Efficient On-the-Fly Robot Dataset Curation with Probabilistic Data Structures

Ryan Diaz
rd88@rice.edu

1 Introduction

Imitation learning from large demonstration datasets has been an increasingly popular strategy to learn robot policies that can perform a wide variety of complex tasks in unstructured environments [1, 2, 3]. Like in any supervised learning problem, the quality of the demonstration dataset can be a significant bottleneck in learning good policies. Recently, large datasets of robot demonstrations have been curated [4, 5, 6, 7] in an effort to train robot policies that can perform a wide variety of tasks. However, when encountering a new task that is out-of-distribution, a human will have to provide a new set of demonstrations to teach the task to the robot. When collecting demonstrations, it can be hard to ensure on-the-fly (i.e. while collecting demonstrations) that each new demonstration provides new information. Repeated demonstrations covering the same states may bias the model and lead to degraded performance, and redundant demonstrations would lead to an inflated dataset size and unnecessary computation during training [8].

Thus, we propose using a computation and memory-efficient probabilistic data structure that, when queried by a stream of incoming human expert demonstrations, is able to quickly determine whether or not these new demonstrations fit within the distribution of an existing dataset of demonstrations. If a demonstration is determined to already be in the dataset's distribution, it is not added to the dataset, and the human demonstrator is notified of the rejection. Otherwise, the demonstration is added to the dataset. We explore two probabilistic data structures in our framework: Locally Sensitive Bloom Filters (LSBFs) [9], a modification of Bloom Filters [10] that uses locally sensitive hash functions [11], and RACE Sketches [12], which provide a means of kernel density estimation with a sketch structure (membership in this case will be determined by thresholding the returned density value). Our goal is to ensure that the dataset of demonstrations contains little to no redundancy without having to extensively search existing demonstrations or fully store them in memory.

2 Related Works

Although probabilistic data structures such as Bloom Filters have not yet been explicitly used in robot learning, previous work has been done in filtering out expert demonstrations for ones that influence policy performance the most. CUPID [13] selects high-influence demonstrations by performing rollouts of a trained policy and using those trajectories to rank each demonstration in terms of their significance via an influence function. In this way, they are able to both remove demonstrations from the existing dataset that harm policy performance and filter incoming demonstrations for ones that can improve performance the most. Although our probabilistic data structure method relies on the simplifying assumption that demonstrations that increase the dataset's coverage are beneficial, it does not require computationally expensive policy training or a time-consuming rollout period to evaluate demonstration influence.

DataMIL [14] attempts to circumvent the costly policy training and evaluation process for determining demonstration influence by using datamodels. Datamodels directly map demonstrations to influence values by quantifying how much policy performance changes when trained on different subsets of the dataset without actually performing the training or evaluation. DataMIL learns these datamodels through two methods: a naive regression-based method, which requires policy training

and evaluation, and a metagradient estimation method, which uses model gradients during training as a sort of influence function to determine demonstration significance. Although DataMIL does get around the cost of policy training and evaluation somewhat, it still requires some computation to learn and apply the datamodel. In contrast, our probabilistic data structure-based method seeks to eliminate training and evaluation costs by curating datasets under a simpler heuristic.

3 Hypothesis

We hypothesize that learning a robot policy from a demonstration dataset filtered by a probabilistic data structure will maintain policy performance compared to a policy trained on the full dataset while decreasing computation and memory demands.

4 Experiment Design

4.1 Environment and Dataset Definition

Our experiments are conducted in the Robosuite [15] simulation environment, and we use the demonstration datasets provided by Robomimic [16] for the *lift*, *can*, and *square* tasks, shown in Figure 1. In the *lift* task, the robot must grasp and lift up a cube from the table. In the *can* task, the robot must pick up a can from a table on one side of the space and place it in a designated area on a table on the other side of the space. Lastly, the robot in the *square* task must pick up a square-shaped ring and place it on the square-shaped peg. For each task, we have a dataset of 200 expert demonstrations, with each demonstration containing around roughly 120 state-action pairs. States are represented by robot end-effector pose, rotation, and gripper state (open or closed), as well as low-dimensional object state information such as object position and rotations. For the purposes of evaluating the performance of filtered datasets, we use 20% of the demonstrations (40 out of the 200 total) as the base dataset to populate the probabilistic data structure and treat the other 80% as a stream of incoming demonstrations to emulate a human expert providing demonstrations in an online fashion.

