Visual Imitation Made Easy

Sarah Young
UC Berkeley

Dhiraj Gandhi
FAIR

Shubham Tulsiani
FAIR

Abhinav Gupta
FAIR, CMU

Pieter Abbeel
UC Berkeley

Lerrel Pinto
UC Berkeley, NYU

Abstract

Visual imitation learning provides a framework for learning complex manipulation behaviors by leveraging human demonstrations. However, current interfaces for imitation such as kinesthetic teaching or teleoperation restrict our ability to efficiently collect large-scale data in the wild. Obtaining such diverse demonstration data is paramount for the generalization of learned skills to novel scenarios. In this work, we present an alternate interface for imitation that simplifies the data collection process while allowing for easy transfer to robots. We use commercially available reacher-grabber tools both as a data collection device and as the robot's end-effector. To extract actions from these visual demonstrations, we use off-the-shelf Structure from Motion (SfM) techniques in addition to training a finger detection network. We experimentally evaluate on two challenging tasks: non-prehensile pushing and prehensile stacking, with 1000 diverse demonstrations for each task. For both tasks, we use standard behavior cloning coupled with data augmentations to learn executable policies from the previously collected offline demonstrations. Finally, we demonstrate the utility of our interface by evaluating on real robotic scenarios with previously unseen objects and achieve a 87% success rate on pushing and a 62% success rate on stacking. Robot videos are available at our project website.

1 Introduction

A powerful technique to learn complex robotic skills is to imitate them from humans [1, 2, 3, 4]. Recently, there has been a growing interest in learning such skills from visual demonstrations, since it allows for generalization to novel scenarios [5, 6]. Prominent works in Visual Imitation Learning (VIL) have demonstrated utility in intricate manipulation skills such as pushing, grasping, and stacking [5, 7]. However, a key bottleneck in current imitation learning techniques is the use of interfaces such as kinesthetic teaching or teleoperation, which makes it harder to collect large-scale manipulation data. But more importantly, the use of such interfaces leads to datasets that are constrained to be in restrictive lab settings. This severely limits the generalizability of the learned skills in novel, previously unseen situations [8].

It is thus important to find a way to simplify data collection for imitation learning to allow both data collection at scale and real world diversity. One of the cheapest 'robots' that is highly prevalent, easy to control, and requires little to no human training is the reacher-grabber depicted in Fig. 1. This assistive tool is commonly used for grasping trash among other activities of daily living and has recently been shown to be a scalable interface for collecting grasping data in the wild by Song et al. [9]. However, unlike teleoperation [5] or kinesthetic [10] interfaces where the demonstrations are collected on the same platform as the robot, assistive tools are significantly different from robotic manipulators. Song et al. [9] bridges this gap by first extracting grasp points from demonstrations and then transferring them to robot in order to achieve closed-loop grasping of novel objects. A key problem, however, still lies in scaling this to enable imitation of general robotics tasks. One

NeurIPS 2020 3rd Robot Learning Workshop: Grounding Machine Learning Development in the Real World.

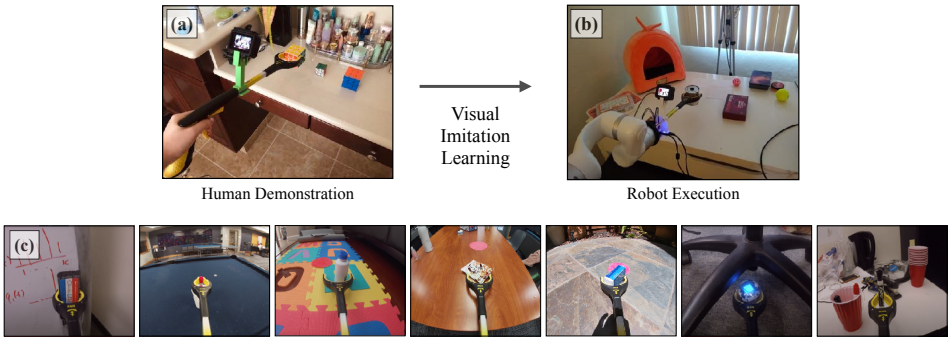


Figure 1: We present a framework for visual imitation learning, where demonstrations are collected using commercially available reacher-grabber tools (a). This tool is also instrumented as an end-effector and attached to the robot (b). This setup allows us to collect and learn from demonstration data across diverse environments (c), while allowing for easy transfer to our robot.

possible solution is to extract full tool configuration and learn a mapping between grabber and the robot hardware. An alternative is to run domain adaptation based techniques for transfer. Instead, why not simply use the assistive tool as an end-effector?

In this work, we propose an alternate paradigm for providing and learning from demonstrations. As seen in Fig. 1 (a,c), the user collects Demonstrations using Assistive Tools (DemoAT) to solve a task. During the collection of this data, visual RGB observations are collected from a camera mounted on the DemoAT tool. Given these visual demonstrations, we extract tool trajectories using off-the-shelf Structure from Motion (SfM) methods and the gripper configuration using a trained finger detector. Once we have extracted tool trajectories, corresponding skills can be learned using standard imitation learning techniques. Finally, these skills can be transferred to a robot that has the same tool setup as the end-effector. Having the same end-effector as the demonstration tool coupled with a 6D robotic control (Fig. 1 (b)) allows for a direct transfer of learning from human demonstrations to the robot.

To study the effectiveness of this tool, we focus on two challenging tasks: (a) non-prehensile pushing [11, 12], and (b) prehensile stacking [13, 14]. For both tasks, we collect 1000 demonstrations in multiple home and office environments with various different objects. This diversity in objects and environments allows our learned policies to generalize and be effective across novel objects.

