OCBENCH: A CONTROLLABLE BENCHMARK FOR STUDYING BEHAVIORAL CLONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Behavioral cloning (BC), despite its simplicity, exhibits many counterintuitive phenomena in the real world. For example, the performance of BC often keeps increasing as the model overfits more to the dataset, and fully closed-loop policies often completely fail without action chunking. Unfortunately, properly studying these empirical phenomena is challenging: in the real world, datasets and experiments are costly and not fully controllable; in simulation with synthetic data, these phenomena are often not easily observed partly due to the discrepancy between scripted policies and human demonstrations. In this work, we propose OCBench, a robotic manipulation benchmark with controllable scripted policies that have similar properties to human demonstrations. We show that, by mimicking key properties of human demonstrations, OCBench reproduces many anecdotal BC-related phenomena in controlled settings. With its GPU-accelerated environments and scripted policies, we demonstrate how OCBench enables scientific studies of previously reported BC-related phenomena by analyzing and refuting various hypotheses.

1 INTRODUCTION

Behavioral cloning (BC) is a foundational technique for large-scale robotic control (Brohan et al., 2023; Physical Intelligence et al., 2026). Despite its simplicity, behavioral cloning exhibits many counterintuitive phenomena, especially when trained on complex, large-scale, human-collected data. For example, researchers have anecdotally found that BC performance often increases as it overfits more (Mandlekar et al., 2021; Allshire et al., 2026), fully closed-loop policies often fail completely whereas partially open-loop control with action chunking works well (Zhao et al., 2023; Lazzati et al., 2026; Zeng et al., 2026), and history conditioning can hurt performance despite providing strictly more information (de Haan et al., 2019; Wen et al., 2020; Sobol Mark et al., 2026).

Unfortunately, properly studying these empirical phenomena is challenging. In the real world, robotic experiments are not perfectly reproducible, human-collected data are expensive and less controllable, and different factors cannot be easily isolated. Simulated benchmarks with scripted policies provide more control over data, but they often do not reproduce all these real-world phenomena (see Section 2), likely because previous scripted policies do not have the same properties as actual human demonstrations.

The main goal of this work is to provide a suite of simulated tasks and datasets, which we call **OCBench**, that has similar *structural properties* to human demonstrations, to enable scientific, controlled studies of realistic behavioral cloning and robotic control. We show that by mimicking the key properties of human demonstrations (*e.g.*, trajectories being very narrow, non-Markovian, and smoothly random), we can reproduce many anecdotal BC-related phenomena in controlled settings, including those that are not clearly observed in previous synthetic benchmarks.

We implement the OCBench tasks and scripted policies in MJWarp, a GPU-accelerated MuJoCo based on NVIDIA Warp (Macklin, 2022). On a single H100 GPU, OCBench can collect synthetic trajectories highly efficiently, at up to 200K environment steps per second. This makes *streaming-data* training practically feasible, meaning that we can train an agent using a continuous stream of freshly generated data. This allows us to study the "limit" of different algorithms and design choices under effectively unlimited data.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In this paper, based on our controlled reproduction in OCBench, we discuss four non-intuitive phenomena that happen with real-world behavioral cloning. We provide (and refute) possible hypotheses and discuss implications of each phenomenon, demonstrating how OCBench's controlled datasets and streaming-data training enable scientific studies of anecdotal phenomena reported in the literature.

Contributions. The main contributions of this work are twofold. First, we introduce OCBench, a new GPU-accelerated benchmark suite accompanied by scripted policies that mimic the structural properties of human demonstrations. This enables *controlled* studies of diverse phenomena observed in real-world behavioral cloning. Second, we reproduce four previously reported, non-intuitive phenomena in OCBench, making them scientifically studyable. We also provide analyses and hypotheses for these phenomena with concrete evidence in controlled settings.

2 Motivation

Behavioral cloning, despite its simplicity, exhibits several counterintuitive phenomena, especially when using real-world data in robotics and autonomous driving. Below are some of these phenomena that have been anecdotally reported in the literature.

Phenomenon 1: Overfitting is not necessarily bad. The performance of behavioral cloning often increases with more overfitting (Mandlekar et al., 2021; Allshire et al., 2026), and a smaller dataset can sometimes lead to better performance than a larger one (Codevilla et al., 2019; Bai et al., 2025).

Phenomenon 2: Open-loop[1] is often better than closed-loop. Open-loop policies with action chunking can substantially outperform closed-loop ones even when there is no inference delay (Zhao et al., 2023; Lazzati et al., 2026; Zeng et al., 2026). Adding history conditioning to closed-loop policies does not necessarily fix this issue (de Haan et al., 2019; Wen et al., 2020; Sobol Mark et al., 2026).

Phenomenon 3: Policies should be very large. Behavioral cloning on human-collected data often requires a much larger network than behavioral cloning on synthetic data, even on state-based tasks (Pan et al., 2026).

Phenomenon 4: Features matter even with enough data. Behavioral cloning performance often depends heavily on the input features (de Haan et al., 2019; Mandlekar et al., 2021; Lu et al., 2026). Scaling data does not necessarily reduce this dependence.

The phenomena listed above have mostly been observed with actual human demonstration data. Unfortunately, scientifically studying them is challenging in practice. In the real world, we typically have limited control over such human-generated data. Existing simulated BC and RL benchmarks based on scripted or RL-trained policies provide more control over data collection, but they do not consistently reproduce all these phenomena. For example, in many popular benchmarks based on systematic data generation (*e.g.*, OGBench manipulation, Robomimic MG, etc.), the gap between closed-loop and open-loop policy is often not as significant as in real data, and they do not require as large policies (see Section C).

We hypothesize that this discrepancy mainly stems from the difference in the *structural properties* of scripted (or RL-trained) policies and human demonstrations. For example, previous scripted policies are often not as temporally correlated, not as multi-modal, and mostly rely on Gaussian noise to diversify trajectories (Fu et al., 2020; Park et al., 2025a). Moreover, their action statistics are typically biased toward boundary actions (-1 or 1) or excessively uniform, unlike human demonstrations, as scripted or RL-trained policies often saturate the action range or are often randomized via Gaussian noise (Figure 1). OCBench aims to bridge the gap between scripted policies and human demonstrations.

3 OCBench: A controllable robotic manipulation benchmark

The main goal of OCBench is to provide a diverse set of tasks and datasets generated by controllable scripted policies that have similar *structural properties* to human demonstrations, so that researchers can scientifically study diverse BC- (and potentially RL-) related phenomena in this benchmark.

[1]Throughout this work, we use the terminology "open-loop" to denote *partially* open-loop control with action chunking (Zhao et al., 2023).

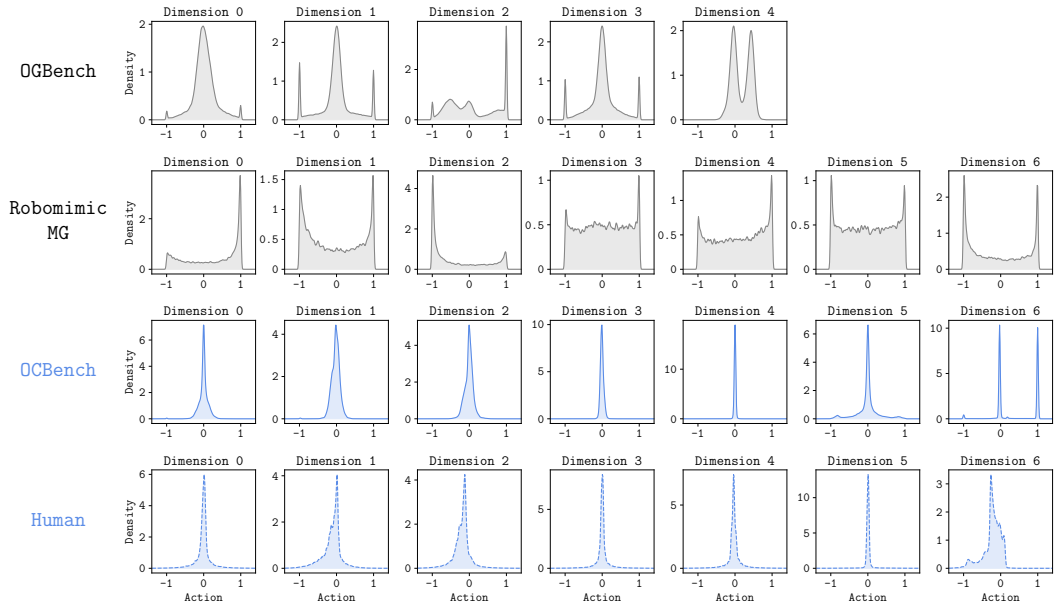


Figure 1: OCBench's scripted policy has more human-like action statistics than previous scripted (or RL-trained) policies, which tend to saturate the action range (*i.e.*, -1 or $+1$) or have overly uniform distributions.

Moreover, we aim to implement *both* the tasks and data-generating scripts of OCBench in GPU-accelerated MuJoCo (MJWarp) to enable highly efficient data generation and simulation.

In this section, we first describe our approach to mimic the properties of human demonstrations (Section 3.1) and the tasks provided in OCBench (Section 3.2).

3.1 Scripted policies and datasets

One of the core contributions of OCBench is its scripted policies designed to mimic the properties of human demonstrations. We hypothesize that human demonstrations have the following key properties: they are (1) very narrowly distributed (as opposed to Gaussian-noise-augmented trajectories), (2) highly non-Markovian and temporally correlated, and (3) semantically diverse (*e.g.*, grabbing objects with diverse poses and strategies).

To mimic these core properties of human demonstrations, we rely on randomized piecewise C^1-continuous Hermite splines as our data-generation backbone. Given a task, we first sample random control points for semantic phases (*e.g.*, grasp, lift, transport, etc.), and smoothly interpolate them with Hermite splines with Catmull-Rom tangent matching to ensure C^1 continuity. Importantly, we randomize a *wide* range of factors, including waypoints, grasping angles and directions, contact points, motion speeds, gripper rolls and pitches, trajectory curvatures, and high-level task-solving strategies, to enhance diversity as much as possible. We also let the scripted policy occasionally make mistakes and recover from them.

As we show in Section 4, we find that scripted policies designed in this way allow us to reproduce many counterintuitive phenomena happening with human-data behavioral cloning in OCBench, unlike previous scripted benchmarks (Section C). Figure 1 also shows that the OCBench data has action statistics more similar to actual human demonstrations than previous scripted or RL-trained policies.

3.2 Tasks

OCBench provides 28 tasks across five types of environments (`block`, `chamber`, `switch`, `hanoi`, and `bowling`) with diverse manipulation challenges. We provide an illustration in Figure 2 and their detailed descriptions in Section B.

All tasks and environments are implemented in MuJoCo (Todorov et al., 2012). We provide both CPU and GPU-accelerated (MJWarp) versions of the tasks and their accompanying scripted policies.

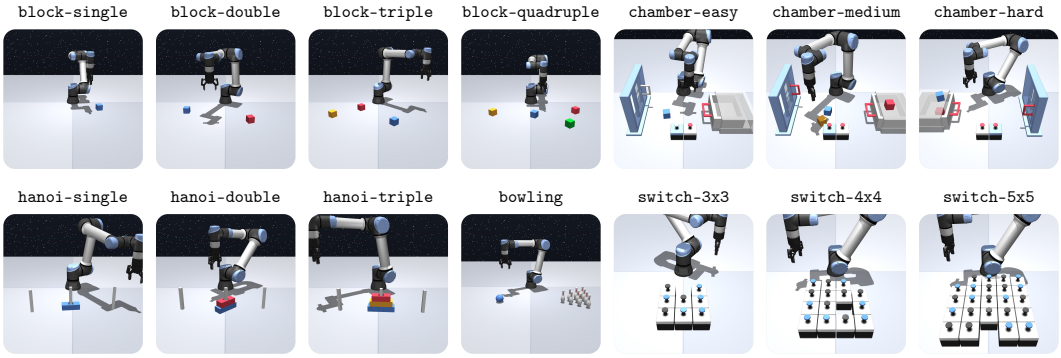


Figure 2: OCBench provides 28 tasks across five types of environments.

The CPU and MJWarp versions have *logically* identical dynamics and data generation strategies, although they are not bit-identical due to differences in numerical solvers and implementation details.