Figure 1: Robomimic tasks (from left to right): *lift*, *can*, and *square*.

4.2 Bloom Filter Analysis

To analyze the effectiveness of the LSBF data structure on robot manipulation datasets, we first conduct an analysis of false positive and false negative rates on the *can*, *square*, and *tool hang* datasets. Our LSBF uses a cosine-similarity locally-sensitive hash function, implemented by projecting the input state-action vector onto 64 randomly generated hyperplanes and outputting a 64-bit bitstring based on the sign of each projection. The bitstring is then converted into a single hash value for the Bloom Filter. To analyze the sensitivity of the LSH in the context of Bloom Filters, we take each state-action pair $x \in \mathbb{R}^D$ in the base dataset ($D = 30$ for our tasks) and generate 5 copies, with each copy consisting of the original state-action pair with a small uniformly sampled perturbation $\hat{x} \in [-T, T]^D$. We report false negative rates for each of the three tasks, defined as

the proportion of perturbed copies that are determined to not have been inserted by the LSBF for $T \in \{0.1, 0.01, 0.001, 0.0001\}$ in Table 1. The LSBF in this experiment has $k = 8$ independent LSH functions and a bitarray length of $m = 2^{15}$.

We then generate a dataset of 10000 randomly generated state-action pairs that are guaranteed to be out-of-distribution from the expert demonstration dataset. We report false positive rates for each of the three tasks, defined as the proportion of these randomly generated state-action pairs that are determined to have been inserted by the LSBF. We repeat this process for LSBFs with parameters $k \in [2, 4, 6, 8, 10]$ and $m \in [2^6, 2^9, 2^{12}, 2^{15}]$ in Figure 2.

4.3 Policy Performance Analysis

We now seek to evaluate the effectiveness of probabilistic data structures to filter incoming demonstrations for state-action pairs to create subsets of state-action pairs that can approximate the policy performance of the full dataset. As mentioned previously, we use the LSBF and RACE Sketch as our data structures. The LSBF has bitarray length $m = 2^{15}$ and $k = 8$ independent LSH functions. The RACE Sketch has $R = 8$ independent LSH functions and arrays of $L = 2^{10}$ for each row in the hash table. These parameters were chosen so that the LSBF and RACE Sketch have approximately the same memory requirements. Both data structures use the cosine-similarity LSH function with 64-bit bitstring outputs as discussed previously. To create a filtered dataset, we first instantiate the probabilistic structure (either an LSBF or RACE Sketch), and insert all state-action pairs of the base dataset. Then for each incoming state-action pair, we query the data structure. If the data structure responds positively (i.e. a similar state-action pair has already been added), then the current state-action pair is discarded. If the data structure responds negatively, then the current-state action pair is appended to the base dataset and is also inserted into the data structure itself for future queries.