In summary, we present three key contributions in this work. First, we propose a new interface for visual imitation learning that uses assistive tools to gather diverse data for robotic manipulation, including an approach for collecting grabber 3-D trajectories and gripper transitions. Second, we demonstrate the utility of this framework on pushing and stacking previously unseen objects, with a success rate of 87.5% and 62.5% respectively. Finally, we present a detailed study on the effects of data augmentations in learning robotic skills, and demonstrate how the combination of random 'crops', 'rotations' and 'jitters' significantly improve our policies over other augmentations.

2 Method

Demonstration tool: Our DemoAT setup is built around a plastic 19-inch RMS assistive tool [15] and a RGB camera [16] to collect visual data. We attach a 3D printed mount above the stick to hold the camera in place. To collect demonstrations, a human user uses the setup shown in Fig. 1 (a), which allows the user to easily push, grab, and interact with everyday objects in an intuitive manner. Examples of demonstrations can be seen in Fig. 1 (c) and Fig. 5. Since a demonstration collected with DemoAT is visual, it can be represented as a sequence of images $\{I_t\}_{t=0}^{T}$.

Robot end-effector: The tool is attached on a 7DoF robot arm with a matching camera and mount setup (Fig. 1 (b)). However, to actuate the fingers, we will need to create an actuating mechanism. Through a compact, lightweight and novel mechanism, we replace the lever from the original reacher grabber tool with a controllable interface. Details on this mechanism are presented in Appendix C.

Extracting actions from visual demonstrations Although our framework provides a robust and reliable interface to collect visual demonstrations, it does not have explicit sensors to collect information

Table 1: Real robot evaluation results (average success rate) for pushing and stacking on different amounts of data.

		Naive BC 100%	BC with augment 100%	BC with augment 50%	BC with augment 10%
Push	reach goal	0.625	**0.875**	0.750	0
Stack	grasp object	0.750	**0.833**	0.792	0
	stack object	0.291	**0.625**	0.416	0

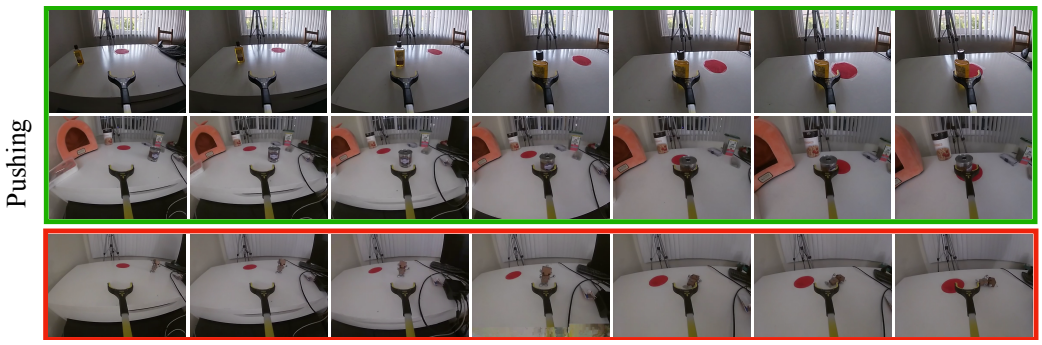


Figure 2: We visualize trajectories executed on the robot using our learned pushing policies trained with augmented data.

about actions such as the end-effector's motion or finger locations. To address this, we recover 6DoF poses of the tool using Structure-from-Motion (SfM) reconstruction and finger configurations via a gripper prediction model. More details on extracting visual information are presented in Appendix D.

Visual behavior cloning: We learn a policy using straightforward behavioral cloning [17, 18]. With the DAT imitation framework, we collect observation-action pairs $D = \{(o_t, a_t)\}$, where o_t is an image and a_t is the action to get from o_t to o_{t+1}. Additional details on training are presented in Appendix F.

3 Experiments

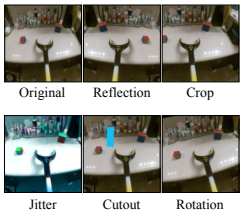
We aim to answer the following key questions: (a) Can DemoAT be used to solve difficult manipulation tasks? (b) How important is the scale and diversity of data for imitation learning in the wild? (c) How important is data augmentation for visual imitation?

We evaluate our learned policy on the two tasks (see Appendix E for task details) using two metrics. First, mean squared error (BC-MSE) between predicted actions and ground truth actions on a set of held-out demonstrations that contain novel objects in novel scenes. This offline measure allows for benchmarking different learning methods. Second, we evaluate on real robot executions on previously unseen objects and measure the fraction of successful executions. This captures the ability of our models to generalize on real scenarios.

3.1 Can DemoAT be used for solving difficult manipulation tasks?

To study the utility of our DemoAT framework, we look at both measures of performance, the offline BC-MSE and the real robot success rate. Unless otherwise noted, we train our policies with 100% of training data and using the 'crop'+'jitter' augmentation for pushing and 'rotate'+'jitter' for stacking (Fig. 3). On the BC-MSE metric, we achieve an error of 0.028 on the pushing task and an error of 0.056 on the stacking task. We note that this is more than two orders of magnitude better than random actions, which has error of 0.67 and 0.69 on pushing and stacking respectively. This demonstrates that our policies have effectively learned to generalize to previously unseen demonstrations. Visualizations of how close predicted actions are to ground truth actions are presented in Appendix G.

