UC Berkeley
UC Berkeley Electronic Theses and Dissertations

Title
Multi-task Policy Learning with Minimal Human Supervision

Permalink
https://escholarship.org/uc/item/43q7p9ts

Author
Mahmoudieh, Parsa

Publication Date
2022

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

Multi-task Policy Learning with Minimal Human Supervision

by

Parsa Mahmoudieh

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Trevor Darrell, Chair
Professor Pieter Abbeel
Professor Sergey Levine
Professor Alison Gopnik

Summer 2022

Multi-task Policy Learning with Minimal Human Supervision

Copyright 2022
by
Parsa Mahmoudieh

Abstract

Multi-task Policy Learning with Minimal Human Supervision

by

Parsa Mahmoudieh

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Trevor Darrell, Chair

Multi-task policies enable a user to adjust their desired objective or task parameters without having to train a new policy for every new desired task. In order to train multi-task policies that can generalize to unseen tasks it is common to train them on a large repository of tasks. Tasks are commonly learned with demonstrations or reward functions. However, collecting human demonstrations or instrumenting reward functions for each new task is expensive and limits scaling of multi-task policies. How tasks are specified to multi-task policies is also an important dimension that can result in expensive labor during task communication. In this thesis we explore ways to learn and specify new tasks with minimal human supervision to enable more scalable multi-task policies.

Acknowledgments

I am so grateful to have been given the opportunity to pursue my PhD with my advisor Trevor Darrell. He has been so patient and supportive during all my highs and lows. He's helped me navigate one of the hardest and most important lessons I've learned during my PhD: Not giving up on a problem too early and finding the healthy middle ground between pursuit and moving on. I am grateful for the freedom and encouragement he gave me in pursuing the ideas that spoke most strongly to me. That freedom truly quenched my intellectual curiosity during my PhD. Thanks to my committee: Pieter Abbeel, Sergey Levine, and Alison Gopnik. Your passion and rigor in your research is truly inspiring. I am grateful for all the enriching discussions I have had the opportunity to have with you.

I am grateful to have been mentored by Evan Shelhamer during my 5th Year Masters and early PhD. I credit most of my foundational deep learning intuitions to Evan. His passionate, kind, and patient mentoring was one of the biggest factors in my decision to apply to PhD programs. I am grateful to have been mentored by Deepak Pathak throughout most of my PhD. I credit most of my matured research skills to him. Deepak's organized and consistent pursuit at solving problems is very admirable and has strongly influenced me. I want to also thank Pulkit Agrawal, Abhishek Gupta, and Sayna Ebrahimi for their mentorship during important moments of my PhD. Thanks to Alyosha Efros and Jitendra Malik who have made Computer Vision at UC Berkeley so enriching. Thanks to the rest of my collaborators Michael Luo, Dian Chen, and Fred Shentu for being so great to work with.

I also want to thank Ron Fearing and Cameron Rose who gave me such a great first exposure to research during my undergrad at Berkeley. Thanks to Andy Packard, Melanie Lutz, Zach Hannan, James Dekloe, John Higashi, and Maria Santiago whose passionate and enthusiastic teachings laid strong foundations for me during my undergraduate education.

The friends I made during my PhD is one of the sweetest parts of my PhD that I did not forsee including Vitchyr Pong, Kelvin Xu, Seth Park, Allan Jabri, Avi Singh, Erin Grant, Varun Tolani, Michael Chang, Thanard Kurutach, Suzie Petryk, Armin Askari, Dibya Ghosh, Ashvin Nair, Justin Fu, Kate Rakelly, Ignasi Clavera, Vicenc Rubies Royo, Carlos Florensa, Mostafa Rohaninejad, Melih Elibol, Frederik Ebert, Marvin Zhang, John D. Co-Reyes, Chandan Singh, Jasmine Collins, Sasha Sax, Ilija Radosavovic, Evan Shelhamer, Deepak Pathak, Pulkit Agrawal, Abhishek Gupta, Sayna Ebrahimi, Marcus Rohrbach, Anna Rohrbach, Rowan McAllister, Dinesh Jayaraman, Michael Janner, Dexter Scobee, Forrest Iandola, Somil Bansal, Daniel Fried, David McPherson, Richard Zhang, Angjoo Kanazawa, Weicheng Kuo, Rachit Dubey, Adam Stooke, Mayur Mudigonda, Ashish Kumar, Shubham Tulsiani, Saurabh Gupta, Georgia Gkioxari, Shiry Ginosar, Roberto Calandra, Gregory Kahn, Amir Zamir, Dequan Wang, Samaneh Azadi, Eric Tzeng, Coline Devin, Jon Long, Ronghang Hu, Xin Wang, Jeffrey Donahue, Lisa Hendricks, Kaylee Burns, Devin Guillory, Huazhe Xu, Yang Gao, Rudy Corona, Medhini Narasimhan, Shizhan Zhu, and many others.

Thanks to all of my dear friends outside of gradschool for your love and support. In particular I'd like to thank Danny, Royce, Max, Vishnu, Kunal, Ravi, Evan, Ardavan, Nick, Jimmie, Bryan, Xavier, Alana, Leslie, Esteban, Nathan, Edgar, Lucas, Andisheh, Sepehr,

Pouya, Paymon, Kiavash, Shadpoor, Shadfar, and Daniel Fallah. Thanks to Yasaman, Sina, Ehsan, and Daii for your countless invaluable hours of family bonding and friendship.

Zahra, thank you for always being there for me and for all the love and support you give no matter the circumstance. Thanks to my parents for supporting me all these years and for sacrificing so much to help me grow in every facet of life. Thanks to my little brother Aria for being so patient with me and always having my back. And thanks to the rest of my lovely family. I've been blessed with so much love and support all of my life.

Contents

Chapter 1

Introduction

Advances in deep learning have made significant impact in many domains including computer vision (i.e image classification [57, 42], detection [37, 43], and segmentation [66, 43]), natural language processing (i.e machine translation [114, 10], document sentiment analysis [124]), and controls (human level performance on Atari [73], superhuman performance on boardgame Go [110], datacenter cooling [64], stratospheric balloon control [14]). In all of these domains, supervision in the form of labelled data or rewarded experience is a critical part in training the deep models for their target tasks. In controls, we learn policies, which are models that take in observed state as input and output actions that lead to completing a desired task in some time horizon. Two of the most common approaches for learning tasks in controls is imitation learning and reinforcement learning.

In imitation learning the policy is optimized to imitate actions taken by an expert via supervised learning where expert actions are the supervised output and the observed state in which those actions were taken are the policy input. Collecting such data from an expert can be very expensive due to instrumentation engineering needed for recording expert actions and large number of expert human labor hours executing the target tasks needed for collecting a sufficiently diverse dataset (example shown in figure 1.1).

In reinforcement learning the policy is optimized for a target task via a reward function that provides a reward for every action taken in a observed state. The reward function is designed to provide higher reward for state, action pairs that lead to progressing towards completing the task and lower reward for state, action pairs that don't. Reward functions are not necessarily linear towards progress of the target task and can be designed in many different ways, but they are usually positively correlated towards task completion. Using this reward function, a policy is optimized to maximize expectation of discounted sum of returns which means it is optimized to maximize sum of rewards in an episode of experience in expectation.

One of the limitations of reinforcement learning however is that reward signals are often expensive to provide. One common method for providing reward is using a human scorer which results in expensive human labor. And another common method for providing reward is to instrument the environment to access direct state to be able to compute some measure

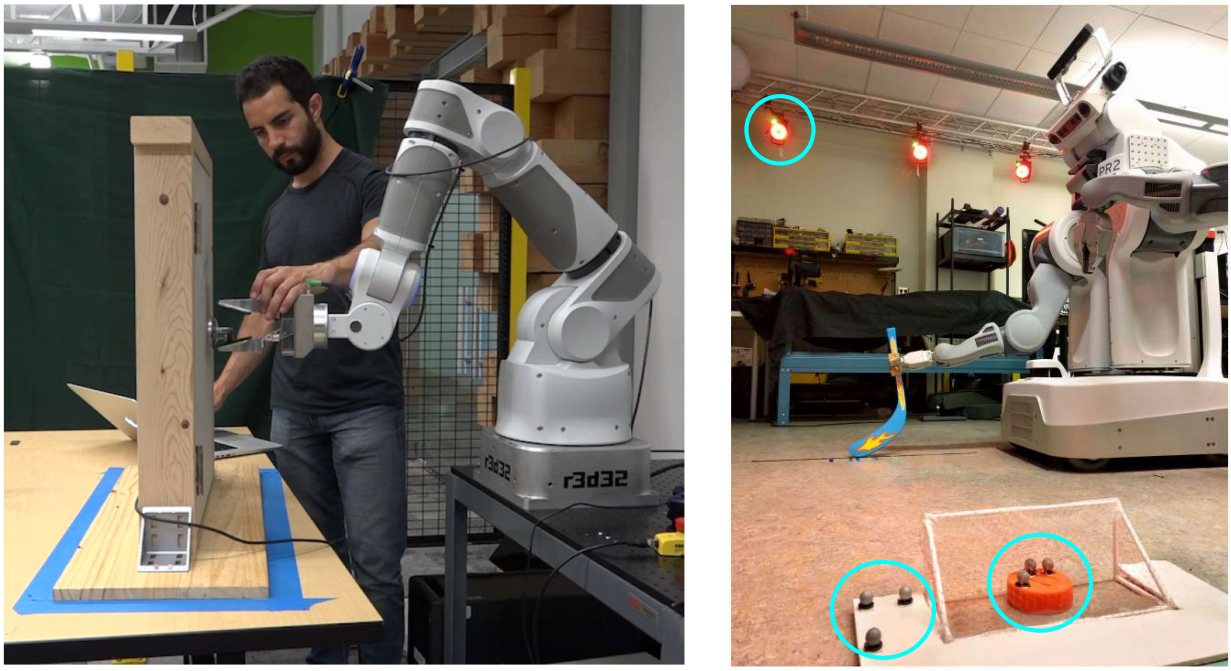


Figure 1.1: On the left image we can observe labor required for collecting expert demonstrations in [122]. On the right image we can observe labor required for instrumenting environment for providing reward signal in [19]. The top cyan circle encircles one of the motion capture cameras and the two bottom cyan circles encircle the gray spheres on the goal post and puck that are tracked by the cameras to compute reward based on the distance between the puck and goal post positions. (Please note that the left image is obtained from a video snapshot of [122] and the right image is obtained from figure 1 in [19] and modified with the cyan circles to highlight instrumentation.)

of progress towards the target task. In figure 1.1, you can see an example where the authors instrument their environment with a motion capture system to access accurate puck and goal positions to compute reward partially based on distance between the two to learn the task of striking the puck into the goal post [19].

In controls, multi-task policies are particularly useful since the user can specify different objectives and task parameters without needing to learn a separate policy per desired task. To learn multi-task policies that can generalize to unseen inputs (unseen observed states and or unseen task parameters), it is important to train them on a large repository of tasks. However, as discussed earlier, collecting expert demonstration or providing reward signals for learning tasks are both usually expensive and limit training on a large repository of tasks. In this thesis we will explore ways we can leverage cheaper sources of signal to enable more scalable multi-task policies.

Another important dimension in multi-task policies is labor required in task specification. For communicating a new task to a multi-task policy we can specify the task with:

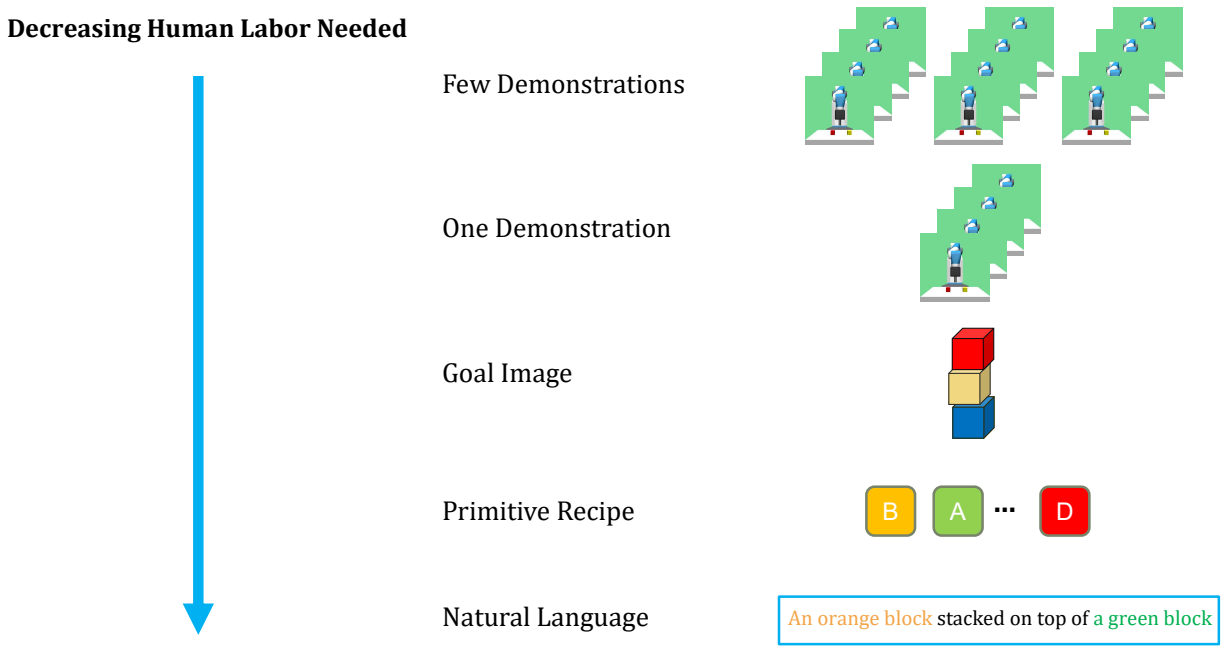


Figure 1.2: Illustration of different ways to specify tasks with decreasing human labor needed for communication from top to bottom.

1. Few demonstrations of the new task
2. One demonstration of the new task
3. Goal image(s) of the new task
4. Primitive recipe of the new task (Do B, then A, . . .)
5. Natural language description of the new task

This list is in order of decreasing human labor needed for communicating a target task. Demonstrations need recording instrumentation and human labor hours to collect. One demonstration needs strictly fewer human labor hours. A goal image only needs the target task configuration of the environment to be setup from which to capture the goal image. A primitive recipe only needs careful consideration of which primitives and in what order to construct recipe of the target task. Natural language can just be a free form description of the target task. This is not an exhaustive list for task specification but it covers some of the most common ways to communicate different tasks to a multi-task policy. We explore task specification in the 3 cheapest forms of this list in this work.

Outline

In this thesis we explore ways to learn and specify new tasks with minimal human supervision to enable more scalable multi-task policies by leveraging cheaper sources of signal both for supervision and specification. We begin in Chapter 2 discussing how demonstrations of old

tasks can be leveraged for learning new tasks to minimize need for humans collecting new demonstrations for new tasks. This chapter explores task specification in the primitive recipe domain. In Chapter 3 we discuss how self-supervised exploration data can be leveraged for learning multi-task policies that use goal images for task specification. In Chapter 4 we discuss how large vision-language models could be leveraged to ground reward models in natural language to remove the need of a human scorer or environment instrumentation for reward computation. We explore natural language conditioned multi-task policies in this chapter. We conclude in Chapter 5 with a discussion of the implications of these works.

Summary of Publications

This thesis incorporates the following publications:

Material in Chapter 2 is published as:

> [68] Parsa Mahmoudieh, Trevor Darrell, Deepak Pathak. "Weakly-Supervised Trajectory Segmentation for Learning Reusable Skills." *ICLRW, 2020.*

Material in Chapter 3 is published as:

> [88] Deepak Pathak*, Parsa Mahmoudieh*, Guanghao Luo*, Pulkit Agrawal*, Dian Chen, Fred Shentu, Evan Shelhamer, Jitendra Malik, Alexei A. Efros, Trevor Darrell. "Zero-Shot Visual Imitation." *ICLR, 2018.*

Material in Chapter 4 is published as:

> [69] Parsa Mahmoudieh, Deepak Pathak, Trevor Darrell. "Zero-Shot Reward Specification via Grounded Natural Language." *ICML, 2022.*

Chapter 2

Leveraging Demonstrations of Old Tasks for New Tasks

2.1 Introduction

Humans have an uncanny ability to generalize from one task to another using either few, or at times, no new examples. This wouldn't be possible if they were to learn each new task from scratch. Humans rather extract reusable *skills* from already learned tasks and compose them to generalize to new tasks seamlessly. However, learning such repeatable skills has been a long standing challenge in sensorimotor control, partly because it is hard to define what constitutes a useful skill in itself.

One way to layout the scope of a skill is either by designing a corresponding reward function, or collecting expert demonstrations. For instance, consider a skill of reaching for an object. One can easily learn a policy for this skill by either using reinforcement learning with l2 distance as reward [115], or by imitation learning from kinesthetic demonstrations [8, 50]. However, neither of these approaches provide a natural form of supervision because the way this skill is performed in isolation can be drastically different from the way it could be used as part of some end task. For instance, reaching for a cup for pushing is very different from the way one would reach for a cup to pick it up for pouring. Therefore, defining a skill directly in a supervised manner could easily lead to biased set of examples.

A promising alternative is to learn skills that are already embedded in some useful end tasks. Previous works have explored this in the context of an agent's own exploration [29, 78, 88], where, the agent learns goal conditioned skill policies using data collected during its exploration phase. These skills are then used to plan for novel tasks at inference. However, exploration in itself is an open research problem, and hence, such approaches have difficulty in scaling to complex skills.

