Bidirectional Decoding: Robots Should Plan Long And Adapt

Jubayer Ibn Hamid

This blog discusses the paper "Bidirectional Decoding: Improving Action Chunking via Closed-Loop Resampling" with an emphasis on the theoretical analysis and the method.

The past year has seen a tremendous increase in efforts to scale up robotic data collection [3, 4, 2]. Such large datasets of expert demonstrations, coupled with strong behavior cloning algorithms, can help robots learn complex skills, from folding shirts to cooking shrimp [7, 1, 5]. One might ask: What are the challenges that uniquely reveal themselves when robots learn from large datasets?

Large robotic datasets generally encompass a diversity of strategies for every task. They contain demonstrations from a variety of experts with different preferences and proficiency levels—for example, demonstrations from left-handed experts versus right-handed ones, demonstrators that are more adept at a task than others. Once trained on these datasets, multimodal policies [6, 1, 5] will capture a large number of strategies.

The first challenge arises in *choosing* the right strategy to start with, when the robot has already learned multiple strategies. In particular, we would want the robot to prefer the more optimal strategies it has learned and execute them instead of having an almost uniform distribution over all strategies. This is non-trivial; large datasets do not have explicit labels for how good each demonstration is and we would hope that, as we scale up robotic data collection, we would not have to spend efforts to prune/label such datasets. Yet, we want the model to, somehow, be biased towards the "better strategies".

The second challenge is in *execution* of any strategy. We would hope that, for any one rollout, the robot can consistently execute any one of these strategies properly. In other words, once a strategy has been sampled at a time step, the robot needs to commit to that strategy and execute perfectly; it must not hop from one strategy to another. This requires learning temporal dependencies and identifying confounders hiding in the demonstrated trajectories.

Here is an illustration-consider the Push-T task where the agent (the blue circle) needs to push the grey T-shaped block to cover the green-colored target. There are multiple routes

that the agent can take to get to the tail of the block and start pushing it. We want an agent to be able to take any one of these two routes consistently, like the one in Figure 1(left). But Figure 1(right) shows an agent that fails to commit - it often starts taking one route and then switches to the other, leading to jittery trajectories.

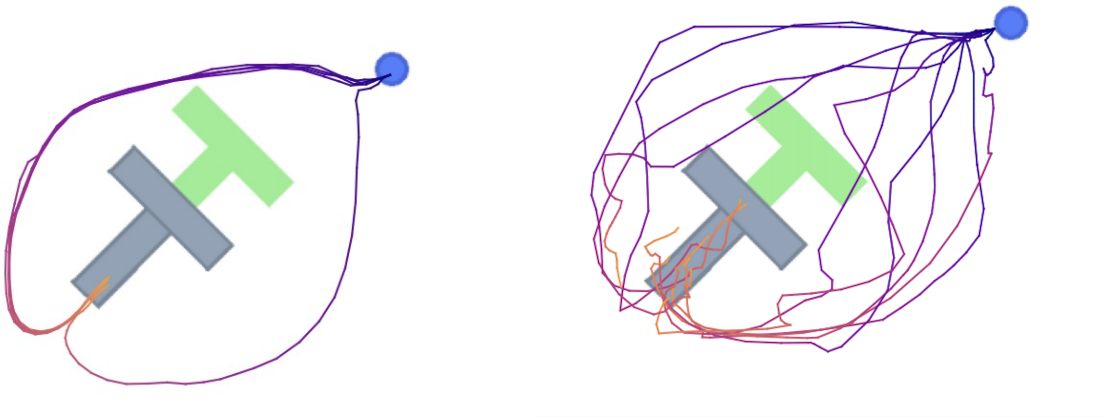

Figure 1: Multimodal policies on the Push-T task.

Therefore, we have a two-pronged challenge. Firstly, we need our robot to learn the *optimal* strategies well. Secondly, we need the robot to *consistently* execute them.

1 Action Chunking

This is not a novel goal by any means. In fact, over the past year, researchers have discovered interesting techniques to achieve some of these goals. One interesting practice has been something called "action chunking". Under this paradigm, at a state s_t, our agent does not only generate the next action. Instead, it models the join distribution of the next p actions $\pi(a_t, a_{t+1}, \ldots, a_{t+p-1}|s_t, s_{t-1}, \cdots, s_{t-c})$. Having modelled this joint probability of the next p actions, the agent executes the next h (where $1 \leq h \leq p$) actions straight-away without replanning at each timestep. We call this an "open-loop operation"–since we do not get feedback from state observation and, instead, "blindly" execute the h actions. In contrast, "closed-loop operations" refers to sampling processes where the agent observes every state at every timestep and plans all over again.

