FBC: Filtered Behavior Cloning for Self-Training Vision–Language–Action Policies

Jeffrey Wei
Department of Computer Science
Yale University
jeffrey.wei@yale.edu

Phuc Duong
Department of Computer Science
Yale University
phuc.duong@yale.edu

Kunwoo Min
Department of Computer Science
Yale University
kunwoo.min@yale.edu

Abstract

Vision-Language-Action (VLA) policies often require task and domain-specific fine-tuning to achieve strong manipulation performance, yet many pipelines lack a simple mechanism for continual improvement from deployment-time interaction. We propose **Filtered Behavior Cloning (FBC)**, a lightweight self-training recipe that executes a pretrained policy, filters its rollouts to retain only successful episodes, and fine-tunes on these self-generated demonstrations using parameter-efficient LoRA updates. Using the Pi0.5-LIBERO checkpoint and evaluating on LIBERO-90, FBC yields measurable gains in overall success rate and improves performance on a majority of non-trivial tasks under a constrained rollout budget. Our results suggest that success-filtered self-training is a practical and scalable primitive for refining large VLA policies, motivating future work that increases self-training and adds safeguards to prevent over-specialization under repeated self-training. Our code can be found here. Our dataset and model checkpoint can be found here.

1 Introduction

Vision–Language–Action (VLA) models have recently emerged as a promising paradigm for robotic manipulation: a single policy network conditions on visual observations and natural-language instructions to generate motor actions. In practice, however, strong performance in a target environment typically requires task- and domain-specific fine-tuning, even when the underlying vision–language backbone is pretrained on large-scale data. This creates a gap between (i) broad pretrained priors and (ii) the adaptive, iterative learning behavior that characterizes how humans acquire skills through repeated attempts.

A central missing ingredient in many current VLA training pipelines is an explicit mechanism for *self-improvement from interaction*, or more generally, *continual learning*. Once a policy is deployed, it will naturally generate a mixture of successful and failed rollouts. Humans treat this stream of experience as informative: successful executions reinforce stable strategies, while failures shape future attempts. By contrast, modern VLAs are often trained once (or in a small number of offline fine-tuning stages) and then evaluated, with limited emphasis on closing the loop between deployment-time trajectories and subsequent adaptation. There is no self correction method that guides the model to "remember" its successes to guide future trials.

Reinforcement learning (RL) provides a principled framework for learning from trial-and-error feedback, but it is frequently expensive in robotics settings due to environment interaction costs, long-

Preprint. Under review.

horizon credit assignment, and engineering complexity. At the same time, imitation-style objectives can be highly data-efficient when demonstrations are available, because they directly optimize for reproducing behaviors that are known to work. This motivates a simple question: can we obtain a practical form of continual adaptation for VLA policies by *reusing the policy's own successful behavior as additional training signal*, without introducing the full machinery of online RL?

In this work, we propose **Filtered Behavior Cloning** (FBC), a self-training recipe based on learning from the model's own interactions. We start from a pretrained and LIBERO-finetuned policy checkpoint (Pi0.5-LIBERO) in the `openpi` framework. We then execute this policy on the LIBERO-90 task suite, collecting multiple trials per task. Because LIBERO-90 spans diverse manipulation skills (including spatial reasoning, object-centric manipulation, and goal-directed behaviors) and specifies tasks through natural language instructions, it provides a natural setting to probe whether a VLA policy can improve by iteratively learning from its own interaction data. We show that this simple self-training loop can improve task performance by turning the policy's own successful rollouts into additional demonstrations, allowing the model to refine its action distribution on tasks it can partially solve while preserving the pretrained vision–language priors.

The remainder of the paper is organized as follows. Section 2 reviews related work on VLAs, self-imitation, and continual adaptation in robotics. Section 3 describes our data collection and filtering pipeline, dataset representation, and LoRA fine-tuning procedure. Section 5 discusses limitations and outlines directions for extending filtered self-training toward more general online learning.