OCBench environments operate at 50Hz and have a 7-dimensional joint delta (relative) action space. We mainly consider state-based environments in this work, but we provide multi-view pixel-based observations as well. For convenience, we also provide `lite` versions of the environments (except `bowling`), which operate at 10Hz and have demonstrations that run twice as fast, to enable lightweight experimental iterations with smaller networks.

4 Studying Behavioral Cloning in OCBench

Now, we study the four counterintuitive phenomena mentioned in Section 2, based on our reproductions in OCBench. Here, our goal is **not** to completely demystify these phenomena, which we leave for future work on OCBench. Rather, we aim to demonstrate examples of how OCBench's controlled, GPU-accelerated scripts enable scientific studies of them.

Behavioral cloning methodology. We mainly consider flow behavioral cloning (Black et al., 2025b) with action chunking (Zhao et al., 2023) in this work, as it is a standard objective in current large-scale robotic systems (NVIDIA et al., 2025; Physical Intelligence et al., 2026). Given a dataset $\mathcal{D} = \{\tau^{(i)}\}_i$ consisting of state-action trajectories $\tau = (s_0, a_0, s_1, \ldots, s_T)$, flow behavioral cloning trains an action-chunking policy $\pi(a_{t:t+k-1} \mid s_t) : \mathcal{S} \to \Delta(\mathcal{A}^k)$ with flow matching (Lipman et al., 2023; Albergo & Vanden-Eijnden, 2023; Liu et al., 2023b), where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\Delta(\mathcal{X})$ denotes the set of probability distributions over a space $\mathcal{X}$, and $a_{t:t+k-1} = (a_t, a_{t+1}, \ldots, a_{t+k-1})$ denotes the length-k action chunk. At test time, we by default execute a full action chunk $a_{t:t+k-1}$ sampled from the policy in an open-loop manner, without re-planning. For our experiments, we average over two random seeds and denote 95% confidence intervals in the plots, unless mentioned otherwise.

4.1 Phenomenon 1: Overfitting is not necessarily bad

The first counterintuitive phenomenon we study is that overfitting is not always harmful in behavioral cloning. In fact, it turns out that overfitting is often *beneficial*. This phenomenon has been anecdotally reported in previous works in real-world robotics and next-token prediction (Ouyang et al., 2022; Allshire et al., 2026).

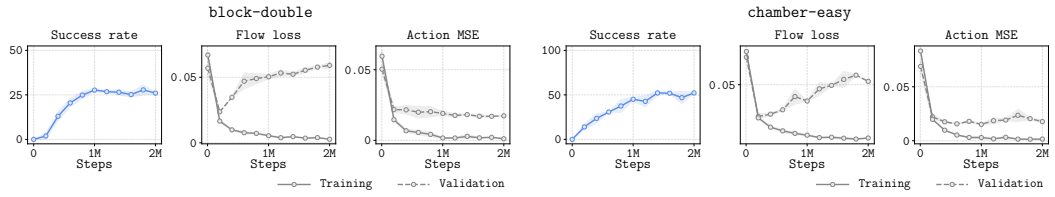


Figure 3: Performance can increase as the agent overfits more.

Figure 3 shows the performance and loss curves of flow BC agents trained on `block-double-task1` with 30K episodes and `chamber-easy-task2` with 10K episodes. The result shows that the performance generally does not drop over time, even though the validation flow loss keeps growing. In particular, the performance when the agent starts to overfit (*i.e.*, the step with the minimum validation loss) is substantially lower than the final performance, on both tasks.

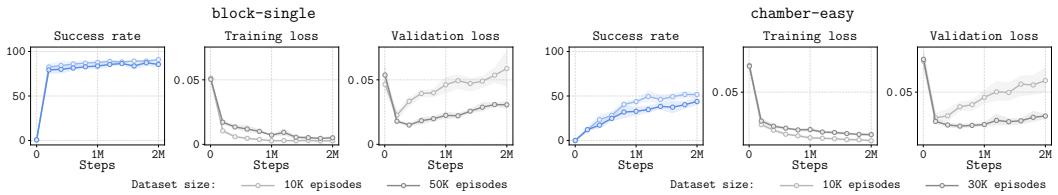


Figure 4: Using a larger dataset can *hurt* performance, even when sampled from the same data distribution.

Figure 4 shows an even more counterintuitive phenomenon. Here, we compare the performance of flow BC trained on two datasets that differ *only* in sizes (10K and 50K for `block-single-task1` and 10K and 30K for `chamber-easy-task2`). The results show that **using a larger dataset can hurt performance** by a small yet statistically significant margin, even when the datasets are sampled from the *same* distribution. In this plot, the curves are averaged over 4 independently sampled datasets for each setting. This indicates that overfitting might potentially be *desirable*, at least in some cases.

This size-reversal phenomenon was anecdotally reported in BC in autonomous driving (Codevilla et al., 2019) and robotics (Bai et al., 2025). Although the authors of these works (Codevilla et al., 2019; Bai et al., 2025) suspect that it may be due to non-identical data distributions and biases, our result shows that this is a *reproducible* phenomenon that happens even in a fully controlled setting with independent and identically distributed samples. This is an example of how OCBench enables a scientific study of BC-related phenomena.

While we do not have a definitive answer to this overfitting phenomenon, especially to the size-reversal one (which we leave for future studies on OCBench), we discuss three potential hypotheses and some refutations below.

Hypothesis 1: Flow loss alone does not capture performance. One hypothesis is that flow matching's velocity loss is not necessarily correlated with performance. For example, Allshire et al. (2026) recently reported that action mean squared error (MSE) (*i.e.*, the squared distance between dataset actions and generated actions) correlates more strongly with performance than flow loss.

The third and sixth plots in Figure 3 show the MSE metric. While this metric indeed correlates more strongly with performance than flow loss, it still does not perfectly explain performance (*e.g.*, the performance keeps increasing beyond 200K steps while MSE flatlines). Moreover, it does not fully explain the size-reversal phenomenon (Figure 17), so we believe this hypothesis alone cannot fully account for our observations.

Hypothesis 2: It is a network capacity issue. Another hypothesis is that the size-reversal issue is due to a network capacity issue (*i.e.*, the model size is the confounder). To examine this possibility, we train 4× larger, 540M-sized models in the same setting (2 seeds each). Figure 5 shows the results. Even with half-billion-sized models, the size-reversal phenomenon does not vanish. This suggests that limited model capacity alone is insufficient to explain this phenomenon.

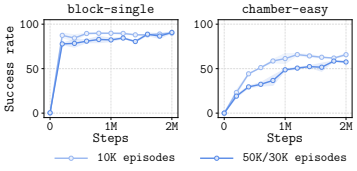


Figure 5: Figure 4 with larger models.

Hypothesis 3: Overfitting prevents test-time distribution shift. The third hypothesis is that it is mainly related to the test-time distribution shift issue ("DAgger issue" (Ross et al., 2011)) in behavioral cloning. By overfitting to the dataset, the agent can more closely follow the "nearest-neighbor" segment in the dataset. Although the task is randomized in each episode, following the nearest-neighbor motion often provides reasonable behavior (Pari et al., 2022), which could save the agent from compounding errors and distribution shift. As long as this neural nearest-neighbor retrieval is based on *learned*, generalizable representations, the agent could still perform well at test

time despite overfitting. While we leave quantitative investigations about this point for future work, Figure 13 in Section 4.4 indirectly supports this hypothesis.

4.2 Phenomenon 2: Open-loop is often better than closed-loop

The second phenomenon is that partially open-loop control (*i.e.*, action chunking) is often *crucial* in behavioral cloning, even when there is no inference delay. In the literature, Zhao et al. (2023) show that a closed-loop policy completely fails while action chunking can achieve reasonable performance. More recent works (Zhang et al., 2026; Lazzati et al., 2026; Zeng et al., 2026) report similar phenomena and discuss them in more diverse settings. We reproduce this on two OCBench tasks, `block-single-task1` and `hanoi-single-task1`, with streaming data.

Figure 6 compares the performance of closed-loop policies $\pi(a_t \mid s_t)$ and open-loop (action-chunking) policies $\pi(a_{t:t+24} \mid s_t)$. The results suggest that open-loop control is indeed essential on these tasks, whereas closed-loop policies almost completely fail. Note that the policies are trained with *streaming data* in this experiment, meaning that the failure of closed-loop policies is not due to data scarcity or overfitting.

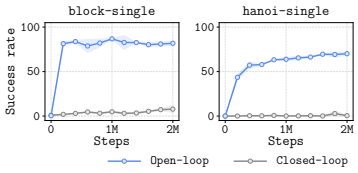


Figure 6: Closed-loop control fails.

There are two main hypotheses for why open-loop control is essential: one is about compounding errors and the other is about non-Markovianity.

Hypothesis 1: Closed-loop control compounds errors. One hypothesis is that closed-loop policies sample (stochastic) actions more often, so errors can accumulate faster than under open-loop control. On the other hand, the policy becomes more complex as the action chunk length increases, so there is a trade-off. The results suggest that the sweet spot might be around length-25 action chunks.

While plausible, this hypothesis is challenged by a recent work (Lazzati et al., 2026), which shows that a closed-loop, "delayed" policy $\pi(a_t \mid s_{t-k})$ can match an open-loop policy's performance on human-collected data. Such a policy still samples an action at every step and thus has the compounding error problem, but can capture temporal dependencies by conditioning on a past state.

We test whether this finding also holds in OCBench. Figure 7 compares the performance of the following five different control strategies on `block-single-task1`:

- `Closed-loop`, a standard closed-loop policy $\pi(a_t \mid s_t)$.
- `Open-loop`, a standard open-loop (action-chunking) policy $\pi(a_{t:t+24} \mid s_t)$.
- `Re-planning`, a closed-loop policy derived from an open-loop policy $\pi(a_{t:t+24} \mid s_t)$ by executing only the first action and re-planning at every step.
- `Delayed`, a delayed policy $\pi(a_t \mid s_{t-24})$.
- `Randomly-delayed`, a randomly delayed policy $\pi(a_t \mid s_{t-k})$, where k is sampled between 0 and 9 (inclusive).

The results show that a randomly delayed closed-loop policy can achieve strong performance (though it still falls behind open-loop control), suggesting that compounding errors alone may not fully explain the gap between closed-loop and open-loop control. This also shows that OCBench's synthetic demo-like trajectories can reproduce a similar result to that observed with real human-demonstration data (Lazzati et al., 2026).

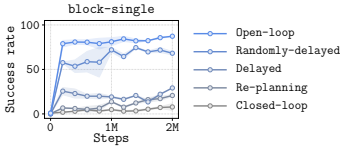


Figure 7: Diverse control strategies.

Hypothesis 2: Closed-loop policies do not capture non-Markovianity in data. Another hypothesis for the necessity of open-loop control is that closed-loop policies fail to capture non-Markovianity in temporally correlated demonstrations. This expressivity mismatch can lead to distribution shift at test time, which in turn leads to poor performance.

One way to test this hypothesis is to compare closed- and open-loop control on "Markovianized" data, where the scripted policy randomly generates a new plan at every step, breaking the temporal dependencies (Zeng et al., 2026).

Figure 8 shows the training curves of closed- and open-loop policies trained on the original and Markovianized datasets. The results suggest that Markovianizing the data can indeed reduce the gap between closed- and open-loop control. However, there still exists a gap on Markovianized data, suggesting that both compounding errors *and* non-Markovianity might contribute to the performance difference.

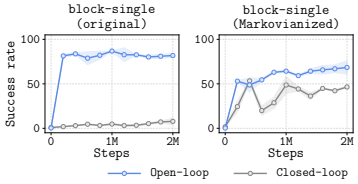


Figure 8: Markovianized datasets.

4.2.1 Aside: What about history conditioning?

A different way to handle non-Markovianity with a closed-loop policy is to condition on *history*, $\pi(a_t \mid s_{t-24:t})$. This allows the policy to capture a similar degree of non-Markovianity to that of the corresponding open-loop policy $\pi(a_{t:t+24} \mid s_t)$.