Data augmentations



Results on pushing

	Rotation	Crop	Color Jitter	Reflection	Cutout-color
None	0.0381				
Rotation	0.0352	0.0340	0.0342	0.0368	0.0334
Crop	0.0340	0.0356	0.0315	0.0365	0.0346
Color Jitter	0.0342	0.0315	0.0355	0.0350	0.0348
Reflection	0.0368	0.0365	0.0350	0.0376	0.0341
Cutout-color	0.0334	0.0346	0.0348	0.0341	0.0398

Results on stacking

	Rotation	Crop	Color Jitter	Reflection	Cutout-color
None	0.0695				
Rotation	0.0636	0.0700	0.0599	0.0646	0.0672
Crop	0.0700	0.0684	0.0721	0.0712	0.0696
Color Jitter	0.0599	0.0721	0.0711	0.0614	0.0648
Reflection	0.0646	0.0712	0.0614	0.0670	0.0733
Cutout-color	0.0672	0.0696	0.0648	0.0733	0.0749

Figure 3: On the left, we show the five data augmentations used in this work. The tables show an analysis of MSE error of different combinations of data augmentations for pushing and stacking.

3.2 How important is data for imitation learning in the wild?

A key promise of the DemoAT setup is the ability to collect large-scale, diverse demonstrations. But how important is this diversity of data? To study this, we train policies on different fractions of sequential and random splits of training data - 1, 5, 10, 25, 50, 75, 100% and evaluate their performance. We present details of BC-MSE in Appendix I and real robot evaluation below.

Real Robot Evaluation: In Table 1, we report the performance of robotic execution on models trained on 10%, 50%, and 100% of the collected data for each task on an unseen test set of 24 different objects. In both tasks, we see that with only 10% of the data (100 trajectories), the robot is unable to even reach the first object. When we increase to 50% of the data, we see a huge improvement and the robot starts to learn to reach the objects and complete the tasks. Particularly, with just 50% of the data, the robot can successfully reach the object 100% of the time in the non-prehensile pushing task. When we evaluate with all our data, we still see considerable performance improvements in completing the tasks, with 12.5% in pushing and 20.9% in stacking. This improvement is significantly higher than what we see with the BC-MSE metric.

Data Diversity vs Size: To further understand the effects of diversity and size, we compare performance on the same fractional split, but different amounts of diversity in the data. Given a quota of 100 demonstrations, we train on two splits of data: (A) many observations of the same objects and scenes (B) sparse observations across a diverse set of objects and scenes. We find that the test error for dataset (A) [0.081] is on average 1.4% higher than that of dataset (B) [0.067]. More detailed analysis is presented in Appendix I.2.

3.3 Does augmented data help?

To improve the performance of our learned policies, we employ data augmentations in training. But, how important are these augmentations in imitation learning? We discuss results on real robot experiments in Appendix H.1 and more details about augmentations in Appendix H.

Behavioral Cloning Evaluation: We compare the application of different augmentations: crop, color jitter, rotate, horizontal reflection, random cutout, and all permutations of two augmentations. We find that data augmentations allow for better generalization to unseen objects and scenes on the BC-MSE metric. As shown in Fig. 3, the best augmentation performs 0.7% better than naive behavioral cloning in pushing and 0.9% better in stacking.

4 Conclusion

We present Demonstrations using Assistive Tools (DemoAT). In contrast to traditional imitation methods that rely on domain adaptation techniques or kinesthetic demonstrations, our proposed method allows for both easy large-scale data collection and direct visual imitation learning. We have shown that using a universal reacher-grabber tool that can act as an end-effector for virtually any robot, smarter data collection methods coupled with simple behavior cloning methods and data augmentations can lead to better out of distribution performance. We hope that this interface is a step towards more efficient robot learning, since it opens up directions for wide scale data collection and re-use.

References

[1] J. Piaget. *Play, dreams and imitation in childhood*, volume 25. Routledge, 2013.

[2] A. N. Meltzoff and M. K. Moore. Imitation of facial and manual gestures by human neonates. *Science*, 198(4312):75–78, 1977.

[3] A. N. Meltzoff and M. K. Moore. Newborn infants imitate adult facial gestures. *Child development*, pages 702–709, 1983.

[4] M. Tomasello, S. Savage-Rumbaugh, and A. C. Kruger. Imitative learning of actions on objects by children, chimpanzees, and enculturated chimpanzees. *Child development*, 64(6):1688–1705, 1993.

[5] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *ICRA*, 2018.

[6] B. C. Stadie, P. Abbeel, and I. Sutskever. Third-person imitation learning. *ICLR*, 2017.

[7] Y. Zhu, Z. Wang, J. Merel, A. A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, and N. Heess. Reinforcement and imitation learning for diverse visuomotor skills. *CoRR*, abs/1802.09564, 2018. URL `http://arxiv.org/abs/1802.09564`.

[8] A. Gupta, A. Murali, D. P. Gandhi, and L. Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. In *Advances in Neural Information Processing Systems*, pages 9094–9104, 2018.

[9] S. Song, A. Zeng, J. Lee, and T. Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *Robotics and Automation Letters (RA-L)*, 2020.

[10] P. Sharma, L. Mohan, L. Pinto, and A. Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. *CoRL*, 2018.

[11] K. M. Lynch. *Nonprehensile robotic manipulation: Controllability and planning*. Citeseer, 1996.

[12] F. Ruggiero, V. Lippiello, and B. Siciliano. Nonprehensile dynamic manipulation: A survey. *IEEE Robotics and Automation Letters*, 3(3):1711–1718, 2018.

[13] Y. Zhang, B. K. Chen, X. Liu, and Y. Sun. Autonomous robotic pick-and-place of microobjects. *IEEE Transactions on Robotics*, 26(1):200–207, 2009.

[14] F. Furrer, M. Wermelinger, H. Yoshida, F. Gramazio, M. Kohler, R. Siegwart, and M. Hutter. Autonomous robotic stone stacking with online next best object target pose planning. In *ICRA*, 2017.

[15] Grabber pick up tool. `https://www.amazon.com/RMS-Grabber-Reacher-Rotating-Gripper/dp/B00THEDNL8/ref=sr_1_16?dchild=1&keywords=grabber+tool&qid=1595965487&sr=8-16`.