2 Related Works

Vision-Language-Action (VLA) and foundation robot policies. A major driver of recent progress in robotic generalization is scaling up models that jointly condition on vision and language while producing continuous control actions. Early work combined large language models with learned affordances and value functions for instruction following [Ahn et al., 2022]. Subsequent systems moved toward end-to-end transformer policies trained on large, diverse robot datasets [Brohan et al., 2022] and then demonstrated that web-scale vision-language pretraining can substantially improve semantic reasoning and generalization when transferred into robot control [Brohan et al., 2023]. Related efforts integrate embodied sensor streams directly into large multimodal models to support broad downstream embodied reasoning and control [Driess et al., 2023]. Most closely aligned with our setup are recent open foundation-policy efforts that couple a pre-trained VLM backbone with an action-generation module suitable for dexterous control, enabling both strong zero-shot behavior and efficient task adaptation via fine-tuning [Black et al., 2024, 2025]. In parallel, open-source VLA releases emphasize reproducibility and practical fine-tuning recipes, including low-rank adaptation, to make large VLA policies accessible to the community [Kim et al., 2025].

Lifelong learning benchmarks and evaluation suites. Evaluating continual or lifelong improvement requires task suites that expose distribution shift across tasks while keeping evaluation systematic. LIBERO formalizes this setting through a standardized suite of language-conditioned manipulation tasks designed to study knowledge transfer and continual learning behavior in robotics [Liu et al., 2023]. Our experiments use LIBERO-90 as a broad task pool for rollouts and post-hoc filtering, aligning with the benchmark's goal of assessing cross-task generalization and adaptation.

Learning from demonstrations, self-generated data, and offline datasets. Behavior cloning (BC) is a standard and effective approach for policy learning when high-quality demonstrations exist, but it is sensitive to covariate shift under rollout. Dataset aggregation methods such as DAgger address this by iteratively collecting data under the learner's state distribution and retraining on aggregated datasets [Ross et al., 2011]. Our approach can be viewed as a lightweight, self-improving variant in which the policy generates rollouts and we selectively retain successful trajectories for subsequent supervised updates, conceptually related to exploiting an agent's own high-return experience as training signal [Oh et al., 2018]. This also connects to offline reinforcement learning, which aims to learn improved policies from fixed logs without additional environment interaction; representative approaches include conservative value learning objectives [Kumar et al., 2020] and implicit policy improvement paired with advantage-weighted cloning [Kostrikov et al., 2021]. In contrast to full offline RL optimization, we focus on a simple filtered-BC procedure that is inexpensive to implement and compatible with large VLA backbones and short project timelines.

Parameter-efficient adaptation. Adapting billion-parameter backbones with limited robot data is often constrained by memory and overfitting risk. Low-Rank Adaptation (LoRA) freezes the base model and trains small rank-decomposed updates in selected layers, enabling efficient fine-tuning with minimal additional parameters [Hu et al., 2021]. This strategy has become a practical default for adapting large multimodal backbones, including recent VLA releases [Kim et al., 2025], and we adopt it to specialize a strong pre-trained checkpoint to the filtered LIBERO rollouts with limited compute.

3 Methods and Data

We study a simple self-training loop for VLA policies based on *Filtered Behavior Cloning* (FBC). Starting from a strong pretrained checkpoint (Pi0.5-LIBERO) in the `openpi` codebase, we (i) execute the policy on a suite of language-conditioned manipulation tasks, (ii) retain only trajectories that achieve task success, and (iii) fine-tune the policy on the resulting set of successful self-generated demonstrations using parameter-efficient adaptation (LoRA). The goal is to operationalize a lightweight form of continual adaptation without requiring online RL or external human demonstrations beyond the initial checkpoint.

3.1 Base Policy

We initialize from the Pi0.5-LIBERO checkpoint provided by `openpi`, which is a Pi0.5 model variant fine-tuned on LIBERO-format data. The policy is an instruction-conditioned transformer architecture with a frozen vision–language backbone and an action prediction component. We use the standard LIBERO configuration with an action horizon of $H = 10$, i.e., the model predicts a short action chunk conditioned on the current observation and instruction.

