Training a Robot Arm to Scoop Beans

Isaac Hirsch, Matthew LeRobot, Chad Schmerling

May 29, 2025

Abstract

We successfully trained a low-cost robot arm to autonomously dig holes in a plastic bin filled with uncooked pinto beans. This was accomplished by using teleoperation to collect a small set of expert data and then training an ACT policy using imitation learning. After training for 13,500 steps, our policy manages to dig a hole in the beans every time. We also demonstrate that our policy has learned to incorporate visual feedback.

1 Introduction

Manipulations of robot arms has been an area of significant research. Recent availability of low-cost robotics arms[2, 1] ($< \$500$) has caused a demand for policies that run on this hardware. Our research investigates the intersection of low-cost robotics with recent work on dexterous manipulation of granular environments. Most research on dexterous policies for robotics arms have been limited to the manipulation of rigid-body objects due to their simplicity.
We employ imitation learning, which formalizes learning from demonstrations as a supervised learning objective where policies are trained to predict next expert actions. Imitation learning has been an area of heavy research and encapsulates many algorithms.[4]

2 Task and Problem Formulation

We chose a simple task, scooping out beans, that we thought was reasonable in scope. We placed uncooked pinto beans into a small plastic bin and a neon green piece of construction paper underneath them. The goal was to scoop out beans from the center of the bin to expose the bright construction paper underneath. Our plan was to use a base of imitation learning with teleoperation to cold start reinforcement learning; however, we found that getting imitation learning to work was quite complex and time consuming. Had we continued on to perform reinforcement learning, our plan was to use the amount of green pixels in the image (a proxy for the amount of paper uncovered) as a reward for the agent.
Our project used the following major resources:

1. Koch robot follower arm
2. Koch robot leader arm
3. GoPro Hero Black 7
4. TriPod
5. Plastic bin
6. Uncooked pinto beans

The Koch robot arms are standard, small, low-cost arms for simple robotics projects.[1]

3 Environment Setup

We used an open table in the TTIC Duckietown Robotics Lab to set up our environment. See Figure 1. As the figure shows, we clamped down the follower arm next to the bin, which was velcroed down for stability. Insufficient bin security in early setups resulted in a high volume of beans unexpectedly translocating to the ground. The GoPro was set up in position to observe the beans and arm, with tape to mark its position. The GoPro required several specialized connectors in order to stream images; this generation of GoPros was not designed to act as a webcam, so we had to use a backdoor. The leader arm was setup close by for easy observation, with a 3D printed support beam to prop it up to a similar height as the follower arm (which had a larger platform). The leader arm was also clamped to the table for stability.
On the software side, we made use of Hugging Face's open source repository LeRobot.[2] LeRobot provided scripts for calibrating, teleoperating, training, and evaluating the robot arms, which allowed us to focus on getting the physical setup and hardware working correctly, along with the ability to modify the code for our particular purposes.

4 Data Collection

To teach our robot arm to scoop, we ran 29 training episodes, each 20 seconds long. During each episode, one of us would teleoperate the follower arm using the leader arm. We varied the starting position and movements to give the arm a reasonably wide base of movements to learn from. After 29 episodes, the follower arm motor started to overheat, and combining multiple sequences of training episodes in the LeRobot repo proved untenable. Thus, we decided to train on what we had already, which amounted to almost 10 minutes of content. We had confidence in our decision since the LeRobot repo suggested 50 episodes

[1] https://github.com/AlexanderKoch-Koch/low_cost_robot
[2] https://github.com/huggingface/lerobot