In this work, we follow an alternative paradigm where we extract skills from a collection of human demonstrations gathered to perform different end tasks. A straightforward way to extract reusable skills would be to get an expert to label each time-step of demonstrations

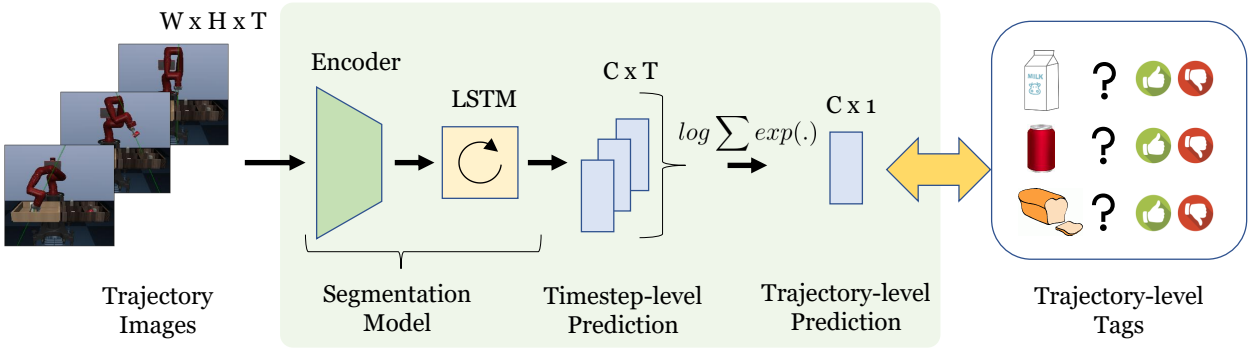


Figure 2.1: We propose a weakly-supervised approach for segmenting demonstrations into skill primitives, aka, sub-tasks. Our approach is end-to-end trainable, works directly from raw sensory data (e.g., images) and only requires the knowledge of class of primitive sub-tasks performed during the demonstration to accomplish the end task, without any need of segmentation or ordering of primitives. The key idea is to make per-time step prediction corresponding to each sub-task and accumulate them to generate a trajectory-level predictions which are then trained to match trajectory-level tags. Our segmentation model consists of an encoder followed by a recurrent LSTM model to capture time-series dependency. We use `log-sum-exp` to perform smooth accumulation of time-step predictions into trajectory-level predictions.

with the corresponding skill label. However, this per-step segmentation supervision is tedious for the expert and too expensive to scale. Moreover, such a labeling would be biased towards what expert thinks is the right segmentation than towards a segmentation which helps learn the skills better. This leads to the question: is it possible to use the expert knowledge to only know the types of skills present in demonstration and figure out the segmentation from the data itself?

Inspired by the classic work in multiple instance learning (MIL) [6], we propose a weakly-supervised approach for segmentation of human demonstration trajectories into primitive repeatable skills. Our approach assumes the access to only trajectory-level labels, i.e., what primitive skills, aka sub-tasks, are present in the demonstration. The key insight is to learn a primitive skills classifier model conditioned on the input sensory data (e.g. demonstration images), and incorporate a per-time step reasoning structure in this classifier. An overview of our approach is shown in Figure 2.1. Our model generates per time-step primitive skill label prediction estimates which are then accumulated via differentiable function to generate trajectory-level predictions. In contrast to classic MIL, where only the most confident prediction across time-steps is trained, our full model is trained end-to-end using trajectory-level multi-class loss directly from raw sensory images.

However, we are training with only trajectory-level supervision, then why should our per-step predictions of segmentation model converge to meaningful skill primitives? Data comes to rescue! Since our model trains across a variety of demonstrations, and hence, it would have seen plenty of demonstration trajectories that contain a certain skill primitive (positives)

and plenty that do not (negatives). The classification loss would force the segmentation model to focus on discriminative cues that are common across all positives, and absent from negatives. These discriminative cues corresponding to each skill would encourage the per time-step predictions to gradually correspond to the correct ground truth time-step labels.

We evaluate our approach in four different environments: (a) As proof of concept, we start with a simple 2D navigation task in grid-world setup where the demonstrations are programatically generated. (b) We then discuss result in robotics setup with continuous control actions, in particular, Jaco robotic arm performing button pushing demonstrations in a touch pad with procedurally generated demonstrations. (c) We test our approach in a robotic setup with actual human demonstrations collected on the RoboSuite benchmark. (d) Finally, we evaluate on a real robot dataset with actual human demonstrations collected kinesthetically. Across all these environments, our approach outperforms the other variants of MIL and achieves decent segmentation accuracy. We then show zero-shot generalization to tasks containing novel permutations of skills in Jaco environment.

2.2 Weakly-Supervised Trajectory Segmentation for Learning Reusable Skills

Given a collection of human demonstration trajectories, our goal is to learn a labeling for skill primitives at each time-step of the sequence, i.e., per time-step skill segmentation. Let X be a human demonstration trajectory denoted by $X = \{x_1, x_2, x_3 \dots x_T\}$, where T is the length of the demonstration and x_t denotes a tuple of observation (which is raw sensory image in our case) at time t and action taken from it. Note that the action data is optional for the purpose of skill segmentation, but can be useful post segmentation for learning skill policies via imitation. Let $Y = \{y_1, y_2, y_3 \dots y_T\}$ be the latent ground truth labeling of skill primitives in the sequence. Each label y_t belongs to one of the k labels from a set of all skill classes $\mathcal{C} = \{1, \dots, k\}$, i.e., $y_t \in \mathcal{C}$. These per time-step labels are not only tedious for expert to annotate, but also difficult to scale. In this work, we do not assume access to y_t during training, and learn the per time-step segmentation in a weakly-supervised fashion by only using trajectory-level 1-bit label during training, i.e., whether a skill class is present in the trajectory or not. After training, our model is able to directly segment demonstrations at inference, without requiring any labels of any kind. An overview of our method is shown in Figure 2.1.

The marginal probability of a skill primitive at each time-step of demonstration can be written as $P(y_t|\theta, \{x_t, x_{t-1} \dots, x_1\})$ where θ is the parameter vector of the segmentation model represented by a neural network in our case. If we had access to the true sequence labels Y, the network parameters θ can be easily learned via maximum log-likelihood by representing the probability as follows:

$$P(y_t|\theta, \{x_t, x_{t-1} \dots, x_1\}) = \frac{1}{Z_t} \exp\left(f(y_t; \theta, \{x_t, x_{t-1} \dots, x_1\})\right) \tag{2.1}$$

where Z_t is the partition function at t, defined as $Z_t = \sum_{k \in \mathcal{C}} \exp\big(f_t(k; \theta, \{x_t, x_{t-1} \ldots, x_1\})\big)$. The output of the function f_t corresponds to the logit score value generated by the neural network. In order to model temporal dependency on across observation time-steps x_t, we represent $f(.)$ via a recurrent neural network, in particular, LSTM [47].

Weakly-Supervised Trajectory Segmentation

We are given a dataset of demonstration trajectories during training, $\mathcal{D} = \{X^1, \ldots, X^n\}$, where n is total number of demonstrations available for training. Each demonstration trajectory is weakly labelled with what skill primitives are contained within the trajectory. Neither do we have access to which time-step densely correspond to which skill primitive, nor to the permutation in which the skills are executed in. Instead, we are only given a set of skill primitive labels $C_X \in \mathcal{C}$ present in the demonstration trajectory X.

Although our supervision is only at trajectory-level, we do not directly predict output labeling C_X from input demonstration X. Instead, we instill the structure of per-step prediction in our weakly supervised segmentation model by first computing the per-step classification score $f(y_t; \theta, \{x_t, x_{t-1} \ldots, x_1\})$ and then accumulate it across all time-steps to compute the probability of a class in the whole trajectory. This weakly-supervised setup is captured by classical paradigm of multiple instance learning (MIL) [6]. At inference, we use this per time-step score to compute the probability of skill primitives at each time t as described in Equation (2.1). There are multiple ways one could accumulate these per time-step scores discussed as follows.

Accumulation of Time-step Predictions

Intuitively, we would like an estimator that could generate an aggregated score for the class depending on how much each time-step votes for that class. We would ideally like the highest score value across time-steps to contribute most to the decision whether a class label $\hat{Y}_X$ is present in the trajectory X or not. One simple way to achieve that is to employ element-wise max operator, which is also the de facto approach in MIL to accumulate element-level scores. However, this would amount to passing gradients only to the most confident time-step and will completely eradicate the role of other time-steps. This is especially problematic in case of sequential trajectories because no skill primitive will be of only 1 time-step long. Hence, instead of max, we use a soft approximation to it which can take into account the contribution of all time-steps. In particular, we use `log-sum-exp` operator. Given the the logit score $f(y_t; \theta, \{x_t, x_{t-1} \ldots, x_1\})$ at each time-step, the trajectory-level logit score g for class $c \in \mathcal{C}$ is computed as follows:

$$g(c; \theta, X) = \log\Big(\sum_{t=1}^{T} \exp\big(f(y_t = c; \theta, \{x_t, x_{t-1} \ldots, x_1\})\big)\Big) \tag{2.2}$$

We perform this operation for all $c \in \mathcal{C}$ and use softmax over $g(c; \theta, X)$ to compute trajectory-level probability distribution $Q(c|X, \theta)$. Finally, the parameters θ are optimized to maximize

$Q(c|X, \theta)$ for each class c with respect to the ground truth trajectory level tags C_X. Note that this optimization is fully differentiable through Equation 2.2, and hence can be optimized via stochastic gradient descent in the end-to-end fashion. [91] has also showed the effectiveness of a temperature-based variant of `log-sum-exp` operation for semantic segmentation in images.

However, these per-step scores f would be almost uniformly random in the beginning of the training process due to the absence of per-step supervision. Since we are training with only trajectory-level supervision, why should these per-step predictions should ever converge to meaningful skill segmentation? It turns out to be the case because we are learning across large variety of demonstration examples. Hence, for each skill primitive we would have seen plenty of positive as well as negative trajectories. The loss suppresses the negative classes and encourages the positive ones, hence our segmentation model would be forced to focus on discriminative cues that are exclusively common among the trajectories containing the skill primitives and not common to the cues that help distinguish other skills. Since our trajectory-level segmentation is based on a deterministic transformation of per-step predictions, each per-step score will then be forced to focus on those discriminative cues. The discriminative nature encourages the per time-step predictions to slowly drift towards the true latent ground truth segmentation which are not available directly.

2.3 Problem Setup Details

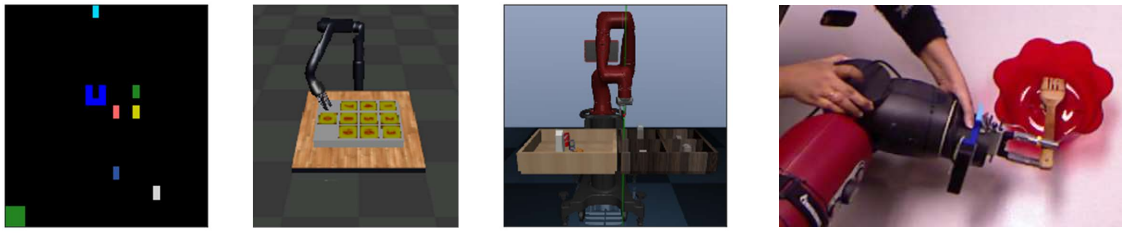
Training and Evaluation Details

Our segmentation model consists of a convolutional neural network encoder of each image of a trajectory and a one layer fully connected encoder of the action to a 32 dimensional feature that is concatenated to the image feature before being fed into an LSTM with 100 hidden units.

We train with a batch size of 64 for the Dial, RoboSuite, and MIME environments and a batch size of 128 for the Grid-world environment. All models were trained with Adam with learning rate of 1e-4. For training, we use 50000, 2000, 1000, and 1600 trajectories for 2D Navigation, Dial, RoboSuite, and MIME Environments respectively. We evaluate our method on the training set, a validation set that consists of the same number of skill primitives (sub-tasks) per trajectory as in training, and a test set that consists of more skill primitives per trajectory than seen in training. The segmentation quality is measured by classification accuracy of the learned model averaged across all time-step. The time-step ground truth is only used for evaluation and not for training.

Baselines

We compare our approach to different formulations of weakly-supervised classification proposed earlier. None of these methods have been applied to a temporal trajectory segmentation before. In our work, we adapt them for sub-task segmentation. In particular, we compare to:



(a) 2D Navigation (b) Dial Env (c) RoboSuite Env (d) MIME Env

Figure 2.2: We evaluate across four environments with different properties: (a) Discrete 2D Navigation as proof of concept. (b) Dial continuous control: Jaco robotic arm based manipulation with procedurally generated demonstrations. (c) RoboSuite environment: Sawyer robot in simulation with human-collected demonstrations. (d) MIME environment: Baxter robot in real world with human demonstrations. Result videos at `https://sites.google.com/view/trajectory-segmentation/`.

(a) MIL [6]: In this baseline, we train the deep segmentation model using Classic MIL formulation as proposed by [6]. We penalize the most confident time-step in the output corresponding to each sub-class to correctly predict the trajectory-level classes.
(b) FCN-MIL [87]: This approach is an adaptation of MIL for deep-networks. The MIL objective is treated as the loss function for training the segmentation network, but instead of training the most confident output, we train a neighborhood of k-elements near the most confident time-step, where k is a hyper-parameter chosen on the validation set. We found k=3, i.e., one extra time-step on each side of argmax, to work the best across all datasets.
(c) CCNN [84]: This is an alternative approach to tackle the weakly-supervised setup. Instead of training the most confident output, a pseudo ground-truth per time-step truth is generated in a way that ensures that it is the closest pseudo ground-truth possible which contains only the classes as specified by the trajectory-level tags. One could additionally put threshold constraints on the lower-bound of time-steps devoted to the present classes in the pseudo ground truth. The segmentation is then pushed to this per time-step pseudo labels, and a new pseudo ground-truth is generated after each iteration. This approach provides per time-step supervision using only trajectory level-tags. The CCNN approach was originally applied for per-pixel segmentation of images in computer vision literature [84], and we adapt it for temporal trajectories.
(d) Random: For sanity check, we randomly pick a sub-task class at every time-step with uniform probability.
(e) Random-Cls: This is a random baseline with privileged class information even at test time. In this case, we only sample random uniformly from the set of classes that are already present in the trajectory. This is not a fair comparison but provides an estimate for the difficulty of the problem.
Note that we use the same network architecture and training details across all baseline implementations. However, we tune each baseline separately in a thorough manner on validation sets. This is crucial to ensure an "apples-to-apples" comparison.

2.4 Experiments

We evaluate our approach and other baselines on four different datasets with very diverse characteristics. The 2D Grid-world environment has a discrete action space, while the Dial, RoboSuite, and MIME environments have a continuous action space. The Grid-world and Dial environments have demonstrations collected procedurally by hand-designed controllers, while the RoboSuite and MIME environments have demonstrations collected by humans. Human demonstration in RoboSuite are collected via teleoperation and kinesthetically in MIME. Grid world, Dial and Robosuite are in simulation while MIME is from real robot. A snapshot of these environments is shown in Figure 2.2.

Proof of Concept: 2D Navigation in Discrete Toy Grid-world

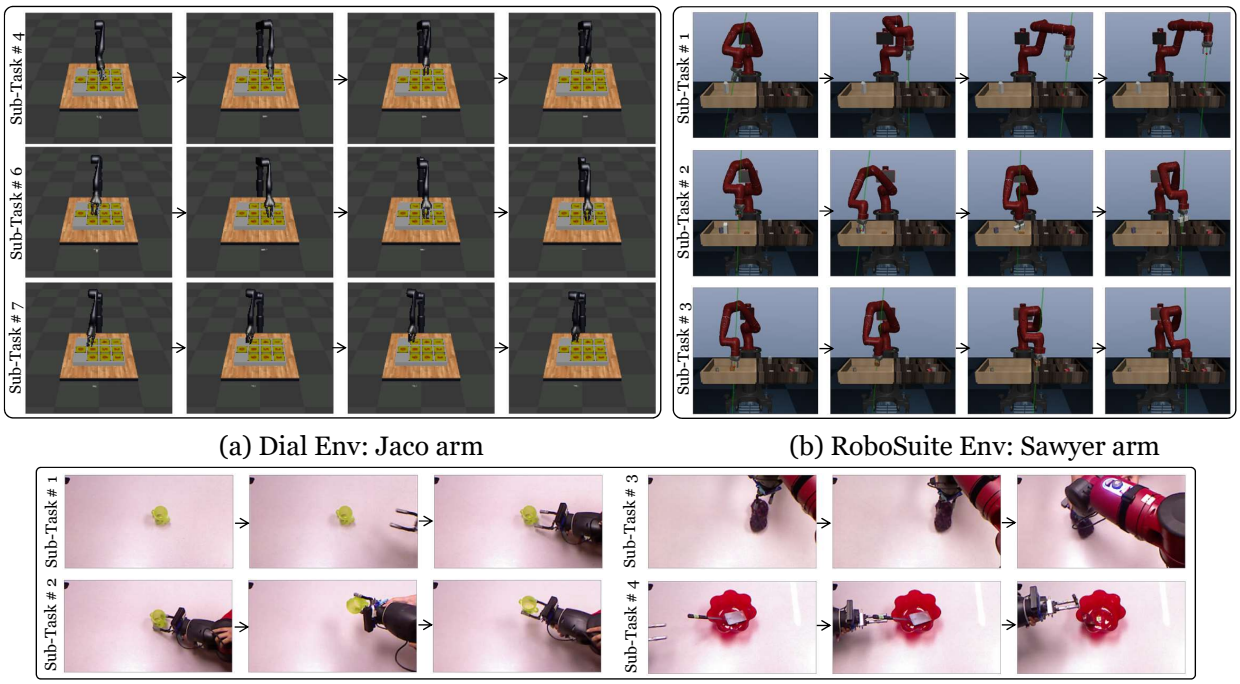
In the grid-world environment, the action space consists of moving up, moving down, moving left, moving right, and picking up object it is hovering over. There are 5 different types of objects uniquely identified by their color and an end task would consist of picking up some subset of all the objects in a particular order. The primitives are defined as picking up a specific type of object. We train our segmentation model on trajectories with 2-4 skill primitives and test with 5 skill trajectories. Each instantiation of the environment has a different starting position of the agent, different starting position of the objects, and different set of objects needed to be grabbed. The image inputs used are 33 by 30 resolution color images, and the max trajectory lengths are 50. This environment serves as toy scenario for proof of concept. At the bottom-left of the image, there is an indicator which suggests which skill is being executed. Hence, an efficient approach should achieve 100% accuracy, as is the case with our method as shown in Table 2.1.

Method	**Train**	**Val**	**Test**
Random	20.00	20.00	20.00
Random-Cls	36.00	36.00	20.00
CCNN	65.27	64.63	60.24
FCN-MIL	91.28	91.88	91.58
MIL	90.78	91.08	91.08
Ours	**100.00**	**100.00**	**100.00**

Table 2.1: This environment serves as only a proof of concept. Since the environment is a simple 2D grid with skill primitive indicator cell at the bottom, an efficient approach should be able to achieve almost full accuracy. Our method is able to perfectly segment the data into ground truths, while other baselines can not despite easily separable skills.

Dial control environment: Jaco robotic arm based manipulation

In the Dial environment, proposed in [108], there is a torque-controlled JACO 6 DoF arm and a dial pad that the arm can interact with which is simulated in MuJoCo. There are naturally 10 different types of primitives available in this environment corresponding to pressing numbers zero through nine. We train our segmentation model on trajectories with two to four sub-tasks and test with five sub-task trajectories. Each instantiation of the environment has a different sequence of numbers it expects to be dialed in the correct order.