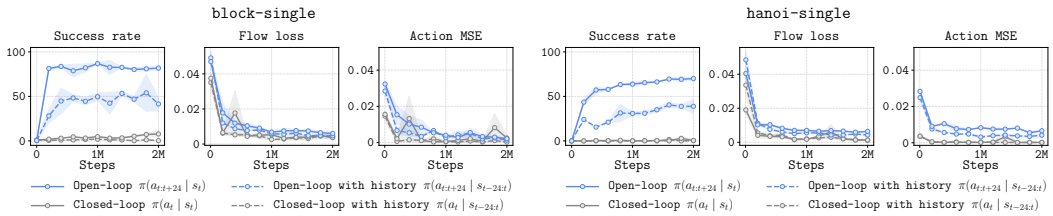


Figure 9: History conditioning can hurt performance despite the lower loss.

Figure 9 compares the performance of closed- and open-loop policies with and without history conditioning, under the streaming-data setting. Perhaps a bit unexpectedly, the results show that history conditioning can *hurt* in practice. Note that this is not due to overfitting, as we use streaming data in this experiment. In fact, history-conditioned policies achieve lower flow loss and action MSE, meaning that they *do* fit the dataset better. However, this does not lead to better performance at test time.

History conditioning hurting performance is a previously reported phenomenon in robotics (de Haan et al., 2019; Mandlekar et al., 2021; Sobol Mark et al., 2026). There are two possible explanations. One is causal confusion (de Haan et al., 2019): a history-conditioned policy might learn to "cheat" by simply outputting the previous action, which is often highly (yet spuriously) correlated with the next action. Another explanation is that history-conditioned policies are more susceptible to test-time distribution shift in general, as they simply have a larger input space. While we do not have a conclusive answer yet, we note that this issue happens even in this basic task with streaming data. We expect that our minimal reproduction will make it easier to study history conditioning in future work.

4.3 Phenomenon 3: Policies should be (very) large

The next phenomenon we study is that we generally need *much* larger networks to clone human behaviors than to clone synthetic trajectories generated by (previous) scripted policies. A typical network used to handle synthetic data in D4RL (Fu et al., 2020) and OGBench (Park et al., 2025a) is a $[512, 512, 512]$-sized multi-layer perceptron (MLP) (Park et al., 2025a), which has roughly 0.5M trainable parameters. In such tasks, we typically see limited performance improvement with larger models (Figure 16). On the other hand, cloning *human* behaviors often benefits from much larger networks (*e.g.*, up to 100M-sized models), especially when using flow matching or diffusion, even in *state-based* environments of similar dimensionality and difficulty to OGBench tasks (Pan et al., 2026). We test whether OCBench is similar to previous synthetic data or human-generated data in terms of the best model size.

Figure 10 presents a model size ablation on two OCBench tasks. The results show that OCBench tasks require much larger networks than previous synthetic benchmarks, as is typically the case with behavioral cloning from human demonstrations (Pan et al., 2026). Figure 10 shows performance keeps improving up to $[8192] \times 8$-sized residual MLPs, which have about 0.5 *billion* parameters, even on these basic state-based manipulation tasks.

We discuss two hypotheses for this phenomenon.

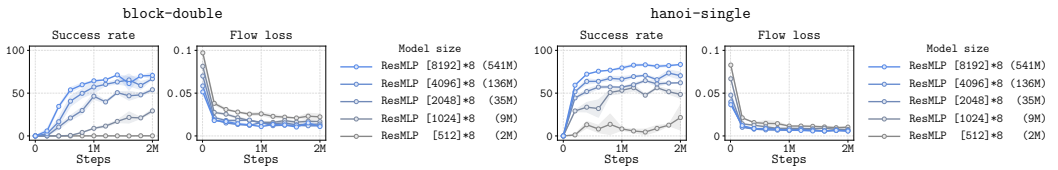


Figure 10: Behavioral cloning from human-like data can require 500M-sized models even on state-based tasks.

Hypothesis 1: Human-data behavioral cloning is simply hard. One hypothesis is that BC from human (or human-like) data is just difficult. Since human behaviors are highly complex and diverse, BC may naturally require a large model to fit the dataset, especially given that overfitting is often beneficial (Section 4.1). This hypothesis could help explain why vision-language-action (VLA) models require a large "action expert," as also reported by Black et al. (2025b). Our result suggests that we may need a billion-scale action expert even with *perfect* perception and task understanding and in a single task.

Hypothesis 2: Flow-matching requires larger networks in general. Another hypothesis is that flow matching (or diffusion) is slower to train and requires a larger capacity (Pertsch et al., 2025; Driess et al., 2025) compared to alternatives (*e.g.*, autoregressive action prediction), possibly due to the high variance of noise coupling.

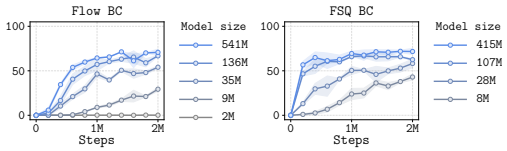


Figure 11: Flow BC vs. FSQ BC.

To test this hypothesis, we perform a similar model size ablation with FSQ (Mentzer et al., 2024), a discrete variant of VAE (Kingma & Welling, 2014). Specifically, we tokenize action chunks with FSQ and train an autoregressive next-token policy over the tokenized actions. We present the ablation results in Figure 11. The results suggest that smaller models indeed perform better with autoregressive policies than with flow-based ones. However, performance still increases with network size even with FSQ tokens, suggesting that the use of flow matching alone does not fully explain the need for large-capacity networks.

4.4 Phenomenon 4: Features matter even with enough data

The last phenomenon we study is that input features substantially affect BC performance, even with virtually unlimited data. A number of works have reported the sensitivity of behavioral cloning to input features (de Haan et al., 2019; Mandlekar et al., 2021; Lu et al., 2026). Since the results from these works are mostly drawn from a fixed set of human-collected trajectories, it is less clear whether this is an issue that will go away with more data or one that is fundamental to behavioral cloning. To understand this phenomenon, we study how different features affect performance under the streaming-data setting in OCBench. In particular, we consider input features that contain identical information but differ only in their relative scales.

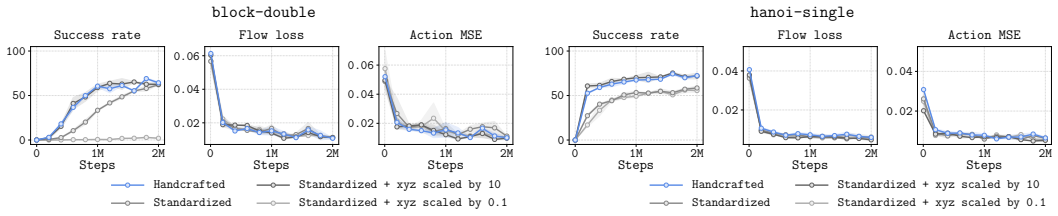


Figure 12: Different input features can lead to different performance despite almost identical training metrics.

Figure 12 shows the performance curves of the following input feature configurations on `block-double-task1` and `hanoi-single-task1` with streaming data:

- `Handcrafted` means that we manually choose the scale of each state dimension.

- `Standardized` means that each state dimension is standardized to the same range.
- `Standardized + xyz scaled by {0.1, 10}` means that after standardizing each state dimension, we scale the dimensions corresponding to xyz coordinates by 0.1 or 10.

The results suggest that performance depends *heavily* on which input features are emphasized, even with unlimited data. Note that all four input feature configurations contain the *same* information, as they only differ by scaling. Indeed, their training and validation metrics (*e.g.*, flow losses and MSEs) are almost indistinguishable. However, their task performance differs substantially.

For this phenomenon, we are relatively confident in the following hypothesis.

Hypothesis: Different features lead to different test-time generalization. Our hypothesis is that this is related to *test-time* policy generalization (de Haan et al., 2019; Park et al., 2024). Even when two policies have identical training metrics, they could generalize differently to unseen states at test time. Ultimately, BC performance is determined by how the policy behaves at states it encounters at test time. For example, in manipulation tasks, one can imagine that policies that focus on the object would generalize better than those that focus on internal joint angles.

To support this point, we measure the MSE between policy and oracle actions on *test-time* states of `hanoi-single-task1`. Specifically, at each test-time state, we generate a fresh oracle plan with OCBench's scripted policy, and measure the difference against the policy action. We present the results in Figure 13. The results suggest that policies with different input features do behave differently at test time, despite their almost identical training metrics. Moreover, test-time MSE correlates better with performance than training or validation MSEs (Park et al., 2024). Although test-time MSE is still an imperfect metric (as it ignores multimodality), these results nevertheless suggest that different features can lead to different *generalization* behavior at test time.

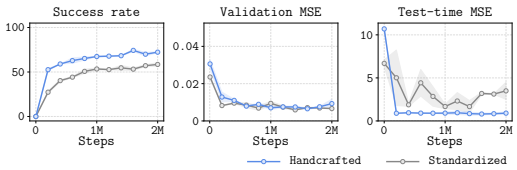


Figure 13: Test-time MSE correlates better with performance than validation MSE.

5 Closing Remarks

In this work, we have introduced OCBench, a controllable robotic manipulation benchmark, accompanied by GPU-accelerated scripted policies that have similar structural properties to human demonstrations. Throughout our analyses, we have experimentally shown that OCBench reproduces diverse real-world BC-related phenomena and enables their scientific studies via its controllable and efficient data generation scripts.

What about reinforcement learning? While RL is outside the scope of this work, OCBench's scripted policies and datasets are designed to enable future studies in RL too. For example, OCBench's scripted policies generate trajectories at different speeds (the fastest is around $2\times$ faster than the slowest), just like how human robot operators have different levels of proficiency. We expect that this makes it easier to study RL in terms of not only success rates but also throughput (*i.e.*, task completion speed). We also provide oracle subtask boundaries and labels to help studies in hierarchical control and RL (see Section B). Given the current discrepancy between real-world and synthetic tasks in data-driven RL (Mandlekar et al., 2021; Huang et al., 2026), we hope that OCBench can help close this gap in robotics RL research too.

Limitations. One limitation is that we mainly use two seeds for the analyses in this paper, due to the high computational cost of training 100M-scale models for 2M steps under a limited compute budget. However, we note that the observed differences are mostly clear enough (*i.e.*, exceed the error ranges) to support our conclusions, and we added more seeds whenever needed (*e.g.*, Figure 4). Another limitation is that, while we design OCBench's scripted policies to have similar properties to human demonstrations, they are still scripted and do not perfectly mimic human behavior. Robotics research should ultimately be done in the real world using real data. Nonetheless, we believe OCBench complements real-world research by serving as a fast, convenient playground where robotics researchers can quickly test and verify their ideas across diverse tasks and conduct controlled studies that would be difficult with real data.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods and cleaning and reformatting datasets. We have not used generative AI tools for helping develop theoretical models or conceptual frameworks, designing or providing feedback on research methodology or experiments, supporting qualitative and thematic data analysis, interpreting results, and generating synthetic datasets.[2] Formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, and assisting with translation are not applicable to this work. We have reviewed all AI-assisted work. For example, we manually checked the core implementation logic generated by LLMs. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the code and exact command-line arguments for reproducing datasets and benchmark table results in the supplementary material. We also describe full implementation details in Section E.

REFERENCES

Michael S Albergo and Eric Vanden-Eijnden. Building Normalizing Flows with Stochastic Interpolants. In *International Conference on Learning Representations (ICLR)*, 2023.

Arthur Allshire, Himanshu Gaurav Singh, Ritvik Singh, Adam Rashid, Hongsuk Choi, David McAllister, Justin Yu, Yiyuan Chen, Huang Huang, Pieter Abbeel, Xi Chen, Rocky Duan, Phillip Isola, Jitendra Malik, Fred Shentu, Guanya Shi, Philipp Wu, and Angjoo Kanazawa. Scalable Behavior Cloning with Open Data, Training, and Evaluation. In *Conference on Robot Learning (CoRL)*, 2026.

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *ArXiv*, abs/1607.06450, 2016.

Shuanghao Bai, Wanqi Zhou, Pengxiang Ding, Wei Zhao, Donglin Wang, and Badong Chen. Rethinking Latent Redundancy in Behavior Cloning: An Information Bottleneck Approach for Robot Manipulation. In *International Conference on Machine Learning (ICML)*, 2025.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. In *Conference on Robot Learning (CoRL)*, 2025a.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Robotics: Science and Systems (RSS)*, 2025b.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, 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.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov,

[2] We used generative AI tools to help implement scripted policies, but synthetic data was generated by code, not directly by AI.

Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *Conference on Robot Learning (CoRL)*, 2023.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research (IJRR)*, 2025.

Felipe Codevilla, Eder Santana, Antonio M. López, and Adrien Gaidon. Exploring the Limitations of Behavior Cloning for Autonomous Driving. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Sudeep Dasari, Oier Mees, Sebastian Zhao, Mohan Kumar Srirama, and Sergey Levine. The Ingredients for Robotic Diffusion Transformers. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal Confusion in Imitation Learning. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, and Sergey Levine. Knowledge Insulating Vision-Language-Action Models: Train Fast, Run Fast, Generalize Better. In *Neural Information Processing Systems (NeurIPS)*, 2025.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A. Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit Behavioral Cloning. In *Conference on Robot Learning (CoRL)*, 2021.

Justin Fu, Aviral Kumar, Ofir Nachum, G. Tucker, and Sergey Levine. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. *ArXiv*, abs/2004.07219, 2020.

Kevin Huang, Rosario Scalise, Cleah Winston, Ayush Agrawal, Yunchu Zhang, Rohan Baijal, Markus Grotz, Byron Boots, Benjamin Burchfiel, Masha Itkina, Paarth Shah, and Abhishek Gupta. Using Non-Expert Data to Robustify Imitation Learning via Offline Reinforcement Learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The Robot Learning Benchmark & Learning Environment. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):3019–3026, 2020.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model. In *Conference on Robot Learning (CoRL)*, 2024.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Filippo Lazzati, Kyle Stachowicz, William Chen, Alberto Maria Metelli, Andrew Wagenmaker, and Sergey Levine. Why Does Action Chunking Improve Behavioral Cloning Performance in Robotic Control? In *Conference on Robot Learning (CoRL)*, 2026.

Qiyang Li, Zhiyuan (Paul) Zhou, and Sergey Levine. Reinforcement Learning with Action Chunking. In *Neural Information Processing Systems (NeurIPS)*, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. In *Neural Information Processing Systems (NeurIPS)*, 2023a.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *International Conference on Learning Representations (ICLR)*, 2023b.

Cong Lu, Philip J. Ball, Tim G. J. Rudner, Jack Parker-Holder, Michael A. Osborne, and Yee Whye Teh. Challenges and Opportunities in Offline Reinforcement Learning from Visual Observations. *Transactions on Machine Learning Research (TMLR)*, 2023.

Jingxian Lu, Wenke Xia, Yuxuan Wu, Zhiwu Lu, and Di Hu. When would Vision-Proprioception Policies Fail in Robotic Manipulation? In *International Conference on Learning Representations (ICLR)*, 2026.

Miles Macklin. Warp: A High-performance Python Framework for GPU Simulation and Graphics, 2022. URL https://github.com/NVIDIA/warp.

Ajay Mandlekar, Danfei Xu, J. Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What Matters in Learning from Offline Human Demonstrations for Robot Manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. MimicGen: A Data Generation System for Scalable Robot Learning using Human Demonstrations. In *Conference on Robot Learning (CoRL)*, 2023.

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite Scalar Quantization: VQ-VAE Made Simple. In *International Conference on Learning Representations (ICLR)*, 2024.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An Open Foundation Model for Generalist Humanoid Robots. *ArXiv*, abs/2503.14734, 2025.

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open X-Embodiment: Robotic Learning Datasets and RT-X Models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. Training language models to follow instructions with human feedback. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Chaoyi Pan, Giridharan Anantharaman, Nai-Chieh Huang, Claire Jin, Daniel Pfrommer, Chenyang Yuan, Frank Permenter, Guannan Qu, Nicholas Boffi, Guanya Shi, and Max Simchowitz. Much Ado About Noising: Dispelling the Myths of Generative Robotic Control. In *International Conference on Learning Representations (ICLR)*, 2026.

Jyothish Pari, Nur Muhammad (Mahi) Shafiullah, Sridhar Pandian Arunachalam, and Lerrel Pinto. The Surprising Effectiveness of Representation Learning for Visual Imitation. In *Robotics: Science and Systems (RSS)*, 2022.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is Value Learning Really the Main Bottleneck in Offline RL? In *Neural Information Processing Systems (NeurIPS)*, 2024.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking Offline Goal-Conditioned RL. In *International Conference on Learning Representations (ICLR)*, 2025a.

Seohong Park, Qiyang Li, and Sergey Levine. Flow Q-Learning. In *International Conference on Machine Learning (ICML)*, 2025b.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient Action Tokenization for Vision-Language-Action Models. In *Robotics: Science and Systems (RSS)*, 2025.

Physical Intelligence, Bo Ai, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Greg Balke, Kevin Black, George Bokinsky, Shihao Cao, Thomas Charbonnier, Vedant Choudhary, Foster Collins, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Maitrayee Dhaka, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachlan Groom, Haroun Habeeb, Hunter Hancock, Karol Hausman, Gashon Hussein, Victor Hwang, Brian Ichter, Connor Jacobsen, Szymon Jakubczak, Rowan Jen, Tim Jones, Gregg Kammerer, Ben Katz, Liyiming Ke, Mairbek Khadikov, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Brendon LeCount, Sergey Levine, Xinyu Li, Adrian Li-Bell, Vladislav Lialin, Zhonglin Liang, Wallace Lim, Yao Lu, Enyu Luo, Vishnu Mano, Nandan Marwaha, Aikys Mongush, Liam Murphy, Suraj Nair, Tyler Patterson, Karl Pertsch, Allen Z. Ren, Gavin Schelske, Charvi Sharma, Baifeng Shi, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Jiaming Tang, Jimmy Tanner, Shalom Tekeste, Marcel Torne, Kyle Vedder, Quan Vuong, Anna Walling, Haohuan Wang, Jason Wang, XuDong Wang, Chris Whalen, Samuel Whitmore, Blake Williams, Charles Xu, Sukwon Yoo, Lili Yu, Wuming Zhang, Zhuoyang Zhang, and Ury Zhilinsky. $\pi_{0.7}$: a Steerable Generalist Robotic Foundation Model with Emergent Capabilities. *ArXiv*, abs/2604.15483, 2026.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

Nur Muhammad (Mahi) Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior Transformers: Cloning k modes with one stone. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Max Simchowitz, Daniel Pfrommer, and Ali Jadbabaie. The Pitfalls of Imitation Learning when Actions are Continuous. In *Conference on Learning Theory (COLT)*, 2025.

Max Sobol Mark, Jacky Liang, Maria Attarian, Chuyuan Fu, Debidatta Dwibedi, Aviral Kumar, and Dhruv Shah. Long-Context Robot Imitation Learning by Focusing on Key History Frames. In *Robotics: Science and Systems (RSS)*, 2026.

Jonathan Spencer, Sanjiban Choudhury, Arun Venkatraman, Brian Ziebart, and J. Andrew Bagnell. Feedback in Imitation Learning: The Three Regimes of Covariate Shift. *ArXiv*, abs/2102.02872, 2021.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-Kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Viswesh Nagaswamy Rajesh, Yong Woo Choi, Yen-Ru Chen, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Demonstrating GPU Parallelized Robot Simulation and Rendering for Generalizable Embodied AI with ManiSkill3. In *Robotics: Science and Systems (RSS)*, 2025.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

Chuan Wen, Jierui Lin, Trevor Darrell, Dinesh Jayaraman, and Yang Gao. Fighting Copycat Agents in Behavioral Cloning from Observation Histories. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. ConvNeXt V2: Co-Designing and Scaling ConvNets With Masked Autoencoders. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2023.

Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, P. Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don't Change the Algorithm, Change the Data: Exploratory Data for Offline Reinforcement Learning. *ArXiv*, abs/2201.13425, 2022.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning. In *Conference on Robot Learning (CoRL)*, 2019.

Kevin Zakka, Yuval Tassa, and MuJoCo Menagerie Contributors. MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo, 2022. URL http://github.com/google-deepmind/mujoco_menagerie.

Michael Zeng, Abhinav Agarwal, Ajay Bati, Brian Lee, Siddharth Ancha, and Russ Tedrake. Revisiting Open-Loop Execution in Robotics: Toward Reactive, Higher-Performing Policies. *ArXiv*, abs/2608.15938, 2026.

Thomas T. Zhang, Daniel Pfrommer, Chaoyi Pan, Nikolai Matni, and Max Simchowitz. Action Chunking and Data Augmentation Yield Exponential Improvements in Behavior Cloning for Continuous Spaces. In *International Conference on Learning Representations (ICLR)*, 2026.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Robotics: Science and Systems (RSS)*, 2023.

A Related work

Behavioral cloning. Modern large-scale robotic control is mainly based on behavioral cloning and imitation learning (Brohan et al., 2023; Kim et al., 2024; NVIDIA et al., 2025; Physical Intelligence et al., 2026). Among diverse behavioral cloning techniques, including those based on multi-layer perceptrons (MLPs) (Fu et al., 2020; Mandlekar et al., 2021), tokenized actions (Shafiullah et al., 2022; Brohan et al., 2023), and energy-based models (Florence et al., 2021), the most popular approaches to date are diffusion (Chi et al., 2025) and flow matching (Black et al., 2025b). Researchers have scaled up flow behavioral cloning with large transformers, demonstrating its effectiveness across a wide array of robotic tasks (NVIDIA et al., 2025; Physical Intelligence et al., 2026). Accordingly, we mainly focus on flow behavioral cloning in this work, although many of the phenomena studied in this work may apply to other behavioral cloning techniques as well.

Analyses of behavioral cloning. Several works have analyzed diverse phenomena and failure modes in behavioral cloning. One line of work has focused on test-time distribution shift and DAgger from both theoretical and empirical perspectives (Ross et al., 2011; Spencer et al., 2021; Park et al., 2024; Simchowitz et al., 2025). Another body of work has analyzed the effect of modern techniques, such as action chunking and open-loop control (Zhang et al., 2026; Lazzati et al., 2026; Zeng et al., 2026), as well as history conditioning (de Haan et al., 2019; Wen et al., 2020; Sobol Mark et al., 2026). Many of the aforementioned works rely primarily on relatively small, fixed, human-collected datasets (*e.g.*, 50–300 demonstrations per task). In contrast, recent large-scale robotic systems train on orders of magnitude more robot data (Open X-Embodiment Collaboration et al., 2024; Physical Intelligence et al., 2026). The difference in data scale often serves as a confounder, making it unclear whether these issues will eventually go away with more data or are fundamental to behavioral cloning. In this work, rather than relying on limited human-collected data, we mimic the structural properties of human demonstrations with GPU-accelerated scripted policies, enabling effectively unlimited data generation at up to 200K steps per second, while reproducing many phenomena observed in real-world behavioral cloning. To the best of our knowledge, this is the first work to systematically reproduce multiple real-world BC phenomena using GPU-accelerated, fully controllable scripted policies.

Behavioral cloning and reinforcement learning benchmarks. A variety of simulated benchmarks have been proposed for behavioral cloning and data-driven reinforcement learning (RL). Previous behavioral cloning benchmarks typically provide simulated tasks with tens to a few hundred human-collected demonstrations per task (Mandlekar et al., 2021; Liu et al., 2023a). Unlike them, we instead design scripted policies that mimic human data's properties to enable unlimited generation of controlled trajectories. Among existing BC benchmarks, MimicGen (Mandlekar et al., 2023) is particularly relevant, which supports systematic human-like data generation. Unlike MimicGen, which generates trajectories by first recording about 10 human trajectories per task and "replaying" their segments in new randomized configurations, OCBench generates data using fully scripted policies having human-like properties and provides a host of controllable knobs (*e.g.*, motion speed, mistake probability, coverage degree, etc.), enabling both diverse data generation and scientific analyses. Another line of simulated benchmarks (Fu et al., 2020; James et al., 2020; Yarats et al., 2022; Lu et al., 2023; Tao et al., 2025; Park et al., 2025a) provides scripted or RL-trained policies to generate data. Compared to these previous benchmarks, OCBench provides more human-like scripted policies, enabling us to reproduce diverse phenomena that happen in real-world behavioral cloning but are not easily observed in previous scripted benchmarks (Section 2). We also provide GPU-accelerated environments and scripted policies based on MJWarp. OCBench's closest predecessor is OGBench (Park et al., 2025a), and we compare them in detail in Section C.