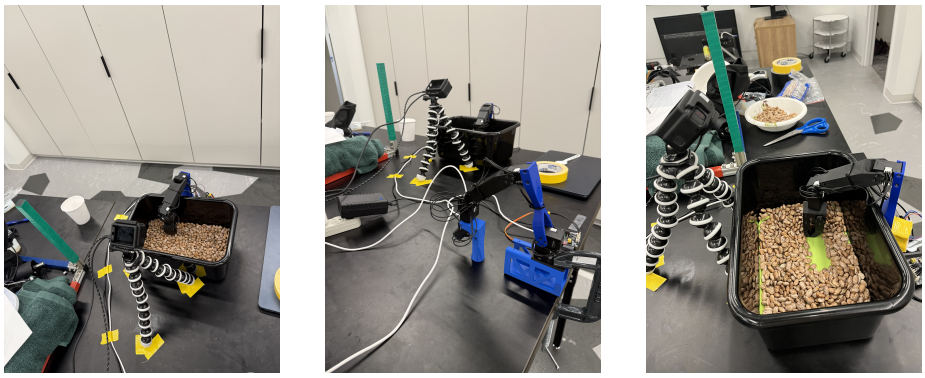

Figure 1: Several views of the environment setup

for picking up a block and placing it in a bin (a more precise task than ours), so we believed that 29 training episodes was reasonable for our less difficult task.

5 Learning Method

We used Action Chunking with Transformers (ACT)[7] as our policy model. ACT is a transformer-based architecture[6] that relies on ResNET-18[3] with pretrained weights to embed observations. We trained our model to predict the next k actuator poses given the current actuator pose and image caputured by the GoPro. We used an L2 loss between the predicted and recorded posses, as well as an L2 weight decay term and the β-VAE loss term.
Our model was trained using the AdamW optimizer[5] with a learning rate of 10^{-5} and a weight decay of 10^{-4}. We trained our model for 13,500 steps, which took 4 hours on a single NVIDIA A100 GPU.

6 Results

To expedite the training process, we initially employed a reduced-complexity version of the ACT model. Despite still containing 11 million parameters, this simplified model proved incapable of effective learning. We trained it up to 7,000 training steps. Figure 2 shows the lackluster training results in graph form at training steps 1,500 to 7,00; Figure 3 indicates the "goal" to which the graphs should be converging. The graphs of the robot's motion did not make distinctively sinusoidal patterns like the ones present during imitation learning. Instead, the graphs stayed flat, indicating that the robot failed to move in a meaningful way. This is confirmed by visual inspection — the robot jittered rapidly but largely remained in place, as if it were shivering on a cold Chicago day.

Consequently, we reverted to the author's original specifications, which in-

volved training 45 million parameters. The parameter increase came overwhelmingly from the transformer, which we had previously shrunk considerably for computational simplicity. This more comprehensive configuration resulted in substantially improved learning performance. Figure 4 shows that the robot started to move meaningfully by training step 6,200, and its graph started to match the goal around step 9,600. By step 11,300, the graph shows distinctive sinusoidal patterns — the result of a robot performing our desired shoveling motions (as shown in Figure 5). However, while the robot was moving correctly at this stage, it would avoid making contact with the beans (i.e., shoveling). This result surprised us because the follower arm moved the beans in each of our 29 training episodes very intentionally. In addition, we confirmed that our calibration was correct by replaying a training episode and observing the robot moving as intended.

The only place that we could have made a mistake was the camera, so we compared the image generated in evaluation and the ones recorded in training. To give a sense for this process of discovery, compare the camera view from epoch 11,300 in Figure 5 and the one from an episode of training in Figure 6. There is a clear difference in the angle! The camera had to be tilted up to return to its original training state.

Figure 7 shows the camera's view after adjusting it at training step 13,500, when the robot finally begins scooping the beans. Note that the camera angle in Figure 7 is distinct from that in prior evaluation steps (e.g., Figure 5) and closely matches the angle during training (e.g., Figure 6). This imitation is illustrated graphically in Figure 8, where the robot very closely matches the movement during our manual training over the training episode. Our resounding success shows the power of imitation learning using only 10 minutes of movements, a few hours of training, and less than $10 of GPU on Google Colab.

7 Conclusion

There were two major limitations of our project. First, we used a fairly small set of training episodes. While this was enough to get good performance, it's likely that a larger set would have resulted in better performance. Secondly, and relatedly, we did not have the opportunity to perform reinforcement learning to improve the robot's behavior. It would have been interesting to see if an RL trained robot would come up with scooping strategies that are counterintuitive or surprising to humans, like AlphaGo's infamous move 37. Additionally, it would have been illuminating to produce our own code to further hone our understanding of reinforcement learning; unfortunately, time constraints prevented all three of these further directions.