The effects of action chunking are not well-understood. While some works [1, 7] have found

them to be critical, others have found them to be counterproductive at times [6, 5].

We first follow this trail.

2 Understanding action chunking and its inherent trade-offs

Our first goal in our paper is to understand the strengths and weaknesses of action chunking. The main finding is a formalization of how action chunking helps our robot attain greater temporal consistency in its actions. Suppose an agent executes action chunks of length h and suppose this action chunk is generated on the observations of the last c states. Then, we will refer to this agent as a (c, h)-model. In contrast, the expert demonstrator clearly only executes one action at a time, so their action chunk is of length 1 but the action is assumed to be dependent on, at most, the last k observations. We would like to compare between a (c, h)-model and a $(c, h + d)$-model (i.e., an agent with larger action chunks) - in particular, we want to determine which model can imitate the expert distribution better. We illustrate the two models in figure 2.

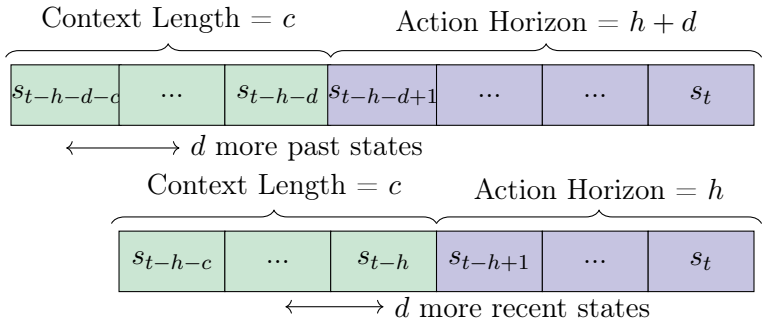

Figure 2: Illustrating the (c, h)-model and the $(c, h + d)$-model. The green and blue cells represent the context and action chunk, respectively

Note that the $(c, h + d)$-learner remembers d more past states while observing d less recent states compared to the (c, h)-learner.

Note that the larger action chunk model has to simulate the states $s_{t-h-d+1:t-h}$, which the smaller action chunk model directly observes. Let the maximum error from simulating these unobserved states wrong be ϵ_f and let the temporal dependency of a_t on these states be α_f. In contrast, the smaller action chunk model has to simulate the states $s_{t-h-d-c:t-h-c-1}$. Let the maximum error from simulating these unobserved states wrong be ϵ_b and let the temporal

dependency of a_t on these states be α_b. A more formal derivation of these terms can be found in the paper.

Lastly, we define stochasticity in the environment. Define the forward probability of being in the ground truth state of the deterministic case to be $P_f := P(S_{t'} = g_{t'}|s_{t'-1}, a_{t'-1})$. In the absence of any noise, the action would deterministically cause the agent to reach this ground truth *i.e.* $P_f = P(S_{t'} = g_{t'}|s_{t'-1}, a_{t'-1}) = 1$ whereas in a noisy environment, $P_f = P(S_{t'} = g_{t'}|s_{t'-1}, a_{t'-1})$ is lower as the entropy increases. Similarly, we define the backward probability of being in the ground truth, unconditional on any action, to be $P_b := P(S_{t'} = g_{t'}|s_{t+1})$. Since this backward probability is not conditioned on any action, this has higher entropy. Notably, in a noisy environment, $P(S_{t'} = s_{t'}|s_{t+1})$ is small for any $s_{t'}$.

Our main finding is the following:

Proposition 1 (Consistency-Reactivity Inequalities)**.** Let $\mathcal{L}$ be a non-linear, convex loss function. Let $\mathcal{S}^+ \subset \{s_{t-k:t}\}$ be the states both the (c, h) and the $(c, h+d)$ models observe and let $\mathcal{S}^- := \{s_{t-k:t}\} \setminus \mathcal{S}^+$. Let $C := \{a_{t-h-d:t-1}\} \cup \mathcal{S}^+$, $G := \{a_t, z_{t-k:t}\} \cup \mathcal{S}^-$. Then, we can bound the expected loss of the $(c, h+d)$-policy and the (c, h)-policy as:

$$\alpha_f - \epsilon_b(1 - P_b^{2d}) \leq \min_{\pi_{h+d}} \mathbb{E}_G \left[\mathcal{L}(\pi_{h+d}, \pi^*)|C\right] - \min_{\pi_h} \mathbb{E}_G \left[\mathcal{L}(\pi_h, \pi^*)|C\right] \leq -\alpha_b + \epsilon_f(1 - P_f^{2d}) \quad (1)$$

This result enables a general comparison of the performance of the two policies. Intuitively, the advantage of each policy stems from the additional information it has access to (*i.e.* α_f for π_h and α_b for π_{h+d}) while the disadvantage is bounded by the divergence arising from simulating missing information incorrectly (*i.e.* $\epsilon_b(1 - P_b^{2d})$ for π_h and $\epsilon_f(1 - P_f^{2d})$ for π_{h+d}).