3.2 Task Suite and Rollout Collection

We collect interaction data on the LIBERO-90 benchmark suite, which contains 90 language-specified manipulation tasks spanning categories such as spatial reasoning, object manipulation, and goal-oriented behaviors. Each task is specified by a natural-language instruction (e.g., "close the top drawer of the cabinet", "put the black bowl in the top drawer").

For each of the 90 tasks, we run the Pi0.5-LIBERO policy for $K = 10$ independent trials, yielding:

- Total episodes: 900 (90 tasks $\times$ 10 trials per task)
- Successful episodes: 251 (success rate 27.9%)
- Successful frames: $\sim 7.5 \times 10^4$ timesteps across all successful episodes
- Control frequency: 10 Hz

We treat the environment-provided task success indicator as the binary label for each trajectory.

3.3 LeRobot Dataset Representation

To fine-tune using the same data pipeline as standard LIBERO/LeRobot datasets, we store collected rollouts in the LeRobot dataset format. Each timestep consists of:

- **Visual observations:** two RGB streams: a third-person view (`image`) and a wrist-mounted view (`wrist_image`), each at 256×256 resolution.
- **Proprioceptive state:** an 8-dimensional vector (`state`) containing 7-DoF arm joint positions and a gripper state scalar.
- **Actions:** a 7-dimensional continuous vector (`actions`) representing end-effector deltas with a gripper command (LIBERO control convention).
- **Language conditioning:** the task instruction string (`task`).

Episodes are stored with explicit boundaries and associated metadata (task identifier, instruction text, episode length), enabling filtering and per-task analysis. This representation supports direct reuse of LeRobot/OpenPI tooling for normalization and training.

3.4 Success Filtering and Postprocessing

Raw rollouts contain both successful and failed trajectories. Training on failures can teach incorrect behaviors (e.g., near-misses, oscillations, or failure-specific recovery motions) and can degrade imitation-based objectives when the dataset is small. We therefore apply *episode-level filtering*: only trajectories with success label $y = 1$ are retained.

Concretely, we construct a filtered dataset `libero_rollouts_success` containing only the 251 successful episodes. The filtering step also records summary statistics (per-task success counts, episode lengths) to preserve traceability and enable reproducibility.

3.5 Normalization Statistics

We found that recomputing observation/action normalization statistics on our self-collected training set substantially degraded downstream performance, while keeping the original normalization used by the pretrained Pi0.5-LIBERO checkpoint produced the expected behavior. This failure is consistent with a normalization mismatch: our filtered "successful-only" rollouts induce a shifted, lower-variance distribution, so the resulting mean/standard-deviation scaling can distort state/action magnitudes and introduce systematic biases (particularly for delta-style actions). Consequently, we retain the checkpoint-provided normalization scheme for all fine-tuning and evaluation to ensure the model's inputs and outputs remain in the units it was originally trained on.

3.6 Parameter-Efficient Fine-Tuning with LoRA

LoRA formulation. We adapt the pretrained policy using Low-Rank Adaptation (LoRA), which freezes the original weight matrices and injects trainable low-rank updates. For a pretrained weight matrix $W \in \mathbb{R}^{d\times k}$, LoRA parameterizes an update $\Delta W = BA$ where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ with rank $r \ll \min(d, k)$. The effective linear map becomes

$$h = Wx + BAx,$$

where only (A, B) are updated during training. This substantially reduces trainable parameters and GPU memory usage relative to full fine-tuning.

Where LoRA is applied. We apply LoRA adapters to transformer attention projection matrices (query, key, value, and output projections) in both the vision–language and action-expert components, using the LoRA-enabled variants specified by the `openpi` configuration (`gemma_2b_lora` for the backbone and `gemma_300m_lora` for the action expert). We use rank $r = 8$ and a LoRA scaling of $\alpha/r = 2.0$.