(a) Dial Env: Jaco arm

(b) RoboSuite Env: Sawyer arm

(c) MIME Env: Baxter arm

Figure 2.3: Figure shows qualitative visualization of the skills (sub-sampled) discovered by our approach on held-out test set. (a) Dial Env: learned primitives using Jaco arm manipulation are shown. Skill # N corresponds to the arm dialing number N on touch pad. (b) RoboSuite Env: Three predicted primitives are shown where the primitives are to pick and place an object into corresponding bin (top to bottom: cereal, milk, box) (c) MIME Env: Four discovered primitives are shown starting from top left clockwise: reach to object, wipe, stir inside object, and pour out object.

The image inputs used are 112 by 112 resolution gray-scale images, and the max trajectory lengths are 100.

Our method performs similar to FCN-MIL on the train/val, and better on test. It significantly outperforms the other baselines (Table 2.2). However, we show that our segmentation are more useful for zero-shot execution performance as explained in Section 2.4. Learning perfect segmentation in the Dial environment is very challenging because there is little signal in most of the trajectory for each skill to signify exactly which digit will be pressed until the arm reaches proximity of the digit.

RoboSuite environment: Sawyer robotic arm based object pick and place

The RoboSuite environment [70] has a Sawyer 7 DoF arm and four different objects (bread, can, cereal, and milk) that the arm can interact with simulated with MuJoCo physics engine. There are four different types of primitives available in this environment corresponding to

pick and place of bread, can, cereal, and milk to correct corresponding bin. We train for trajectories with two to three skill primitives and test on trajectories with four skill primitives. Each instantiation of the environment has a different sequence and set of objects that need to be picked up and placed into their corresponding bins. The image inputs used are 128 by 128 resolution color images and the max trajectory lengths are 100.

Our method significantly outperforms all baselines in full trajectory segmentation by a significant margin. Only MIL performs above random present class on validation and test datasets. Learning perfect segmentation in the RoboSuite environment is also very challenging because there is very little signal in most of the trajectories of each subtask to signify exactly which object will be picked up until near the end of the primitive where the object has been picked up. The "pick object" portion of human demonstrations is usually much longer than the "place object" part because with the tele-operation setup the human stumbles a little bit until fully gripping the object. After the object is in the gripper, placing object in bin is a quick reach to the correct bin for the object.

Method	**Dial Jaco Manipualtion**			**RoboSuite Manipulation**		
	Train	Val	Test	Train	Val	Test
Random	10.00	10.0	10.0	25.00	25.00	25.00
Random-Cls	36.00	36.00	20.00	33.00	33.00	25.00
CCNN	9.05	9.58	10.70	31.92	22.24	21.80
FCN-MIL	**64.42**	**59.64**	57.81	29.79	23.76	22.21
MIL	19.02	15.89	14.00	35.95	28.81	28.69
Ours	**61.57**	**59.52**	**59.93**	**44.96**	**37.67**	**33.88**

Table 2.2: We show the segmentation performance across different methods on train, validation, and test datasets for **Dial Jaco and RoboSuite Object Manipulation**. (a) Dial Jaco: we train our segmentation model on trajectories with three to four subtasks and test with five subtask trajectories. Our approach beats all baselines by significant margin except for FCN-MIL which performs similar to ours on training and validation, but is outperformed on the test set. (b) Robosuite: we train our segmentation model on trajectories with three subtasks and test with four subtask trajectories. Our approach outperforms all baselines by significant margin.

MIME environment: Baxter Robotic Arm Based Manipulation

MIME is a robotic-demonstration dataset that contains 8260 human-robot video demonstrations of 20 different robotic tasks [107]. We defined the following primitives for a subset of this dataset: reach for object, pour out object, stir inside object, stack objects, place object in box, wipe with rag (6 primitives). All videos have two primitives where one is to reach for object and the other is the action to do with or on the object. There is a held out test dataset for each robotic task which we use for evaluation. The image inputs used are 120 by 320 resolution grayscale images and the max trajectory lengths are 100. Our method beats all other baselines in full trajectory segmentation by at least 1.8x on the test set (Table 2.4).

Method	Seg Acc. without Reject	Seg Acc with Reject	Zeroshot Success Rate
CCNN	9.58	28.08	-
FCN-MIL	**59.64**	68.58	38.8
MIL	15.89	50	39.6
Ours	**59.52**	**79.01**	**50.4**

Table 2.3: **Quantitative Zero-Shot results on Dial Jaco Manipulation**: This table shows the comparison of performance across different methods. The first two columns reports the segmentation accuracy on validation set without and with minimum of 5 time-step segment rejection. The last column shows the zero-shot subtask success rate on 5 subtask tasks. The CCNN baseline collapsed to predicting only one class making it unusable for zero-shot results. Our method has significantly better post 5 time-step segment rejection segmentation accuracy and zero-shot subtask performance.

Method	Test Accuracy
CCNN	11.17
FCN-MIL	19.10
MIL	24.37
Ours	**44.22**

Table 2.4: Quantitative results on MIME dataset. We test our segmentation model on held out test trajectories not seen during training.

Zero-Shot results: Jaco Manipulation

We use our segmentation model to create sub-datasets for each of our primitives to train a behavior cloned skill policy for each. We then test our skill policies on performing higher sequence length tasks not seen in training data. During the creation of the sub-datasets, we rejected all segments smaller than 5 consecutive timesteps of the same labelled primitive. We applied gaussian smoothing on the segmentation prior to extraction to filter out noisy predictions. The gaussian smoothing had a very minor effect on the segmentation accuracy of the total original dataset. The smoothing helped connect smaller segments into larger contiguous segments which made a critical difference on what segments were rejected since we rejected segments smaller than 5 consecutive timesteps for our behavior cloning dataset aggregation of each skill. With smoothing fewer correctly labelled segments were rejected which led to higher segmentation accuracy behavior cloning data as shown in first two columns of table 4.

We demonstrate the zero-shot capability of our model and baselines on the Dial control environment in Table 2.3. Our model performs at least 1.25x better than all baselines. We also show that although FCN-MIL had the same segmentation accuracy as our method, after rejecting smaller than 5 timestep segments our method has a significantly higher post rejection segmentation accuracy. We speculate this is due to our model committing less to a wrong prediction than the baselines. Therefore wrong predictions are more easily rejected with our segmentation model.

2.5 Related Work

Multiple Instance Learning is the sub-field that deals with learning with weak-labels [24], formulated mostly as max-margin. Classic formulations include MI-SVM [6] or LSVM [31]. There have been boosting [4, 126] and Noisy-OR models [44] formulations as well. This has been extensively explored in image segmentation [91, 84]. There is also work on learning to segment videos into primitive components that take place in each of the videos using human narration of the videos [3], [76], [130], [97]. However collecting human narrations is an expensive process that cannot scale to very large datasets. In our setup, we need to label each timestep of a video with a further preferable constraint of having the labels as contiguous as possible.

Training goal conditioned policies have become very popular recently since they learn policies that are reusable for new tasks. The goals can be defined in state space [99], [7] or in image space [88], [78]. These methods however suffer from not being able to reach goals that are out of the training distribution. They also tend to greedily reach goals which make them suffer for long horizon tasks. This leads to having to break the problem down to checkpoint states and this can still be expensive and cumbersome for a human to define each new task [88]. Similar way to define a new task is one-shot imitation learning where a new task is defined with one demonstration of the task; this can also be expensive if one need to learn new tasks that are cumbersome for the human to collect [27].

Hierarchical policies have been used to solve long horizon tasks by having a meta-policy or manager that takes in states at a sparser time scale and chooses which sub-policy or short term goal for worker to achieve at dense time scale [23], [118], [116], [9]. However, the subpolicies learned in this type of options framework are not intrepretable in their specialization and therefore tend to be hard to reuse for new tasks.

Learning reusable and interpretable primitives from scratch has been very challenging. One way to tackle this is for humans to define the high level primitives we care about beforehand. We can then decompose complex tasks into subtasks / primitives and learn the primitives first [82]. By learning the primitives first, an agent can then perform any new task that can be decomposed into those primitives without needing any new human demonstration data [5]. The human only needs to specify the order in which the primitives need to be done. In the presence of only the order of high level tags of what primitives were performed at the trajectory level, the optimization is non-trivial. [108] provide one approach using dynamic programming to efficiently compute gradients [39] through all possible trajectories of a given high level permutation of primitives performed in each trajectory. We however have presented an approach where we would not need the order in which the primitives were performed thereby making the labeling process of videos much cheaper. [55] and [80] also tackle the problem of primitive segmentation, however, both the approaches use state-space input to their model (former only use image input for the grid-world) while our approach is trained with high dimensional image input in all the environments.

2.6 Discussion

Obtaining primitives from existing experience and composing them to perform novel tasks presents a viable paradigm for tackling generalization to novel tasks. Due to the combinatorial nature of composition, this approach allows generalization to exponentially many scenarios with a linear number of primitives. This work provides an end-to-end trainable formulation for extracting primitives. Our results demonstrate strong segmentation accuracy, and show early evidence for zero-shot generalization via compositionality. Our primitive segmentations, obtained from demonstrations of end tasks, are more naturally extracted 'options' than those defined by an expert, which may be prone to being biased. Another alternative, besides zero-shot composition, is to treat these primitives as new atomic actions to develop a hierarchical framework for modeling long-horizon tasks. Deciding the threshold for segmenting actions into different levels of controller is one of the main bottlenecks in hierarchical control, and current approaches usually resort to using domain knowledge to solve the issue. These extracted primitives, a.k.a 'macro' actions, provide an alternative scaffolding which could bootstrap the hierarchy in a bottom-up manner. To promote these follow-up ideas, we will publicly release our code and environments.

Chapter 3

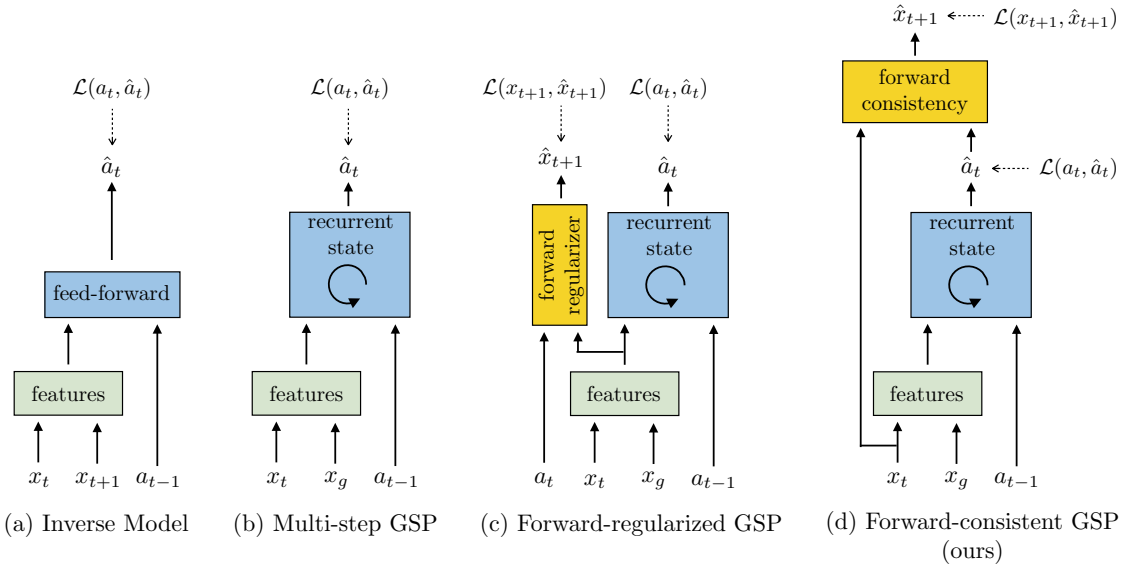
Leveraging Self-Supervised Exploration Data

3.1 Introduction

Imitating expert demonstration is a powerful mechanism for learning to perform tasks from raw sensory observations. The current dominant paradigm in learning from demonstration requires the expert to either manually move the robot joints (i.e., kinesthetic teaching) or teleoperate the robot to execute the desired task. The expert typically provides multiple demonstrations of a task at training time, and this generates data in the form of observation-action pairs from the agent's point of view. The agent then distills this data into a policy for performing the task of interest. Such a heavily supervised approach, where it is necessary to provide demonstrations by controlling the robot, is incredibly tedious for the human expert. Moreover, for every new task that the robot needs to execute, the expert is required to provide a new set of demonstrations.

Instead of communicating *how* to perform a task via observation-action pairs, a more general formulation allows the expert to communicate only *what* needs to be done by providing the observations of the desired world states via a video or a sparse sequence of images. This way, the agent is required to infer how to perform the task (i.e., actions) by itself. In psychology, this is known as *observational learning* [12]. While this is a harder learning problem, it is a more interesting setting, because the expert can demonstrate multiple tasks quickly and easily.

An agent without any prior knowledge will find it extremely hard to imitate a task by simply watching a visual demonstration in all but the simplest of cases. Thus, the natural question is: in order to imitate, what form of prior knowledge must the agent possess? A large body of work [60, 59, 51, 15, 25, 123] has sought to capture prior knowledge by manually pre-defining the state that must be inferred from the observations. The agent then infers how to perform the task (i.e., plan for imitation) using this state. Unfortunately, computer vision systems are often unable to estimate the state variables accurately and it has proven



(a) Inverse Model (b) Multi-step GSP (c) Forward-regularized GSP (d) Forward-consistent GSP (ours)

Figure 3.1: The goal-conditioned skill policy (GSP) takes as input the current and goal observations and outputs an action sequence that would lead to that goal. We compare the performance of the following GSP models: (a) Simple inverse model; (b) Mutli-step GSP with previous action history; (c) Mutli-step GSP with previous action history and a forward model as regularizer, but no forward consistency; (d) Mutli-step GSP with forward consistency loss proposed in this work.

non-trivial for downstream planning systems to be robust to such errors.

In this work, we follow [2, 63, 92] in pursuing an alternative paradigm, where an agent explores the environment without any expert supervision and distills this exploration data into goal-directed skills. These skills can then be used to imitate the visual demonstration provided by the expert [77]. Here, by skill we mean a function that predicts the sequence of actions to take the agent from the current observation to the goal. We call this function a *goal-conditioned skill policy (GSP)*. The GSP is learned in a self-supervised way by re-labeling the states visited during the agent's exploration of the environment as goals and the actions executed by the agent as the prediction targets, similar to [2, 7]. During inference, given goal observations from a demonstration, the GSP can infer how to reach these goals in turn from the current observation, and thereby imitate the task step-by-step.

One critical challenge in learning the GSP is that, in general, there are multiple possible ways of going from one state to another: that is, the distribution of trajectories between states is multi-modal. We address this issue with our novel *forward consistency loss* based on the intuition that, for most tasks, reaching the goal is more important than how it is reached. To operationalize this, we first learn a forward model that predicts the next observation given an action and a current observation. We use the difference in the output of the forward model for the GSP-selected action and the ground truth next state to train the GSP. This loss has the effect of making the GSP-predicted action *consistent* with the ground-truth action instead of exactly matching the actions themselves, thus ensuring that actions that are different from

the ground-truth—but lead to the same next state—are not inadvertently penalized. To account for varying number of steps required to reach different goals, we propose to jointly optimize the GSP with a goal recognizer that determines if the current goal has been satisfied. See Figure 3.1 for a schematic illustration of the GSP architecture.

We call our method *zero-shot* because the agent never has access to expert actions, neither during training of the GSP nor for task demonstration at inference. In contrast, most recent work on one-shot imitation learning requires full knowledge of actions and a wealth of expert demonstrations during training [27, 32]. In summary, we propose a method that (1) does not require any extrinsic reward or expert supervision during learning, (2) only needs demonstrations during inference, and (3) restricts demonstrations to visual observations alone rather than full state-actions. Instead of learning by imitation, our agent learns to imitate.

We evaluate our zero-shot imitator on real-world robots for rope manipulation tasks using a Baxter and office navigation using a TurtleBot. We show that the proposed forward consistency loss improves the performance on the complex task of knot tying from 36% to 60% accuracy. In navigation experiments, we steer a simple wheeled robot around partially-observable office environments and show that the learned GSP generalizes to unseen environments. Furthermore, using navigation experiments in *VizDoom* environment, we show that (GSP) learned using curiosity-driven exploration [83, 100, 85] can more accurately follow demonstrations as compared to using random exploration data for learning the GSP. Overall our experiments show that the forward-consistent GSP can be used to imitate a variety of tasks without making environment or task-specific assumptions.

3.2 Zero-Shot Visual Imitation

Let $\mathcal{S} : \{x_1, a_1, x_2, a_2, ..., x_T\}$ be the sequence of observations and actions generated by the agent as it explores its environment using the policy $a = \pi_E(s)$. This exploration data is used to learn the goal-conditioned skill policy (GSP) π takes as input a pair of observations (x_i, x_g) and outputs sequence of actions $(\vec{a}_\tau : a_1, a_2...a_K)$ required to reach the goal observation (x_g) from the current observation (x_i).

$$\vec{a}_\tau = \pi(x_i, x_g; \theta_\pi) \tag{3.1}$$

where states x_i, x_g are sampled from the $\mathcal{S}$. The number of actions, K, is also inferred by the model. We represent π by a deep network with parameters θ_π in order to capture complex mappings from visual observations (x) to actions. π can be thought of as a variable-step generalization of the inverse dynamics model [52], or as the policy corresponding to a universal value function [33, 99], with the difference that x_g need not be the end goal of a task but can also be an intermediate sub-goal.

Let the task to be imitated be provided as a sequence of images $\mathcal{D} : \{x_1^d, x_2^d, ..., x_N^d\}$ captured when the expert demonstrates the task. This sequence of images $\mathcal{D}$ could either be temporally dense or sparse. Our agent uses the learned GSP π to imitate the sequence of visual observations $\mathcal{D}$ starting from its initial state x_0 by following actions predicted by

$\pi(x_0, x_1^d; \theta_\pi)$. Let the observation after executing the predicted action be x_0'. Since multiple actions might be required to reach close to x_1^d, the agent queries a separate *goal recognizer* network to ascertain if the current observation is close to the goal or not. If the answer is negative, the agent executes the action $a = \pi(x_0', x_1^d; \theta_\pi)$. This process is repeated iteratively until the *goal recognizer* outputs that agent is near the goal, or a maximum number of steps are reached. Let the observation of the agent at this point be $\hat{x}_1$. After reaching close to the first observation (x_1^d) in the demonstration, the agent sets its goal as (x_2^d) and repeats the process. The agent stops when all observations in the demonstrations are processed.

