Closing the Domain Gap for Data-Efficient Robotic Learning

Sarah Young

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-62
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-62.html

May 26, 2020

Acknowledgement

I would like to thank my advisor, Professor Pieter Abbeel, for his continuing support, advice, and the opportunity to work in the Robot Learning Lab. I am also extremely grateful to Lerrel Pinto for his mentorship, guidance, and assistance. I've been able to learn so much in lab alongside the PR2! This experience has helped me grow both as a researcher and as an engineer. Lastly, thank you to Dhiraj Gandhi and Shubham Tulsiani for helping me on this project.

Closing the Domain Gap for Data-Efficient Robotic Learning

by

Sarah Young

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Pieter Abbeel, Chair

Spring 2020

Closing the Domain Gap for Data-Efficient Robotic Learning

by Sarah Young

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Pieter Abbeel
Research Advisor

26-MAY-2020

(Date)

* * * * * * *

Professor Sergey Levine
Second Reader

May 26, 2020

(Date)

Closing the Domain Gap for Data-Efficient Robotic Learning

Acknowledgments

I would like to thank my advisor, Professor Pieter Abbeel, for his continuing support, advice, and the opportunity to work in the Robot Learning Lab. I am also extremely grateful to Lerrel Pinto for his mentorship, guidance, and assistance. I've been able to learn so much in lab alongside the PR2! This experience has helped me grow both as a researcher and as an engineer. Lastly, thank you to Dhiraj Gandhi and Shubham Tulsiani for helping me on this project.

Abstract

Closing the Domain Gap for Data-Efficient Robotic Learning

by

Sarah Young

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Pieter Abbeel, Chair

Object manipulation has always been an important task in robotics. Though there are various methods for learning to do such tasks, imitation learning is one method that has been immensely successful in allowing agents to learn through human demonstrations. One of the key challenges of learning such tasks is working with the domain gap. Getting large scale kinesthetic data on robots doing real world tasks is tedious, and while getting third person data is much easier, there is a domain gap to resolve. In this work, we present a method to simplify the data collection process while eliminating the domain gap present in third-person demonstrations. We focus on using one universal grasping tool that can attach to a variety of robots and perform a variety of tasks. We present a trash-bot setup where we are able to easily collect first-person demonstration videos. Our goal is to learn to reach, push, and place objects in the scene in diverse environments. To train this task, we collected demonstration data in various different environments and objects using our grasping tool. We then learn a policy to output the appropriate actions for completing the task.

To my family for their endless support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Learning to manipulate objects are vital and challenging tasks for robots. While there has been significant work done on grasping objects [4], there has not been a focus on the way in which we can allow robots to universally grasp, push, or place objects easily. Furthermore, many of these object manipulation setups are very complicated. In this paper, we tackle the challenge of using a universal grasping tool to simplify the data collection process while preserving the domain on a simple push and place task.

Imitation learning has been proven to be successful in learning such tasks. This method involves an agent learning to perform a task from demonstration rather than learning a reward function or learning a policy directly. One of the key challenges of learning via imitation learning is working with the domain gap present in demonstration data. Traditionally, first-person demonstrations are provided via kinesthetic teaching. However, getting kinesthetic demonstration data on real robots is extremely costly and inefficient. To avoid this, we can use third-person imitation learning [6]. Instead of using kinesthetic teaching, this method provides a teacher demonstration achieving the same goal, but from a different viewpoint. While this mitigates the inefficient and difficult data collection process, it introduces a new challenge – a domain gap in the viewpoints of data seen from the robot's point of view and the data collected from the human.

In this work, we propose a setup that can perform various tasks and collect more efficient demonstrations without any domain gap. Furthermore, we do not need to work with a specific robot and train based on its gripper – we learn these tasks on the tool instead and the tool can attach to any robot and learn from a single agent. The tasks we focus on in this work are a reach and push and place task. In the reach task, sing the grasping tool, the agent learns to reach a specified goal location. In the push task, the agent must initially move towards the object before pushing it towards the goal. Data collection involves a human holding the grasping tool and performing the same tasks. The camera is attached onto the tool for both the robot and the human, such that the viewpoint is consistent. With this data, the agent learns a policy which takes in a single image as input and outputs the correct action to take.

Chapter 2

Background and Related Work

2.1 Imitation Learning

Imitation learning is a branch of AI in which an agent learns decision policies through expert demonstrations, similar to how humans often learn. Imitation learning comes in two main forms, behavioral cloning and inverse reinforcement learning. Inverse reinforcement learning learns a reward function, while behavioral cloning uses supervised learning to extract actions. In our case, we learn a such a policy through supervised learning. Given a set of observation and action pairs (o_t, a_t) from human demonstrations, we can learn a policy π_θ.