Overall, we were very pleased with the performance of the arm based on just imitation learning. While simulations are a fantastic tool for RL, it was extremely satisfying to see the results of our work in the real world. This provided an exciting way to put the theory we learned in class to use in a physical environment.

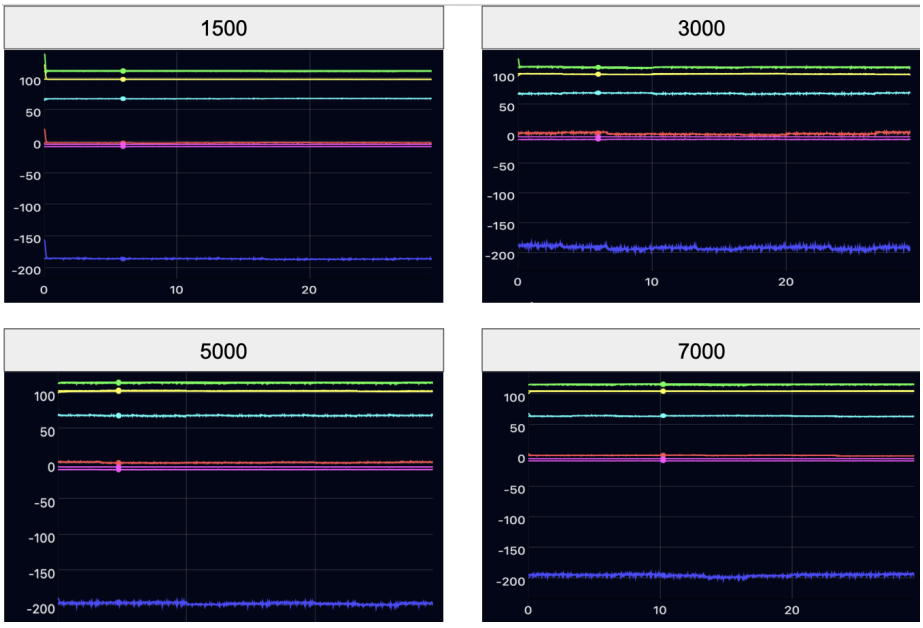

Figure 2: Graph of poor runs in first iteration

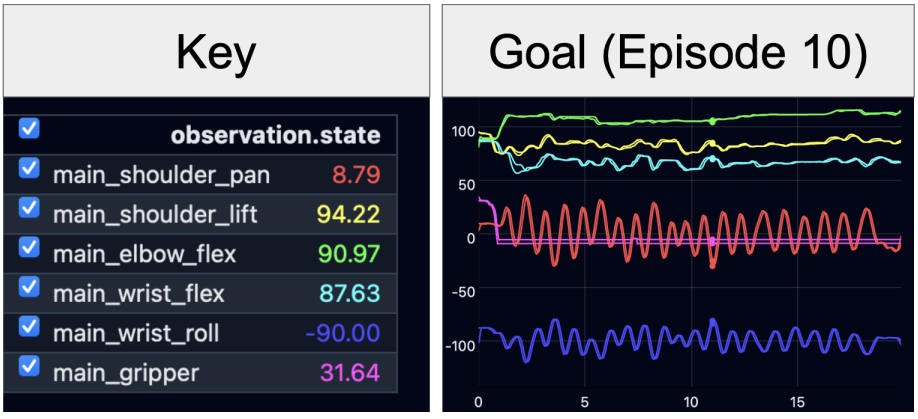

Figure 3: Graph of goal (1 sample of a training episode)