[16] Gopro hero7 silver. `https://gopro.com/en/us/shop/cameras/hero7-silver/CHDHC-601-master.html`.

[17] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

[18] S. Ross, G. Gordon, and D. 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*, pages 627–635, 2011.

[19] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[20] A. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. *ISER*, 2004.

[21] P. K. Pook and D. H. Ballard. Recognizing teleoperated manipulations. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 578–585. IEEE, 1993.

[22] J. D. Sweeney and R. Grupen. A model of shared grasp affordances from demonstration. In *2007 7th IEEE-RAS International Conference on Humanoid Robots*, pages 27–35. IEEE, 2007.

[23] T. I. M. I. H. Inoue, M. Inamura, and H. Inaba. Acquisition of probabilistic behavior decision model based on the interactive teaching method. In *ICAR*, 1999.

[24] W. D. Smart. *Making reinforcement learning work on real robots*. Brown University Providence, 2002.

[25] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert. Learning monocular reactive uav control in cluttered natural environments. In *ICRA*, 2013.

[26] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[27] A. Handa, K. Van Wyk, W. Yang, J. Liang, Y.-W. Chao, Q. Wan, S. Birchfield, N. Ratliff, and D. Fox. Dexpilot: Vision based teleoperation of dexterous robotic hand-arm system. *arXiv preprint arXiv:1910.03135*, 2019.

[28] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz. Keyframe-based learning from demonstration. *International Journal of Social Robotics*, 4(4):343–355, 2012.

[29] K. Muelling, A. Boularias, B. Mohler, B. Schölkopf, and J. Peters. Learning strategies in table tennis using inverse reinforcement learning. *Biological cybernetics*, 108(5):603–619, 2014.

[30] I. Lenz, R. A. Knepper, and A. Saxena. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*. Rome, Italy, 2015.

[31] P. Sermanet, K. Xu, and S. Levine. Unsupervised perceptual rewards for imitation learning. *arXiv preprint arXiv:1612.06699*, 2016.

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[34] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, 2019.

[35] D. Ho, E. Liang, X. Chen, I. Stoica, and P. Abbeel. Population based augmentation: Efficient learning of augmentation policy schedules. In *ICML*, 2019.

[36] B. Zoph, E. D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, and Q. V. Le. Learning data augmentation strategies for object detection. *arXiv preprint arXiv:1906.11172*, 2019.

[37] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *NeurIPS*, 2019.

[38] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2019.

[39] F. Sadeghi and S. Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[40] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, 2017.

[41] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel. Asymmetric actor critic for image-based robot learning. *RSS*, 2018.

[42] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020.

[43] I. Kostrikov, D. Yarats, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[44] xarm 7. `https://www.xarm.cc/products/xarm-7-2020`.

[45] J. L. Schönberger and J.-M. Frahm. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[46] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[47] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019.

[48] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

A Related Work

In this section, we briefly discuss prior research in the context of our work. For a more comprehensive review of imitation learning, we point the readers to Argall et al. [19].

Interfaces for Imitation: In imitation learning, a robot tries to learn skills from demonstrations provided by the expert. There are various interfaces through which these demonstration can be recorded. One option is teleoperation, in which the human controls the robot using a control interface. This method has been successfully applied to a large range of robotic tasks including flying a robotic helicopter [20], grasping objects [21, 22], navigating robots through cluttered environments [23, 24, 25], and even driving cars [26]. Teleoperation has been successful in solving a wide variety of tasks because of the availability of control interfaces through which human operators can perform high-quality maneuvers. However, it is challenging to devise such interfaces for robotic manipulation [27]. Kinesthetic demonstrations, in which the expert actively controls the robot arm by exerting force on it, is an effective method [28, 10] of collecting robot manipulation demonstrations for playing ping pong [29] and cutting vegetables [30]. However, for visuomotor policies, which map from pixels to actions, these demonstrations are inappropriate due to the undesirable appearance of human arms. One way to overcome this problem is by mounting an assistive tool on the robot end effector that is being used to record demonstration in isolation [9]. We take this idea a step further by using it as an end-effector on the robot as well. This eliminates the domain gap between the human-collected demonstrations and the robot executions, which enables easier imitation.

Behavior Cloning in Imitation: Behavior cloning is the simplest form of imitation learning, where the agent learns to map observations to actions through supervised learning. It has been successfully applied in solving a wide range of tasks including playing games [18], self-driving [26], and navigating drones through cluttered environments [25]. However, it has not been widely applicable to learning visuomotor policies for robotic manipulation tasks due to unwanted visual artifacts collected in kinesthetic demonstrations. To overcome this problem, Zhang et al. [5] propose a Virtual Reality (VR) setup to collect robot manipulation data. They showed that behavior cloning can be used to learn complex manipulation tasks, such as grasping and placing various objects. There have also been recent efforts to imitate from visual demonstrations collected from a different space e.g. from a different viewpoint or an agent with a different embodiment [6, 31] from the robot. This is a promising direction as it allows for data collection outside the lab. However, learning from such demonstrations is still an active research problem, as there is a significant domain gap between training and testing. In our setup, we use behavior cloning to learn challenging tasks such as pushing and stacking. But instead of relying on a costly VR setup which can only be deployed in constrained lab environments, we rely on cheap assistive tools to collect diverse data in the wild. Further, we eliminate the domain gap present in previously mentioned lines of work by attaching the same tool on the robot to match the demonstration and imitation space.