In near-deterministic environments, simulating the recent state $s_{t-h-d+1:t-h}$ is easy for the larger chunk because it has access to the actions taken during these time steps. Therefore, the larger chunk model has strictly *more* information as it also remembers the distant past states $s_{t-h-d-c:t-h-c-1}$. When the action a_t is temporally dependent over long context, a larger action horizon is going to perform better. This is shown in the following result:

Corollary 2 (Consistency in Deterministic Environments)**.** In a highly deterministic environment, if a_t is temporally dependent on at least one state in $\{s_{t-h-c-d:t-h-c-1}\}$ and ϵ_f is finite,

$$\min_{\pi_{h+d}} \mathbb{E}_G \left[\mathcal{L}(\pi_{h+d}, \pi^*)|C\right] < \min_{\pi_h} \mathbb{E}_G \left[\mathcal{L}(\pi_h, \pi^*)|C\right] \quad (2)$$

In stochastic environments, simulating the recent states $s_{t-h-d+1:t-h}$ is challenging for the larger chunk, despite knowing the actions taken during these time steps, due to significant environmental noise. Furthermore, it is likely that temporal dependencies decrease over time steps *i.e.* actions depend more on recent past states than distant past ones. In this case, action chunking is likely to be detrimental. This is shown in the following result:

Corollary 3 (Reactivity in Stochastic Environments)**.** In a highly stochastic environment, if temporal dependency decreases over the number of time steps *i.e.* $\alpha_f > \epsilon_b$, then

$$\min_{\pi_h} \mathbb{E}_G \left[\mathcal{L}(\pi_h, \pi^*)|C\right] < \min_{\pi_{h+d}} \mathbb{E}_G \left[\mathcal{L}(\pi_{h+d}, \pi^*)|C\right] \tag{3}$$

This suggests why action chunking has performed so well in many of the static environments. Note that some of the experiments were carried out in dynamic environments too but the injection was noise was not high frequency i.e., the pizza sauce spreading experiment [1] used for diffusion policies where the pizza was moved fairly infrequently and the task itself did not require high precision. In contrast, we found that in environments where the noise was injected much more frequently and the robot needed more precise control, action chunking significantly degraded performance.

Corollary 3 can be made even stronger if we consider $h = 1$ *i.e.* if we compare a strictly closed-loop policy and an open-loop one. In this case, under the same conditions, the closed-loop policy will have lower divergence across the entire trajectory, not just at the time t we considered in the prior section.

Corollary 4. In a highly stochastic environment, if the context length c is long enough such that $\alpha_f > \epsilon_b$ at all time steps t, then divergence between the closed-loop policy $\pi_{(c,1)}$ over the full trajectory of T steps is lower than that between the open-loop policy $\pi_{(c,1+d)}$ and the expert.

3 Improving action chunking

One way forward from here could be to always select the length of action chunks on the basis of how stochastic the environment is and the length of temporal dependencies in demonstrations. However, both of these are largely intractable. One could choose action chunking in dynamic environments only but it's certainly possible that in many of the timesteps, there is no noise in which case we would want the temporal consistency action chunking brings whereas in other timesteps, where we encounter noise, we would want higher reactivity.

Our approach to solving this problem was to think about increasing temporal consistency within closed-loop action chunking. In other words, we maximize reactivity by using closed-loop operations and *then* think about how to increase temporal consistency.

3.1 Backward Coherence

The first step is to think about being consistent with a strategy we selected in a previous timestep. Going back to the task of picking up a jug of water, if the robot has decided to pick it up with its right hand, we would want it to continue to do so instead of randomly deciding midway that it should use its left hand. One simple yet effective way to do this is to use backward coherence: suppose we are at time t and we want to sample action a_t. One way to do this is to first sample an action chunk $(a_t, \cdots, a_{t+p})$ that is "coherent" with action chunks taken previously. To do so, we first sample n action chunks at timestep t and then select the one that is most coherent with our previous action chunk. Once we have done so, we only take the first action in the chosen chunk.