B TASKS AND DATASETS

OCBench provides 28 tasks across five types of environments (`block`, `chamber`, `switch`, `hanoi`, and `bowling`). We provide basic specifications for the tasks in Tables 3 and 4 and the datasets in Table 5.

- `block` (8 tasks in total) is a standard block manipulation environment (Figure 19). We provide four variants (`single`, `double`, `triple`, and `quadruple`) with one to four blocks, with each environment variant defining up to three different tasks (`task1`–`task3`).

  Across four environments, `task1` is a fixed pick-and-place task, where the agent must place the blocks at pre-defined positions, starting from randomized initial states. `task2` is a "stack anywhere" task, where the goal is to stack the blocks anywhere on the table in any order. `task3` is available only in `block-quadruple` and requires the agent to arrange the blocks in a grid at a fixed position.
- `chamber` (7 tasks in total) is an environment with multiple objects (blocks, a drawer, a window, and button locks) that require different manipulation strategies (Figure 20). We provide three variants (`easy`, `medium`, and `hard`) with up to three different tasks (`task1`–`task3`). The position and orientation of the objects are randomized, the drawer and window are randomly swapped, and demonstrations complete the subtasks in a random order, with occasional high-level mistakes. Hence, the agent must deal with multimodal demonstrations over a long horizon (up to 5500 steps).

  `chamber-easy-task1` is the simplest task and does not require block manipulation; its goal is to open both the drawer and window (after unlocking them if necessary), and lock them again. `chamber-easy-task2` and `chamber-{medium, hard}-task1` require the agent to put all blocks in the drawer and lock it, and open the window and lock it. `chamber-easy-task3` and `chamber-{medium, hard}-task2` require the agent to pull all blocks out of the drawer, stack them anywhere on the table in any order, and leave both the window and drawer closed and locked.
- `switch` (6 tasks in total) is an environment based on the Lights Out puzzle, which requires combinatorial reasoning (Figure 21). We provide three board sizes (`3x3`, `4x4`, and `5x5`) and two variants: `task1` provides a fixed grid, and `task2` provides a randomized grid where at most one (`3x3`) or two (`4x4` and `5x5`) randomly selected buttons are omitted. When the agent presses a button, it toggles the color of the pressed button and up to four adjacent buttons. The goal is to make all buttons the same color (either all blue or all black). The demonstrations for these tasks are *highly* suboptimal, unlike those for the other environments in OCBench, so this environment examines the ability to learn from suboptimal data.
- `hanoi` (6 tasks in total) is an environment based on the Tower of Hanoi puzzle (Figure 22). We provide three variants (`single`, `double`, and `triple`) with one to three disks, with each variant defining two tasks: `task1` uses fixed pegs, and `task2` randomizes the positions of the pegs. The disks are initially stacked on the middle peg, and the goal is to move them to *either* the first or third peg, while following the rules of the Tower of Hanoi. Since the disks fit tightly around the pegs, these tasks require highly precise manipulation maneuvers over up to 7000 environment steps.
- `bowling` (1 task) is a task whose goal is to push a bowling ball to knock down as many pins as possible (Figure 23). Since the dynamics are not easily predictable due to the shape of the bowling pins, this task requires understanding of the environment dynamics from highly diverse outcomes.

Among them, the first three types of environments are modified from the `cube`, `scene`, and `puzzle` environments in OGBench (Park et al., 2025a), although we made substantial modifications to make them more suitable for standard robotic manipulation studies (see Section C). As in OGBench, we use assets from MuJoCo Menagerie (Zakka et al., 2022) and Meta-World (Yu et al., 2019) for the UR5e robot arm and objects.

Lite tasks. In OCBench, we also provide `lite` versions of all original tasks except for `bowling`. These tasks have shorter horizons and simplified scripted policies. The main differences are threefold: (1) `lite` tasks use 10Hz control whereas the original ones use 50Hz, (2) the scripted policies for `lite` tasks move approximately $2\times$ as fast as the original ones (so if we combine this with the

reduction from 50Hz to 10Hz, `lite` tasks have approximately $10\times$ shorter horizons in total), and (3) the scripted policies for `lite` tasks are simpler (they do not randomize the gripper's roll and pitch). As a result, `lite` tasks can be learned with a smaller network (*e.g.*, $[1024] \times 8$-sized networks, as opposed to $[4096] \times 8$ for full versions). We expect `lite` tasks to be useful for faster initial iteration on algorithmic ideas.

Pixel-based tasks. We also provide `visual` versions of OCBench tasks for pixel-based control. While we only provide datasets for selected tasks due to storage constraints, one can easily generate data using OCBench's data generation scripts. A visual observation consists of three camera images of size $224\times224\times3$ (front, side, and wrist). In the provided datasets, images are recorded every 25 steps (*i.e.*, one image set per length-25 action chunk) to save space. We found that this still allows reasonably high performance while substantially reducing the dataset size and speeding up data generation.

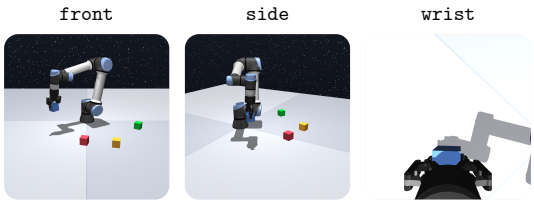


Figure 14: Camera observations ($224 \times 224 \times 3$).

Rewards and metrics. All tasks except for `bowling` use binary sparse rewards. If the agent successfully completes the task, it receives a reward of 1 and the episode immediately terminates. Otherwise, it receives a reward of 0. On `bowling`, the agent receives a reward of 0.1 for each pin it knocks down for the first time. Across all environments, the sum of rewards is always bounded by $[0, 1]$.

For all tasks except for `bowling`, we mainly consider two metrics: success rate and throughput. Success rate is the ratio of successful episodes to total episodes. Throughput is defined as follows:

$$(\text{Throughput}) = (\text{Success rate}) \times \frac{(\text{Maximum episode length})}{(\text{Mean episode length})}, \tag{1}$$

where the mean episode length is computed over all episodes, including failures. For `bowling`, we consider the normalized average return, defined as the average return multiplied by 100. Throughout the tables (Tables 5 and 6), `bowling` numbers denote this normalized average return instead of the success rate.

Auxiliary information. OCBench datasets provide additional information beyond standard observations and actions. First, we provide full simulation states (`qpos`, `qvel`, `button_states`, and `dynamics_info`) from which observations can be perfectly reconstructed. Second, we provide segment information for each atomic subtask in a trajectory (*e.g.*, opening a drawer, pressing a button, etc.), including subtask types, target information, subtask success indicators, and high-level mistake indicators. We expect this auxiliary information to be useful for research in RL and hierarchical control as well as for more fine-grained analyses.

C How is OCBench different from previous synthetic benchmarks?

One of the main contributions of OCBench is that it features more human-like scripted policies. This enables reproducing many BC-related phenomena (see Section 2) that happen in the real world. In this section, we discuss how OCBench's scripted policies differ from those of previous benchmarks based on synthetic data (collected by scripted or RL-trained policies) in detail, showing how previous synthetic benchmarks might not fully reproduce all these phenomena.

Open-loop vs. closed-loop control (Section 4.2). Figure 15 compares the performance of fully closed-loop policies and partially open-loop (action-chunking) policies on standard single-cube pick-and-place tasks from three benchmarks: OGBench (Park et al., 2025a), Robomimic (Mandlekar et al., 2021), and OCBench. For Robomimic, we use machine-generated (MG) datasets, as our goal in this experiment is to compare policies trained on synthetic data. The results suggest that the gap between closed- and open-loop policies is much less pronounced in previous benchmarks, than in OCBench or with human-generated data (Zhao et al., 2023; Lazzati et al., 2026; Zeng et al., 2026). Although action chunking does benefit long-horizon tasks in OGBench (Li et al., 2025), synthetic policies in previous benchmarks typically generate trajectories that have (much) less tempo-

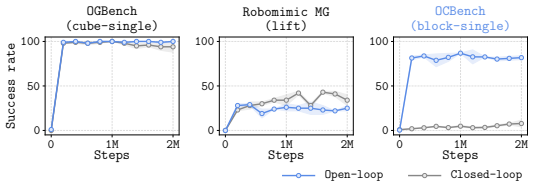


Figure 15: The difference between closed-loop and open-loop policies is less pronounced in previous synthetic benchmarks.

ral correlation than human demonstrations, which could explain why the gap between closed- and open-loop control is generally smaller in these previous benchmarks.

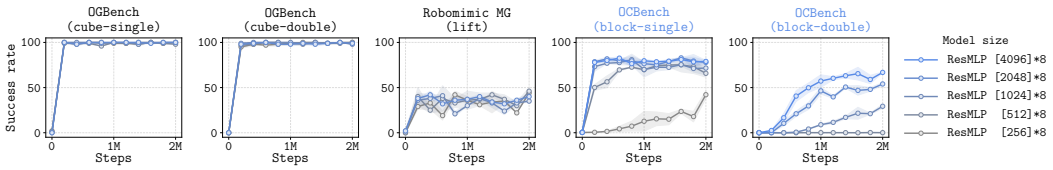


Figure 16: A smaller network ($[256] \times 8$) can saturate performance on previous synthetic benchmarks. In contrast, OCBench requires larger networks (as is the case with human demonstrations) even for tasks of similar semantic difficulty.

Model size ablation (Section 4.3). Figure 16 shows how different model sizes affect performance on five single- or double-cube pick-and-place tasks across three benchmarks. We use `cube-{single, double}` from OGBench (Park et al., 2025a), `lift` from Robomimic MG (Mandlekar et al., 2021), and `block-{single, double}` from OCBench. The results show that relatively small networks ($[256] \times 8$) already saturate performance on previous benchmarks, whereas OCBench tasks benefit from larger networks of up to hundreds of millions of parameters even for tasks of similar semantic complexity (*e.g.*, double pick-and-place). This model-size scaling property resembles that observed with human demonstrations (Pan et al., 2026). We believe this is mainly because OCBench's trajectories are more narrowly distributed and temporally correlated than the Gaussian noise-augmented trajectories from the previous benchmarks.

Action distribution comparison (Figure 1). In the main paper, we qualitatively compare the action distributions from previous synthetic benchmarks (OGBench and Robomimic MG), OCBench, and human demonstrations (Figure 1). Specifically, we plot per-dimension action distributions computed on `cube-double-play` for OGBench, `can` for Robomimic MG, `block-double-task1` for OCBench, and BiPlay (Dasari et al., 2025) for human demonstrations. For OGBench, since the original dataset is designed for goal-conditioned RL, we re-generate a single-task dataset on OCBench's `block-double-task1` using OGBench's scripted policy to ensure compatibility. For BiPlay, we convert the original absolute action targets into relative ones to make them compatible with the relative action representations in the other benchmarks.

While the absolute action values in the action distributions in Figure 1 may not be directly comparable due to the difference in joint ranges from different robots, the results nevertheless highlight that previous synthetic data often have qualitatively different *properties* from those of human demonstrations. For example, they often have highly bimodal actions peaked at -1 or $+1$ or have overly uniform distributions, unlike human demonstrations. On the other hand, OCBench's action distributions more closely resemble those of human demonstrations.

OGBench vs. OCBench. OGBench (Park et al., 2025a) is the closest predecessor of OCBench. Here, we highlight three key differences between OGBench and OCBench. Unlike OGBench, (1) OCBench's scripted policies are explicitly designed to mimic the properties of human demonstrations (see Figures 1, 15 and 16), (2) it supports MJWarp-based GPU acceleration, and (3) it provides standard robotic tasks not limited to goal-conditioned ones. We summarize the main differences in Table 1.

Table 1: **OGBench vs. OCBench.** OCBench provides more human-like scripted policies, is not limited to goal-conditioned tasks, and supports GPU acceleration.