Data Augmentation in Learning: Data augmentation is widely used in machine learning to inject additional knowledge in order to overcome the challenges of overfitting. This technique has been shown to greatly benefit deep learning systems for computer vision. Its use can be found as early as LeNet-5 [32], which was used to classify hand written digits. In AlexNet [33], data augmentations such as random flip and crop were used to improve the classification accuracy. More recently, learning augmentation strategies from data has emerged as a new paradigm to automate the design of augmentation [34, 35, 36]. For unsupervised and semi-supervised learning, several unsupervised data augmentation techniques have been proposed [37, 38]. It has also been extensively used in context of RL, where domain randomization is proposed to transfer learning from simulation to real world [39, 40, 41]. Although the effects of augmentations have been extensively studied in image-based RL [42, 43], to the best of our knowledge, we are the first to study the effects of data augmentations in real-robot applications.

B Details on DemoAT Demonstration Tool

The DemoAT steup consists of the following parts:

- 19-inch RMS Handi Grip Reacher
- GoPro HERO7 Silver camera
- 3D printed mount

Our simple setup makes it easy to start collecting demonstrations. We directly attach the angled 3D mount on the tool as shown in Fig. 4 (a) and insert the camera. This can easily be replaced by a different GoPro camera or even a phone with a modified mount. To close the fingers on the tool, the human user simply needs to press the lever. Although our tasks do not require rotating the fingers, this tool is capable of doing so.

Figure 4: In part (a), we show the setup for collecting human demonstrations. This includes a 3D printed mount, a camera, and the reacher-grabber stick. Part (b) displays the corresponding setup on the robot.

C Details on DemoAT Robot End Effector

On the robot's end, we have a similar setup. We use the same reacher-grabber tool on the robot and attach it using metal studs to the robot end effector. We modified the 3D printed mount used for collecting human demonstrations with a similar one that includes an actuator to control the fingers on the tool, shown in Fig. 4 (b). While we use an xArm7 robot [44] as our robotic arm, we note that this end-effector setup can be attached to any standard commercial-grade robotic arm.

D Extracting actions from Visual Demonstrations

We recover 6DoF poses of the tool using Structure-from-Motion (SfM) reconstruction. Specifically, we use the publicly available COLMAP [45, 46] software for SfM. Once we have the end-effector pose p_t for every image I_t, we extract the relative translation and rotation Δp_t between consecutive frames and use them as the action for training.

COLMAP gives us the relative change in pose across frames, however, we also need to obtain the finger configurations for tasks that require moving the fingers. To do this, we use a neural network that extracts the location of the gripper fingers in our observations. This network is trained on a small human-labeled dataset of 155 frames from the DemoAT setup. Given these gripper finger locations predicted by our gripping model, we can generate labels for "close" or "open" states $g_t \in \{0, 1\}$. Through this procedure we can now obtain visual demonstrations with actions a_t, which is represented as $(o_t, a_t = (\Delta p_t, g_{t+1}))_{t=0}^{T}$. Note that the grasping action at a given timestep is the grasp state at the next timestep. Visualizations of actions can be seen in Fig. 5.

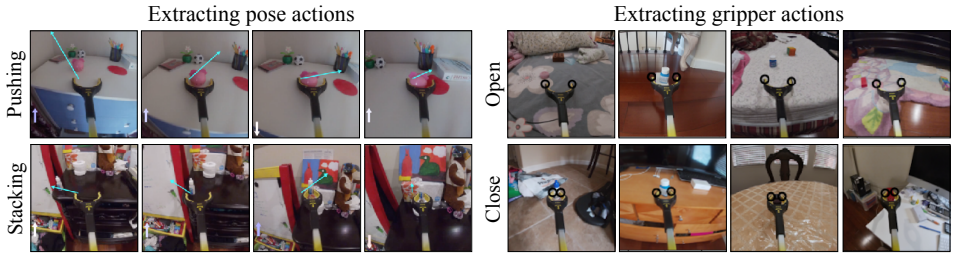


Figure 5: Extracting labels: (a) COLMAP translation arrows. (b) Gripper finger predictions.

D.1 Accuracy of Extracted Actions

Our method for extracting labels can be noisy. Specifically, COLMAP reconstructions are significantly less accurate in lightly textured, clean, and high dynamic range scenes. However, since our demonstrations are collected in cluttered real-world scenarios, our reconstructions are reasonably accurate for the purposes of learning. Nevertheless, to reduce the effect of noisy action labels, we visually inspect the reconstructions and discard $\sim 6\%$ of aberrant demonstrations. The model we use to generate grasping actions by detecting finger achieves $\sim 95\%$ accuracy on held-out testing set, which is empirically sufficient for downstream learning.

E Pushing and Stacking Task Details

Non-prehensile Pushing: This task requires the robot to push an object to a red circle by sliding it across the table. Such contact-rich manipulation has been extensively studied and known to be challenging to solve [11, 12].

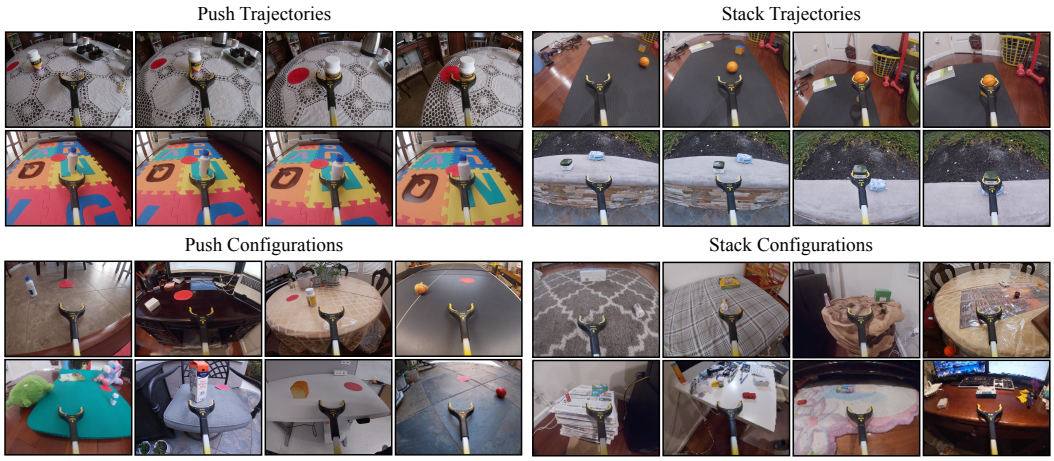


Figure 6: We collect 1000 trajectories for each task with diverse objects and scenes. Here we show examples of the diversity in data.