1. Collect human demonstrations (τ trajectories).

$$\{(o_1, o_2, o_3, ...), (o_1, o_2, o_3), ...\}$$

2. Label the observations with the ground truth to form state-action pairs.

$$D = \{(o_1, a_1), (o_2, a_2), ...(a_n, o_n)\}$$

3. Learn a policy $\pi_\theta(a_t|o_t)$ by minimizing a loss function $L(a, \pi_\theta(s))$

In the environment we have set up, the agent learns two tasks, reaching and pushing. Reaching involves the agent learning to reach a target location, and pushing adds an object into the scene, and the agent must reach towards the object first before pushing it into the goal, in our case a red circle. We provide human demonstrations in the form of videos, which gives the agent the current and goal observations.

2.2 Related Work

The following papers detail similar lines of work.

Rope Manipulation[3]

There has been significant work done in imitation learning and object manipulation. Much of the work we have done is based off of key ideas in this paper. We began with implementing a low-level inverse model and training with high level human demonstrations. This paper uses a pre-trained Alex-net to retrieve actions from images for the trash-bot environment. However, our action space is different and we are training on a variety of different environments, so our model dynamics are slightly different as well. A big takeaway from this paper is that human imitation is important, leading to 16% more successes.

Grasping in the Wild[5]

This paper has a similar but much more complicated setup with the same grasping tool that we use. It has a servo motor, compute stick, and one camera in addition to other parts in an attempt to minimize the domain gap. Despite the difference in setup, we use a similar approach of recovering true actions. Another difference is that their goal is to learn to grasp objects rather than place them, which introduces different challenges. This paper's approach is to a 6DoF closed-loop grasping model, which uses off-policy Q-learning to train visual grasping value functions. Their experimental setup also requires them to work with a small domain gap, since their grasping tool is used to simulate a robot's arm and there are small differences.

Learning from Virtual Reality Teleoperation[5]

This work addresses the same challenge of obtaining demonstration data more efficiently using a Virtual Reality teleoperation system. They learn a variety of tasks, including reaching, pushing, grasping, and dropping objects among others. However, each task is trained on the same set of objects and environments, whereas we are looking to learn a policy that can perform well in more diverse environments. Another key difference is that their input data includes color as well as depth images, whereas we only provide color images from a simple GoPro camera.

Chapter 3

Setup

3.1 Trash-stick

Our experimental setup consists of a simple 19-inch RMS (Royal Medical Solutions) plastic grabber tool. We attach an angled mount above the stick to hold a Go-Pro camera in place. There is a lever behind the mounts, which allows the tool to close on an object. We use this setup for all our data collection. However, in the task we present in this project, we do not make use of the lever. The setup is illustrated in Figure 3.1 below.

Figure 3.1: Trash-stick setup

It is important to note that this setup allows us to eliminate the domain gap that is present in [3] and [5]. The Go-Pro attachment is consistent across our data collection and

testing time on the robot. As seen in figure 3.1, the observation from the robot is the same as what the Go-Pro would see if a human were collecting data. This makes it much easier for the model to learn the task at hand.

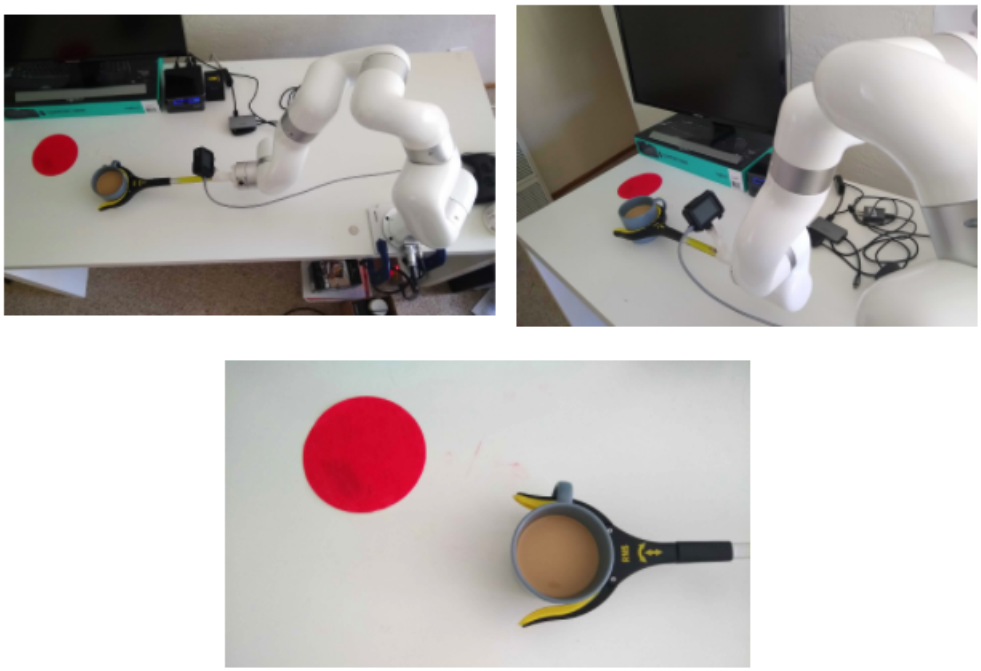

Figure 3.2: Robot setup

3.2 Task

The task is to learn how to push objects to a goal location. In our experiments, we use various different objects and a single goal type, a red circle (approximately 5 inches in diameter). We vary the starting locations of the trash stick, the object locations, as well as the goal locations. Figure 3.3 shows an example trajectory.