Note that in the method of imitation described above, the expert is never required to convey to the agent what actions it performed. In the following subsections we describe how we learn the GSP, *forward consistency loss*, *goal recognizer* network and various baseline methods.

Learning the Goal-conditioned Skill Policy (GSP)

We first describe the one-step version of GSP and then extend it to variable length multi-step skills. One-step trajectories take the form of (x_t, a_t, x_{t+1}) and GSP, $\hat{a_t} = \pi(x_t, x_{t+1}; \theta_\pi)$, is trained by minimizing the standard cross-entropy loss $\mathcal{L}(a_t, \hat{a}_t)$,

$$\mathcal{L}(a_t, \hat{a}_t) = p(a_t | x_t, x_{t+1}) \log(\hat{a_t}) \tag{3.2}$$

with respect to parameters θ_π, where p and $\hat{a_t}$ are the ground-truth and predicted action distributions. While we do not have access to true p, we empirically approximate it using samples from the distribution, a_t, that are executed by the agent during exploration. For minimizing the cross-entropy loss, it is common to assume p as a delta function at a_t. However, this assumption is notably violated if p is inherently multi-modal and high-dimensional. If we optimize say a deep neural network assuming p to be a delta function, the same inputs will be presented with different targets (due to multi-modality) leading to high-variance in gradients which in turn would make learning challenging.

In our setup, such multi-modality can occur because multiple actions can lead the agent to the same future observation from the initial observation. For instance, in navigation, if the agent is stuck against a corner, turning or moving forward all collapse to the same effect. The issue of multi-modality becomes more critical as the length of trajectories grow, because more and more paths may take the agent from the initial observation to the goal observation given more time. Furthermore, it would require many samples to even obtain a good empirical estimate of a high-dimensional multi-modal action distribution p.

Forward Consistency Loss

One way to account for multi-modality is by employing the likes of variational auto-encoders [54, 96]. However, in many practical situations it is not feasible to obtain ample data for each mode. In this work, we propose an alternative based on the insight that in many

scenarios, we only care about whether the agent reached the final state or not and the exact trajectory is of lesser interest. Instead of penalizing the actions predicted by the GSP to match the ground truth, we propose to learn the parameters of GSP by minimizing the distance between observation $\hat{x}_{t+1}$ resulting by executing the predicted action $\hat{a}_t = \pi(x_t, x_{t+1}; \theta_\pi)$ and the observation x_{t+1}, which is the result of executing the ground truth action a_t being used to train the GSP. In this formulation, even if the predicted and ground-truth action are different, the predicted action will not be penalized if it leads to the same next state as the ground-truth action. While this formulation will not explicitly maintain all modes of the action distribution, it will reduce the variance in gradients and thus help learning. We call this penalty the *forward consistency loss*.

Note that it is not immediately obvious as to how to operationalize *forward consistency loss* for two reasons: (a) we need the access to a good *forward dynamics* model that can reliably predict the effect of an action (i.e., the next observation state) given the current observation state, and (b) such a dynamics model should be differentiable in order to train the GSP using the state prediction error. Both of these issues could be resolved if an analytic formulation of forward dynamics is known.

In many scenarios of interest, especially if states are represented as images, an analytic forward model is not available. In this work, we learn the forward dynamics f model from the data, and is defined as $\tilde{x}_{t+1} = f(x_t, a_t; \theta_f)$. Let $\hat{x}_{t+1} = f(x_t, \hat{a}_t; \theta_f)$ be the state prediction for the action predicted by π. Because the forward model is not analytic and learned from data, in general, there is no guarantee that $\tilde{x}_{t+1} = \hat{x}_{t+1}$, even though executing these two actions, $a_t, \hat{a}_t$, in the real-world will have the same effect. In order to make the outcome of action predicted by the GSP and the ground-truth action to be *consistent* with each other, we include an additional term, $\|x_{t+1} - \hat{x}_{t+1}\|_2^2$ in our loss function and infer the parameters θ_f by minimizing $\|x_{t+1} - \tilde{x}_{t+1}\|_2^2 \; + \; \lambda \|x_{t+1} - \hat{x}_{t+1}\|_2^2$, where λ is a scalar hyper-parameter. The first term ensures that the learned forward model explains ground truth transitions (x_t, a_t, x_{t+1}) collected by the agent and the second term ensures consistency. The joint objective for training GSP with forward model consistency is:

$$\min_{\theta_\pi, \theta_f} \; \|x_{t+1} - \tilde{x}_{t+1}\|_2^2 + \lambda \|x_{t+1} - \hat{x}_{t+1}\|_2^2 \; + \; \mathcal{L}(a_t, \hat{a}_t) \tag{3.3}$$

$$\begin{aligned} \text{s.t.} \quad & \tilde{x}_{t+1} = f(x_t, a_t; \theta_f) \\ & \hat{x}_{t+1} = f(x_t, \hat{a}_t; \theta_f) \\ & \hat{a}_t = \pi(x_t, x_{t+1}; \theta_\pi) \end{aligned}$$

Note that learning θ_π, θ_f jointly from scratch is precarious, because the forward model f might not be good in the beginning, and hence could make the gradient updates noisier for π. To address this issue, we first pre-train the forward model with only the first term and GSP separately by blocking the gradient flow and then fine-tune jointly.

Generalization to feature space dynamics Past work has shown that learning forward dynamics in the feature space as opposed to raw observation space is more robust and leads to better generalization [85, 2]. Following these works, we extend the GSP to make

predictions in feature representation $\phi(x_t), \phi(x_{t+1})$ of the observations x_t, x_{t+1} respectively learned through the self-supervised task of action prediction. The forward consistency loss is then computed by making predictions in this feature space ϕ instead of raw observations. The optimization objective for feature space generalization with mutli-step objective is shown in Equation (3.4).

Generalization to multi-step GSP We extend our one-step optimization to variable length sequence of actions in a straightforward manner by having a multi-step GSP π_m model with a step-wise forward consistency loss. The GSP π_m maintains an internal recurrent memory of the system and outputs actions conditioned on current observation x_t, starting from x_i to reach goal observation x_T. The forward consistency loss is computed at each time step, and jointly optimized with the action prediction loss over the whole trajectory. The final multi-step objective with feature space dynamics is as follows:

$$\min_{\theta_\pi, \theta_f, \theta_\phi} \sum_{t=i}^{t=T} \Big(\|\phi(x_{t+1}) - \tilde{\phi}(x_{t+1})\|_2^2 \; + \; \lambda \|\phi(x_{t+1}) - \hat{\phi}(x_{t+1})\|_2^2 \; + \; \mathcal{L}(a_t, \hat{a}_t) \Big) \tag{3.4}$$

$$\begin{aligned} \text{s.t.} \quad & \tilde{\phi}(x_{t+1}) = f(\phi(x_t), a_t; \theta_f) \\ & \hat{\phi}(x_{t+1}) = f(\phi(x_t), \hat{a}_t; \theta_f) \\ & \hat{a}_t = \pi(\phi(x_t), \phi(x_T); \theta_\pi) \end{aligned}$$

where $\phi(.)$ is represented by a CNN with parameters θ_ϕ. The number of steps taken by the multi-step GSP π_m to reach the goal at inference is variable depending on the decision of goal recognizer; described in next subsection. Note that, in this objective, if ϕ is identity then the dynamics simply reduces to modeling in raw observation space. We analyze feature space prediction in *VizDoom* 3D navigation and stick to observation space in the rope manipulation and the office navigation tasks.

The multi-step *forward-consistent GSP* π_m is implemented using a recurrent network which at every step takes as input the feature representation of the current ($\phi(x_t)$) state, goal ($\phi(x_T)$) states, action at the previous time step (a_{t-1}) and the internal hidden representation h_{t-1} of the recurrent units and predicts $\hat{a}_t$. Note that inputting the previous action to GSP π_m at each time step could be redundant given that hidden representation is already maintaining a history of the trajectory. Nonetheless, it is helpful to explicitly model this history. This formulation amounts to building an auto-regressive model of the joint action that estimates probability $P(a_t|x_1, a_1, ...a_{t-1}, x_t, x_g)$ at every time step. It is possible to further extend our forward-consistent GSP π_m to build multi-step forward model, but we leave that direction of future work.

Goal Recognizer

We train a goal recognizer network to figure out if the current goal is reached and therefore allow the agent to take variable numbers of steps between goals. Goal recognition is especially critical when the agent has to transit through a sequence of intermediate goals, as is the

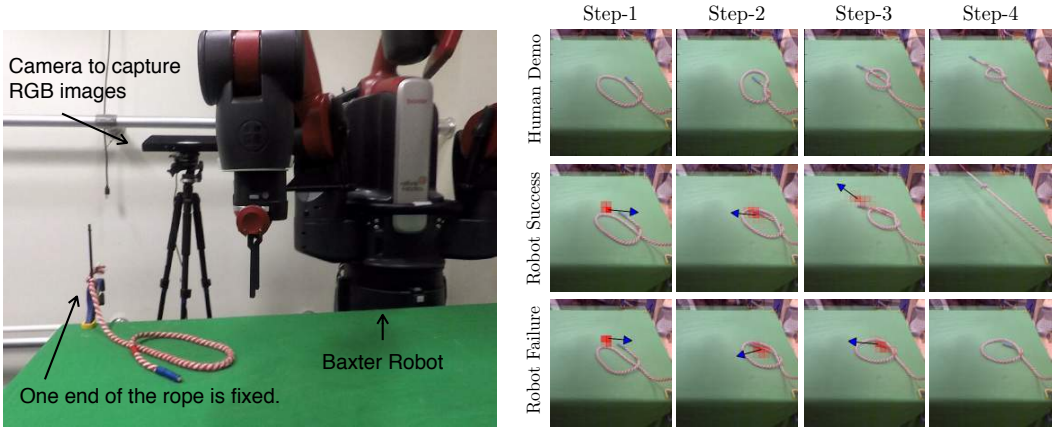
case for visual imitation, as otherwise compounding error could quickly lead to divergence from the demonstration. This recognition is simple given knowledge of the true physical state, but difficult when working with visual observations. Aside from the usual challenges of visual recognition, the dependence of observations on the agent's own dynamics further complicates goal recognition, as the same goal can appear different while moving forward or turning during navigation.

We pose goal recognition as a binary classification problem that given an observation x_i and the goal x_g infers if x_i is close to x_g or not. Lacking expert supervision of goals, we draw goal observations at random from the agent's experience during exploration, since they are known to be feasible. For each such pseudo-goal, we consider observations that were only a few actions away to be positives (i.e., close to the goal) and the remaining observations that were more than a fixed number of actions (i.e., a margin) away as negatives. We trained the goal classifier using the standard cross-entropy loss. Like the skill policy, our goal recognizer is conditioned on the goal for generalization across goals. We found that training an independent goal recognition network consistently outperformed the alternative approach that augments the action space with a "stop" action. Making use of temporal proximity as supervision has also been explored for feature learning in the concurrent work of [104].

Ablations and Baselines

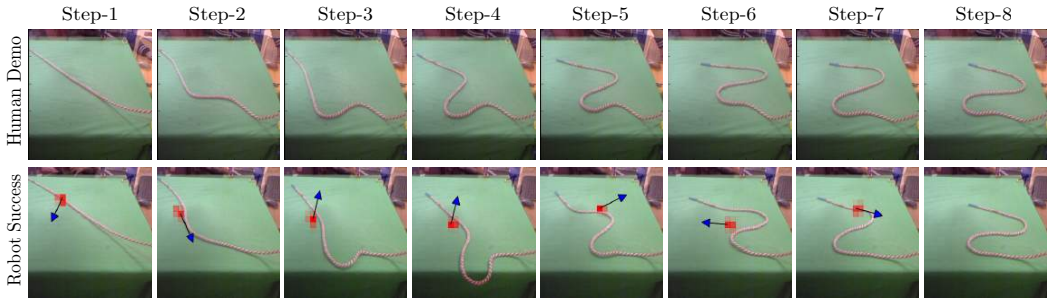
Our proposed formulation of GSP composed of following components: (a) recurrent variable-length skill policy network, (b) explicitly encoding previous action in the recurrence, (c) goal recognizer, (d) forward consistency loss function, and (w) learning forward dynamics in the feature space instead of raw observation space. We systematically ablate these components of forward-consistent GSP, to quantitatively review the importance of each component and then perform comparisons to the prior approaches that could be deployed for the task of visual imitation.

The following methods will be evaluated and compared to in the subsequent experiments section: (1) Classical methods: In visual navigation, we attempted to compare against the state-of-the-art open source classical methods, namely, ORB-SLAM2 [75, 22] and OpenSFM [71]. (2) Inverse Model: [77] leverage vanilla inverse dynamics to follow demonstration in rope manipulation setup. We compare to their method in both visual navigation and manipulation. (3) GSP-NoPrevAction-NoFwdConst is the ablation of our recurrent GSP without previous action history and without forward consistency loss. (4) GSP-NoFwdConst refers to our recurrent GSP with previous action history, but without forward consistency objective. (5) GSP-FwdRegularizer refers to the model where forward prediction is only used to regularize the features of GSP but has no role to play in the loss function of predicted actions. The purpose of this variant is to particularly ablate the benefit of consistency loss function with respect to just having forward model as feature regularizer. (6) GSP refers to our complete method with all the components. We now discuss the experiments and evaluate these baselines.



(a) Robotics Setup

(b) Manipulating rope into tying a knot

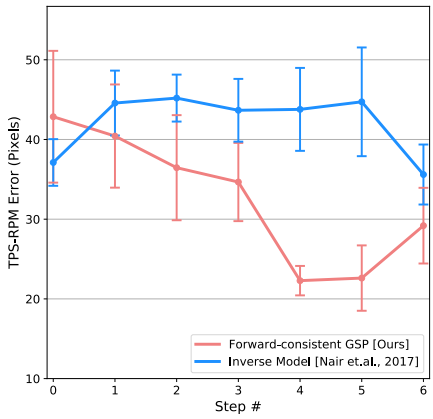


(c) Manipulating rope into 'S' shape

Figure 3.2: Qualitative visualization of results for rope manipulation task using Baxter robot. (a) Our robotics system setup. (b) The sequence of human demonstration images provided by the human during inference for the task of knot-tying (top row), and the sequences of observation states reached by the robot while imitating the given demonstration (bottom rows). (c) The sequence of human demonstration images and the ones reached by the robot for the task of manipulating rope into 'S' shape. Our agent is able to successfully imitate the demonstration.

3.3 Experiments

We evaluate our model by testing its performance on: rope manipulation using Baxter robot, navigation of a wheeled robot in cluttered office environments, and simulated 3D navigation. The key requirements of a good skill policy are that it should generalize to unseen environments and new goals while staying robust to irrelevant distractors in the observations. For rope manipulation, we evaluate generalization by testing the ability of the robot to manipulate the rope into configurations such as knots that were not seen during random exploration. For navigation, both real-world and simulation, we check generalization by testing on a novel building/floor.



(a) TPS-RPM error for 'S' shape manipulation

Method	Success %
Inverse Model [Nair et.al. 2017]	36% ± 9.6%
Forward-regularized GSP	44% ± 9.9%
Forward-consistent GSP [Ours]	**60**% ± **9.8**%

(b) Success rate for Knot-tying

Figure 3.3: GSP trained using forward consistency loss significantly outperforms the baselines at the task of (a) manipulating rope into 'S' shape as measured by TPS-RPM error and (b) knot-tying where we report success rate with bootstrap standard deviation.

Rope Manipulation

Manipulation of non-rigid and deformable objects, e.g., rope, is a challenging problem in robotics. Even humans learn complex rope manipulation such as tying knots, either by observing an expert perform it or by receiving explicit instructions. We test whether our agent could manipulate ropes by simply observing a human perform it. We use the data collected by [77], where a Baxter robot manipulated a rope kept on the table in front of it. During exploration, the robot interacts with the rope by using a pick and place primitive that chooses a random point on the rope and displaces it by a randomly chosen length and direction. This process is repeated a number of times to collect about 60K interaction pairs of the form (x_t, a_t, x_{t+1}) that are used to train the GSP.

During inference, our proposed approach is tasked to follow a visual demonstration provided by a human expert for manipulating the rope into a complex 'S' shape and tying a knot. Our agent, Baxter robot, only gets to observe the image sequence of intermediate states, as human manipulates the rope, without any access to the corresponding actions. Note that the knot shape is never encountered during the self-supervised data collection phase and therefore the learned GSP model would have to generalize to be able to follow the human demonstration. More details follow in the supplementary material, Section A.

Metric The performance of the model is evaluated by measuring the non-rigid registration cost between the rope state achieved by the robot and the state demonstrated by the human at every step in the demonstration. The matching cost is measured using the thin plate spline robust point matching technique (TPS-RPM) described in [21]. While TPS-RPM provides a good metric for measuring performance for constructing the 'S' shape, it is not an appropriate metric for knots because the configuration of the rope in a knot is 3D due to

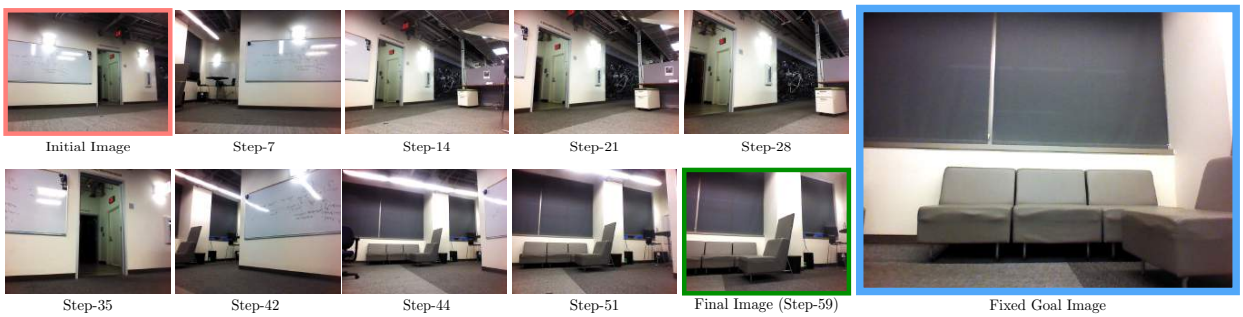


Figure 3.4: Visualization of the TurtleBot trajectory to reach a goal image (right) from the initial image (top-left). Since the initial and goal image have no overlap, the robot first explores the environment by turning in place. Once it detects overlap between its current image and goal image (i.e. step 42 onward), it moves towards the goal. Note that we did not explicitly train the robot to explore and such exploratory behavior naturally emerged from the self-supervised learning.

intertwining of the rope, and it fails to find the correct point correspondences. We, therefore, use success rate as the metric in knot tying where the completion of a successful knot is judged by human verification.