Particularly in our case, we operate with diverse objects in diverse scenes, which makes accurately manipulating objects difficult. For robotic experiments, we evaluate robotic success rate as $\frac{\#\text{trajectories where object reaches goal}}{\#\text{total trajectories}}$ on a set of 24 different objects unseen in training.

Prehensile Stacking: In this task, the goal is to grasp an object and stack it on top of an equally sized or larger object. We set it up such that the smaller object is always in front of the larger object to reduce ambiguity in learning (Fig. 6). We evaluate robotic success rate as $\frac{\#\text{trajectories where object is grasped and stacked}}{\#\text{total trajectories}}$ on a set of 24 configurations unseen in training.

F Training Details

Let Ck denote convolutional layers with k filters and Fk denote fully connected layers of size k.

Our architecture is a set of convolutional layers followed by fully connected layers. The first part of our network takes in an image $I_t \in \mathbb{R}^{3x224x224}$ and outputs the latent representation of the observation. It consists of the first five layers of the AlexNet followed by C256 layer. We feed the latent representation into an additional net of size F512-F256 to output a relative translation vector. To get the relative rotation, we concatenate the latent representation and the translation vector and feed the result into a F256 layer before projecting to a 6D vector. For tasks that require using the gripper, we train an additional classification model that takes in an image $I_t \in \mathbb{R}^{3x224x224}$ and outputs a gripper open/close label $g_{t+1} \in \{1, 0\}$. This architecture is illustrated in Fig. 7.

We train on a 6D rotation representation [47] because it is continuous in the real Euclidean space and thus more suitable for learning as opposed to more commonly used axis-angle and quaternion based representations.

To train our model, we use a combination of L1, L2, and a direction loss. We care more about the direction between the prediction and ground truth actions than the magnitude, so we add the following loss to encourage this directional alignment [5].

$$L_d = \arccos\left(\frac{\Delta x_t^T \pi_\theta(\Delta x_t | o_t)}{||\Delta x_t|| ||\pi_\theta(\Delta x_t | o_t)||}\right) \tag{1}$$

G Visualizations of Predicted Actions

We overlay predicted actions and COLMAP-generated labels on the images to qualitatively evaluate our results. Fig. 8 displays examples of test time results for both the pushing and stacking task.

The arrows on the images show the relative translations across the transverse plane of the camera between I_t and I_{t+1}. The aqua arrow represents the true action as output by COLMAP, and the yellow arrow represents our model's prediction. At the bottom left corner are two color-map arrows representing the up-down movement. The first arrow is the label action and the second arrow is the predicted action. The intensity of the color represents the magnitude of the action.

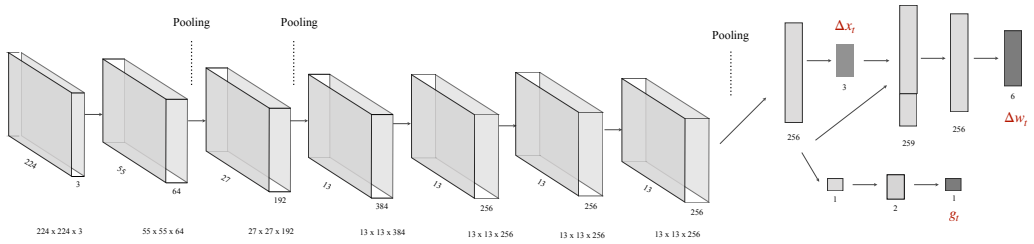


Figure 7: This is our network architecture. The input to the network is an image $I_t \in \mathbb{R}^{3x224x224}$ and it outputs $\Delta p_t = (\Delta x_t, \Delta w_t)$ and g_t predictions.

The angle plot shows the predicted relative frame rotation between I_t and I_{t+1}. The blue and green arrows represent true and predicted rotations respectively as rotation matrices multiplied by the unit vector $< 1, 0, 0 >$. We apply minimal rotation in these tasks, so these arrows are very close to $< 1, 0, 0 >$.

The bar chart in the stacking task shows the predicted probability of the status of the gripper at the next timestep. The true gripper label is green.

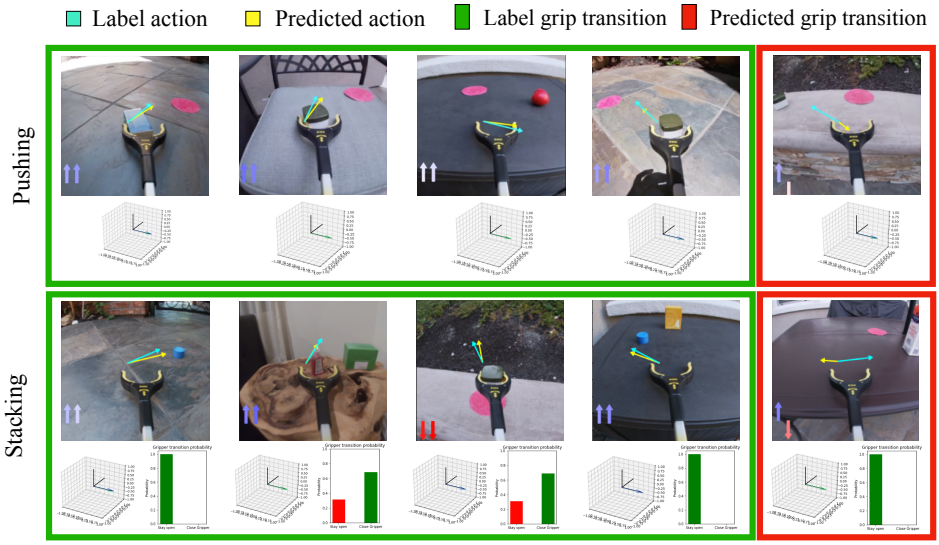


Figure 8: Examples of pushing and stacking result visualizations. The aqua arrow represents the true action across the transverse plane of the camera, while the yellow arrow represents the predicted action. The arrows in the corner are the true and predicted up-down actions. In the stacking task, the heights of the bars show the probabilities of the predicted gripper status at the next timestep, and the true gripper transition is shown in green.