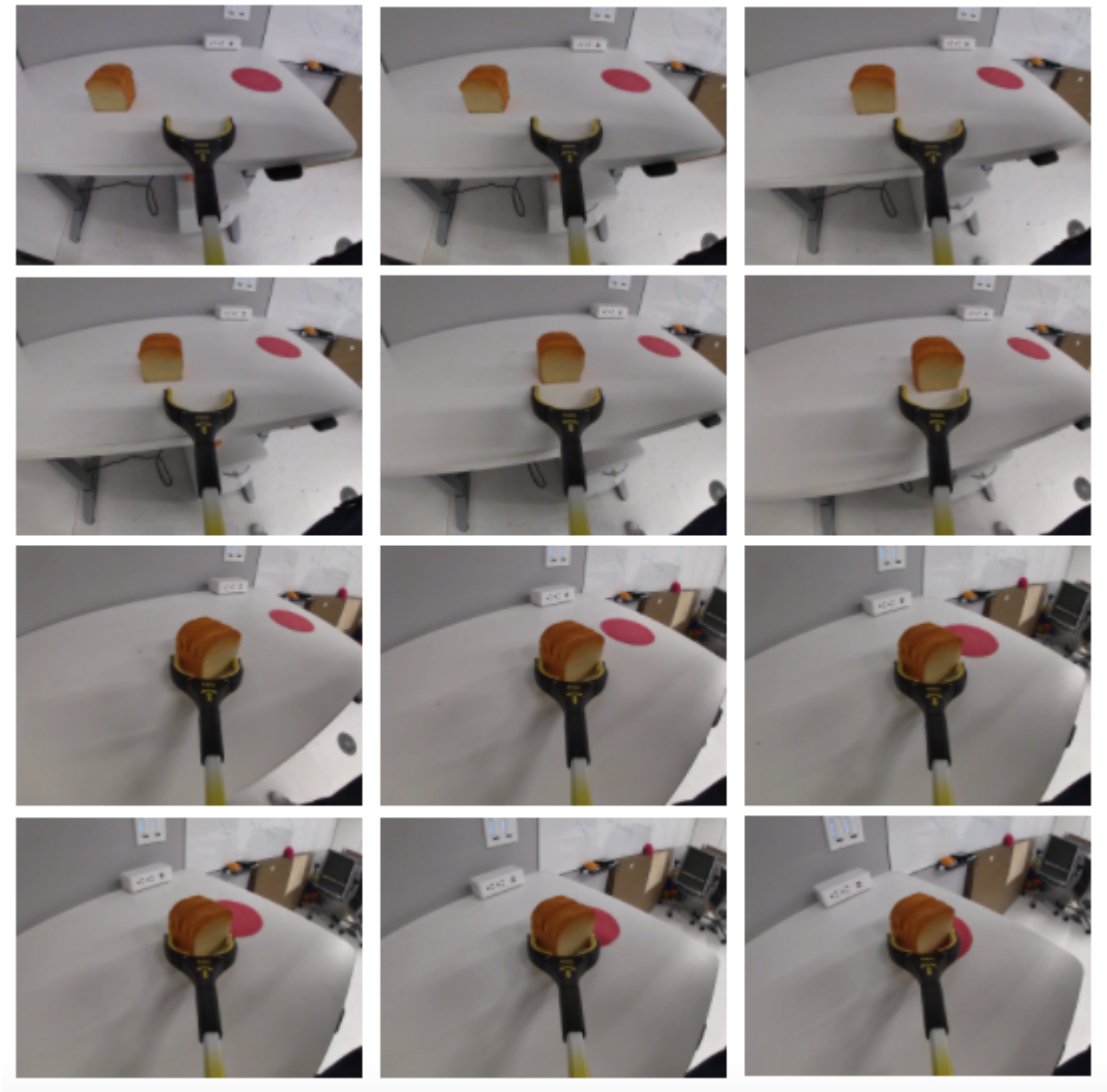

Figure 3.3: Example trajectory

Chapter 4

Data Collection

4.1 Demonstrations

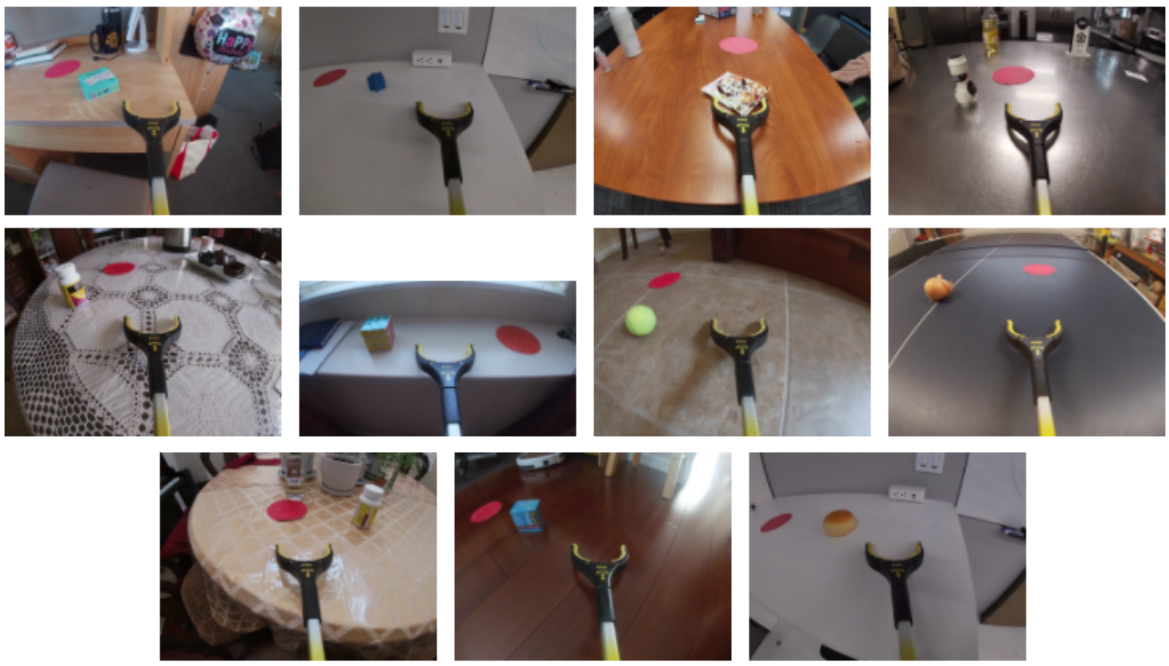

Figure 4.1: Push Task: Some different example environments where we collected data. Each environment consists of a red goal location, an object, and a start position.

We collected training data with the trash-stick setup in various different environments. In this report, we will mainly focus on the push task, as it is a more difficult version of the reach task. We have training data of over 600 trajectories for the push task. As seen in the examples in Figure 4.1, every trajectory consists of a goal location (red circle), an object (for the push task only), and a starting location for the trash stick. We took short 5 - 10 second videos of a human using the grasping tool to push different objects onto the red circle. The videos begin with the grasping tool at a start location, then moving towards the object, and then pushing the object towards the red circle. For the reach task, we directly move the tool towards the goal location. We first learn to do the simpler reach task before trying our model on the push task. The videos were then split into frames at a frame rate of 4 frames per second to use as observations.

Figure 4.2 displays the distribution of translations in the x direction after each translation is scaled by the largest absolute translation in its trajectory for the push task. We want to keep the distribution of positive and negative x-values roughly balanced. About 50% of the data has x values very close to 0, meaning that a large portion of movements were moving the stick forward in the y direction.

4.2 Data Labeling and Cleaning

We recovered the true actions for our training data for both the push and reach task by running COLMAP [1], a Structure-from-Motion (SfM) software on our data. The COLMAP reconstruction gives us the pose of each frame in the form of a quaternion (QW, QX, QY, QZ) and a translation vector (TX, TY, TZ). From this, we calculate the relative translations and rotations between consecutive images. Figure 4.3 is an example of a trajectory labeled with ground truth actions for the push task.

[1]COLMAP is a general-purpose Structure-from-Motion (SfM) and Multi-View Stereo (MVS) pipeline. https://colmap.github.io/index.html