For policy training, our policy architecture takes in end-effector and object state information as input and processes it with an MLP containing 2 hidden layers with 1024 nodes each. The MLP outputs a 7-dimensional action representing the desired end-effector position and orientation as well as gripper state (open or closed) for the next timestep. We train policies for 50000 gradient steps total using an L_2 behavior cloning loss with the AdamW [17] optimizer and a linearly decaying learning rate that starts at 0.001 and decays to 0.0001 by the end of training. For every 10000 gradient steps during training, we save a model checkpoint and conduct 50 rollouts in the same environment that demonstrations were collected in. The policy checkpoint that achieves the highest success rate out of 50 rollouts during the training phase will be selected for evaluation in the experiment. We evaluate our two probabilistic data structure filters against three baselines: a dataset containing just the 20% of demonstrations used to populate the data structure (meant to represent bottom-line performance), the full original dataset with 100% of the demonstrations (meant to represent top-line performance), and a random filter baseline that randomly responds positively to a query according to a specific rate. We set the acceptance rate of the random filter to generate dataset sizes that are directly comparable to the LSBF in order to evaluate the effectiveness of the actual state-action pairs chosen by a probabilistic filter. We measure policy performance by recording the success rate of each trained policy over 50 rollouts. We also measure dataset size (in number of state-action pairs) to quantify training and memory efficiency. These results are reported in Table 2.

5 Experiment Results

5.1 Bloom Filter Analysis

We report the false negative and false positive analysis results from the experiments described in Section 4.2. False negative rates for the *lift*, *can*, and *square* tasks are shown in Table 1 below:

T	False Negative Rates		
	Lift	**Can**	**Square**
0.1	0.9256	0.8660	0.8361
0.01	0.4293	0.3241	0.2711
0.001	0.0670	0.0448	0.0314
0.0001	0.0082	0.0042	0.0030

Table 1: False negative rates for the *lift*, *can*, and *square* tasks for different scales T of perturbations.

We see that for all three tasks, the false negative rates of the LSH functions drop drastically at $T = 0.0001$, suggesting that this is close to the threshold that determines what state-action pairs are similar to each other. Since the sensitivity of the cosine-similarity LSH depends on the length of its output bitstring, we anticipate that the similarity threshold would rise as the bitstring length decreases, and fall as the bitstring length increases.

Plots of false positive rates for the *lift*, *can*, and *square* tasks are shown in Figure 2 below:

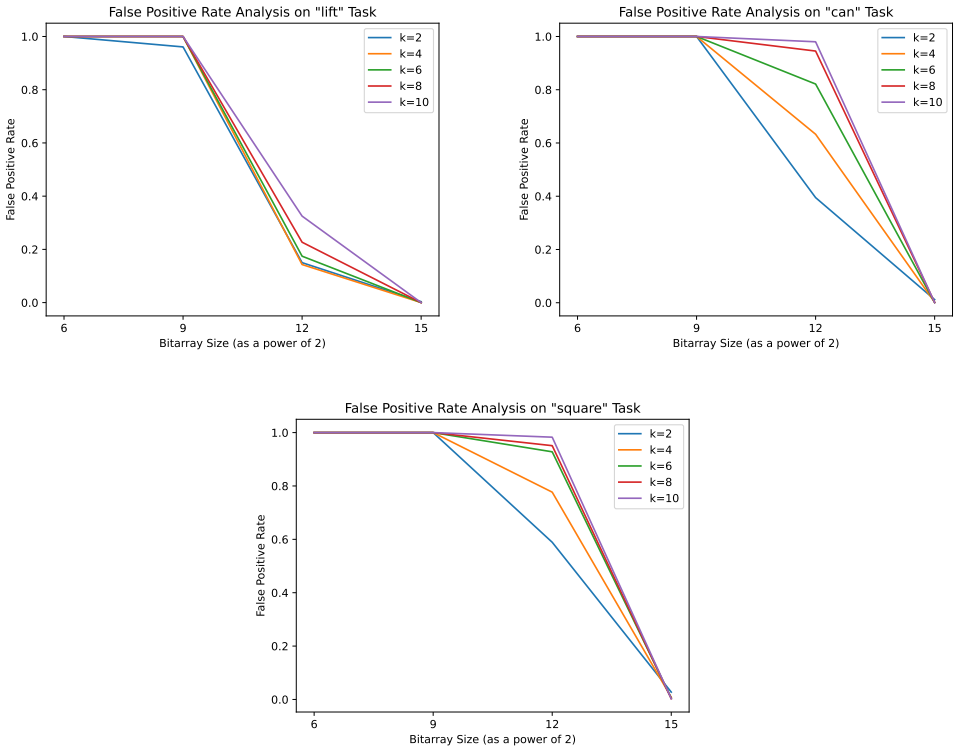

Figure 2: False positive rates for the *lift*, *can*, and *square* tasks for LSBFs with varying numbers of LSH functions k and bitarray lengths m.