More formally, we use the action chunk selected at the previous time, $a_{t-1}^{(t-1)}, \cdots, a_{t+p-1}^{(t-1)}$, and select the action chunk that minimize the weighted sum of Euclidean distances across the $p-1$ overlapping steps:

$$\mathcal{L}_B(a^{(t)}) = \sum_{\tau=0}^{p-1} \rho^\tau \left\| a_{t+\tau}^{(t)} - a_{t+\tau}^{(t-1)} \right\|_2 . \tag{4}$$

Here, ρ is a decay hyperparameter to account for the uncertainty in longer-horizon action predictions (e.g., $\rho = 0.9$).

We found other approaches like exponential moving average (EMA) to be fairly powerful as well. However, the issue with simply averaging is that the resulting action could end up being one that is very low probability in our policy distribution. Our method, instead, takes an "arg min" over actions we sampled from the policy so we only take an action that the policy does choose and is consistent. Empirically, we observed that averaging leads to loss of precision [2].

3.2 Forward Contrast

The backward coherence incentivizes our policy to take actions that are *consistent* with previous actions. However, consistency is only one part of the problem. Consistently executing suboptimal trajectories would continue to be a problem. We would want our robot to, generally, select more optimal strategies and then stick to it.

The notion of "optimal strategies" is subjective. One aspect of it is the observation that expert demonstrators tend to make long-term plans and so, we would want our robot to select more long-term strategies too. We wouldn't want it to select a trajectory to pick up the jug of water only to find out after 10 steps that there is an obstruction on this trajectory - it should plan ahead to avoid that obstruction in the first place. One naive approach for

this could be to just train policies that are exceptionally long-horizon. However, that is also exceptionally data and compute-expensive.

A simple, yet effective approach to do this is to use contrastive sampling, fairly similar to contrastive decoding in NLP. we introduce a forward contrast objective to identify and reject these suboptimal plans. At the core of the forward contrast is to compare each candidate plan with two sets of reference samples: one set from a stronger policy and the other set from a weaker one. We use a well-trained model built with a long prediction horizon as the stronger policy, and a model with a shorter prediction horizon or from an early underfitting checkpoint as the weaker policy. Intuitively, either choice of the weaker policy cannot capture long-term planning as effectively as the stronger one. Our forward contrast loss is thus framed as minimizing the average distance between a candidate plan and a set of positive samples while maximizing its average distance from the negative ones,

$$\mathcal{L}_F = \sum_{a^+ \in \mathcal{A}^+} \sum_{\tau=0}^{p'} \left\| a_{t+\tau}^{(t)} - a_{t+\tau}^{+} \right\|_2 - \sum_{a^- \in \mathcal{A}^-} \sum_{\tau=0}^{p'} \left\| a_{t+\tau}^{(t)} - a_{t+\tau}^{-} \right\|_2, \tag{5}$$

where $\mathcal{A}^+ = \mathcal{A} \setminus \{a\}$ is the positive set predicted by the strong policy π, $\mathcal{A}^-$ is the negative set predicted by the weaker one π', and p' denotes the shortest prediction horizon between the two policies.

3.3 Bidirectional Decoding (BID)

Putting it all together, we have our final sampling algorithm. Suppose, we have a generative policy with context length c and prediction horizon l and an action chunk sampled at time t

$$a \sim \pi_\theta(a_t, a_{t+1}, \cdots, a_{t+l} | s_{t-c}, s_{t-c+1}, \cdots, s_t) \tag{6}$$

We cast the problem of closed-loop action chunking as searching for the optimal action among a batch of plans sampled at each time step,

$$a^* = \arg\min_{a \in \mathcal{A}} \mathcal{L}_B(a) + \mathcal{L}_F(a), \tag{7}$$

where $\mathcal{A}$ is the set of sampled action chunks, $\mathcal{L}_B$ and $\mathcal{L}_F$ are the two criteria described above.

4 Experiments

We go over some of the experiments that we conducted to test BID. We want to **(1) validate our theory i.e., test whether closed-loop operations are truly better in stochastic environments (2) test whether BID outperforms vanilla closed-loop operations in both static and noisy environments.**

4.1 Simulation Experiments

We first test BID on Vector-Quantized Behavior Transformers [4] in the Push-T task [1]. The results are summarized in Table 1. We observe that in the static setting, open-loop operations tend to be better than closed-loop operations, as suggested by Corollary 2. BID still outperforms Vanilla open-loop. However, in the noisier settings, closed-loop operations always seem to be the best choice wherein BID gains a significant performance boost compared to the vanilla alternatives.