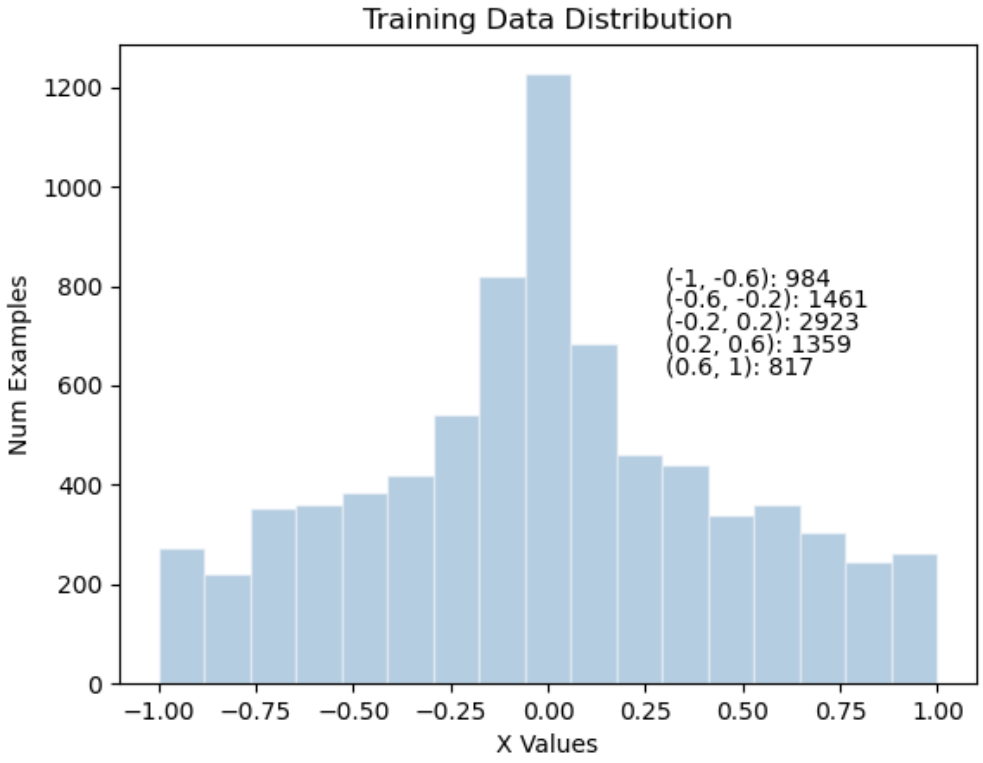

Figure 4.2: A histogram of the data distribution of translations in the x-direction for the push task.

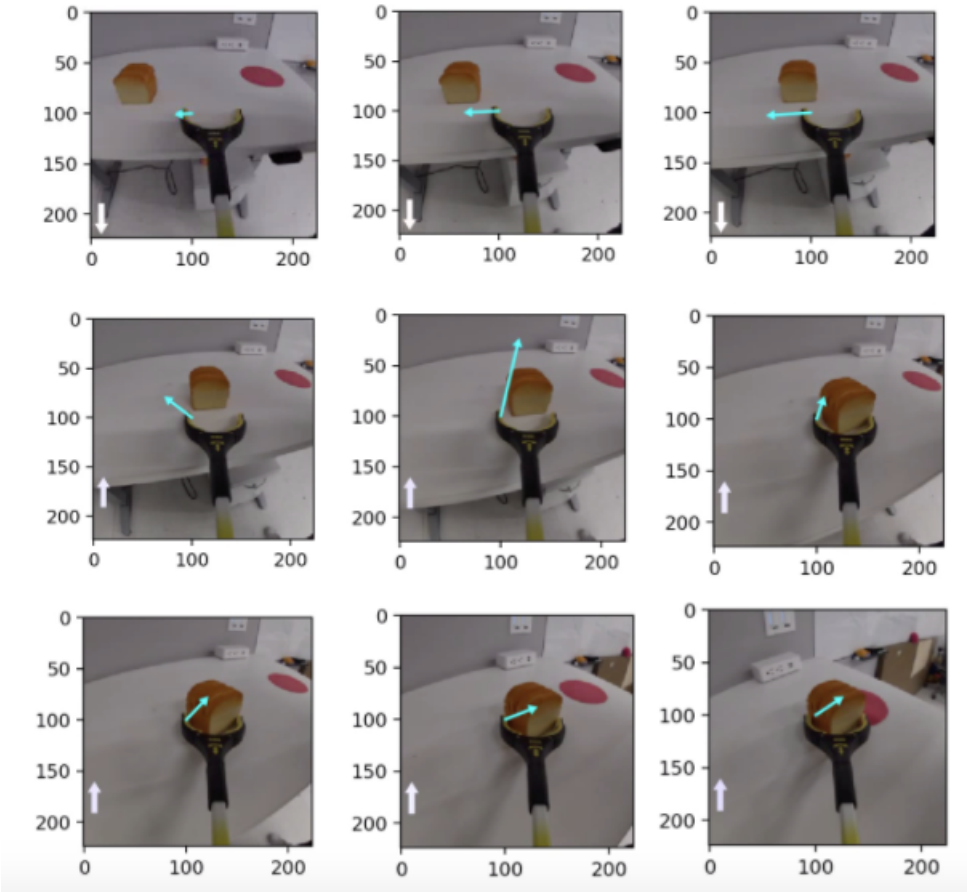

Figure 4.3: Frames labeled with ground truth actions.

Chapter 5

Methods

5.1 Architecture

We chose to use the following simple dynamics model.

$$u_t = F(I_t) \tag{5.1}$$

The notation is as follows: C-x is a convolutional layer with x filters, and F-x is a fully-connected layer with x filers.