Property	**OGBench** (Park et al., 2025a)	**OCBench (ours)**
Task type	Goal-reaching tasks	General robotic tasks
Action space	5-D end effector control	7-D joint position control
Action frequency	20Hz	50Hz
Environment randomization	No	Yes
GPU acceleration	No	Yes
Scripted policy	More rule-based	More human-like

D ADDITIONAL RESULTS

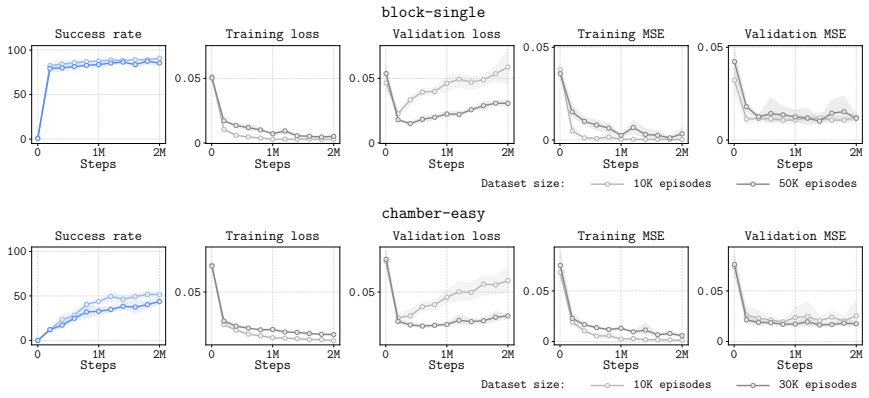
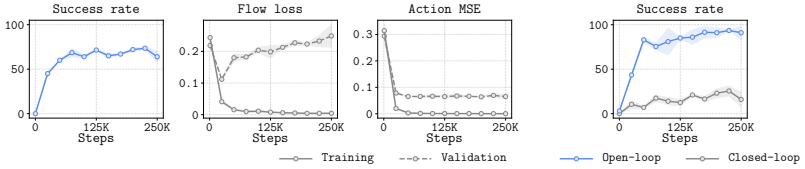


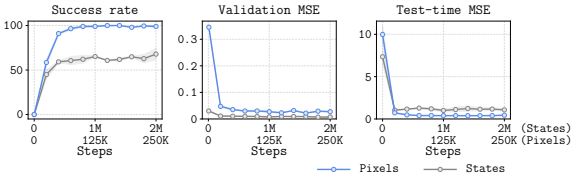
Figure 17: The action MSE metric does not fully explain the dataset size-reversal phenomenon.

Additional plots. Figure 17 compares performance, training and validation flow losses, and training and validation action MSEs on datasets of different sizes. The results show that the MSE metric still does not fully explain the size-reversal phenomenon (see Section 4.1 for the context).



(a) Overfitting is not necessarily bad.

(b) Closed-loop control fails.



(c) Features matter.

Figure 18: Pixel-based OCBench tasks reproduce BC-related phenomena too.

Results with pixel-based tasks. While we focus on state-based tasks for computational reasons in this work, we also present preliminary results showing that pixel-based OCBench tasks can be used to study the same BC-related phenomena too. After all, the phenomena studied in this work are "reproductions" of real-world phenomena observed with visual data (Zhao et al., 2023; Sobol Mark et al., 2026; Allshire et al., 2026).

Figure 18 reproduces three phenomena on visual OCBench tasks (`visual-block-single-task1` for (a) and (b), and `visual-hanoi-single-task1` for (c)). Figure 18a shows that overfitting does not necessarily hurt performance (Figure 3). Figure 18b shows that open-loop control substantially improves performance over closed-loop control (Figure 6). Figure 18c shows that test-time MSE can have an opposite trend to validation MSE for two different features (states vs. pixels), and that this test-time generalization metric can better explain the actual performance (Figure 13).

Benchmark results. While the main focus of this paper is analysis rather than benchmarking and comparing different algorithms, we provide a preliminary benchmark result for three BC algorithms (BC, flow BC, and FSQ BC) in Tables 6 and 7 as a reference. The results are averaged over two seeds and $\pm$ denotes a standard deviation.

E Implementation details

We describe the full implementation details in this section. We also provide the code and exact command-line arguments for reproducing the benchmark tables (Tables 6 and 7) in the supplementary material. For datasets, as we were unable to find an anonymous file hosting service that can accommodate several terabytes of data, we instead provide the exact data generation commands during the review period. We will host our datasets once the paper is de-anonymized.

E.1 Methods

We implement all methods in JAX (Bradbury et al., 2018). By default, we train BC agents only using successful episodes (except for `bowling`) with length-25 action chunks (length-3 action chunks for `lite` tasks). We use residual MLPs (ResMLPs) with residual streams over every two adjacent layers. We follow the action normalization scheme used in $\pi_{0.5}$ (Black et al., 2025a): we first compute the 1st and 99th percentiles for each action dimension except for the gripper dimension, map them to -1 and 1 respectively, and clip normalized values to $[-5, 5]$. For the gripper dimension, we simply clip the values to $[-1, 1]$ without normalization. In the benchmark table, we train agents with 2M steps, evaluate each agent with 500 rollouts every 50K steps, and report the performance averaged over the last 5 evaluation steps ($\{1.8\text{M}, 1.85\text{M}, 1.9\text{M}, 1.95\text{M}, 2\text{M}\}$ steps). For streaming-data runs, we collect 25K episodes every 100K steps (and discard old data) by default.

Behavioral cloning (BC). Vanilla behavioral cloning trains a deterministic policy with a standard MSE loss.

Flow behavioral cloning (FBC). Flow behavioral cloning trains a velocity field with standard flow matching (Lipman et al., 2023; Liu et al., 2023b; Albergo & Vanden-Eijnden, 2023). We use $\text{Beta}(1, 1.5)$ (*i.e.*, emphasizing the noisy part) for the flow time distribution, similarly to Black et al. (2025b). This differs from previous flow-based works on OGBench (Park et al., 2025b; Li et al., 2025), which use a uniform time distribution. We found that the use of the beta distribution substantially improves training speed in OCBench, whereas it makes no clear difference in OGBench.

FSQ behavioral cloning (FSQBC). FSQ behavioral cloning jointly trains two models: (1) a state-conditional finite scalar quantization (FSQ) (Mentzer et al., 2024) autoencoder that encodes an action chunk into discrete tokens and reconstructs it using an ℓ_1 loss, and (2) an autoregressive BC network that predicts these tokens given a state. By default, we encode each action chunk into four tokens, each with quantization levels of $[8, 5, 5, 5]$. This gives us up to $(8 \times 5 \times 5 \times 5)^4 = 1000^4$ possible action chunks given a state.

Pixel-based behavioral cloning. To handle visual observations, we use a ConvNeXt encoder (Woo et al., 2023). Specifically, we encode each of the three images using a shared ConvNeXt V2-N encoder, concatenate the three 640-dimensional representations into a 1920-dimensional vector, and feed it to a standard $[4096] \times 8$-sized residual MLP. For training, we apply random-crop image augmentation by default.

E.2 Hyperparameters

We provide the default hyperparameters used in our experiments in Table 2. We refer the reader to the code provided in the supplementary material for details.

Table 2: **Default hyperparameters.**

Hyperparameter	**Value**
Gradient steps	2M (default), 250K (`visual`)
Optimizer	Adam (Kingma & Ba, 2015)
Learning rate	0.0001
Batch size	1024 (default), 256 (FSQBC, `visual`)
ResMLP size (BC, FBC)	$[4096] \times 8$ (default), $[1024] \times 8$ (`lite`)
ResMLP size (FSQBC)	$[2048] \times 8$ (default), $[512] \times 8$ (`lite`)
MLP normalization	Layer normalization (Ba et al., 2016)
Action chunk length	25 (default), 3 (`lite`)

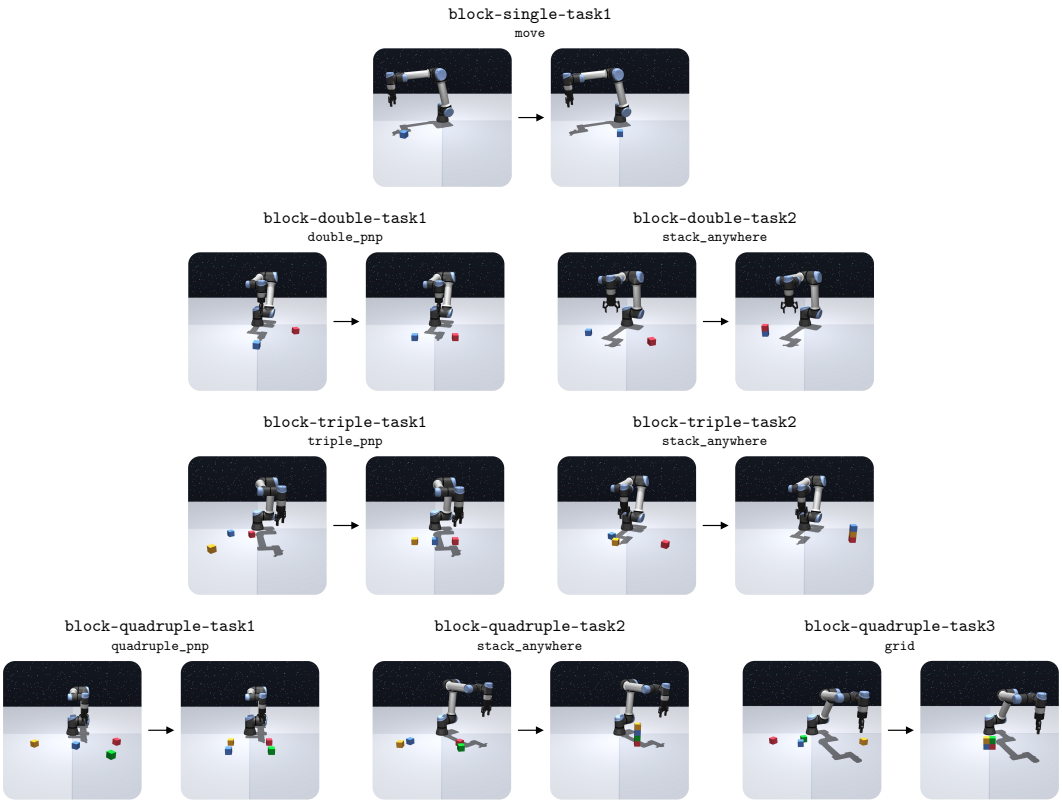


Figure 19: `block` tasks.

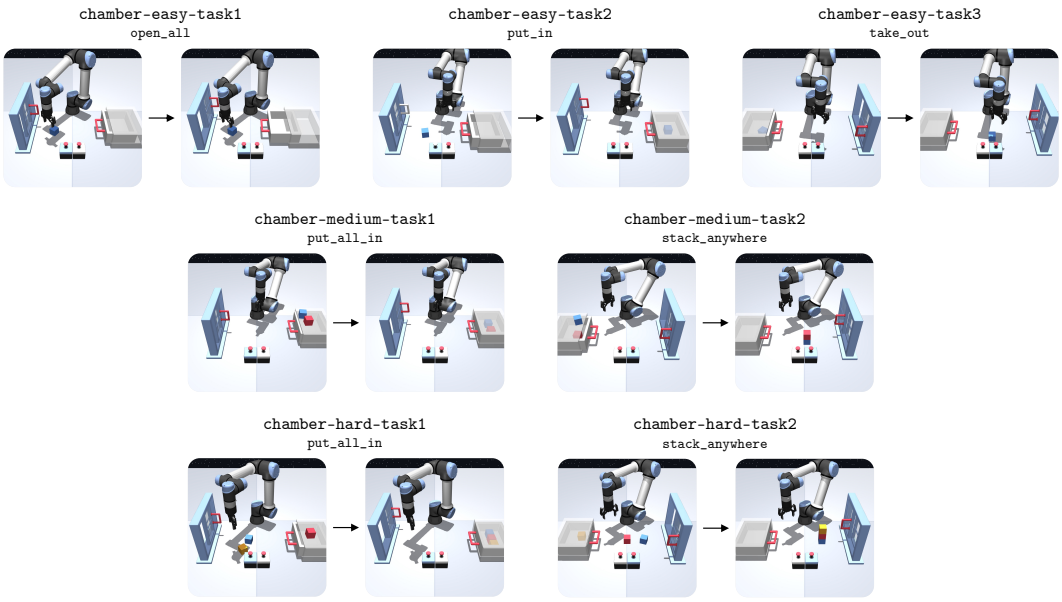


Figure 20: `chamber` tasks.