Training configuration. We fine-tune on `libero_rollouts_success` using the `openpi` training stack with the following configuration:

- Batch size: 32
- Optimizer: AdamW with gradient norm clipping at 1.0
- Learning rate schedule: cosine decay with 2,000 warmup steps; peak LR 5×10^{-5}
- Number of training steps: 10,000
- Checkpoint initialization: `gs://openpi-assets/checkpoints/pi05_libero/params`

All non-LoRA parameters are frozen via the model's freeze filter, so optimization updates only the low-rank adapters.

Compute. We run fine-tuning on a single NVIDIA H100 GPU. The use of LoRA makes the adaptation feasible with limited compute and reduces the risk of overfitting given the modest number of successful episodes. Our training schema takes around 2-3 hours.

3.7 Filtered Behavior Cloning Objective

Let $\tau = (o_{1:T}, a_{1:T}, \ell)$ denote a trajectory with observations o_t (including vision and state), actions a_t, and language instruction ℓ. Let $y(\tau) \in \{0, 1\}$ be the binary success label. Filtered BC constructs

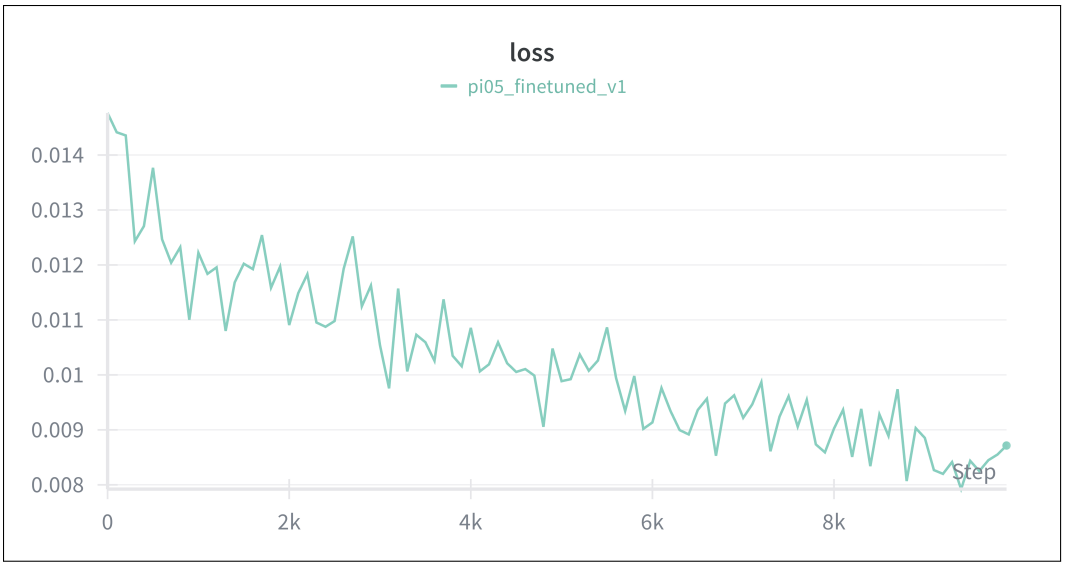

Figure 1: Loss curve of Pi05-LIBERO checkpoint finetuned on successful trajectories. Loss curve generated via `wandb`.

a training set

$$\mathcal{D}_{\text{succ}} = \{\tau \in \mathcal{D} \ : \ y(\tau) = 1\},$$

and minimizes the standard imitation loss over successful trajectories:

$$\mathcal{L}(\theta) = \mathbb{E}_{\tau \sim \mathcal{D}_{\text{succ}}} \left[\sum_{t=1}^{T} - \log \pi_\theta(a_t \mid o_t, \ell) \right],$$

where π_θ is the VLA policy with LoRA parameters θ (base weights frozen). This objective explicitly reinforces behaviors that the current policy has already demonstrated can solve the task suite.

Further Implementation Details. For more implementation details, please refer to our Github repository `README.md` for the for further details

4 Experiments & Results

4.1 Evaluation protocol

We evaluate on LIBERO-90 using the environment-provided binary success indicator. For each task, we run $K = 10$ independent rollouts and compute the per-task success rate as the fraction of successful trials. Aggregating across the 90 tasks yields 900 total rollouts per model. Due to compute budget constraints, we limit evaluation to these 900 episodes.

