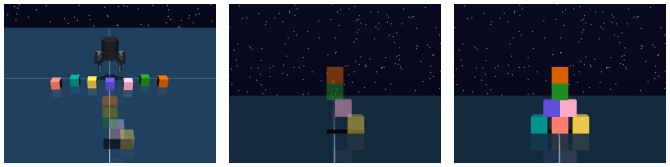
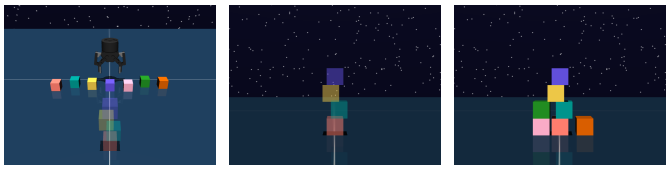
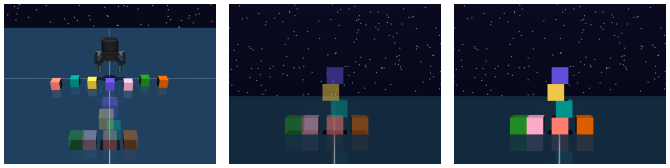
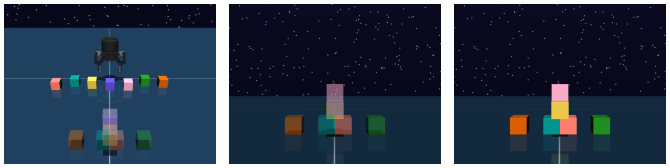
Task name	Task visualization	Abilities
cube-7-task1		1. Sequential-logic 2. Motor skills
cube-7-task2		1. Building a support structure 2. Sequential-logic 2. Motor skills
cube-7-task3		1. Building a support structure 2. Sequential-logic 2. Motor skills
cube-7-task4		1. Temporary bi-scaffolding 2. Sequential-logic 3. Motor skills
cube-7-task5		1. Temporary bi-scaffolding 2. Sequential-logic 3. Motor skills

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

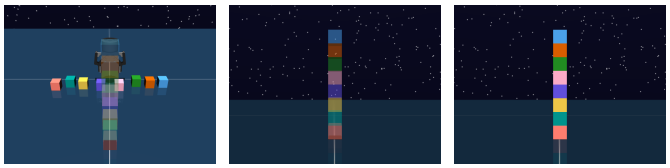
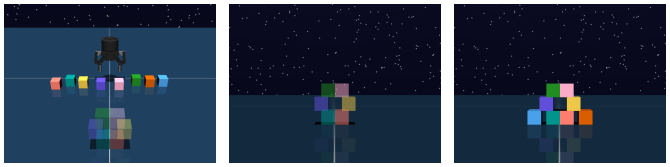
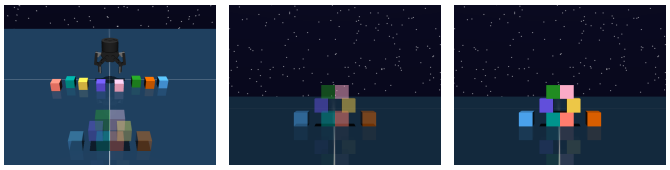
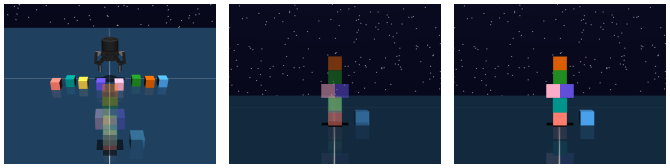
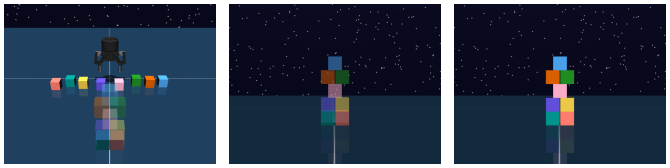
Task name	Task visualization	Abilities
cube-8-task1		1. Sequential-logic 2. Motor skills
cube-8-task2		1. Building a support structure 2. Sequential-logic 2. Motor skills
cube-8-task3		1. Temporary bi-scaffolding 2. Sequential-logic 2. Motor skills
cube-8-task4		1. Temporary bi-scaffolding 2. Sequential-logic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