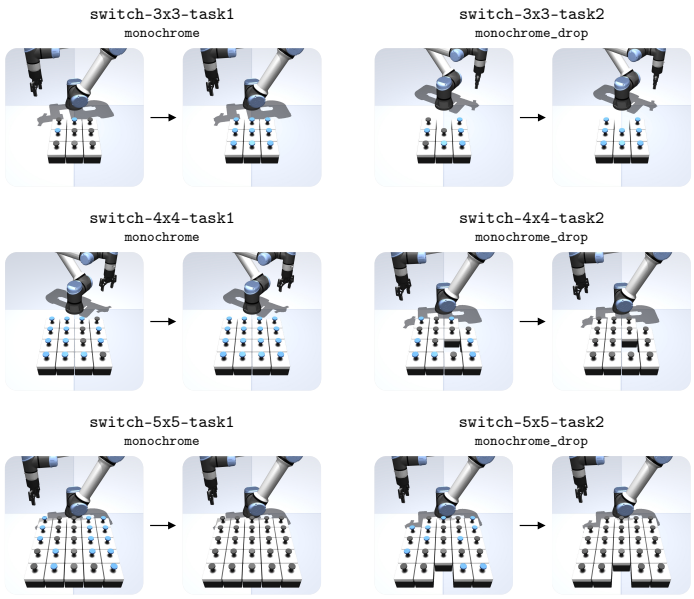


Figure 21: `switch` tasks.

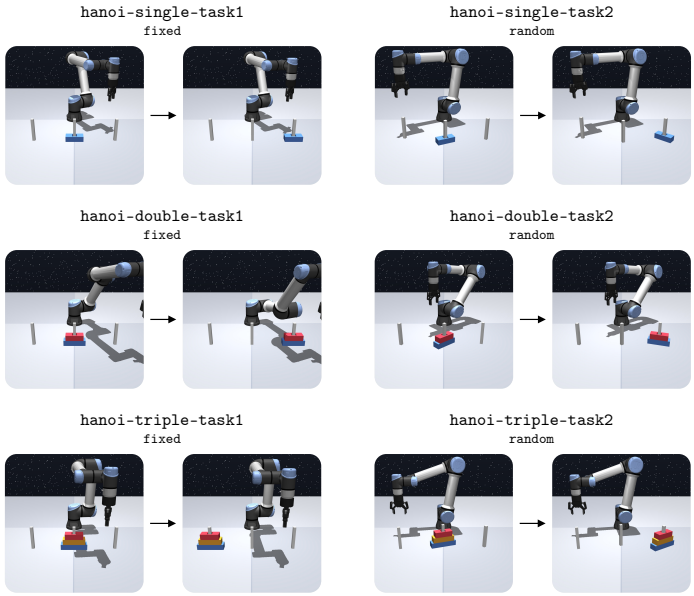


Figure 22: `hanoi` tasks.

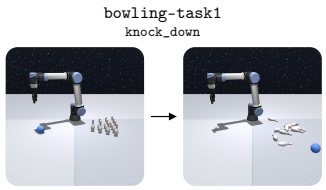


Figure 23: `bowling` task.

Table 3: **State-based task specifications.**

Task	State dimensions	Action dimensions	Maximum episode length
block-single-task1-v0	28	7	1250
block-double-task1-v0	37	7	2500
block-double-task2-v0	37	7	2500
block-triple-task1-v0	46	7	3750
block-triple-task2-v0	46	7	3750
block-quadruple-task1-v0	55	7	5000
block-quadruple-task2-v0	55	7	5000
block-quadruple-task3-v0	55	7	5000
chamber-easy-task1-v0	53	7	4000
chamber-easy-task2-v0	53	7	4000
chamber-easy-task3-v0	53	7	4000
chamber-medium-task1-v0	62	7	5000
chamber-medium-task2-v0	62	7	5000
chamber-hard-task1-v0	71	7	5500
chamber-hard-task2-v0	71	7	5500
switch-3x3-task1-v0	64	7	1500
switch-3x3-task2-v0	64	7	1500
switch-4x4-task1-v0	99	7	1500
switch-4x4-task2-v0	99	7	1500
switch-5x5-task1-v0	144	7	4000
switch-5x5-task2-v0	144	7	4000
hanoi-single-task1-v0	39	7	1000
hanoi-single-task2-v0	39	7	1000
hanoi-double-task1-v0	44	7	3000
hanoi-double-task2-v0	44	7	3000
hanoi-triple-task1-v0	49	7	7000
hanoi-triple-task2-v0	49	7	7000
bowling-task1-v0	102	7	500
block-lite-single-task1-v0	28	7	200
block-lite-double-task1-v0	37	7	400
block-lite-double-task2-v0	37	7	400
block-lite-triple-task1-v0	46	7	600
block-lite-triple-task2-v0	46	7	600
block-lite-quadruple-task1-v0	55	7	800
block-lite-quadruple-task2-v0	55	7	800
block-lite-quadruple-task3-v0	55	7	800
chamber-lite-easy-task1-v0	53	7	500
chamber-lite-easy-task2-v0	53	7	500
chamber-lite-easy-task3-v0	53	7	500
chamber-lite-medium-task1-v0	62	7	600
chamber-lite-medium-task2-v0	62	7	600
chamber-lite-hard-task1-v0	71	7	700
chamber-lite-hard-task2-v0	71	7	700
switch-lite-3x3-task1-v0	64	7	300
switch-lite-3x3-task2-v0	64	7	300
switch-lite-4x4-task1-v0	99	7	300
switch-lite-4x4-task2-v0	99	7	300
switch-lite-5x5-task1-v0	144	7	800
switch-lite-5x5-task2-v0	144	7	800
hanoi-lite-single-task1-v0	39	7	300
hanoi-lite-single-task2-v0	39	7	300
hanoi-lite-double-task1-v0	44	7	900
hanoi-lite-double-task2-v0	44	7	900
hanoi-lite-triple-task1-v0	49	7	2100
hanoi-lite-triple-task2-v0	49	7	2100

Table 4: **Pixel-based task specifications.**

Task	State dimensions	Action dimensions	Maximum episode length
visual-block-single-task1-v0	3×224×224×3	7	1250
visual-block-double-task1-v0	3×224×224×3	7	2500
visual-block-double-task2-v0	3×224×224×3	7	2500
visual-block-triple-task1-v0	3×224×224×3	7	3750
visual-block-triple-task2-v0	3×224×224×3	7	3750
visual-block-quadruple-task1-v0	3×224×224×3	7	5000
visual-block-quadruple-task2-v0	3×224×224×3	7	5000
visual-block-quadruple-task3-v0	3×224×224×3	7	5000
visual-chamber-easy-task1-v0	3×224×224×3	7	4000
visual-chamber-easy-task2-v0	3×224×224×3	7	4000
visual-chamber-easy-task3-v0	3×224×224×3	7	4000
visual-chamber-medium-task1-v0	3×224×224×3	7	5000
visual-chamber-medium-task2-v0	3×224×224×3	7	5000
visual-chamber-hard-task1-v0	3×224×224×3	7	5500
visual-chamber-hard-task2-v0	3×224×224×3	7	5500
visual-switch-3x3-task1-v0	3×224×224×3	7	1500
visual-switch-3x3-task2-v0	3×224×224×3	7	1500
visual-switch-4x4-task1-v0	3×224×224×3	7	1500
visual-switch-4x4-task2-v0	3×224×224×3	7	1500
visual-switch-5x5-task1-v0	3×224×224×3	7	4000
visual-switch-5x5-task2-v0	3×224×224×3	7	4000
visual-hanoi-single-task1-v0	3×224×224×3	7	1000
visual-hanoi-single-task2-v0	3×224×224×3	7	1000
visual-hanoi-double-task1-v0	3×224×224×3	7	3000
visual-hanoi-double-task2-v0	3×224×224×3	7	3000
visual-hanoi-triple-task1-v0	3×224×224×3	7	7000
visual-hanoi-triple-task2-v0	3×224×224×3	7	7000
visual-bowling-task1-v0	3×224×224×3	7	500
visual-block-lite-single-task1-v0	3×224×224×3	7	200
visual-block-lite-double-task1-v0	3×224×224×3	7	400
visual-block-lite-double-task2-v0	3×224×224×3	7	400
visual-block-lite-triple-task1-v0	3×224×224×3	7	600
visual-block-lite-triple-task2-v0	3×224×224×3	7	600
visual-block-lite-quadruple-task1-v0	3×224×224×3	7	800
visual-block-lite-quadruple-task2-v0	3×224×224×3	7	800
visual-block-lite-quadruple-task3-v0	3×224×224×3	7	800
visual-chamber-lite-easy-task1-v0	3×224×224×3	7	500
visual-chamber-lite-easy-task2-v0	3×224×224×3	7	500
visual-chamber-lite-easy-task3-v0	3×224×224×3	7	500
visual-chamber-lite-medium-task1-v0	3×224×224×3	7	600
visual-chamber-lite-medium-task2-v0	3×224×224×3	7	600
visual-chamber-lite-hard-task1-v0	3×224×224×3	7	700
visual-chamber-lite-hard-task2-v0	3×224×224×3	7	700
visual-switch-lite-3x3-task1-v0	3×224×224×3	7	300
visual-switch-lite-3x3-task2-v0	3×224×224×3	7	300
visual-switch-lite-4x4-task1-v0	3×224×224×3	7	300
visual-switch-lite-4x4-task2-v0	3×224×224×3	7	300
visual-switch-lite-5x5-task1-v0	3×224×224×3	7	800
visual-switch-lite-5x5-task2-v0	3×224×224×3	7	800
visual-hanoi-lite-single-task1-v0	3×224×224×3	7	300
visual-hanoi-lite-single-task2-v0	3×224×224×3	7	300
visual-hanoi-lite-double-task1-v0	3×224×224×3	7	900
visual-hanoi-lite-double-task2-v0	3×224×224×3	7	900
visual-hanoi-lite-triple-task1-v0	3×224×224×3	7	2100
visual-hanoi-lite-triple-task2-v0	3×224×224×3	7	2100