Visual Imitation Qualitative examples of our agent trying to manipulate rope are shown in Figure 3.2. We compare our approach to the baseline that deploys an inverse model which takes as input a pair of current and goal images to output the desired action to reach the goal [77]. We re-implement the baseline and train in our setup for a fair comparison. To further ablate the importance of consistency loss, we compare to a baseline that just uses a forward model as a regularizer of features. The results in Figure 3.3 show that our method significantly outperforms the baseline at task of manipulating the rope in the 'S' shape and achieves a success rate of 60% in comparison to 36% achieved by the baseline.

Navigation in Indoor Office Environments

A natural way to instruct a robot to move in an indoor office environment is to ask it to go near a certain location, such as a refrigerator or a someone's office. Instead of using language to command the robot, in this work, we communicate with the robot by either showing it a single image of the goal, or a sequence of images leading to faraway goals. In both scenarios, the robot is required to autonomously determine the motor commands for moving to the goal. We used TurtleBot2 for navigation using an onboard camera for sensing RGB images. For learning the GSP, an automated self-supervised scheme for data collection was devised that doesn't require human supervision. The robot collected a number of navigation trajectories from two floors of a academic building which in total contain 230K interactions data, i.e. (x_t, a_t, x_{t+1}). We then deployed the learned model on a separate floor of a building with substantially different textures and furniture layout for performing visual imitation at test

Model Name	Run Id-1	Run Id-2	Run Id-3	Run Id-4	Run Id-5	Run Id-6	Run Id-7	Run Id-8	Num Success
Random Search	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	0
Inverse Model [Nair et. al. 2017]	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	0
GSP-NoPrevAction-NoFwdConst	39 steps	34 steps	Fail	Fail	Fail	Fail	Fail	Fail	2
GSP-NoFwdConst	22 steps	22 steps	39 steps	48 steps	Fail	Fail	Fail	Fail	4
GSP (Ours)	119 steps	66 steps	144 steps	67 steps	51 steps	Fail	100 steps	Fail	6

Table 3.1: Quantitative evaluation of various methods on the task of navigating using a *single image* of goal in an unseen environment. Each column represents a different run of our system for a different initial/goal image pair. Our full GSP model takes longer to reach the goal on average given a successful run but reaches the goal successfully at a much higher rate.

time. The details of the robotic setup, data collection, and network architecture of GSP are described in supplementary material, Section A.

1) Goal Finding We first tested if the GSP learned by the TurtleBot can enable it to find its way to a goal that is within the same room from just a *single image* of the goal. To test the extrapolative generalization, we keep the Turtlebot approximately 20-30 steps away from the target location in a way that current and goal observations have no overlap as shown in Figure 3.4. We test the robot in an indoor office environment on a different floor that it has never encountered before. We judge the robot to be successful if it stops close to the goal and failure if it crashed into furniture or does not reach the goal within 200 steps. Since the initial and goal images have no overlap, classical techniques such as structure from motion that rely on feature matching cannot be used to infer the executed action. Therefore, in order to reach the goal, the robot must explore its surroundings. We find that our GSP model outperforms the baseline models in reaching the target location. Our model learns the exploratory behavior of rotating in place until it encounters an overlap between its current and goal image. Results are shown in Table 3.1 and videos are available at the website [1].

2) Visual Imitation In the previous paragraph, we saw that the robot can reach a goal that's within the same room. However, our agent is unable to reach far away goals such as in other rooms using just a single image. In such scenarios, an expert might communicate instructions like go to the door, turn right, go to the closest chair etc. Instead of language instruction, in our setup we provide a sequence of *landmark* images to convey the same high-level idea. These landmark images were captured from the robot's camera as the expert moved the robot from the start to a goal location. However, note that it is not necessary for the expert to control the robot to capture the images because we don't make use of the expert's actions, but only the images. Instead of providing the image after every action in the demonstration, we only provided every fifth image. The rationale behind this choice is that we want to sample the demonstration sparsely to minimize the agent's reliance on the expert. Such sub-sampling (as shown in Figure 3.5) provides an easy way to vary the complexity of the task.

[1] `https://pathak22.github.io/zeroshot-imitation/`

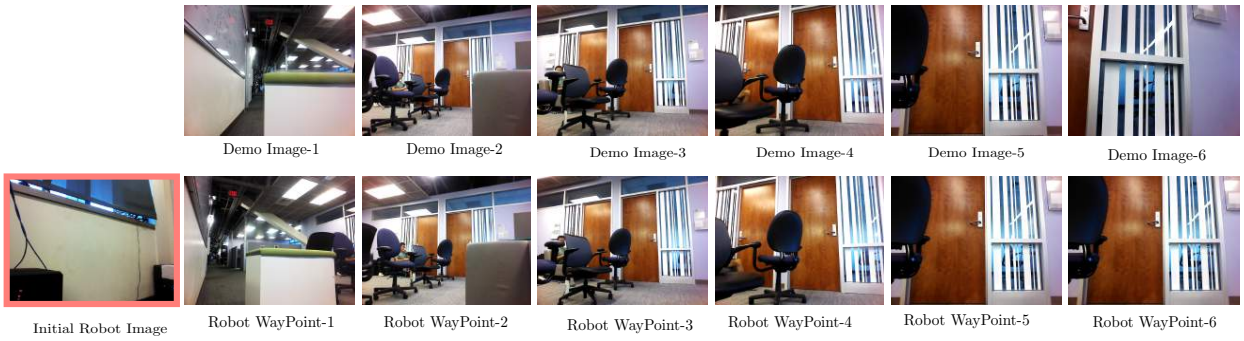


Figure 3.5: The performance of TurtleBot at following a visual demonstration given as a sequence of images (top row). The TurtleBot is positioned in a manner such that the first image in demonstration has no overlap with its current observation. Even under this condition the robot is able to move close to the first demo image (shown as Robot WayPoint-1) and then follow the provided demonstration until the end. This also exemplifies a failure case for classical methods; there are no possible keypoint matches between WayPoint-1 and WayPoint-2, and the initial observation is even farther from WayPoint-1.

We evaluate via multiple runs of two demonstrations, namely, maze demonstration where the robot is supposed to navigate through a maze-like path and perturbed loop demonstration, where the robot is supposed to make a complete loop as instructed by demonstration images. The loop demonstration is longer and more difficult than the maze. We start the agent from different starting locations and orientations with respect to that of demonstration. Each orientation is initialized such that no part of the demonstration's initial frame is visible. Results are shown in Table 3.2. When we sample every frame, our method and classical structure from motion can both be used to follow the demonstration. However, at sub-sampling rate of five, SIFT-based feature matching approaches did not work and ORBSLAM2 [75] failed to generate a map, whereas our method was successful. Notice that providing sparse landmark images instead of dense video adds robustness to the visual imitation task. In particular, consider the scenario in which the environment has changed since the time the demonstration was recorded. By not requiring the agent to match every demonstration image frame-by-frame, it becomes less sensitive to changes in the environment.

3D Navigation in VizDoom

We have evaluated our approach on real-robot scenarios thus far. To further analyze the performance and robustness of our approach through large scale experiments, we setup the same navigation task as described in previous subsection in a simulated VizDoom environment. Our goal is to measure: (1) the robustness of each method with proper error bars, (2) the role of initial self-supervised data collection for performance on visual imitation, (3) the quantitative difference in modeling forward consistency loss in feature space in comparison to

Model Name	Maze Demonstration			Loop Demonstration		
	Run-1	Run-2	Run-3	Run-1	Run-2	Run-3
SIFT	10%	5%	15%	—	—	—
GSP-NoPrevAction-NoFwdConst	60%	70%	100%	—	—	—
GSP-NoFwdConst	65%	90%	100%	0%	0%	0%
GSP (ours)	100%	60%	100%	0%	100%	100%

Table 3.2: Quantitative evaluation of TurtleBot's performance at following visual demonstrations in two scenarios: maze and the loop. We report the % of landmarks reached by the agent across three runs of two different demonstrations. Results show that our method outperforms the baselines. Note that 3 more trials of the loop demonstration were tested under significantly different lighting conditions and neither model succeeded. Detailed results are available in the supplementary materials.

raw visual space.

In VizDoom, we collect data by deploying two types of exploration methods: random exploration and curiosity-driven exploration [85]. The hypothesis is that if the initial data collected by the robot is driven by a better strategy than just random, this should eventually help the agent follow long demonstrations better. Our environment consists of 2 maps in total. We train on one map with 5 different starting positions for collecting exploration data. For validation, we collect 5 human demonstrations in a map with the same layout as in training but with different textures. For zero-shot generalization, we collect 5 human demonstrations in a novel map layout with novel textures. Exact details for data collection and training setup are in the supplementary, Section A.

Metric We report the median of maximum distance reached by the robot in following the given sequence of demonstration images. The maximum distance reached is the distance of farthest landmark point that the agent reaches contiguously, i.e., without missing any intermediate landmarks. Measuring the farthest landmark reached does not capture how efficiently it is reached. Hence, we further measure efficiency of the agent as the ratio of number of steps taken by the agent to reach farthest contiguous landmark with respect to the number of steps shown in human demonstrations.

Visual Imitation The task here is same as the one in real robot navigation where the agent is shown a sparse sequence of images to imitate. The results are in Table 3.3. We found that the exploration data collected via curiosity significantly improves the final imitation performance across all methods including the baselines with respect to random exploration. Our baseline GSP model with a forward regularizer instead of consistency loss ends up overfitting to the training layout. In contrast, our forward-consistent GSP model outperforms other methods in generalizing to new map with novel textures. This indicates that the forward consistency is possibly doing more than just regularizing the policy features. Training forward consistency loss in feature space further enhances the generalization even

Model Name	Same Map, Same Texture Median %	Same Map, Same Texture Efficiency %	Same Map, Diff Texture Median %	Same Map, Diff Texture Efficiency %	Diff Map, Diff Texture Median %	Diff Map, Diff Texture Efficiency %
Random Exploration for Data Collection:						
GSP-NoFwdConst	63.2 ± 5.7	36.4 ± 3.3	32.2 ± 0.7	28.9 ± 4.0	34.5 ± 0.6	23.1 ± 2.4
GSP (ours pixels)	62.2 ± 5.1	43.0 ± 2.6	32.4 ± 0.8	30.9 ± 2.9	35.4 ± 1.1	29.3 ± 3.9
GSP (ours features)	68.9 ± 6.9	53.9 ± 4.0	32.4 ± 0.7	47.4 ± 7.6	39.1 ± 2.0	30.4 ± 2.5
Curiosity-driven Exploration for Data Collection:						
GSP-NoFwdConst	78.2 ± 2.3	63.0 ± 4.3	43.2 ± 2.6	33.9 ± 3.0	40.2 ± 4.0	27.3 ± 1.9
GSP-FwdRegularizer	78.4 ± 3.4	59.8 ± 4.1	50.6 ± 4.7	30.9 ± 3.0	37.9 ± 1.1	28.9 ± 1.7
GSP (ours pixels)	78.2 ± 3.4	65.2 ± 4.2	47.1 ± 4.7	32.4 ± 3.0	44.8 ± 4.0	29.5 ± 1.9
GSP (ours features)	78.2 ± 4.6	67.0 ± 3.3	49.4 ± 4.8	26.9 ± 1.5	47.1 ± 3.0	24.1 ± 1.7

Table 3.3: Quantitative evaluation of our proposed GSP and the baseline models at following visual demonstrations in VizDoom 3D Navigation. Medians and 95% confidence intervals are reported for demonstration completion and efficiency over 50 seeds and 5 human paths per environment type.

when both pixel and feature space models perform similarly on training environment.

3.4 Related Work

Our work is closely related to imitation learning, but we address a different problem statement that gives less supervision and requires generalization across tasks during inference.

Imitation Learning The two main threads of imitation learning are behavioral cloning [93, 8], which directly supervises the mapping of states to actions, and inverse reinforcement learning [79, 1, 131, 63, 46], which recovers a reward function that makes the demonstration optimal (or nearly optimal). Inverse RL is most commonly achieved with state-actions, and is difficult to extend to fitting the reward to observations alone, though in principle state occupancy could be sufficient. Recent work in imitation learning [27, 32, 41] can generalize to novel goals, but require a wealth of demonstrations comprised of expert state-actions for learning. Our approach does not require expert actions at all.

Visual Demonstration The common scenario in LfD is to assume full knowledge of expert states and actions during demonstrations, but several works have focused on relaxing this supervision to visual observations alone. [77] observe a sequence of images from the expert demonstration for performing rope manipulations. [104, 103] imitate humans with robots by self-supervised learning but require expert supervision at training time. Third person imitation learning [112] and the concurrent work of imitation-from-observation [65] learn to translate expert observations into agent observations such that they can do policy optimization to minimize the distance between the agent trajectory and the translated demonstration, but they require demonstrations for learning. Visual servoing is a standard

problem in robotics [56] that seeks to take actions that align the agent's observation with a target configuration of carefully-designed visual features [125, 120] or raw pixel intensities [18]. Classical methods rely on fixed features or policies, but more recently end-to-end learning has improved results [61, 62].

Forward/Inverse Dynamics and Consistency Numerous prior works, such as [119, 81, 28], have learned forward dynamics model for planning actions. The works of [52, 121, 2, 85] jointly learn forward and inverse dynamics model but do not optimize for consistency between the forward and inverse dynamics. We empirically show that learning models by our forward consistency loss significantly improves task performance. Enforcing consistency as a meta-supervision has also been successful in finding visual correspondences [128] or unpaired image translations [129].

Goal Conditioning By parameterizing the value or policy function with a goal, an agent can learn and do multiple tasks. The idea of learning goal-conditioned policies has been explored in [99, 2, 77, 7]. Similarly to hindsight experience replay [7] we draw goals from experience, but our policy optimization has better sample efficiency through supervised learning and dynamics modeling instead of reinforcement learning. Moreover, we work from high-dimensional visual inputs instead of knowledge of the true states and do not make use of a task reward during training. In our setting, all of the expert goals are followed zero-shot since they are only revealed after learning.

3.5 Discussion

In this work, we presented a method for imitating expert demonstrations from visual observations alone. In contrast to most work in imitation learning, we never require access to expert actions. The key idea is to learn a GSP using data collected by self-supervised exploration. However, this limits the quality of the learned GSP as per the exploration data. For instance, we deploy random exploration on our real-world navigation robot, which means that it would almost never follow trajectories that go between rooms. Consequently, the learned GSP is unable to navigate towards a goal image taken in another room without requiring intermediate sub-goals. [85] show that the agent learns to move along corridors and transition between rooms purely driven by curiosity in VizDoom. Training GSP on such a structured data could equip the agent with more interesting search behaviors, e.g., going across rooms to find a goal. In general, using better methods of exploration for training the GSP could be a fruitful direction toward generalizing zeroshot imitation.

One limitation of our approach is that we require first-person view demonstrations. Extension to third-person demonstrations [112, 65] would make the method applicable in more general scenarios. Another limitation is that, in the current framework, it is implicitly assumed that the statistics of visual observations when the expert demonstrates the task and the agent follows it are similar. For e.g., when the expert performs a demonstration in one setting, say in daylight and the agent needs to imitate say in the evening, the change in the lighting conditions might result in worse performance. Making the GSP robust to such

nuisance changes or other changes in environment by domain adaptation would be necessary to scale the method to practical problems. Another thing to note is that, in the current framework, we do not learn from expert demonstrations, but simply imitate them. It would be interesting to investigate ways for an agent to learn from the expert to bias its exploration to more useful parts of the environment.

While we used a sequence of images to provide a demonstration, our work makes no image-specific assumptions and can be extended to using formal language for communicating goals. For instance, after training the GSP, instead of transforming an image into features ϕ as described in section 3.2, one could possibly learn a mapping to transform language instructions into this feature space.

Chapter 4

Leveraging Large Vision-Language Models

4.1 Introduction

Previous efforts have explored image-based goal specification, with significant successes in visual navigation and manipulation tasks [89, 78, 36, 111]. Yet existing image-based goal specification paradigms are limited because they are typically limited to a particular scene *instance* in an environment, whereas a *semantic* goal comprises multiple possible scene configurations. Reinforcement learning offers one of the most appealing premises in the study of AI: from a reward signal alone, algorithms which learn optimal policies that maximize expected reward can learn to perform navigation, dexterous manipulation, and host of other impactful tasks. However, discovering or specifying a reward function for a given task is often a very challenging problem, especially when one is considering agents that can learn from un-instrumented environments, e.g., from raw image observations alone. We wish to have an agent that can learn purely from pixels, with no access to the underlying state of the environment at any point during learning or task execution. Achieving this goal

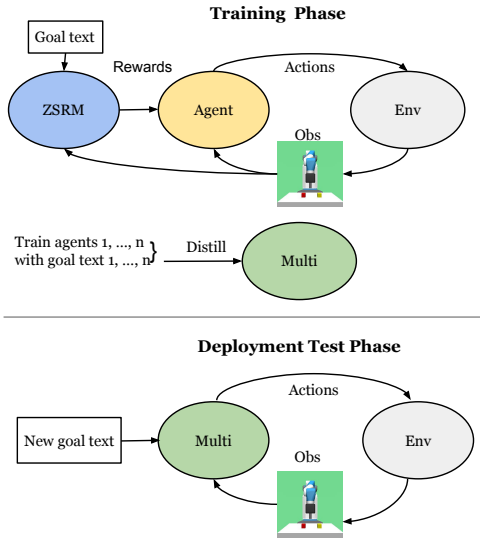


Figure 4.1: Our method uses only goal text from user and images from the environment at training time. We train several agents on several tasks and distill their policies into a single goal text conditioned policy that can generalize to new tasks with an unseen goal text description in a zero-shot manner. We assume no access to environment reward, state, demonstrations, or goal images at either train or test time.

without access to an instrumented reward function has been exceedingly challenging.

One can use image-based reward specification to cause a robot agent to navigate to a particular chair next to a specific tall plant, but that agent may not always succeed at the generic task of "go to a chair next to a tall flowering plant": e.g., if the goal specification image shows a red chair next to a plant with a yellow flower the agent may navigate away from a scene with a blue chair next to a red flower, depending on the model's underlying image representation. To be sure that the true goal is properly specified irrespective of the invariances of the model's underlying perceptual representation, a user may have to provide a set of goal image examples that cover the variation of the target concept, potentially a very expensive undertaking to collect or generate.