4.2 Baseline vs. filtered-BC fine-tuning

Using the Pi0.5-LIBERO checkpoint as our baseline, we obtain an overall success rate of 0.2756 across 900 rollouts. After fine-tuning on the policy's own successful rollouts, the success rate increases to 0.2933, corresponding to a relative improvement of approximately 6.4% over baseline.

To better isolate the effect of self-training on tasks where improvement is plausible, we additionally report results on a filtered subset of tasks that excludes (i) tasks already mastered by the baseline (baseline success rate $= 1.0$) and (ii) tasks for which baseline model is completely incompetent at (baseline success rate $= 0$). This focuses analysis on "non-trivial" tasks where the baseline exhibits partial competence and where additional successful experience could reasonably refine the

Table 1: Summary of evaluation results on LIBERO-90. Overall metrics aggregate across 90 tasks.

	Baseline	**FBC**
Total rollouts (LIBERO-90)	900	900
Overall success rate	0.2756	0.2933
Filtered task subset size	24 tasks	
Subset success rate	0.392	0.480
Tasks improved / subset	15 / 24 (62.5%)	

policy. On this subset, the average success rate increases from 0.392 (baseline) to 0.480 (fine-tuned), corresponding to a relative improvement of approximately 22.4% over baseline.

$$\mathcal{T}_{\text{mid}} = \{0 < s_{\text{base}}(t) < 1\}.$$

Figure 2 shows per-task success rates (each computed from 10 trials) for tasks included in the comparison plot. Across the filtered task set ($n = 24$ tasks), the fine-tuned policy improves on 15 tasks (62.5%), while the remaining tasks exhibit comparable performance or regress slightly, highlighting both the potential and the instability of self-training under limited interaction budgets.

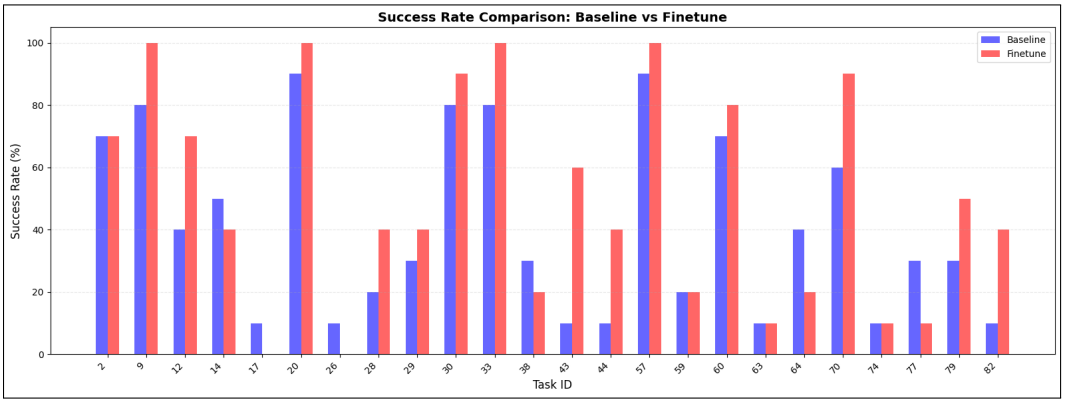

Figure 2: Per-task success rate comparison between the Pi0.5-LIBERO baseline and FBC. Each bar reports success percentage over $K = 10$ rollouts for a given LIBERO-90 task ID.

4.3 Discussion

Given 62.5% of the tasks improving using the finetuned policy, and a total of 22.4% episode success gained in the filtered tasks, we contend that our method works to improve the quality of policy outputs on a task. We believe that this method works as a potential start to a general continual learning method, where one systematically performs offline finetuning of its previous experience.