H Data Augmentations for imitation:

To improve the performance of our networks with limited data, we experiment with using the following data augmentations in training [42, 43, 48]:

- Color Jitter: Randomly adds up to $\pm 20\%$ random noise to the brightness, contrast and saturation of each observation.
- Crop: Randomly extracts a 224 x 224 patch from an original image of size 240 x 240.
- Cutout-color[rad]: Randomly inserts a colored box of size [10, 60] into the image.
- Rotation: Randomly rotates original image [-5, 5] degrees.
- Horizontal Reflection: Mirrors image across the y-axis. Action labels are reflected as well.

H.1 Real Robot Evaluation with Augmented Data

Our second method of evaluation is to compare the success rates of stacking and pushing with data augmentations to naive behavioral cloning. Table 1 shows that for both tasks we achieve significant improvements with data augmentations. We see the biggest increases in performance in the second part of each task (after the initial object has been reached): a 12.5% improvement for reaching the goal in pushing and a 33.4% improvement in stacking. Interestingly, using augmentations with just 50% of training data surpasses the performance of not using augmentation with 100% of training data on both pushing and stacking. This ability to improve performance in robotics is in line with recent research in RL [42, 43] and computer vision [48].

I Data Size and Diversity Analysis

I.1 Data Size: Behavioral Cloning Evaluation

In Fig. 9 we illustrate the effects of changing dataset size on BC-MSE performance. In both the pushing and stacking task, we see increasing data size significantly improves performance especially in the low-data regime. We note that improvements diminish with larger data on the BC-MSE metric with just $\sim 0.9\%$ performance gain when increasing our training data from 500 to 1000 trajectories.

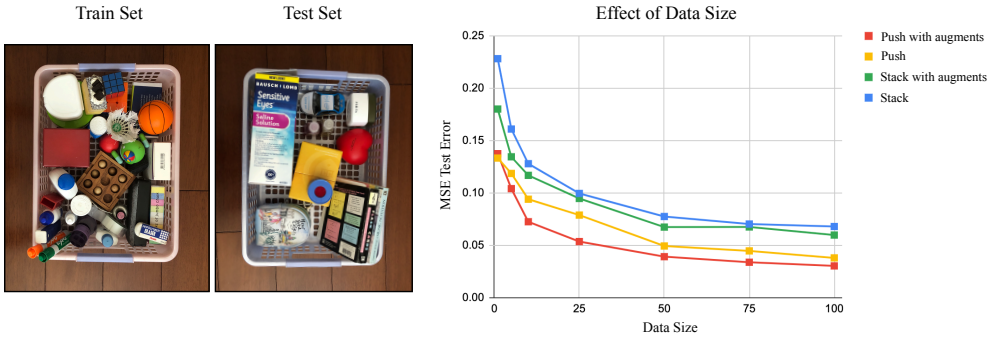


Figure 9: On the left, we show examples of objects used in training and testing for behavioral cloning evaluation. On the right, we evaluate the MSE error on held-out testing objects with and without random data augmentations. Note that as we increase the amount of data, our models improve and achieves lower error.

I.2 Data Diversity Analysis

We provide more detail on the results of comparing random and sequential splits. We let dataset (A) be many observations of the same objects and scenes (sequential split) and dataset (B) be sparse observations across a diverse set of objects and scenes (random split). We showed that in the most extreme case, when using 10% of the data, we see an average of 1.4% increase in performance when comparing the sequential dataset (A) to the diverse dataset (B). We analyze these results in more detail by running naive behavioral cloning without data augmentations for the following splits of data: 10, 25, 50, and 75%.

In Fig. 10, we compare error rates for both dataset types in both pushing and stacking. The most prominent performance increase of using diverse data is when we only use 10% of the data, and as the amount of data increases, the gap between dataset (A) and dataset (B) starts to decrease. Even at 75% data, we still see a 0.005 and a 0.001 increase in accuracy in the pushing and stacking task, respectively. As we train on more data, it follows that the diversity of data increases and thus the difference in performance decreases.

I.3 Study of Data Augmentations on Random Data Splits

We show the same analysis in Fig. 9 using random, diverse data splits instead of sequential data splits to demonstrate that data augmentations are effective and amount of data is important in both cases. Similar to the case with sequential data, we note that the performance gains diminish as we include more data in our training set. In Fig. 10, we see that random splits perform better the sequential splits across every fraction of data. We analyze this difference in more detail in Appendix I.2.