We advocate *semantic reward specification via grounded natural language*, where a user describes a goal configuration in the world using a natural language description referring to entities in the world. This direction has long been a "holy grail" of AI research and a presumed capability of AI science fiction—the ability to instruct a robot with natural language—yet attempts have been limited by the state of the art in grounded language perception. It also falls under the general umbrella of leveraging large-scale passive data to bootstrap embodied learning, where we rely on language as a means to provide necessary reward signal which might be missing in visual data alone.

Prior work has explored reward functions or policies that take natural language as for goal description [82, 11, 127, 38, 35, 45, 106]. However, they rely on reward signals that have access to state of the system or demonstrations of the task distribution they are training on. There are works that use human videos to learn reward functions to train their agent [106, 105, 103], but they require a curated dataset of humans performing the tasks.

Recently however, the advent of large-scale multimodal training data together with large capacity language and vision deep learning models has significantly advanced the state of the art. A steady series of innovations have advanced grounded language modeling, from early work on multimodal translation and fusion models [13, 94, 40], to large-scale joint embedding models [34, 95], to the plethora of multimodal transformer models currently under investigation [113, 67, 20, 48]. CLIP, in particular, demonstrated a transformative advance on zero-shot object recognition [95].

One way to communicate text-based goal to a robot is by simply offering a description of the goal configuration in natural language and using the CLIP embedding dot product with an observed image to evaluate proximity to goal state. Surprisingly, this can work in simple cases, for examples see the top example in figure 4.5. However, for more complex goals, such as those involving spatial relationships, the simple solution has poor performance as shown in the bottom example of figure 4.5.

To overcome these limitations and scale to complex manipulation tasks, we spatially ground the natural language goal in the image. We factor 'what' vs 'where/how' aspects of goal state, and offer a novel spatial-salience scheme to generate data using this factorization (Figure 4.2 top). We argue that existing (e.g., CLIP-like) models can be used to ground the *what* aspects of a goal quite effectively, including appropriate attribute and concept-level generalization, while a separate *where/how* module can ground spatial relationship

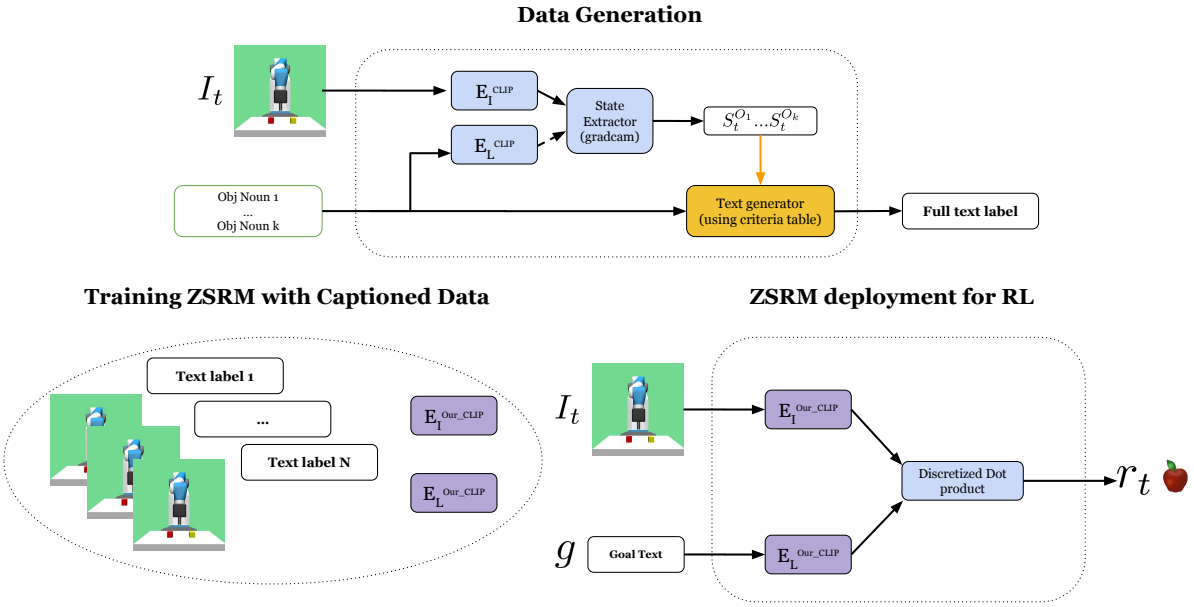


Figure 4.2: Our Zero-Shot Reward Model (ZSRM) is made by first generating spatial text labels of random exploration collected images with random initial states of arm and objects. We train ZSRM with the generated data in the same style of CLIP. We then use a discretized dot product of current image observation embedding and goal text embedding as our zero-shot reward. We train an agent via Reinforcement Learning with our reward model that is computed using only goal text and current observation as input. We assume no access to environment reward, state, demonstrations, or goal images at train or test time.

aspects of goal configuration. On this generated data, we first learn our Zero-Shot Reward Model (ZSRM) and then use ZSRM to learn individual text-conditioned policies. Finally, the individual task policies are distilled into a single goal-text-conditioned policy which is multi-task in nature and can execute unseen tasks in a zero-shot fashion – from a natural language description of task – without having to train a new policy for every new task.

4.2 Zero-Shot Reward Specification via Grounded Natural Language

In our work we assume an agent only has access to a text description of the goal desired by the user and image observations from the environment. At no point during training or testing does the agent have access to demonstrations, goal images, or reward from the environment. The agent only has access to a reward model that takes images and goal text as input to provide progress towards goal text description with a reward score output. The reward is then used to teach the agent how to achieve the goal described by the text with

online reinforcement learning. Given these assumptions we will now describe how we provide a zero-shot reward model by leveraging CLIP and how we learn a text conditioned policy with this model.

Vanilla Dot-product Visuolinguistic Base Reward Model

Our base model is the most intuitive way to use the CLIP model to compute reward. Our base model simply computes reward by taking a dot product between the goal text feature and image observation feature through CLIP's language and image encoders respectively: $r_t = E_I(I_t) \cdot E_L(g)$ A subset of tasks can be learned with this reward model. We visualize the limits of this model on two tasks in Figure 4.5. One significant limitation of CLIP is that it cannot distinguish spatial relationship of objects in images. This limits our base reward model from being useful for tasks that have spatial goals. Our full zero-shot reward model remedies this issue by leveraging CLIP in a very different way.

Spatially Grounded Visuolinguistic Zero-Shot Reward Model (ZSRM)

Data Generation via CLIP-Saliency Phrase Grounding We use a simple method that generates texts with spatial information for images using phrase grounding and spatial relationship processing. We assume we have access to the noun phrases that will be used by the user to specify their full goal text. Figure 4.2 illustrates how the object noun phrases are passed through CLIP's language encoder and the current image observation is passed through CLIP's image encoder. We use the language encodings as class embeddings of each object noun phrase to perform a saliency analysis (using Grad-CAM [102]) on the image encoding of the observation on the last convolutional layer (specifically, the ReLU layer of CLIP's ResNet-50 backbone). Saliency models such as Grad-CAM generally output a heatmap of the features that indicate a class exists in the current image input. The state extractor in Figure 4.2 computes the "state" of each object using the argmax of the saliency heatmap (see Figure 4.3). Once we have the the object states (object pixel coordinates), we use the criteria described in the next section to generate a full text description of the image describing the spatial relationship between the objects. The images we use to label are randomly sampled robot arm and object locations with random actions taken by the agent to move the arm.

Spatial Language Grounding Criteria We generate a spatial text label using the object pixel coordinates of one or more camera views as input to match various criteria. The criteria that matches our object state determines the text label of the image. The text labels used are the simplest spatial descriptions we could think of. We did not engineer what text labels to use.

Our spatial language grounding criteria is defined in Table 4.1. The first set (left of, right of, on top of, below, and in between) assume coordinates in a front camera view and the

semantics are defined in that camera view with positive y in the upward direction and positive x in the right direction. The second set (in front of & behind) has access to a left camera view with coordinate x2 pointing towards the right which is towards the front in the first camera view and y2 pointing upward similar to the first camera view. The second camera is needed to know if the object is placed in the front or behind another object correctly. The third set (close to & inside of) require access to a front camera view with a 45° downward tilt towards the ground. This camera view is needed to see if the object is getting closer to another object in two orthogonal directions at once where as a left only or front view only allows you to determine one dimension of closeness. For "inside of" a 45° camera helps the agent see if the object is going inside another object without occlusion. The ϵ threshold for "inside of" is much smaller than for "close to" since the centroid can come much closer when an object goes inside a container object.

Full Reward Model Training and Usage After generating spatial language labels for randomly collected images using CLIP-saliency phrase grounding, we train our full reward model similar to CLIP [95] with the similar visuolinguistc contrastive loss where the model essentially predicts which caption matches which image in every batch. The cosine similarity of the image and text embeddings of the correct pairs in the batch are maximized while the cosine similarity of the embeddings of the incorrect pairs are minimized. We initialized our model with the pretrained language and image encoders from CLIP. This, what we call 'full reward model', is essentially same as the base reward model but now it has been "fine-tuned" with spatial-grounded text-image data generated automatically. To prevent catastrophic forgetting and overfitting to just the new data, one could maintain a distillation loss with respect to the old model. However, we found that training the model from scratch on this new data to work well. Finally, the dot product of our new model can directly be used as reward but we found thresholding the dot product to obtain binarized reward to work better. In order to avoid engineering the threshold of our binarized reward we computed the dot product of positive and negative goal texts and if the highest dot product (argmax) was our positive goal text we output a reward of 1 and a reward of 0 otherwise. We found this to give us the best results. This finetuned zero-shot reward model can be used for a broad set of manipulation tasks to push or place objects to semantic locations which we showcase in our results.

Spatial Language Label	Label Grounding Criteria
Obj1 on the left of Obj2	$O_x^2 > O_x^1$
Obj1 on the right of Obj2	$O_x^1 > O_x^2$
Obj1 on top of Obj2	$\lvert O_x^1 - O_x^2 \rvert < \epsilon_1$ & $O_y^2 < O_y^1 < O_y^2 + \epsilon_2$
Obj1 below Obj2	$\lvert O_x^1 - O_x^2 \rvert < \epsilon_1$ & $O_y^1 < O_y^2 < O_y^1 + \epsilon_2$
Obj1 in between Obj2, Obj3	$min(O_x^2, O_x^3) < O_x^1 < max(O_x^2, O_x^3)$
Obj1 in front of Obj2	$O_{x2}^1 > O_{x2}^2$
Obj1 behind Obj2	$O_{x2}^2 > O_{x2}^1$
Obj1 close to Obj2	$\lVert O_{xy}^1 - O_{xy}^2 \rVert_2 < \epsilon$
Obj1 inside of Obj2	$\lVert O_{xy}^1 - O_{xy}^2 \rVert_2 < \epsilon$

Table 4.1: Spatial Language Grounding Criteria.

Figure 4.3: a) Mask R-CNN object detection results b) Mask R-CNN detection results on far view c) Mask R-CNN instance segmentation results on far view d) Grad-CAM result with 'red block' text input (white being highest intensity) e) Grad-CAM result with 'yellow block' text input (white being highest intensity).

Language Conditioned Multi-task Policy

Now that we have a method for learning tasks in a zero-shot reward fashion, we would like to be able to not require training a new policy for every new task and ideally have a language conditioned multi-task policy that takes in the goal text description of an unseen task and executes the task without needing any new samples from the environment.

To approach this goal, we first learn several tasks via reinforcement learning using our zero-shot reward model. We then create a large dataset of the rollouts of those polices and pair each trajectory with the goal text description of the tasks it was trained to learn. We then use behavioral cloning (supervised learning for predicting actions) to learn a policy that takes images and text goal task description as input and actions as target outputs. We use CLIP's language encoder as the text goal embedding that is fed to the multi-task policy. In addition we use image augmentation techniques to aid behavioral cloning in learning more robust policies.

4.3 Experiments

Phrase grounding and Base Zero-shot Reward Model Visualization

Object detectors are one way to extract object states for our spatial text generator, however, they are usually not used off the shelf and need to be fine-tuned with in domain data. In Figure 4.3, we show pretrained Mask R-CNN [43] outputs on different camera views for the block stack environment. As you can see in the first two subfigures, the blocks are not proposed as objects with Mask R-CNN from both far and close camera views. This is a demonstration of the need for in-domain fine-tuning of object detectors to work in the environment you want to use. Our Grad-CAM output from CLIP however, highlights exactly the objects we are interested in from the object noun phrases that describes each object in the last two subfigures. Another limitation with Mask R-CNN is that even if the object proposals were good, it would not necessarily classify objects it has not been trained for and therefore not output a filtered set of object proposals. In other words, we would not know

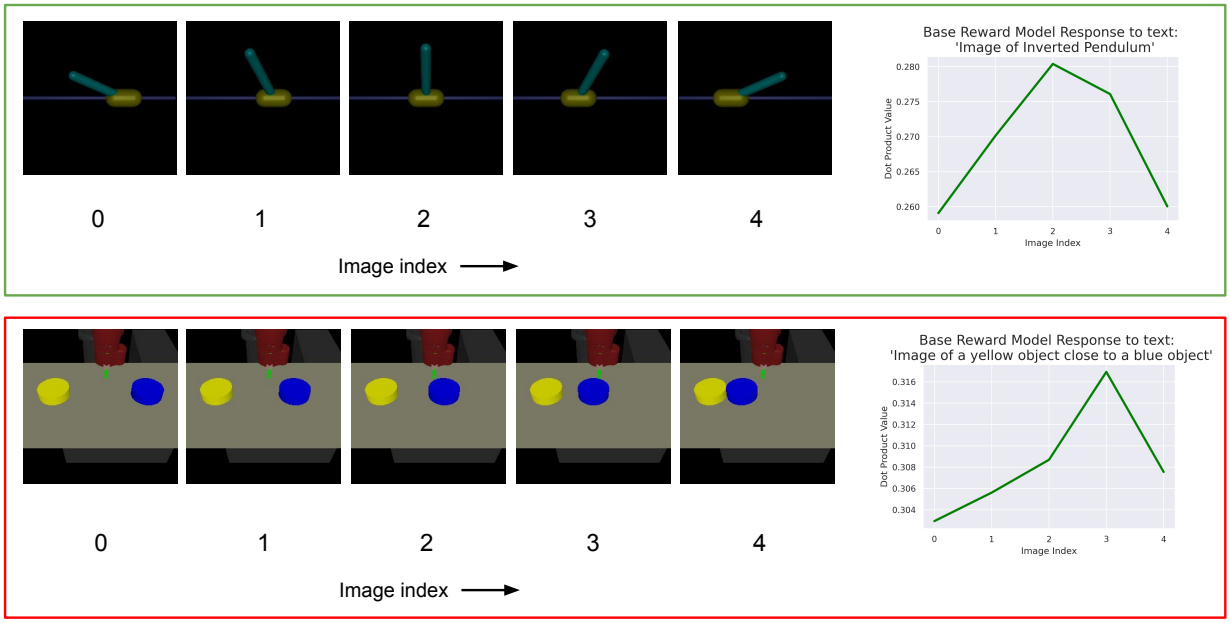


Figure 4.5: We visualize the results of the base reward model which is a trivial dot product between the goal language description and image observation. The top row (green box) displays a successful utilization of the base reward model and the bottom row (red box) shows a failure case. The x-axis represents image index.

which object is which if the detector hasn't been trained with the label of objects we care about.

In Figure 4.5 we show what our base reward model outputs on two goal descriptions: 1. inverted pendulum 2. yellow object close to a blue object. For the first goal description we observe that the dot product increases as the pendulum becomes more inverted from either side as desired. For the second goal description we observe that as the blue object gets closer to the yellow object the dot product increases except for the closest image where it dips which results in an undesired output. We observed this example and many others that the base reward model is not sufficient for recognizing object spatial relationships. The encoders are good at identifying what objects are in the image however, which is what we leverage to generate paired language image data for our full reward model.

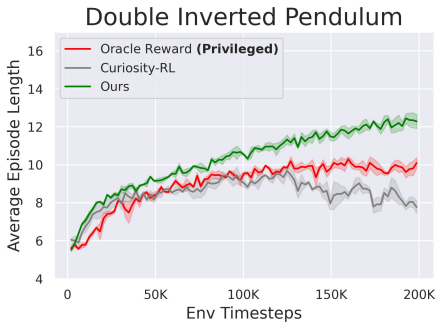


Figure 4.4: We showcase our base reward model trained policy in Double Inverted Pendulum performing slightly better than oracle reward. We average our results over 3 seeds per task.

Full Zero-shot Reward Model Results

In this section, we evaluate our full Zero-shot Reward Model on pushing, picking, and placing manipulation tasks performed in a planar setup. We train each task using our full zero-shot reward model output as reward for the PPO reinforcement learning algorithm [101]. We then train for the same tasks with other types of reward functions as baselines or privileged methods for comparison:

a) Oracle reward (privileged): this reward function has direct access to state to determine task completion. In other words it uses true x,y,z spatial positions of the objects to determine if the desired spatial relationship is reached and outputs 1 to the RL algorithm for every timestep the conditions of the task are met.

b) VICE (privileged): VICE[36] is used as a privileged method that has access to task goal image for comparison. We train the VICE reward model as a binary goal classifier that is trained with true goal images of the task such as "red block on top of yellow block" as positive images and images that aren't in the correct goal configuration as negative images. We discretize the model to output 1 if it determines the current image is positive and 0 otherwise.

c) Ours-base: Our base zero-shot reward model that uses the dot product between the goal text feature and image observation feature of CLIP as reward.

d) Curiosity-RL [86, 17]: Curiosity is used as a baseline since it only has access to images similar to our method for computing reward, but has only been successful for videogames such as Atari and Mario or locomotion where exploring new states leads to progressing through the task (going further in levels of game for example). It is less privileged however, in that it does not use language input for task specification.

Environment Reward Details for Oracle Evaluation: The oracle reward function for Double-Inverted-Pendulum is alive bonus minus distance penalty minus velocity penalty. The oracle for SawyerSimRobot-Pushing is a sparse reward that outputs one when the centroid distance between two pucks are below a threshold. The oracle reward for FetchSimRobot-Stacking is a sparse reward that outputs one when a yellow block is within a horizontal and vertical threshold distance of a red block. The oracle reward for FetchSimRobot-Placing is a sparse reward that outputs one when a yellow block is correctly placed on the right of a red block. See Figs. 4.3 & 4.5 for image observation examples of FetchSimRobot and SawyerSimRobot.