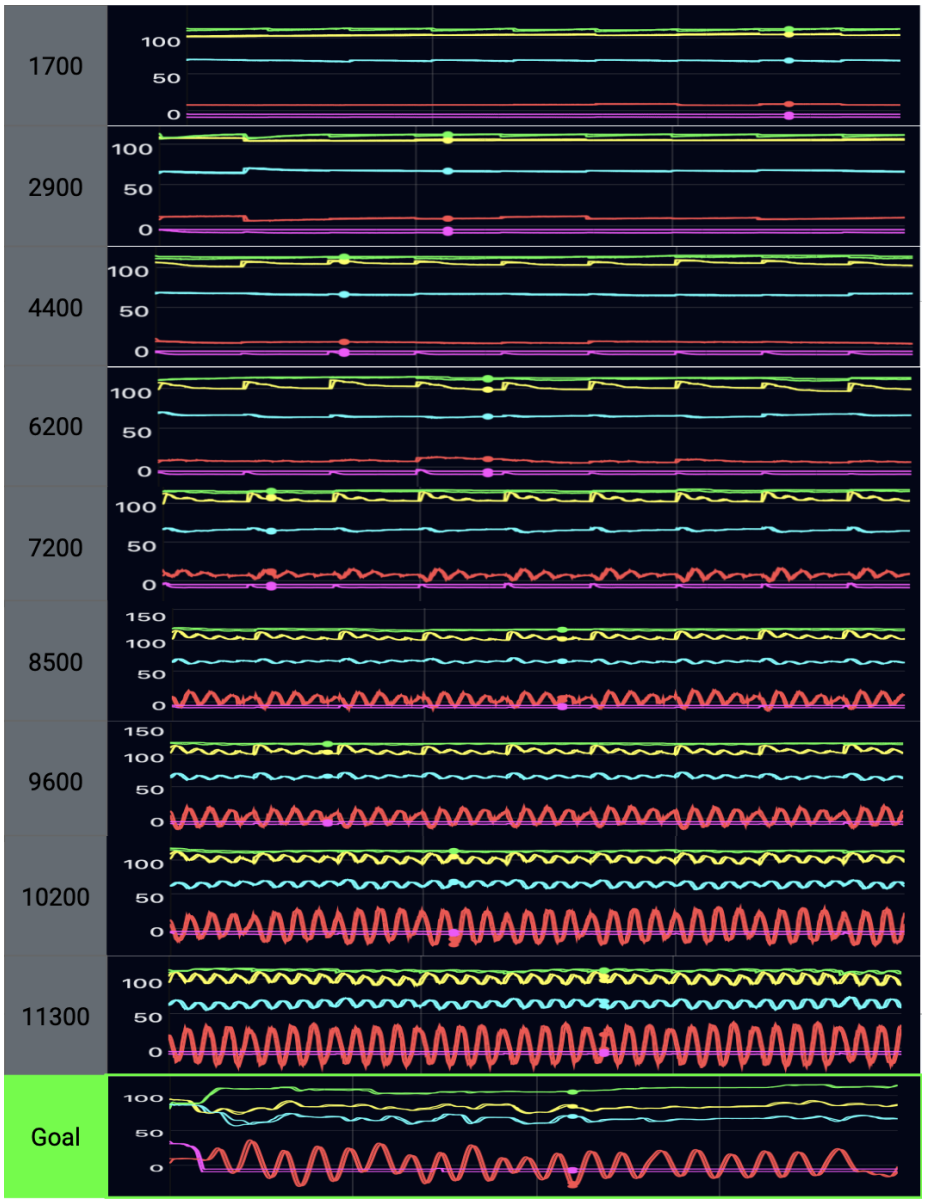

Figure 4: Progression of robot graph in second iteration over various training steps