Predictably, we see the lowest false positive rates for the highest values of m, as higher-capacity Bloom Filters can afford to insert more elements before getting saturated. Interestingly, we see that for a fixed value of $m \in \{2^9, 2^{12}\}$, the false positive rate increases as k increases. We hypothesize that this may be because even though a high number of independent hashes increases the robustness of membership queries through reducing the probability of a multi-hash collision, the small capacity of the bitarrays of these sizes may cause them to be increasingly saturated as more indices in the array are filled in with more hash functions. The pattern is least apparent on the *lift* task, as its base

dataset has the least amount of state-action pairs (see Table 2) and so the bitarray does not get as saturated. However, for larger datasets (such as in the *can* and *square* tasks), larger amounts of hash functions means that the bitarray gets to a state where almost all the indices are filled in, leading to higher false positive rates.

5.2 Policy Performance Analysis

We report in Table 2 success rates over 50 rollouts for policies trained on the *lift*, *can*, and *square* tasks from the experiments described in Section 4.3 below:

Task	Filter Method	Success Rate	Dataset Size	% Filtered
Lift	Base Dataset	0.82	1992	–
	Random Filter	0.96	6734	61.79
	Bloom Filter	0.98	6786	62.00
	RACE Sketch	0.88	2388	5.16
	Full Dataset	0.98	9666	–
Can	Base Dataset	0.38	4617	–
	Random Filter	0.90	13649	48.59
	Bloom Filter	0.88	13543	48.02
	RACE Sketch	0.78	8622	21.54
	Full Dataset	0.94	23207	–
Square	Base Dataset	0.20	6060	–
	Random Filter	0.38	14748	36.06
	Bloom Filter	0.44	14635	35.59
	RACE Sketch	0.36	9815	15.58
	Full Dataset	0.50	30154	–

Table 2: Success rates on the *lift*, *can*, and *square* tasks for various dataset types (either base, filtered, or full). Dataset size is measured in total number of state-action pairs. For the Random Filter, Bloom Filter, and RACE Sketch, % filtered corresponds to the proportion of incoming state-action pairs that were added to the base dataset.

In terms of success rate, we see that there exists a gap between the Base Dataset performance and the Full Dataset performance, suggesting that there is ample room for improvement by adding additional state-action pair samples to the base dataset. The smallest gap exsits in the *lift* task, as it is the simplest task to perform and thus even the small base dataset is able to attain a somewhat acceptable success rate, while the largest gap exists in the *can* dataset where the Base Dataset achieves a very low success rate whereas the Full Dataset achieves near 100% performance. For the probabilistic data structures, we see that the Bloom Filter dataset is able to achieve within 6% of the Full Dataset performance (exactly matching the Full Dataset performance in the case of the *lift* task) despite only using a fraction of the total number of state-action pairs, showcasing its ability to select for optimal incoming state-action pairs in online demonstrations. We also see that the Random Filter, constructed to generate dataset sizes directly comparable to that of the Bloom Filter, performs worse than the Bloom Filter on two of the tasks, showing that the Bloom Filter is able to specifically select for high-quality state-action pairs that lead to higher policy performance.

Although the RACE Sketch achieves lower success rates than the Bloom Filter due to the significantly smaller dataset size for each of the three tasks, we see that the improvement in success rate compared to the Base Dataset is significantly higher than the increase in dataset size. In the *can* task, for example, the dataset size increased by 86%, yet the success rate increased from 0.38 to 0.78, representing a 105% improvement compared to the base performance. This suggests that although this specific RACE Sketch is much more conservative than the Bloom Filter, it is able to find a smaller set of the highest quality state-action pairs to improve policy performance the most. Overall, we recognize that both the Bloom Filter and the RACE Sketch can be overly aggressive with their filtering, leading to completely valid state-action pairs that are not yet in the dataset being

rejected. However, we still see that the state-actions pairs that are accepted lead to significant gains in performance.