We experimented with 3 different architectures to get the latent space representations of the images: a small CNN, a pretrained AlexNet [2], and a pretrained ResNet-18 [1]. We use an image I_t of size 224 x 224 and feed it into one of the following networks, which outputs its latent space representations x_t. During training, we add random jitter and random crops of the images for better generalization.

a) The small neural net has 6 convolutional layers followed by a fully connected layer. Each convolutional layer is followed by a ReLu layer. The convolutional layers have the following architecture:

$$C16 - C32 - C64 - C64 - C128 - C128 \tag{5.2}$$

b) AlexNet: we use the first 5 layers of the AlexNet and append another convolutional layer, followed by a ReLu and MaxPool layer.

c) ResNet18: we use all the convolutional layers of the ResNet and freeze the weights on the first 11 layers.

The latent space representation is then fed into a fully connected layer to output the predicted actions p_t, a vector (x, y, z). The fully connected layer projects from the output size to the feature length, 3, and then is followed by a tanh layer.

For rotations, we concatenate the latent space representation of the image with the predicted translations and feed the result into a linear layer of size 512 and a ReLu, which then gets projected to a 6D representation of the angle. We train on a 6D rotation representation [8] because it is continuous in the real Euclidean space and thus more suitable for learning as

opposed to traditional 3D or 4D rotations. The full architecture is illustrated in the Figure 5.1.

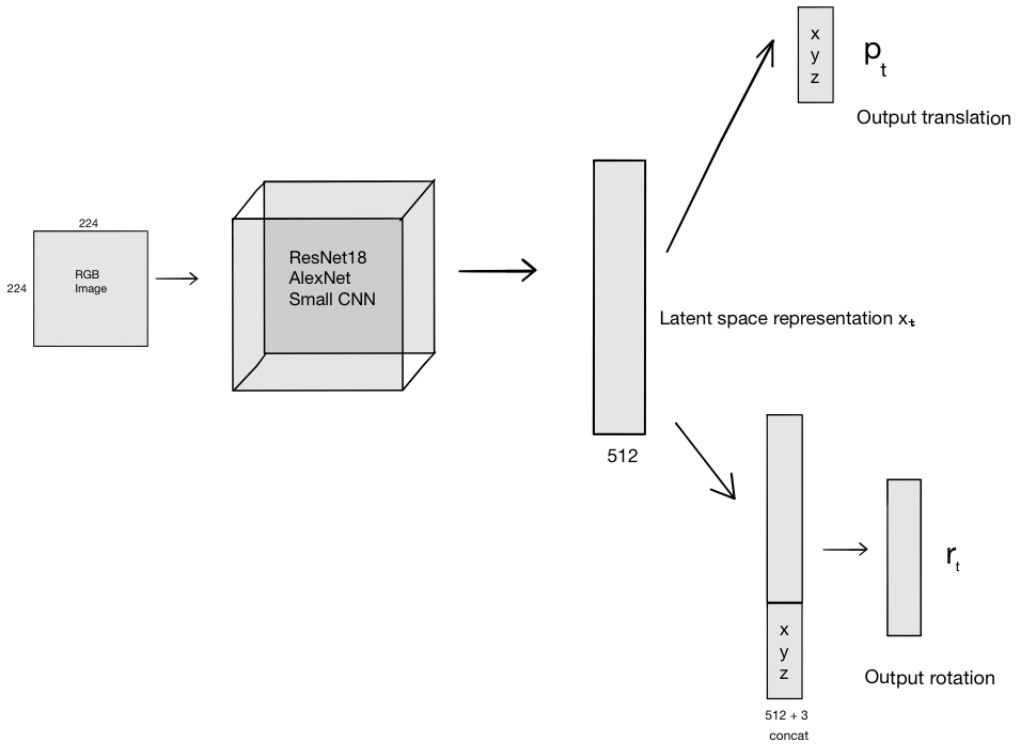

Figure 5.1: Network Architecture: Convolutional layers followed by a fully connected layer. For translations, the latent space representation is projected to a 3-dimensional vector. For rotations, output is latent space representation is concatenated with translations, and then projected to a 6-dimensional vector.

5.2 Action Space

The action space representing the movement from image I_t to I_{t+1} is of the form (x, y, z) and $(q_t, q_u, q_v, q_w, q_x, q_y)$, where x, y, z are the relative translations to get from I_t to I_{t+1} and $(q_t, q_u, q_v, q_w, q_x, q_y)$ are the relative rotations between two consecutive images. We use a continuous 6D angle representation for better learning and output a 3D rotation matrix. The orientation of the translation axes is shown in figure 5.2.