In Figure 4.6 we show how our full reward model performs on three planar manipulation tasks. We observe that our Full Reward Model performs as well as oracle reward and VICE both of which are priviledged on SawyerSimRobot-Pushing (pushing a blue puck close to a yellow puck), FetchSimRobot-Stacking (stacking a yellow block on top of a red block), and FetchSimRobot-Placing (placing a yellow block on the right of a red block). For Curiosity-RL we see that it learns the pushing pucks close together task but then starts learning separation of the pucks which reemphasizes that curiosity is only useful for tasks where exploring new dynamics leads to going farther in the task. Curiosity also has some trouble learning the double inverted pendulum task because the dynamics of the pendulum swinging can be hard

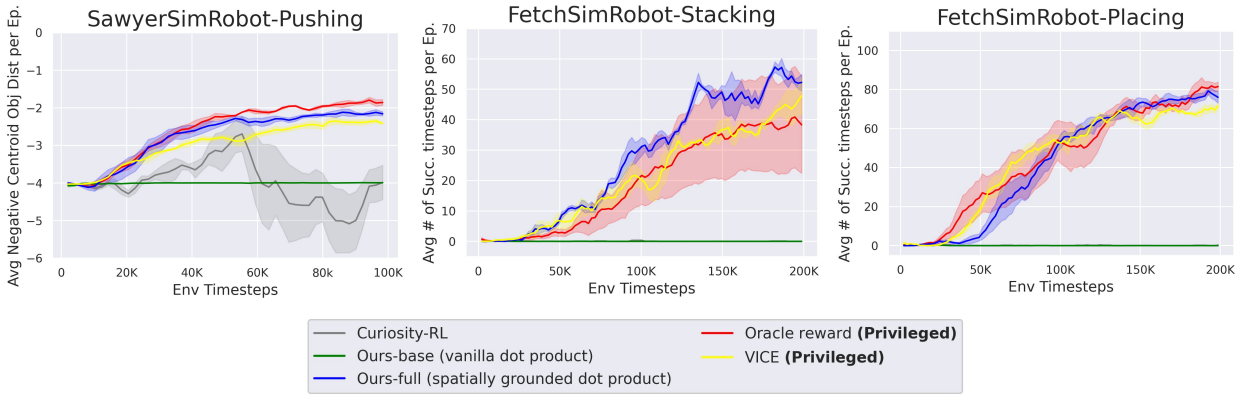


Figure 4.6: We showcase our Full Reward Model performing as well as oracle reward and VICE both of which are priviledged on SawyerSimRobot-Pushing (pushing a blue puck close to a yellow puck), FetchSimRobot-Stacking (stacking a yellow block on top of a red block), and FetchSimRobot-Placing (placing a yellow block on the right of a red block). We average results over 3 seeds per task. Oracle reward is privileged with true state information used to compute proximity to goal. VICE is privileged with true goal images used to train its reward model in classifying observations as close to goal or not.

to predict and therefore have misleading higher reward. Curiosity also does not learn the other manipulation tasks (stacking and placing) as those are more complex tasks that are harder to reach by exploration.

For Double inverted pendulum (Fig. 4.4) our base reward model does better than oracle by chance which we speculate is because the oracle reward was originally designed for state input and was not tuned for learning image to reward mapping. Our base reward model fails for pushing, stacking, and placing, which take "an image of a yellow block on top of a red block", "an image of a yellow object close to a blue object", and "an image of a yellow block on the right of a red block" as language input for those tasks respectively.

Multi-task Policy Results

In order to avoid having to train a new policy for every new task we want to learn, we train a multi-task policy with a set of training tasks and then show that it can generalize to a set of unseen test tasks by leveraging CLIP's language model to encode the goal text description of the tasks as conditioning input to our multi-task policy.

In FetchSimRobot env, we train 18 tasks with PPO for 200K steps with our zero-shot reward model described in the previous section and show generalization results for 18 unseen test tasks by training a language conditioned policy with behavior cloning on rollouts of the 18 training tasks. We collect 5000 steps (i.e., 50 trajectories) per task. We do this for pickplace-left, pickplace-right, and stack tasks for different object combinations. The object

	Seen distribution				Unseen distribution			
	train tasks		test tasks		train tasks		test tasks	
(episode reward stats)	mean	s.e.	mean	s.e.	mean	s.e.	mean	s.e.
No Conditioning	17.91	1.11	14.82	0.97	14.81	1.02	10.79	0.85
Primitive Code Cond.	26.71	1.23	17.20	1.03	17.03	1.07	11.74	0.87
Language Cond.	29.89	1.28	22.41	1.09	21.14	1.16	15.69	0.98

Table 4.2: Multi-task Policy Performance Results: We report multi-task average and standard error performance across all 18 training tasks and all 18 test tasks (unseen object color in goal text or image) for seen and unseen initial state distributions with no conditioning, primitive code conditioning, and language conditioning. The values reported are the episode reward averaged across 50 seeds per task policy rollout. The tasks are robotic arm manipulation of objects to different target semantic relationships of each other in the FetchSimRobot environment. While the policies were trained using our full zero-shot reward model, the reward metric reported is oracle reward to evaluate true performance.

training colors are red, green, and yellow, and the test color is blue. The multi-task policy has never seen blue block in text input or image input. (There are 18 training tasks because there are 3 relationship types and 6 different combinations of choosing two out of three training colors where order matters. There are 18 test tasks because there are 3 relationship types and 6 different combinations of pairing the test color to one of three training colors where order matters.)

In evaluation, we average episode rewards over 50 seeds. We then average across all training tasks to get the train metric and across all test tasks to get the test metric respectively. We compare our language conditioned policy with a policy trained without task labels, and one trained with a primitive code as the conditioning input. The latter baseline simply labels the training tasks with integers 0 to 17. Since the Primitive code conditioned policy has only been trained with primitive codes of the training tasks, when evaluating it for test tasks we allow the policy to use CLIP's language embedding to find the closest corresponding training primitive code. This is done by comparing the text description embedding of all the training tasks to the test task text description embedding and choosing the training task primitive code corresponding to the smallest L2 distance.

We use the same oracle reward that has direct access to state in figure 4.6 to measure the performance of our multi-task models. Every timestep the objects are in the correct target text description the agent is rewarded 1 point. The objects never start in the correct target state. Therefore it always takes several timesteps for the policy to move the objects to the correct state. All episodes timeout after 100 timesteps or if one of the objects falls off the table. In RL training and in policy rollout collection for behavior cloning each object has a one block width of variation in starting point location. We tested the policies on same distribution of start point variation interval (seen initial state distribution) and double the

start point variation interval (unseen initial state distribution).

In Table 4.2 we see that no conditioning policy has the lowest performance as expected since there is full ambiguity in what task to perform given only an image. However, since the no conditioning policy has been trained across many different object colors going to different target states it has learned general displacement of blocks to different semantic locations randomly that can sometimes be moved to the correct semantic location by chance during testing. For the training tasks it has higher performance than testing tasks since it can memorize to execute some of the training tasks that have almost the exact same initial states sampled during testing as those in the behavior cloning dataset and thereby execute correctly as memorized. We observe that the primitive code conditioned policy has much higher training performance than the no conditioning policy since it can disambiguate what task it needs to execute.

We observe that the language conditioned policy performs significantly better on test tasks than the primitive code conditioned policy since it can use the language embedding to infer the target task determined by the goal text description. The primitive code conditioning policy has access to the language embedding only to determine the closest training primitive code to the target task description. It's interesting to observe that the primitive code conditioning performs better than no conditioning on the test tasks because the closest primitive code can sometimes lead to a successful execution of the test task since it has some signal towards the correct task. One such example is that the test goal description task of "a blue block on the right of a yellow block" is mapped to the primitive code of the training goal description task of "a red block on the right of a yellow block". In Table 4.2 we also observe the same trend in the policy performances in unseen intial state distribution: the no conditioning policy performing the poorest on train and test tasks, the primitive code conditioned policy performing significantly better on both, the language conditioned policy performing the best at training and test tasks via its language structured specification of tasks.

We train the language conditioned policy with ResNet18 image encoding that is concatenated with CLIP language encoding both of which are pushed through an fc layer before concatenation. The concatenated vector is then inserted into two fc layers before predicting actions. The primitive code conditioned policy swaps the CLIP language encoding with an integer from 0-17 representing the primitive code input. The no conditioned policy only has two fc layers on top of the same image encoding as the other two policies. We apply an L2 regression loss on the output for predicting continuous actions (behavior cloning). The policy is trained using Adam optimizer with AMS grad with learning rate of 1e-4. The images are augmented with PyTorch RandomResizedCrop of 0.95 to 1.0 area and 0.98 to 1.02 aspect ratio randomization and resized to original image dimensions of 128x128. All the policies are trained for 300 epochs.

4.4 Related Work:

Goal conditioned policies Goal conditioned policies allow a user to specify the agent's goal. States [99, 7] and Images [89, 78, 36, 111, 72] are one way of specifying goal. However, they assume that the user has access to a photo or state of the completed task to give to the agent. We assume no access to goal images or state and use language which provides a natural form of supervision.

Goal text conditioned policies Several previous efforts train reward functions or policies that take natural language as input for goal description [82, 11, 127, 38, 35, 45, 106]. They all however rely on reward signals that have access to state of the system or demonstrations of the task distribution they are training on.

Learning reward functions There are works that use human videos to learn reward functions to train their agent with [105, 103, 106]. We however, don't need a curated dataset of humans performing the tasks we want our agent to train with. Having humans perform all tasks we are interested in may not scale well with the amount of labor needed for recording and curating those datasets. CLIP has the advantage of learning a caption model from 400 million image, text pairs from the publicly available sources on the internet WIT.

No Environment Reward There have been recent work on methods that use no reward signal from the environment to train for specific tasks, for instance, curiosity [86], DIAYN [30], intrinsic goals [78, 72], etc. These methods have different ways to go from self-directed exploration to solving tasks at test time, and rely on reward function or goal images to be given by the user. Such a signal could be unnatural and costly to obtain. Our method however, can learn a subset of manipulation tasks by specifying end goal with language.

Utility of large vision language models in RL Recent works leverage CLIP to do diverse manipulation tasks [109] and using GPT-3 to perform navigation [49]. However, former has access to labeled expert demonstrations for training their policy and latter requires predefined action primitives. We assume no access to demonstrations or goal images at training or test time and operate with direct low-level actions.

4.5 Discussion

In this work, we presented a method for learning a set of object manipulation tasks without access to state of the system by computing reward from pixels conditioned on text goal description alone. Our method doesn't use goal images or demonstrations at either training time or test time. We devised a zero-shot reward model that leverages a language vision model (CLIP) that has been trained on a very large dataset of captioned images on the internet to compute progress (reward) towards a goal text description. We use this zeroshot-reward model to collect data on many tasks to then supervise a language conditioned multi-task

policy that can execute new tasks without need of extra training. There are many future directions that can expand the abilities of our reward model such as taking into account pose of objects and state of objects (such as closed door).

Finally we must address the ethical perils inherent in our leverage of models trained on large-scale vision and language datasets. Such datasets are well known to suffer from dataset bias that can cause failure or unintended harm [16]. While the near-term risks appear to be limited with the robotic applications presently envisioned, practitioners should continuously monitor systems for bias against underrepresented groups and ensure that robotic systems work across all socioeconomic domains. Techniques for bias assessment and debiasing should be employed whenever possible to ensure this remains the case.

Chapter 5

Conclusion

In this thesis we discussed how to make multi-task policies more scalable by using cheaper sources of supervision for learning new tasks and cheaper task specifications.

In Chapter 2 we discussed how we can reuse demonstrations for old tasks to learn new tasks by segmenting those demonstrations into primitives and specifying new tasks with the same primitive vocabulary. One limitation of this work is that primitive vocabularies can be limiting in task expressivity in certain settings. Goal image or natural language task specifications may provide better task expressivity in those settings.

In Chapter 3 we discussed how we can use self-supervised exploration data to learn goal image conditioned recurrent multi-task policies via supervised learning where goal images, actions, observations are sampled from the exploration data. One limitation of this work is that long horizon tasks may be difficult to execute with goal images alone if the exploration data hasn't explored enough of the target task state space. Future work on more effective self-supervised exploration methods can help alleviate this limitation.

In Chapter 4 we discussed how large vision-language models can be used to learn natural language conditioned reward models to replace instrumented rewards or human scorers to significantly decrease human labor needed for learning natural language specified multi-task policies. One limitation of this work was that we were limited in the domain of tasks we were able to solve partially due to the limited quality of complex text and image alignment in current large vision-language models. Future work on higher quality text and image alignment in these models can help enable solving more complex tasks with zero-shot reward models.

Future Directions There are two directions we strongly advocate for to help make mult-task policies more scalable.

- **Improving Exploration**: Collecting self-supervised exploration is usually very cheap and if desired target state distributions are visited during exploration frequently enough, policies that can reach those target state distributions can be learned with minimal human supervision. However, target state distributions for complex tasks occur rarely or not at all with most current self-supervised exploration methods. Part of the problem

is that in many settings exploring the entire state space is infeasible and target state distributions are easily missed. One idea to improve desired state space coverage of exploration methods is to guide and narrow the exploration with natural language as in [74]. We think this is a promising direction for improving exploration methods.

- **Improving Sim to Real Gap**: If the Sim to Real Gap was completely solved, several important obstacles can be significantly averted by training policies only in simulation. First, in simulation since there is direct access to state, expensive reward signal instrumentation can be avoided. Second, in the real world certain desired states occur with low frequency without explicit engineering, but in simulation the environment and agent can be set to desired tail distribution states. Lastly, in the real world data sampling is limited to real-time data collection where as in simulation the only limitation is compute availability for speeding up data sampling. Due to these advantages, we strongly advocate research in this direction and fortunately there has been a lot of promising recent work in this area [98, 117, 90, 58, 26].

These two directions are orthogonal because in high fidelity simulators we don't have enough compute to visit all possible states in complex environments and thereby require better exploration methods to learn complex tasks in a scalable fashion. We are excited to see what future research will accomplish in these directions.

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship learning via inverse reinforcement learning". In: *ICML*. 2004.

[2] Pulkit Agrawal et al. "Learning to poke by poking: Experiential learning of intuitive physics". In: *NIPS* (2016).

[3] Jean-Baptiste Alayrac et al. "Learning from narrated instruction videos". In: *IEEE transactions on pattern analysis and machine intelligence* 40.9 (2018), pp. 2194–2208.

[4] K. Ali and K. Saenko. "Confidence-Rated Multiple Instance Boosting for Object Detection". In: *CVPR*. 2014.

[5] Jacob Andreas, Dan Klein, and Sergey Levine. "Modular multitask reinforcement learning with policy sketches". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 166–175.

[6] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. "Support vector machines for multiple-instance learning". In: *NIPS*. 2002.

[7] Marcin Andrychowicz et al. "Hindsight experience replay". In: *arXiv preprint arXiv:1707.01495* (2017).

[8] Brenna D Argall et al. "A survey of robot learning from demonstration". In: *Robotics and autonomous systems* (2009).

[9] Pierre-Luc Bacon, Jean Harb, and Doina Precup. "The option-critic architecture". In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[10] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[11] Dzmitry Bahdanau et al. "Learning to understand goal specifications by modelling reward". In: *arXiv preprint arXiv:1806.01946* (2018).

[12] Albert Bandura and Richard H Walters. *Social learning theory*. Vol. 1. Prentice-hall Englewood Cliffs, NJ, 1977.

[13] Kobus Barnard et al. "Matching words and pictures". In: *The Journal of Machine Learning Research* (2003).

[14] Marc G Bellemare et al. "Autonomous navigation of stratospheric balloons using reinforcement learning". In: *Nature* 588.7836 (2020), pp. 77–82.

[15] Cynthia Breazeal and Brian Scassellati. "Robots that imitate humans". In: *Trends in cognitive sciences* (2002).

[16] Joy Buolamwini and Timnit Gebru. "Gender shades: Intersectional accuracy disparities in commercial gender classification". In: *Conference on fairness, accountability and transparency*. PMLR. 2018, pp. 77–91.

[17] Yuri Burda et al. "Large-scale study of curiosity-driven learning". In: *arXiv preprint arXiv:1808.04355* (2018).

[18] Guillaume Caron, Eric Marchand, and El Mustapha Mouaddib. "Photometric visual servoing for omnidirectional cameras". In: *Autonomous Robots* 35.2-3 (2013), pp. 177–193.

[19] Yevgen Chebotar et al. "Combining model-based and model-free updates for trajectory-centric reinforcement learning". In: *International conference on machine learning*. PMLR. 2017, pp. 703–711.

[20] Yen-Chun Chen et al. "Uniter: Learning universal image-text representations". In: *ECCV* (2020).

[21] Haili Chui and Anand Rangarajan. "A new point matching algorithm for non-rigid registration". In: *CVIU* (2003).

[22] Andrew J Davison and David W Murray. "Mobile robot localisation using active vision". In: *ECCV*. 1998.

[23] Peter Dayan and Geoffrey E Hinton. "Feudal reinforcement learning". In: *Advances in neural information processing systems*. 1993, pp. 271–278.

[24] Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. "Solving the multiple instance problem with axis-parallel rectangles". In: *Artificial intelligence* (1997).

[25] Rüdiger Dillmann. "Teaching and learning of robot tasks via observation of human performance". In: *Robotics and Autonomous Systems* (2004).

[26] Yuqing Du et al. "Auto-Tuned Sim-to-Real Transfer". In: *ICRA* (2021).

[27] Yan Duan et al. "One-shot imitation learning". In: *NIPS*. 2017.

[28] Frederik Ebert et al. "Self-supervised visual planning with temporal skip connections". In: *arXiv preprint arXiv:1710.05268* (2017).

[29] Benjamin Eysenbach et al. "Diversity is All You Need: Learning Skills without a Reward Function". In: *arXiv preprint* (2018).

[30] Benjamin Eysenbach et al. "Diversity is all you need: Learning skills without a reward function". In: *arXiv preprint arXiv:1802.06070* (2018).

[31] Pedro F Felzenszwalb et al. "Object detection with discriminatively trained part-based models". In: *IEEE Tran. PAMI* (2010).

[32] Chelsea Finn et al. "One-shot visual imitation learning via meta-learning". In: *CoRL* (2017).

[33] David Foster and Peter Dayan. "Structure in the space of value functions". In: *Machine Learning* (2002).

[34] Andrea Frome et al. "Devise: A deep visual-semantic embedding model". In: *Advances in neural information processing systems* (2013).

[35] Justin Fu et al. "From language to goals: Inverse reinforcement learning for vision-based instruction following". In: *arXiv preprint arXiv:1902.07742* (2019).