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

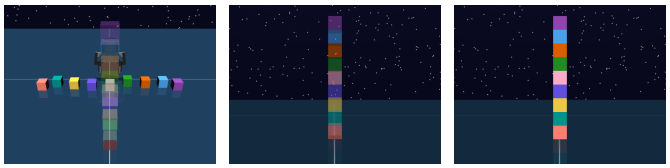
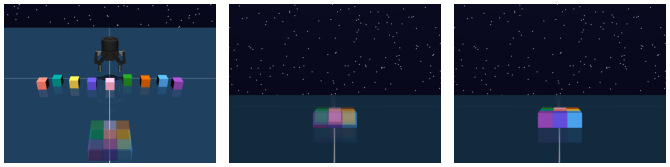
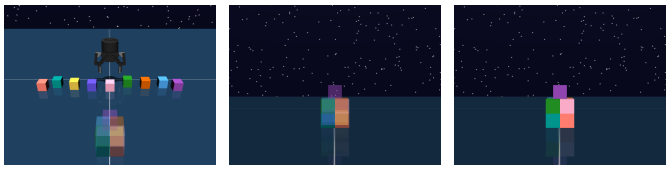
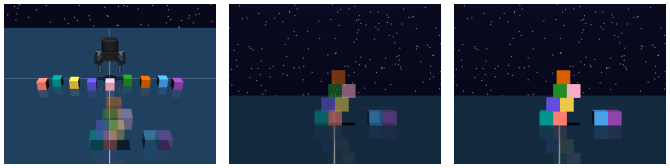
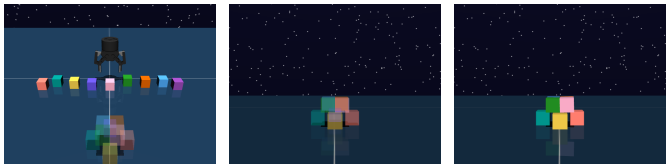
Task name	Task visualization	Abilities
cube-9-task1		1. Sequential-logic 2. Motor skills
cube-9-task2		1. Sequential-logic 2. Motor skills
cube-9-task3		1. Temporary mono-scaffolding 2. Mono-scaffold reuse 3. Sequential-logic 4. Motor skills
cube-9-task4		1. Temporary tri-scaffolding 2. Bi-scaffold reuse 3. Sequential-logic 4. Motor skills
cube-9-task5		1. Building a support structure 2. Sequential-logic 3. Motor skills

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

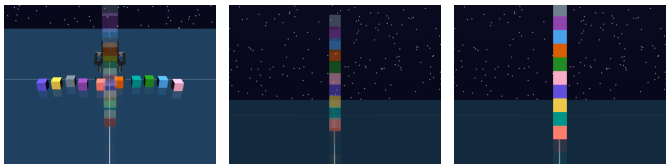
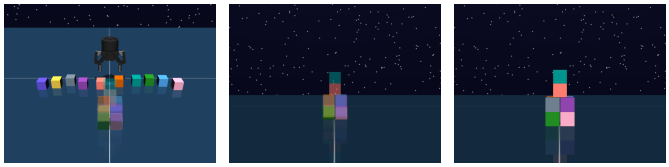
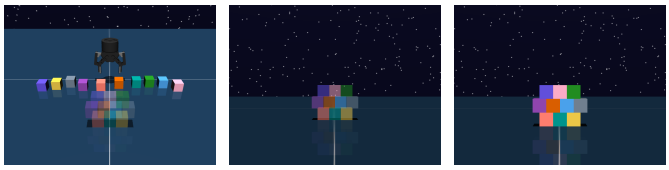
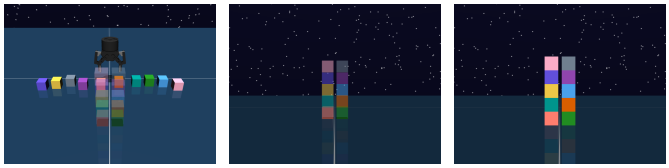
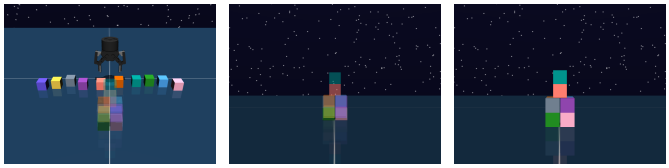
Task name	Task visualization	Abilities
cube-10-task1		1. Sequential-logic 2. Motor skills
cube-10-task2		1. Sequential-logic 2. Motor skills 3. Self-body awareness 3. Temporary scaffolding.
cube-10-task3		1. Temporary mono-scaffolding 2. Mono-scaffold reuse 3. Sequential-logic 4. Motor skills
cube-10-task4		1. Self-body awareness 3. Sequential-logic 4. Motor skills
cube-10-task5		1. Building a support structure 2. Sequential-logic 3. Motor skills

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