Beyond the aggregate gains, we speculate that FBC's success primarily results from concentrating the training signal on trajectories that already correspond to correct task completion. In effect, the model generates a distribution of behaviors under deployment, and FBC "distills" this distribution into a lower-variance dataset by retaining only successful episodes. This narrows the action distribution the policy is asked to model, reinforces stable solution strategies, and reduces the influence of failure modes that can dominate small offline datasets. Empirically, we expect the benefit of FBC to scale with the number of collected rollouts: more interaction yields more successful trajectories per task, improving coverage of successful strategies and enabling more reliable specialization.

5 Limitations & Future Direction

At the same time, FBC may exhibit diminishing returns and potential failure modes at larger scale. Because the fine-tuning distribution is self-generated, repeatedly training only on a policy's own

successes can lead to over-specialization, reduced behavioral diversity, and, in the worst case, a form of "model collapse" where the policy becomes brittle outside the narrow manifold it self-reinforces. This motivates future work that scales interaction while incorporating safeguards such as mixing in broader data, retaining a small fraction of informative failures or near-successes, or regularizing updates toward the base checkpoint to preserve generality. This mirrors the model collapse behavior for language models as described in Gerstgrasser et al. [2024] that occurs when a model trains on its own probabilistically generated data.

A second limitation is the reliance on coarse supervision. We treat environment success as a binary episode-level label, which is convenient in simulation but provides little guidance about *why* a rollout failed or which subskills require improvement. In addition, binary success signals may not transfer cleanly to real-world settings where success can be partial, delayed, or hard to instrument. Future work should explore richer evaluators and learning signals, including graded success metrics, subgoal detectors, or learned reward models via a vision-language-judge that can provide denser feedback and enable learning not only from successes but also from near-success and informative failures.

Finally, our approach performs offline fine-tuning after collecting a batch of rollouts rather than updating online. This limits responsiveness and can be interaction-inefficient when success rates are low, since collecting enough successful demonstrations may require substantial environment time. The grand vision for robotics is to provide real time human correction of preferred robot policy behaviors, a dystopia given current online reinforcement learning methods that lack long term memory of instructions.

6 Conclusion

In this paper, we propose **Filtered Behavior Cloning** (FBC), a simple self-training recipe for vision–language–action policies that learns from the model's own interaction data. Starting from a strong Pi0.5-LIBERO checkpoint, we collected rollouts across LIBERO-90, filtered for successful episodes, and fine-tuned the policy using parameter-efficient LoRA updates. Despite a limited interaction budget, this procedure yielded consistent gains overall and more pronounced improvements on a filtered subset of non-trivial tasks, suggesting that even coarse success filtering can provide a useful learning signal for continual refinement.

Looking forward, the most direct next step is to scale the interaction budget: collecting more rollouts should increase the number and diversity of successful trajectories available per task, strengthening the supervision signal that drives FBC. At the same time, scaling raises important questions about stability, over-specialization, and potential model collapse under repeated self-training. Future work should therefore explore higher-budget regimes alongside safeguards such as replay/mixing with broader data, regularization toward the base checkpoint, and incorporating informative near-success failures to preserve robustness while continuing to improve task success.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. URL https://arxiv.org/abs/2204.01691.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. URL https://arxiv.org/abs/2212.06817.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. URL https://arxiv.org/abs/2307.15818.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023. URL https://arxiv.org/abs/2303.03378.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, 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. a vision-language-action model with open-world generalization. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 17–40. PMLR, 2025. URL https://proceedings.mlr.press/v305/black25a.html.

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 *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635. PMLR, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3878–3887. PMLR, 2018. URL https://proceedings.mlr.press/v80/oh18b.html.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020. URL https://arxiv.org/abs/2006.04779.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021. URL https://arxiv.org/abs/2110.06169.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. URL https://arxiv.org/abs/2106.09685.

Matthias Gerstgrasser, Rylan Schaeffer, Apratim Dey, Rafael Rafailov, Henry Sleight, John Hughes, Tomasz Korbak, Rajashree Agrawal, Dhruv Pai, Andrey Gromov, Daniel A. Roberts, Diyi Yang, David L. Donoho, and Sanmi Koyejo. Is model collapse inevitable? breaking the curse of recursion by accumulating real and synthetic data, 2024. URL https://arxiv.org/abs/2404.01413.