Figure 5: Visualization of robot after 11,300 training steps

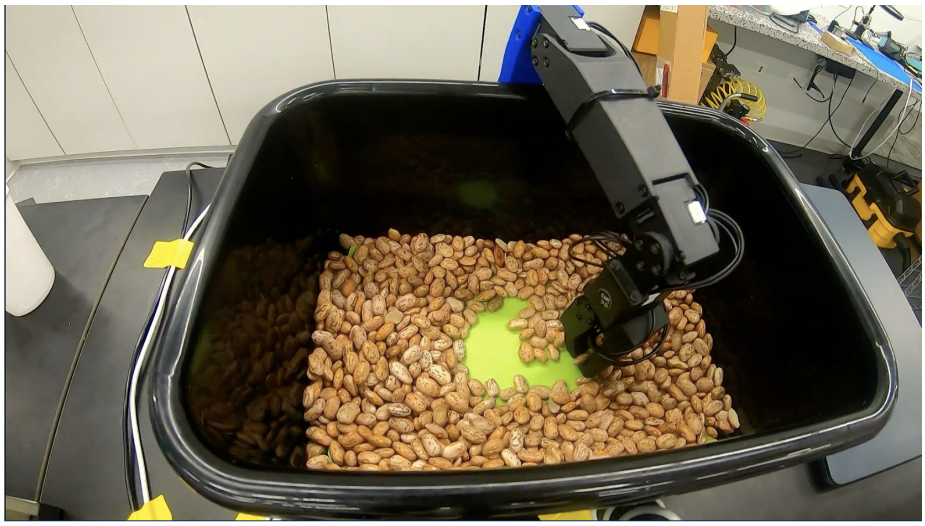

Figure 6: Visualization of expert data from GoPro perspective

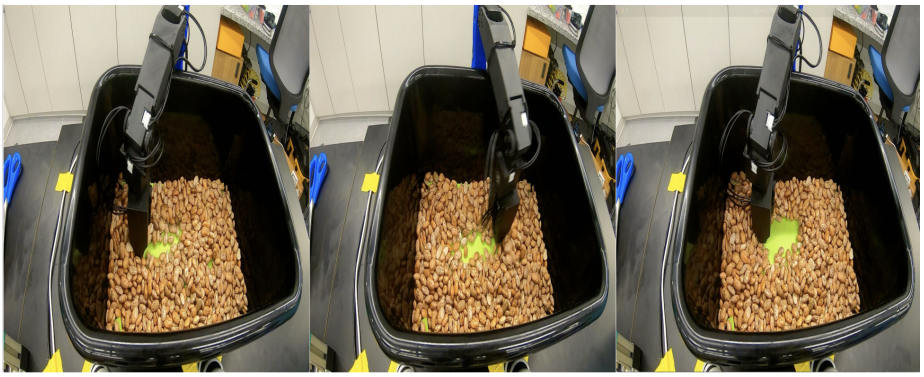

Figure 7: Visualization of robot after 13,500 training steps

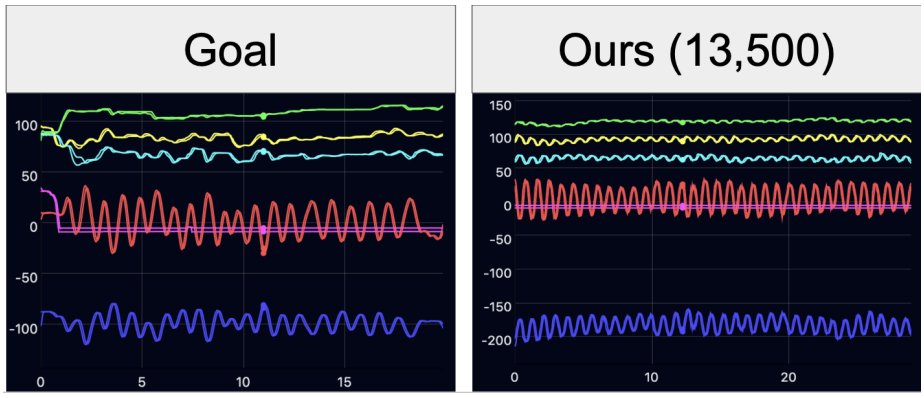

Figure 8: Robot graph at 13,500 training steps vs Goal

References

[1] Zain Ali, Muhammad Sheikh, Ans Al Rashid, Zia Arif, Muhammad Yasir Khalid, Rehan Umer, and Muammer Koç. Design and development of a low-cost 5-dof robotic arm for lightweight material handling and sorting applications: A case study for small manufacturing industries of pakistan. *Results in Engineering*, page 101315, July 2023.

[2] Kasey Moomau Amie Sueann Sommers Markeya S. Peteranetz Eric Markvicka, Jason Daniel Finnegan and Tareq A. Daher. Designing learning experiences with a low-cost robotic arm. In *2023 ASEE Annual Conference & Exposition*, number 10.18260/1-2–42983, Baltimore , Maryland, June 2023. ASEE Conferences. https://peer.asee.org/42983.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[4] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2), April 2017.

[5] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

[7] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023.