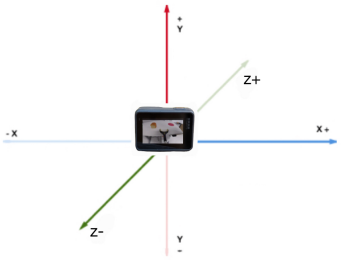

Figure 5.2: Orientation

To account for inconsistencies between COLMAP outputs for different trajectories, we scale the translations by the largest absolute action for each trajectory. Each translation becomes

$$T = \{(\frac{x_1}{t_{max}}, \frac{y_1}{t_{max}}, \frac{z_1}{t_{max}}), (\frac{x_2}{t_{max}}, \frac{y_2}{t_{max}}, \frac{z_2}{t_{max}}), ...\}$$

where t_{max} is the absolute value of the largest action in trajectory t. We then additionally scale each action by its L1-norm to focus more on the directional alignment rather than absolute magnitude.

5.3 Loss Functions

For translations, we use a combination of L1, L2, a direction loss, and a norm-loss. We want to encourage the directional alignment between the prediction and ground truth[7], so we add the following loss to minimize the cosine angle:

$$L_d = arccos(\frac{a_t^T \pi_\theta(o_t)}{||p_t|| ||\pi_\theta(o_t)||}) \tag{5.3}$$

The norm loss encourages translations (x, y, z) to have a norm of 1, since our ground truth labels are normalized as well. Our overall loss function is a weighted combination of the above losses for translation and the angle loss L_a. Because the magnitude of the angle loss is much smaller than that of the translation loss, we scale it by 10.

$$L = \alpha_1 * L1 + \alpha_2 * L2 + \alpha_3 * L_d + 10 * L_a \tag{5.4}$$

Chapter 6

Results

Through our experiments, we found that the AlexNet performs best on our data. We predict on an entirely new dataset with a different object and different background. Our model is able to successfully push objects to a goal location. The training error reaches 0.34, and the validation error reaches 0.45.

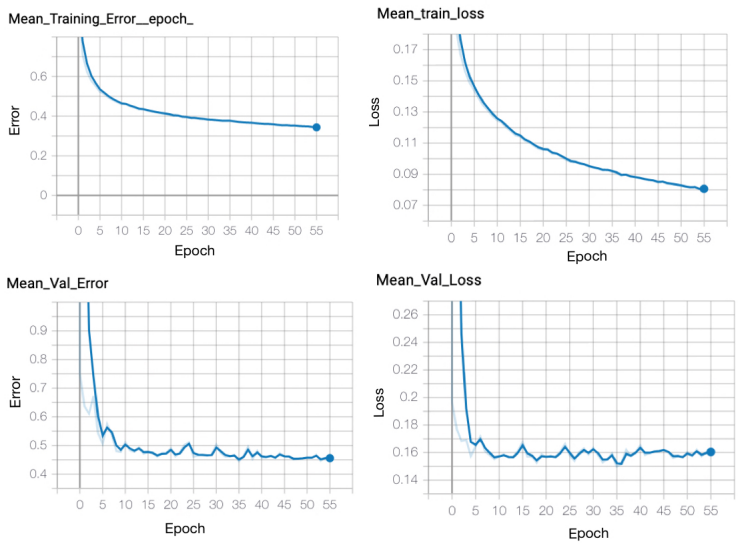

Figure 6.1: Training curves for our experiments using the pretrained AlexNet.

We ran experiments on the other two architectures as well and found that while our agent learns fairly well on the pretrained AlexNet, using a ResNet could give us better results if we have more data. We ran our experiments on a ResNet model, and found that the training

Table 6.1: Architecture Comparison.

Architecture	**Train Error**	**Validation Error**
No training	5.09	5.1
Small CNN	0.40	0.49
Pretrained ResNet18	0.18	3.3
Pretrained AlexNet	0.34	0.45

performs significantly better, but the model severely overfits and just memorizes the training data, and this is likely due to not having enough data. We ran an experiment without any learning to see the baseline error rate. Table 6.1 displays the results we found. To visualize the results qualitatively, we plotted the predicted actions on top of the images.