BID achieves significant improvement compared to other closed-loop operations even in static settings. We tested BID on Diffusion Policies [1] and compared against 4 baseline methods and found that BID is significantly more robust.

Noise	0.0	1.0	1.5
Vanilla Closed-Loop	52.0	50.4	44.2
BID Closed-Loop	**56.6**	**54.8**	**54.4**
Vanilla Open-Loop	61.0	39.0	19.4
BID Open-Loop	**65.2**	39.8	21.4

Table 1: The success rate of different methods for closed-loop and open-loop operations of VQ-BeT. Results are averaged over 500 episodes on the Push-T task.

Method	Push T	Lift	Can	Square	Transport	ToolHang	Kitchen	Average
Vanilla	0.78	0.62	0.89	0.74	0.43	0.50	0.22	0.60
Lowvar	0.79	0.54	0.91	0.75	0.52	0.52	0.25	0.61
Warmstart	0.79	0.53	0.92	0.78	0.47	0.51	0.27	0.61
EMA	0.83	0.77	0.92	0.75	0.59	0.46	0.51	0.69
BID (ours)	**0.85**	**0.91**	**0.96**	**0.79**	**0.62**	**0.56**	**0.60**	**0.76**

Table 2: Comparison of different methods for the closed-loop operation of diffusion policies. Evaluations are based on the mean score over 100 episodes in the Push-T, RoboMimic, and 4-Object Franka Kitchen tasks.

Next, we test BID implemented on top of diffusion policies. We tested it on three simulation benchmarks, including Push-T, RoboMimic, and Franka Kitchen. We compare against other closed-loop sampling methods, the results of which are summarized in Table 2. Critically, we find BID to be much more robust than other methods across different tasks.

	Dynamic	Static
Vanilla Open-Loop	0.10	**0.95**
Vanilla Closed-Loop	0.20	**0.90**
EMA	0.45	0.80
BID (ours)	**0.90**	**0.95**

Table 3: Success rates of different sampling methods on VQ-BeT for cup rearrangement task.

4.2 Real-World Experiments

We tested BID on two real-world experiments in dynamic environments. For both, we deployed BID on top of a pre-trained diffusion policy.

For the first task, we consider a task where the robot is to deliver an object held in its gripper into a cup held by a human. To introduce stochasticity, the cup is moved while the robot completes the task. For both the static and the dynamic settings, we see that BID gains almost 2x the success rate compared to vanilla closed-loop operations, as shown in Figure 3.

For the second experiment, we consider a cup rearrangement task. The robot must pick up a cup that is moved while it is completing the task. After that, the robot must place the cup on a nearby saucer. We use the public checkpoint from UMI [2], without any further fine-tuning, and compare BID with vanilla (both open and closed-loop) and EMA. The results are summarized in Table 4. We observed that open-loop operations degrade very significantly in dynamic settings where BID outperforms all other baselines by at least two times. In static settings, we observe that BID matches the peformance of baseline methods.

The main issue we observed with respect to open-loop operations is still the inability to react to stochasticity. As the videos on our project website demonstrate, the robot often closes/opens the gripper before it reaches target state. Closed-loop random sampling has higher reactivity but severely lacks coherence leading to extremely (and often dangerously) jittery behavior. EMA tends to have high reactivity but lacks precision; this is because naive averaging stops it from committing to any strategy especially in a dynamic environment especially at timesteps when the actions must change fairly fast.

5 Discussion on Context Length

Firstly, I want to stress that the theoretical analysis relies on the assumption that the context length is small compared to the expert's memory horizon. In practice, context length in robotics is small. For example, diffusion policies use context length 2 whereas VQ-BeT use 5.

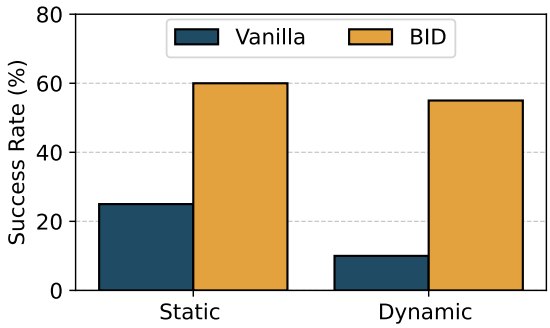

Figure 3: Success rate of object delivery. Each method-setting is evaluated across 20 episodes. BID achieves much higher success rate than the vanilla baseline, effectively handling the diverse demonstrations and dynamic target.