Table 5: **Dataset specifications.**

Task	Episodes	Transitions	Mean episode length	Success rate	Throughput
block-single-task1-v0	10K	7.3M	729.3	88	1.51
block-double-task1-v0	30K	53.4M	1778.9	79	1.11
block-double-task2-v0	30K	45.3M	1511.5	88	1.45
block-triple-task1-v0	100K	287.3M	2873.0	72	0.94
block-triple-task2-v0	100K	279.0M	2790.5	72	0.96
block-quadruple-task1-v0	100K	405.7M	4056.8	65	0.80
block-quadruple-task2-v0	100K	403.3M	4033.1	60	0.75
block-quadruple-task3-v0	100K	452.8M	4527.7	33	0.37
chamber-easy-task1-v0	10K	19.7M	1968.5	96	1.96
chamber-easy-task2-v0	10K	33.0M	3295.4	65	0.79
chamber-easy-task3-v0	10K	33.8M	3377.6	58	0.69
chamber-medium-task1-v0	30K	128.7M	4288.8	58	0.67
chamber-medium-task2-v0	30K	134.0M	4467.8	44	0.49
chamber-hard-task1-v0	100K	502.8M	5028.0	42	0.46
chamber-hard-task2-v0	100K	490.3M	4902.9	42	0.47
switch-3x3-task1-v0	10K	14.3M	1433.9	11	0.11
switch-3x3-task2-v0	10K	13.5M	1352.9	21	0.24
switch-4x4-task1-v0	30K	43.6M	1454.1	9	0.10
switch-4x4-task2-v0	30K	43.9M	1463.6	8	0.08
switch-5x5-task1-v0	100K	395.1M	3951.0	5	0.05
switch-5x5-task2-v0	100K	395.7M	3956.7	4	0.05
hanoi-single-task1-v0	10K	7.1M	706.8	100	1.41
hanoi-single-task2-v0	10K	7.1M	712.1	99	1.40
hanoi-double-task1-v0	30K	77.9M	2595.3	71	0.82
hanoi-double-task2-v0	30K	78.5M	2615.1	69	0.79
hanoi-triple-task1-v0	100K	618.2M	6181.6	58	0.65
hanoi-triple-task2-v0	100K	618.9M	6188.9	53	0.60
bowling-task1-v0	10K	5.0M	500.0	30	-
block-lite-single-task1-v0	10K	0.8M	77.2	97	2.50
block-lite-double-task1-v0	30K	5.8M	193.7	95	1.96
block-lite-double-task2-v0	30K	4.8M	159.6	98	2.45
block-lite-triple-task1-v0	100K	31.6M	315.7	94	1.79
block-lite-triple-task2-v0	100K	32.1M	321.1	89	1.66
block-lite-quadruple-task1-v0	100K	45.5M	454.8	93	1.63
block-lite-quadruple-task2-v0	100K	48.9M	489.0	80	1.30
block-lite-quadruple-task3-v0	100K	62.9M	628.7	52	0.66
chamber-lite-easy-task1-v0	10K	2.1M	210.5	96	2.28
chamber-lite-easy-task2-v0	10K	3.6M	359.0	82	1.14
chamber-lite-easy-task3-v0	10K	3.8M	377.6	72	0.96
chamber-lite-medium-task1-v0	30K	14.0M	467.1	74	0.95
chamber-lite-medium-task2-v0	30K	15.1M	502.2	59	0.70
chamber-lite-hard-task1-v0	100K	57.8M	577.8	65	0.78
chamber-lite-hard-task2-v0	100K	56.9M	569.4	60	0.73
switch-lite-3x3-task1-v0	10K	2.7M	267.6	22	0.25
switch-lite-3x3-task2-v0	10K	2.4M	243.9	34	0.42
switch-lite-4x4-task1-v0	30K	8.1M	271.7	21	0.24
switch-lite-4x4-task2-v0	30K	8.3M	276.7	18	0.20
switch-lite-5x5-task1-v0	100K	74.8M	748.0	18	0.19
switch-lite-5x5-task2-v0	100K	75.5M	754.5	16	0.17
hanoi-lite-single-task1-v0	10K	1.4M	141.5	100	2.12
hanoi-lite-single-task2-v0	10K	1.5M	145.9	97	1.99
hanoi-lite-double-task1-v0	30K	15.2M	506.7	94	1.68
hanoi-lite-double-task2-v0	30K	15.5M	516.4	89	1.56
hanoi-lite-triple-task1-v0	100K	122.5M	1224.8	75	1.29
hanoi-lite-triple-task2-v0	100K	123.4M	1234.2	70	1.20

Table 6: **Full benchmark results (success rate).**

Task	BC	FBC	FSQBC
block-single-task1-v0	88 ±0	90 ±1	89 ±0
block-double-task1-v0	51 ±5	60 ±1	72 ±0
block-double-task2-v0	68 ±3	71 ±2	74 ±2
block-triple-task1-v0	1 ±0	2 ±1	29 ±2
block-triple-task2-v0	27 ±2	8 ±1	30 ±8
block-quadruple-task1-v0	0 ±0	0 ±0	0 ±0
block-quadruple-task2-v0	3 ±0	0 ±0	6 ±2
block-quadruple-task3-v0	2 ±0	8 ±3	25 ±2
chamber-easy-task1-v0	65 ±8	73 ±9	88 ±5
chamber-easy-task2-v0	40 ±0	53 ±0	48 ±5
chamber-easy-task3-v0	42 ±0	48 ±6	38 ±4
chamber-medium-task1-v0	7 ±1	12 ±0	27 ±1
chamber-medium-task2-v0	10 ±3	5 ±0	23 ±1
chamber-hard-task1-v0	0 ±0	1 ±1	2 ±0
chamber-hard-task2-v0	1 ±0	0 ±0	8 ±4
switch-3x3-task1-v0	21 ±0	16 ±1	20 ±2
switch-3x3-task2-v0	25 ±3	16 ±0	19 ±3
switch-4x4-task1-v0	8 ±0	5 ±0	6 ±0
switch-4x4-task2-v0	2 ±0	1 ±0	1 ±0
switch-5x5-task1-v0	0 ±0	0 ±0	0 ±0
switch-5x5-task2-v0	0 ±0	0 ±0	0 ±0
hanoi-single-task1-v0	56 ±1	41 ±31	84 ±10
hanoi-single-task2-v0	62 ±1	74 ±0	66 ±2
hanoi-double-task1-v0	23 ±2	21 ±3	20 ±2
hanoi-double-task2-v0	28 ±2	16 ±11	18 ±3
hanoi-triple-task1-v0	18 ±0	4 ±2	1 ±0
hanoi-triple-task2-v0	14 ±1	8 ±1	2 ±0
bowling-task1-v0	39 ±1	35 ±1	32 ±1
block-lite-single-task1-v0	93 ±0	64 ±9	96 ±0
block-lite-double-task1-v0	26 ±6	41 ±2	73 ±2
block-lite-double-task2-v0	46 ±5	53 ±3	74 ±8
block-lite-triple-task1-v0	1 ±1	1 ±1	0 ±0
block-lite-triple-task2-v0	7 ±1	22 ±6	8 ±2
block-lite-quadruple-task1-v0	0 ±0	0 ±0	0 ±0
block-lite-quadruple-task2-v0	0 ±0	0 ±0	0 ±0
block-lite-quadruple-task3-v0	0 ±0	4 ±4	3 ±1
chamber-lite-easy-task1-v0	27 ±2	78 ±9	66 ±4
chamber-lite-easy-task2-v0	11 ±0	35 ±4	22 ±5
chamber-lite-easy-task3-v0	9 ±1	26 ±6	17 ±4
chamber-lite-medium-task1-v0	1 ±0	3 ±1	1 ±0
chamber-lite-medium-task2-v0	0 ±0	2 ±1	0 ±0
chamber-lite-hard-task1-v0	0 ±0	0 ±0	0 ±0
chamber-lite-hard-task2-v0	0 ±0	0 ±0	0 ±0
switch-lite-3x3-task1-v0	2 ±1	6 ±1	1 ±1
switch-lite-3x3-task2-v0	4 ±0	6 ±1	3 ±1
switch-lite-4x4-task1-v0	1 ±0	2 ±1	1 ±0
switch-lite-4x4-task2-v0	1 ±0	1 ±0	0 ±0
switch-lite-5x5-task1-v0	0 ±0	0 ±0	0 ±0
switch-lite-5x5-task2-v0	0 ±0	0 ±0	0 ±0
hanoi-lite-single-task1-v0	59 ±6	9 ±3	84 ±0
hanoi-lite-single-task2-v0	67 ±2	2 ±1	56 ±2
hanoi-lite-double-task1-v0	9 ±2	0 ±0	20 ±1
hanoi-lite-double-task2-v0	0 ±0	0 ±0	0 ±0
hanoi-lite-triple-task1-v0	0 ±0	0 ±0	0 ±0
hanoi-lite-triple-task2-v0	0 ±0	0 ±0	0 ±0

Table 7: **Full benchmark results (throughput).**

Task	BC	FBC	FSQBC
block-single-task1-v0	1.41 ±0.01	1.45 ±0.04	1.40 ±0.00
block-double-task1-v0	0.58 ±0.07	0.71 ±0.02	0.88 ±0.00
block-double-task2-v0	0.95 ±0.05	1.00 ±0.04	1.07 ±0.05
block-triple-task1-v0	0.01 ±0.00	0.02 ±0.01	0.30 ±0.03
block-triple-task2-v0	0.29 ±0.02	0.09 ±0.01	0.33 ±0.10
block-quadruple-task1-v0	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
block-quadruple-task2-v0	0.03 ±0.00	0.00 ±0.00	0.06 ±0.02
block-quadruple-task3-v0	0.02 ±0.00	0.08 ±0.03	0.26 ±0.02
chamber-easy-task1-v0	0.89 ±0.16	1.07 ±0.18	1.54 ±0.15
chamber-easy-task2-v0	0.43 ±0.00	0.61 ±0.01	0.53 ±0.06
chamber-easy-task3-v0	0.46 ±0.01	0.53 ±0.07	0.42 ±0.05
chamber-medium-task1-v0	0.07 ±0.01	0.12 ±0.00	0.28 ±0.01
chamber-medium-task2-v0	0.10 ±0.03	0.05 ±0.00	0.24 ±0.01
chamber-hard-task1-v0	0.00 ±0.00	0.01 ±0.01	0.02 ±0.00
chamber-hard-task2-v0	0.01 ±0.00	0.00 ±0.00	0.08 ±0.05
switch-3x3-task1-v0	0.23 ±0.00	0.17 ±0.01	0.22 ±0.02
switch-3x3-task2-v0	0.27 ±0.03	0.17 ±0.00	0.21 ±0.03
switch-4x4-task1-v0	0.08 ±0.00	0.05 ±0.00	0.06 ±0.00
switch-4x4-task2-v0	0.02 ±0.00	0.01 ±0.00	0.01 ±0.00
switch-5x5-task1-v0	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
switch-5x5-task2-v0	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
hanoi-single-task1-v0	0.63 ±0.02	0.48 ±0.38	1.08 ±0.16
hanoi-single-task2-v0	0.75 ±0.02	0.93 ±0.00	0.81 ±0.03
hanoi-double-task1-v0	0.24 ±0.03	0.22 ±0.03	0.21 ±0.02
hanoi-double-task2-v0	0.30 ±0.02	0.17 ±0.12	0.19 ±0.03
hanoi-triple-task1-v0	0.21 ±0.00	0.05 ±0.02	0.01 ±0.00
hanoi-triple-task2-v0	0.15 ±0.01	0.08 ±0.01	0.02 ±0.00
block-lite-single-task1-v0	1.93 ±0.00	0.93 ±0.18	2.07 ±0.02
block-lite-double-task1-v0	0.28 ±0.07	0.48 ±0.02	1.02 ±0.07
block-lite-double-task2-v0	0.59 ±0.09	0.72 ±0.07	1.19 ±0.21
block-lite-triple-task1-v0	0.01 ±0.01	0.01 ±0.01	0.00 ±0.00
block-lite-triple-task2-v0	0.07 ±0.01	0.24 ±0.07	0.08 ±0.02
block-lite-quadruple-task1-v0	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
block-lite-quadruple-task2-v0	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
block-lite-quadruple-task3-v0	0.00 ±0.00	0.05 ±0.04	0.03 ±0.01
chamber-lite-easy-task1-v0	0.31 ±0.03	1.24 ±0.25	0.96 ±0.08
chamber-lite-easy-task2-v0	0.11 ±0.00	0.39 ±0.04	0.23 ±0.05
chamber-lite-easy-task3-v0	0.09 ±0.01	0.28 ±0.07	0.18 ±0.04
chamber-lite-medium-task1-v0	0.01 ±0.00	0.03 ±0.01	0.01 ±0.00
chamber-lite-medium-task2-v0	0.00 ±0.00	0.02 ±0.01	0.00 ±0.00
chamber-lite-hard-task1-v0	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
chamber-lite-hard-task2-v0	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
switch-lite-3x3-task1-v0	0.02 ±0.01	0.06 ±0.02	0.01 ±0.01
switch-lite-3x3-task2-v0	0.04 ±0.00	0.07 ±0.01	0.04 ±0.01
switch-lite-4x4-task1-v0	0.01 ±0.00	0.02 ±0.01	0.01 ±0.00
switch-lite-4x4-task2-v0	0.01 ±0.00	0.01 ±0.00	0.00 ±0.00
switch-lite-5x5-task1-v0	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
switch-lite-5x5-task2-v0	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
hanoi-lite-single-task1-v0	0.85 ±0.14	0.11 ±0.04	1.49 ±0.00
hanoi-lite-single-task2-v0	1.02 ±0.05	0.02 ±0.01	0.80 ±0.03
hanoi-lite-double-task1-v0	0.10 ±0.02	0.00 ±0.00	0.23 ±0.01
hanoi-lite-double-task2-v0	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
hanoi-lite-triple-task1-v0	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00
hanoi-lite-triple-task2-v0	0.00 ±0.00	0.00 ±0.00	0.00 ±0.00