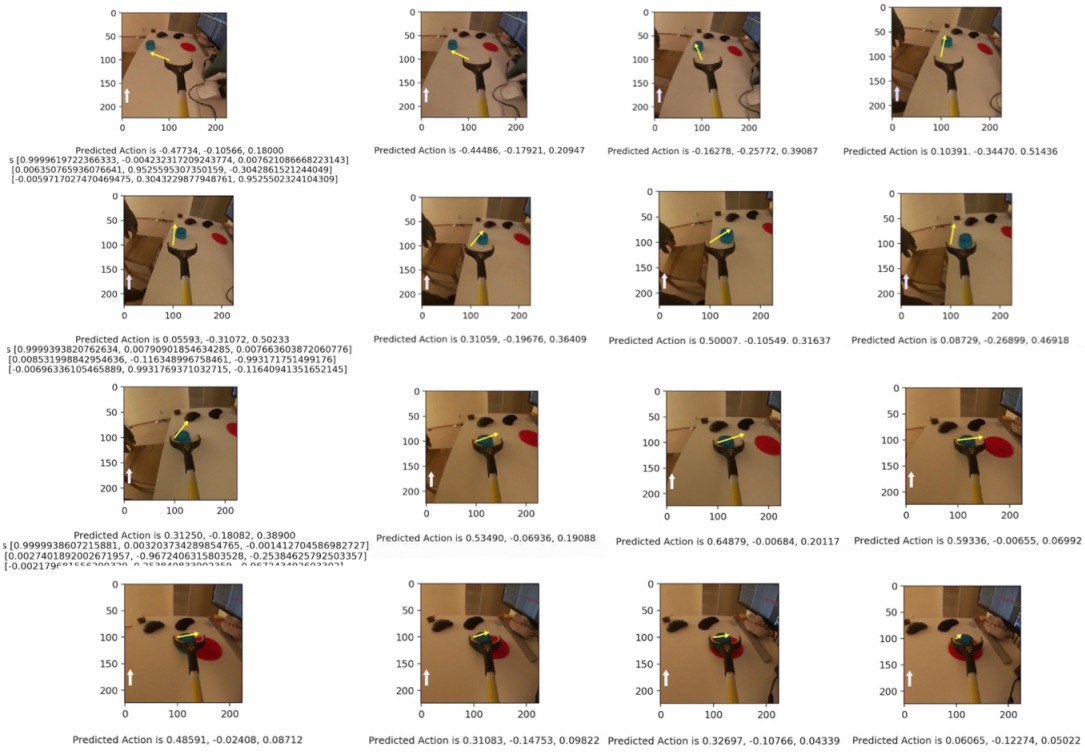

Figure 6.2: Frames from test results on a new environment with labeled translations and rotations (only 3 rotations are shown to save space). The yellow arrow in the images represents the xz translations, and the arrow at the corner of the image is the y (up-down) movement. It is difficult to visualize the relative rotation, but we try to minimize rotation in this trajectory and we can see that the rotation matrices roughly correspond to no rotation.

Chapter 7

Conclusion

In this work, we have trained an agent to learn a policy that can successfully reach and push an object to a goal location. Without introducing any domain-gap, we are able to more easily obtain human demonstration data with our simple grasping tool setup. We have greatly increased the efficiency of data collection, from kinesthetic teaching to holding a universal grasping stick to push around objects, and even pick up and place objects. With only around 40 minutes of training data and 600 trajectories for the pushing task, we are able to learn fairly well. Furthermore, we have shown that this method works in a variety of environments with different colored objects and different objects, start, and goal locations. Our results have shown that the agent has potential to learn even better with more data and a more complex model.

We are continuing to expand this project in several directions. We have found that using a more advanced network, (i.e. ResNet18 or ResNet34), leads to much better training performance. Due to severe over-fitting, we will need to collect more data to be able to use these more complex architectures. In the future, we would also like to expand to multiple different tasks, such as picking up and placing down objects. The final goal would be to combine several different methods to learn end-to-end trash-picking in diverse environments.

Bibliography

[1] K. He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *NIPS*. 2012.

[3] Ashvin Nair et al. "Combining Self-Supervised Learning and Imitation for Vision-Based Rope Manipulation". In: *CoRR* abs/1703.02018 (2017). arXiv: 1703.02018. URL: http://arxiv.org/abs/1703.02018.

[4] Lerrel Pinto and Abhinav Gupta. "Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours". In: *CoRR* abs/1509.06825 (2015). arXiv: 1509.06825. URL: http://arxiv.org/abs/1509.06825.

[5] Shuran Song et al. "Grasping in the Wild: Learning 6DoF Closed-Loop Grasping from Low-Cost Demonstrations". In: *CoRR* abs/1912.04344 (2019). arXiv: 1912.04344. URL: http://arxiv.org/abs/1912.04344.

[6] Bradly C. Stadie, Pieter Abbeel, and Ilya Sutskever. "Third-Person Imitation Learning". In: *CoRR* abs/1703.01703 (2017). arXiv: 1703.01703. URL: http://arxiv.org/abs/1703.01703.

[7] Tianhao Zhang et al. "Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation". In: *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*. IEEE, 2018, pp. 1–8. DOI: 10.1109/ICRA.2018.8461249. URL: https://doi.org/10.1109/ICRA.2018.8461249.

[8] Yi Zhou et al. "On the Continuity of Rotation Representations in Neural Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 5745–5753. DOI: 10.1109/CVPR.2019.00589. URL: http://openaccess.thecvf.com/content%5CCVPR%5C2019/html/Zhou%5COn%5Cthe%5CContinuity%5Cof%5CRotation%5CRepresentations%5Cin%5CNeural%5CNetworks%5CCVPR%5C2019%5Cpaper.html.