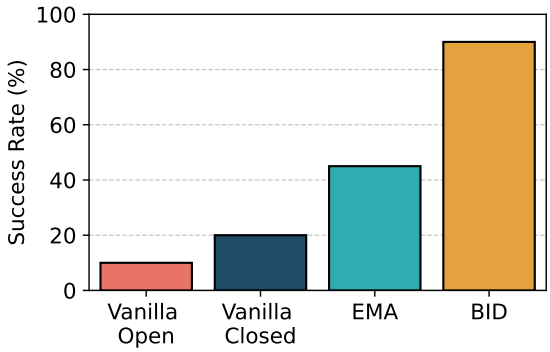

Figure 4: Success rate of cup replacement in the dynamic setting. Each method is evaluated across 20 episodes. Existing methods degrade substantially under slow cup movements, whereas BID retains a strong performance.

Part of the reason for this is that, with insufficient data, long-context models often learn spurious correlations as opposed to the true temporal dependencies. I believe that as robotic data scales up, we can, more comfortably, extend context length and that would be beneficial. This can be easily seen from the following, *extremely* intuitive proposition in the paper:

Proposition 5 (Context is valuable)**.** Let $\mathcal{L}$ be a non-linear, convex function. Let $c < k$. Let $G := \{a_t, s_{t-k:t-c-1}, z_{t-k:t}\}$ and let $C := \{s_{t-c:t}, a_{t-1}\}$. Then,

$$\min_{\pi_{(c+1,h)} \in X_{+,+}} \mathbb{E}_G \left[\mathcal{L}(\pi_{(c+1,h)}, \pi^*) \middle| C \right] \leq \min_{\pi_{(c,h)} \in X_{-,-}} \mathbb{E}_G \left[\mathcal{L}(\pi_{(c,h)}, \pi^*) \middle| C \right].$$

However, I also believe that it is non-trivial to ask *how much* such data needs to scale up; real world robotic tasks are complex and it is possible that the scale of data needs to be significantly larger than that of, say, NLP.

In that realm, it is possible that action chunking itself would not be necessary at all. The optimal learner, after all, is still a $(k, 1)$-policy. However, in that realm, contrastive sampling should still be a powerful method in improving the *quality* of strategies executed.

6 Final Remarks

While test-time sampling has been explored and continues to be so in NLP, it is a large under-explored yet high potential field of research in robotics. Powerful methods in this field

should be heavily generalizable to a broad class of policies and be robust to environment stochasticity.

References

[1] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Robotics: Science and Systems XIX*. Robotics: Science and Systems Foundation, July 2023.

[2] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.

[3] Open X.-Embodiment Collaboration, Abhishek Padalkar, Acorn Pooley, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Animesh Garg, Anthony Brohan, Antonin Raffin, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Brian Ichter, Cewu Lu, Charles Xu, Chelsea Finn, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Chuer Pan, Chuyuan Fu, Coline Devin, Danny Driess, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Federico Ceola, Fei Xia, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Giulio Schiavi, Gregory Kahn, Hao Su, Hao-Shu Fang, Haochen Shi, Heni Ben Amor, Henrik I. Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jan Peters, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jiajun Wu, Jialin Wu, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jitendra Malik, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Joseph J. Lim, João Silvério, Junhyek Han, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Zhang, Krishan Rana, Krishnan Srinivasan, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Lionel Ott, Lisa Lee, Masayoshi Tomizuka, Max Spero, Maximilian Du, Michael Ahn, Mingtong Zhang, Mingyu Ding, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J. Joshi, Niko Suenderhauf, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Pannag R. Sanketi, Paul Wohlhart, Peng Xu, Pierre Sermanet, Priya Sundaresan, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shuran Song, Sichun Xu, Siddhant Haldar, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Sudeep Dasari, Suneel Belkhale, Takayuki Osa, Tatsuya Harada,

Tatsuya Matsushima, Ted Xiao, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xuanlin Li, Yao Lu, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zhuo Xu, and Zichen Jeff Cui. Open X-Embodiment: Robotic Learning Datasets and RT-X Models, December 2023.

[4] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R. Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J. Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. DROID: A Large-Scale In-The-Wild Robot Manipulation Dataset, March 2024.

[5] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior Generation with Latent Actions, March 2024.

[6] Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.

[7] 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*, April 2023.