6 Conclusion

We present a framework that introduces probabilistic data structures as filters to efficiently curate a high-quality robot dataset for imitation learning. We evaluate our framework on three tasks from the Robomimic environment, and compared them to policy performance on the base dataset and full dataset baselines. We see that despite only using a fraction of the available state-action pairs in the demonstrations, the Bloom Filter and RACE Sketch filters are able to closely approximate the performance of the full dataset, and in some cases even matching it exactly, providing support for our hypothesis. With the framework validated, we can easily extend this system to an actual online demonstration setting, in which the human demonstrator may be notified of rejected state-action pairs and adjust their demonstration style on-the-fly accordingly. Future work that builds on this project could involve integrating probabilistic data structure filters with more sophisticated policy architectures and training algorithms such as Transformers [18] and Diffusion Policy [19], and extending evaluation to much more complex and longer-horizon tasks.

References

[1] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

[2] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[3] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. Openvla: An open-source vision-language-action model. In P. Agrawal, O. Kroemer, and W. Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

[4] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du, A. Lee, K. Fang, C. Finn, and S. Levine. Bridgedata v2: A dataset for robot learning at scale. In J. Tan, M. Toussaint, and K. Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 1723–1736. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/walke23a.html.

[5] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems*, 2024.

[6] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi,

G. Berseth, G. Kahn, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. Ben Amor, H. I. Christensen, H. Furuta, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake, J. Peters, J. Schneider, J. Hsu, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Tompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Fan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. Di Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitrano, P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Martín-Martín, R. Baijal, R. Scalise, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Song, S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, and Z. Lin. Open x-embodiment: Robotic learning datasets and rt-x models : Open x-embodiment collaboration0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903, 2024. doi:10.1109/ICRA57147.2024.10611477.

[7] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. Robotics: Science and Systems, 2024.

[8] S. Mirchandani, S. Belkhale, J. Hejna, E. Choi, M. S. Islam, and D. Sadigh. So you think you can scale up autonomous robot data collection? In P. Agrawal, O. Kroemer, and W. Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2791–2810. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/mirchandani25a.html.

[9] Y. Hua, B. Xiao, B. Veeravalli, and D. Feng. Locality-sensitive bloom filter for approximate membership query. *IEEE Transactions on Computers*, 61(6):817–830, 2012. doi:10.1109/TC.2011.108.

[10] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970. ISSN 0001-0782. doi:10.1145/362686.362692. URL https://doi.org/10.1145/362686.362692.

[11] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, page 253–262, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138857. doi:10.1145/997817.997857. URL https://doi.org/10.1145/997817.997857.

[12] B. Coleman and A. Shrivastava. Sub-linear race sketches for approximate kernel density estimation on streaming data. In *Proceedings of The Web Conference 2020*, WWW '20, page 1739–1749, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370233. doi:10.1145/3366423.3380244. URL https://doi.org/10.1145/3366423.3380244.

[13] C. Agia, R. Sinha, J. Yang, R. Antonova, M. Pavone, H. Nishimura, M. Itkina, and J. Bohg. Cupid: Curating data your robot loves with influence functions. *arXiv preprint arXiv:2506.19121*, 2025.

[14] S. Dass, A. Khaddaj, L. Engstrom, A. Madry, A. Ilyas, and R. Martín-Martín. Datamil: Selecting data for robot imitation learning with datamodels. *arXiv preprint arXiv:2505.09603*, 2025.

[15] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

[16] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In A. Faust, D. Hsu, and G. Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1678–1690. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/mandlekar22a.html.

[17] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[19] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.