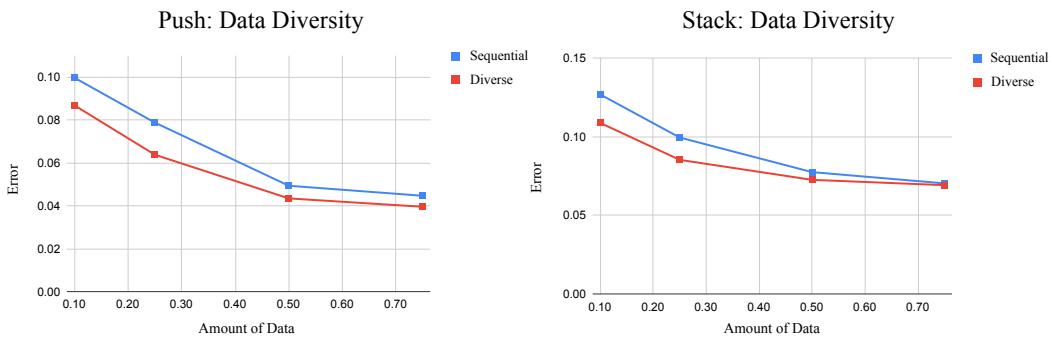


Figure 10: We show a comparison of performance between a sequential split and a random split, dataset (A) and dataset (B) respectively. In both tasks across all fractions of data, diverse data has much better performance.

J Closed-loop Control with Moving Objects

We have shown that our DemoAT framework can solve complex tasks in diverse domains, and further investigate whether our learned policies are robust to disturbances. We perturb the objects and goals during online robot execution and find that our closed-loop policy is still able to successfully complete both tasks. Results are shown in Fig. 11 and in the provided supplementary video on our project website.

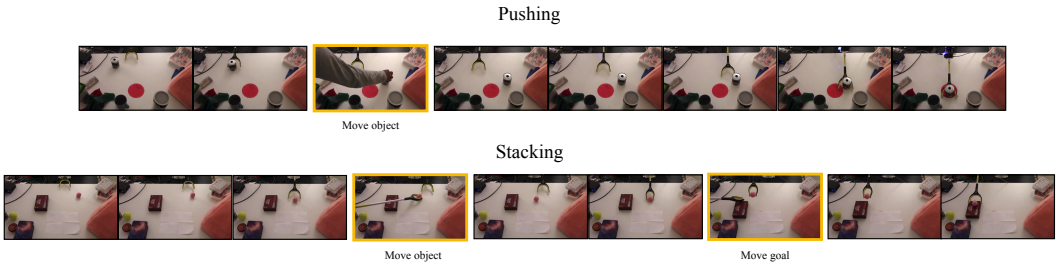


Figure 11: Here, we demonstrate how our learned closed-loop policies are robust to disturbances applied on the objects. When we slightly move the object or goal location, our policy immediately learns to adapt to the new scene. The frames where we apply a perturbation to the scene are highlighted in yellow.

K Third-person Views of Robot Experiments

We show additional robot trajectories for both the pushing and stacking task from a third person point of view in Fig. 12. These experiments are run with the best data augmentations for each respective task. In the pushing task, the most common reason for failure is the gripper not fully wrapping around the object such that it slides out of its fingers during execution. In the stacking task, we note that common causes of failure are that the policy often grasps too late or it does not lift the object high enough to successfully stack onto the second object.

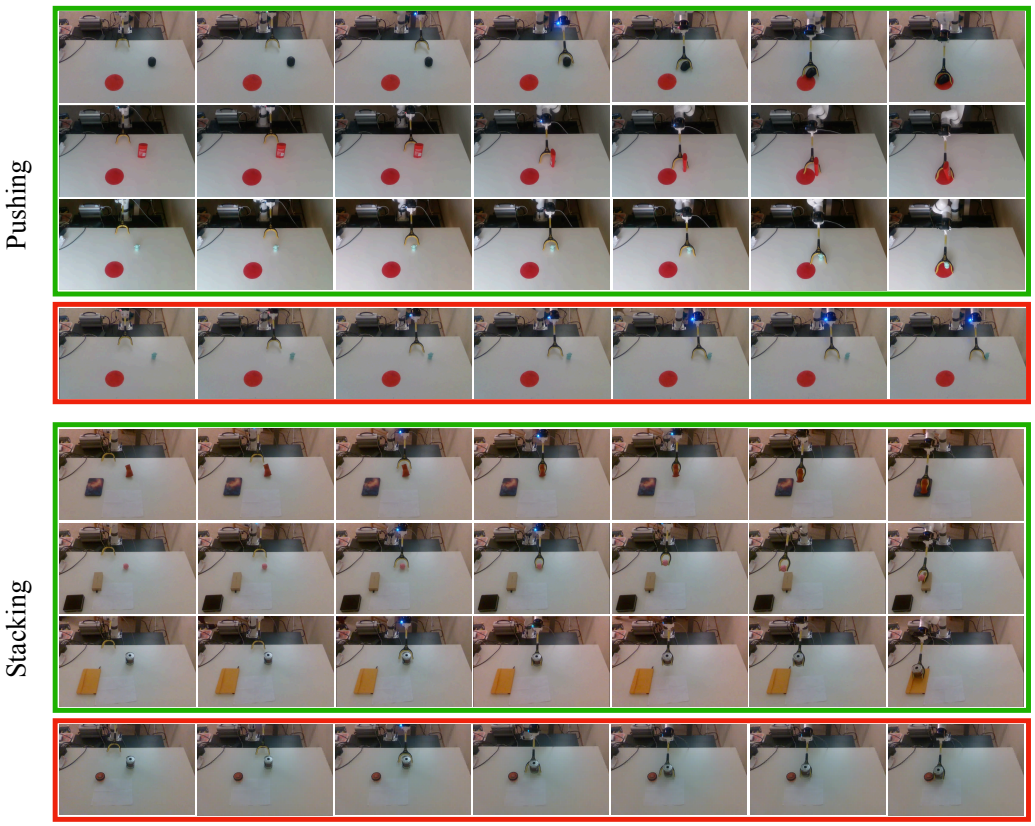


Figure 12: We visualize additional trajectories executed on the robot using our learned pushing and stacking policies from a third person point of view. Successful trajectories are highlighted in green, unsuccessful ones in red.