[36] Justin Fu et al. "Variational inverse control with events: A general framework for data-driven reward definition". In: *arXiv preprint arXiv:1805.11686* (2018).

[37] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[38] Prasoon Goyal, Scott Niekum, and Raymond J Mooney. "PixL2R: Guiding Reinforcement Learning Using Natural Language by Mapping Pixels to Rewards". In: *arXiv preprint arXiv:2007.15543* (2020).

[39] Alex Graves et al. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks". In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 369–376.

[40] Sergio Guadarrama et al. "Understanding object descriptions in robotics by open-vocabulary object retrieval and detection". In: *The International Journal of Robotics Research* 35.1-3 (2016), pp. 265–280.

[41] Saurabh Gupta et al. "Cognitive mapping and planning for visual navigation". In: *CVPR* (2017).

[42] Kaiming He et al. "Deep residual learning for image recognition". In: *CVPR*. 2016.

[43] Kaiming He et al. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[44] David Heckerman. "A tractable inference algorithm for diagnosing multiple diseases". In: *arXiv preprint arXiv:1304.1511* (2013).

[45] Karl Moritz Hermann et al. "Grounded language learning in a simulated 3d world". In: *arXiv preprint arXiv:1706.06551* (2017).

[46] Jonathan Ho and Stefano Ermon. "Generative Adversarial Imitation Learning". In: *NIPS*. 2016.

[47] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* (1997).

[48] Ronghang Hu and Amanpreet Singh. "UniT: Multimodal Multitask Learning with a Unified Transformer". In: *arXiv preprint arXiv:2102.10772* (2021).

[49] Wenlong Huang et al. "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents". In: *arXiv preprint arXiv:2201.07207* (2022).

[50] Ahmed Hussein et al. "Imitation learning: A survey of learning methods". In: *ACM Computing Surveys (CSUR)* (2017).

[51] Katsushi Ikeuchi and Takashi Suehiro. "Toward an assembly plan from observation. I. Task recognition with polyhedral objects". In: *IEEE Transactions on Robotics and Automation* (1994).

[52] Michael I Jordan and David E Rumelhart. "Forward models: Supervised learning with a distal teacher". In: *Cognitive science* (1992).

[53] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *ICLR* (2015).

[54] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[55] Thomas Kipf et al. "CompILE: Compositional Imitation Learning and Execution". In: *International Conference on Machine Learning*. 2019, pp. 3418–3428.

[56] Hashimoto Koichi and Husband Tom. *Visual servoing: real-time control of robot manipulators based on visual sensory feedback*. Vol. 7. World scientific, 1993.

[57] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[58] Ashish Kumar et al. "RMA: Rapid Motor Adaptation for Legged Robots". In: *RSS* (2021).

[59] Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue. "Learning by watching: Extracting reusable task knowledge from visual observation of human performance". In: *IEEE Transactions on Robotics and Automation* (1994).

[60] Yasuo Kuniyoshi, Masayuki Inaba, and Hirockika Inoue. "Teaching by showing: Generating robot programs by visual observation of human performance". In: *International Symposium on Industrial Robots*. 1989.

[61] Thomas Lampe and Martin Riedmiller. "Acquiring visual servoing reaching and grasping skills using neural reinforcement learning". In: *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE. 2013, pp. 1–8.

[62] Alex X Lee, Sergey Levine, and Pieter Abbeel. "Learning visual servoing with deep features and fitted q-iteration". In: *arXiv preprint arXiv:1703.11000* (2017).

[63] Sergey Levine et al. "Learning hand-eye coordination for robotic grasping with large-scale data collection". In: *ISER*. 2016.

[64] Yuanlong Li et al. "Transforming cooling optimization for green data center via deep reinforcement learning". In: *IEEE transactions on cybernetics* 50.5 (2019), pp. 2002–2013.

[65] YuXuan Liu et al. "Imitation from observation: Learning to imitate behaviors from raw video via context translation". In: *ICRA* (2018).

[66] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[67] Jiasen Lu et al. "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks". In: *arXiv preprint arXiv:1908.02265* (2019).

[68] Parsa Mahmoudieh, Trevor Darrell, and Deepak Pathak. "Weakly-Supervised Trajectory Segmentation for Learning Reusable Skills". In: *ICLR 2020 Workshop on Bridging AI and Cognitive Science*. 2020.

[69] Parsa Mahmoudieh, Deepak Pathak, and Trevor Darrell. "Zero-Shot Reward Specification via Grounded Natural Language". In: *ICML* (2022).

[70] Ajay Mandlekar et al. "RoboTurk: A Crowdsourcing Platform for Robotic Skill Learning through Imitation". In: *Conference on Robot Learning*. 2018.

[71] Mapillary. "Open source Structure from Motion pipeline". In: *https://github.com/mapillary/OpenSfM* (2016).

[72] Russell Mendonca et al. "Discovering and achieving goals via world models". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 24379–24391.

[73] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* (2015).

[74] Jesse Mu et al. "Improving intrinsic exploration with language abstractions". In: *arXiv preprint arXiv:2202.08938* (2022).

[75] Raul Mur-Artal and Juan D Tardós. "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras". In: *IEEE Transactions on Robotics* (2017).

[76] Iftekhar Naim et al. "Unsupervised alignment of natural language instructions with video segments". In: *Twenty-Eighth AAAI Conference on Artificial Intelligence*. 2014.

[77] Ashvin Nair et al. "Combining Self-Supervised Learning and Imitation for Vision-Based Rope Manipulation". In: *ICRA* (2017).

[78] Ashvin V Nair et al. "Visual reinforcement learning with imagined goals". In: *NeurIPS*. 2018.

[79] Andrew Y Ng and Stuart J Russell. "Algorithms for inverse reinforcement learning". In: *ICML*. 2000, pp. 663–670.

[80] Scott Niekum et al. "Learning and generalization of complex tasks from unstructured demonstrations". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5239–5246.

[81] Junhyuk Oh et al. "Action-conditional video prediction using deep networks in atari games". In: *NIPS*. 2015.

[82] Junhyuk Oh et al. "Zero-shot task generalization with multi-task deep reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2661–2670.

[83] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. "Intrinsic motivation systems for autonomous mental development". In: *Evolutionary Computation* (2007).

[84] Deepak Pathak, Philipp Krähenbühl, and Trevor Darrell. "Constrained Convolutional Neural Networks for Weakly Supervised Segmentation". In: *ICCV*. 2015.

[85] Deepak Pathak et al. "Curiosity-driven Exploration by Self-supervised Prediction". In: *ICML*. 2017.

[86] Deepak Pathak et al. "Curiosity-driven exploration by self-supervised prediction". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2778–2787.

[87] Deepak Pathak et al. "Fully Convolutional Multi-Class Multiple Instance Learning". In: *ICLR*. 2015.

[88] Deepak Pathak et al. "Zero-Shot Visual Imitation". In: *ICLR*. 2018.

[89] Deepak Pathak et al. "Zero-shot visual imitation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018, pp. 2050–2053.

[90] Xue Bin Peng et al. "Sim-to-real transfer of robotic control with dynamics randomization". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 3803–3810.

[91] Pedro O Pinheiro and Ronan Collobert. "Weakly Supervised Semantic Segmentation with Convolutional Networks". In: *CVPR*. 2015.

[92] Lerrel Pinto and Abhinav Gupta. "Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours". In: *ICRA* (2016).

[93] Dean A Pomerleau. "ALVINN: An autonomous land vehicle in a neural network". In: *NIPS*. 1989.

[94] Ariadna Quattoni, Michael Collins, and Trevor Darrell. "Learning visual representations using images with captions". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2007, pp. 1–8.

[95] Alec Radford et al. "Learning transferable visual models from natural language supervision". In: *arXiv preprint arXiv:2103.00020* (2021).

[96] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic backpropagation and approximate inference in deep generative models". In: *arXiv preprint arXiv:1401.4082* (2014).

[97] Alexander Richard, Hilde Kuehne, and Juergen Gall. "Weakly supervised action learning with rnn based fine-to-coarse modeling". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 754–763.

[98] Fereshteh Sadeghi and Sergey Levine. "Cad2rl: Real single-image flight without a single real image". In: *arXiv preprint arXiv:1611.04201* (2016).

[99] Tom Schaul et al. "Universal value function approximators". In: *ICML*. 2015.

[100] Jurgen Schmidhuber. "A possibility for implementing curiosity and boredom in model-building neural controllers". In: *From animals to animats: Proceedings of the first international conference on simulation of adaptive behavior*. 1991.

[101] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[102] Ramprasaath R Selvaraju et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[103] Pierre Sermanet, Kelvin Xu, and Sergey Levine. "Unsupervised perceptual rewards for imitation learning". In: *arXiv preprint arXiv:1612.06699* (2016).

[104] Pierre Sermanet et al. "Time-Contrastive Networks: Self-Supervised Learning from Video". In: *ICRA*. 2018.

[105] Pierre Sermanet et al. "Time-contrastive networks: Self-supervised learning from video". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1134–1141.

[106] Lin Shao et al. "Concept2robot: Learning manipulation concepts from instructions and human demonstrations". In: *Proceedings of Robotics: Science and Systems (RSS)*. 2020.

[107] Pratyusha Sharma et al. "Multiple interactions made easy (mime): Large scale demonstrations data for imitation". In: *arXiv preprint arXiv:1810.07121* (2018).

[108] Kyriacos Shiarlis et al. "Taco: Learning task decomposition via temporal alignment for control". In: *arXiv preprint arXiv:1803.01840* (2018).

[109] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. "CLIPort: What and Where Pathways for Robotic Manipulation". In: *arXiv preprint arXiv:2109.12098* (2021).

[110] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), pp. 484–489.

[111] Avi Singh et al. "End-to-end robotic reinforcement learning without reward engineering". In: *arXiv preprint arXiv:1904.07854* (2019).

[112] Bradly C. Stadie, Pieter Abbeel, and Ilya Sutskever. "Third-Person Imitation Learning". In: *ICLR*. 2017.

[113] Weijie Su et al. "Vl-bert: Pre-training of generic visual-linguistic representations". In: *arXiv preprint arXiv:1908.08530* (2019).

[114] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems* 27 (2014).

[115] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge, 1998.

[116] Richard S Sutton, Doina Precup, and Satinder Singh. "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning". In: *Artificial intelligence* 112.1-2 (1999), pp. 181–211.

[117] Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.

[118] Alexander Sasha Vezhnevets et al. "Feudal networks for hierarchical reinforcement learning". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 3540–3549.

[119] Manuel Watter et al. "Embed to control: A locally linear latent dynamics model for control from raw images". In: *NIPS*. 2015.

[120] William J Wilson, CC Williams Hulls, and Graham S Bell. "Relative end-effector control using cartesian position based visual servoing". In: *IEEE Transactions on Robotics and Automation* 12.5 (1996), pp. 684–696.

[121] Daniel M Wolpert, Zoubin Ghahramani, and Michael I Jordan. "An internal model for sensorimotor integration". In: *Science-AAAS-Weekly Paper Edition* (1995).

[122] Ali Yahya et al. "Collective robot reinforcement learning with distributed asynchronous guided policy search". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 79–86.

[123] Yezhou Yang et al. "Robot Learning Manipulation Action Plans by "Watching" Unconstrained Videos from the World Wide Web". In: *AAAI*. 2015.

[124] Yichun Yin, Yangqiu Song, and Ming Zhang. "Document-level multi-aspect sentiment classification as machine comprehension". In: *Proceedings of the 2017 conference on empirical methods in natural language processing*. 2017, pp. 2044–2054.

[125] Billibon H Yoshimi and Peter K Allen. "Active, uncalibrated visual servoing". In: *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE. 1994, pp. 156–161.

[126] Cha Zhang, John C Platt, and Paul A Viola. "Multiple instance boosting for object detection". In: *NIPS*. 2005.

[127] Li Zhou and Kevin Small. "Inverse Reinforcement Learning with Natural Language Goals". In: *arXiv preprint arXiv:2008.06924* (2020).

[128] Tinghui Zhou et al. "Learning dense correspondence via 3d-guided cycle consistency". In: *CVPR*. 2016.

[129] Jun-Yan Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *ICCV* (2017).

[130] Dimitri Zhukov et al. "Cross-task weakly supervised learning from instructional videos". In: *arXiv preprint arXiv:1903.08225* (2019).

[131] Brian D. Ziebart et al. "Maximum entropy inverse reinforcement learning". In: *AAAI*. 2008.

Appendix A

Chapter 2 Appendix

We evaluated our proposed approach across number of environments and tasks. In this section, we provide additional details about the experimental task setup and hyperparameters.

Rope Manipuation

Robotic Setup Our setup of Baxter robot for rope manipulation task follows the one described in [77]. We re-use the data that is collected by a Baxter robot interacting with a rope kept on a table in front of it in a self-supervised manner, and consists of approximately 60K interaction pairs.

Implementation Details The base architecture for all the methods consists of a pre-trained AlexNet, whose features are fed into a skill policy network that predicts the location of grasp, direction of displacement, and the magnitude of displacement. For the forward regularizer baseline, a forward model is trained to jointly regularize the AlexNet features along with the skill policy network with loss weight of forward model set to 0.1. For our proposed forward-consistent GSP, a forward consistency loss is then applied to the actions predicted by the skill policy network. The forward consistency loss weight is set to 0.1. Since this is a fully observed setup, we did not use recurrence in any of the skill policy networks. All the models are optimized using Adam [53] with a learning rate of $1e-4$. For the first 40K iterations, the AlexNet weights were frozen, and then fine-tuned jointly with the later layers.

Navigation in Indoor Office Environments

Robotic Setup We used the TurtleBot2 robot comprising of a wheeled Kobuki base and an Orbbec Astra camera for capturing RGB images for all our experiments. The robot's action space had four discrete actions: move forward, turn left, turn right, and stand still (i.e., no-op). The forward action is approximately 10cm forward translation and the turning actions are approximately 14-18 degrees of rotation. These numbers vary due to the use of velocity control. A powerful on-board laptop was used to process the images and infer

Model Name	Maze Runs - Optimal Steps: 100 Run-1	Run-2	Run-3	Loop Runs - Optimal Steps: 85 Run-1	Run-2	Run-3
SIFT	2/20 (10)	1/20 (9)	3/20 (38)	—	—	—
GSP-NoPrevAction-NoFwdConst	12/20 (109)	14/20 (184)	20/20 (263)	—	—	—
GSP-NoFwdConst	13/20 (147)	18/20 (325)	20/20 (166)	0/17 (0)	0/17 (0)	0/17 (0)
GSP (ours)	20/20 (353)	12/20 (194)	20/20 (168)	0/17 (0)	17/17 (243)	17/17 (165)

Table A.1: Quantitative evaluation of TurtleBot's performance at following visual demonstrations in two conditions: maze and the loop. The fraction denotes how many landmarks it reaches out of the total number of landmarks in the full demonstration. The bracketed number represents the number of actions the agent took to reach its farthest landmark.

the motor commands. Several modifications were made to the default TurtleBot setup: the base's batteries were replaced with longer lasting ones, and the default NVIDIA Jetson TK1 embedded board was replaced with a more powerful GigaByte Aero laptop and an accompanying portable charging power bank.

Self-supervised Data Collection We devised an automated self-supervised scheme for data collection which does not require any human supervision. In our scheme, the robot first samples one out of four actions and then the number of times to repeat the selected action (i.e. action repeat). The no-op action is sampled with probability 0.05 and the other three actions are sampled with equal probability. In case the no-op action is chosen, an action repeat of {1, 2} steps is uniformly sampled. In case of other actions, an action repeat of 1-5 steps is randomly and uniformly chosen. The robot autonomously repeated this process and collected 230K interactions from two floors of an academic building. If the robot crashes into an object, it performs a reset maneuver by first moving backwards and then turning right/left by a uniformly sampled angle between 90-270 degrees. A separate floor of the building with substantially different furniture layout and visual textures is then used for testing the learned model.

Implementation Details The data collected by self-supervised exploration is then used to train our *recurrent forward-consistent GSP*. The base architecture of our model is an ImageNet pre-trained ResNet-50 [42] network. Input are the images and output are the actions of robot. The forward consistency model is first pre-trained and then fine-tuned together end-to-end with the GSP. The loss weight of the forward model is 0.1, and the objective is minimized using Adam [53] with learning rate of $5e-4$.

3D Navigation in VizDoom

Self-supervised Data Collection Our environment consists of two map. One map is used for training and validation, with different textures for validation. Second map has different textures than training and validation and is used for generalization experiments. For both curiosity and random exploration, we collect a total of 1.5 million frames each with action

Model Name	Same Map, Same Texture Mean %	Same Map, Same Texture Efficiency %	Same Map, Diff Texture Mean %	Same Map, Diff Texture Efficiency %	Diff Map, Diff Texture Mean %	Diff Map, Diff Texture Efficiency %
Random Exploration for Data Collection:						
GSP-NoFwdConst	61.8 ± 0.9	60.4 ± 2.1	37.6 ± 0.7	68.6 ± 2.5	42.2 ± 0.8	50.6 ± 1.9
GSP (ours pixels)	61.0 ± 1.0	68.0 ± 2.2	38.1 ± 0.7	69.1 ± 2.5	40.3 ± 0.9	64.2 ± 2.3
GSP (ours features)	62.0 ± 1.0	75.8 ± 2.5	37.0 ± 0.7	87.1 ± 2.8	48.7 ± 0.9	52.5 ± 1.8
Curiosity-driven Exploration for Data Collection:						
GSP-NoFwdConst	70.7 ± 0.9	66.9 ± 1.4	49.8 ± 0.8	55.8 ± 2.2	51.2 ± 1.0	39.5 ± 1.3
GSP-FwdRegularizer	70.6 ± 0.9	67.9 ± 1.6	51.9 ± 0.8	49.3 ± 1.6	48.3 ± 1.0	49.3 ± 1.8
GSP (ours pixels)	71.0 ± 0.9	73.1 ± 2.7	53.3 ± 0.9	53.4 ± 2.0	52.2 ± 1.0	44.0 ± 1.5
GSP (ours features)	68.8 ± 1.0	72.0 ± 1.7	53.2 ± 0.8	53.0 ± 2.3	52.8 ± 0.9	37.7 ± 1.3

Table A.2: Quantitative evaluation of our GSP and baselines at following visual demonstrations in VizDoom 3D Navigation. Means and standard errors are reported for demonstration completion and efficiency over 50 seeds and 5 human paths per environment type.

repeat of 4 collected in the standard `DoomMyWayHome` map used for training in [85]. $\sim \frac{2}{3}$ of the data comes from random-room resets, and $\sim \frac{1}{3}$ of the data comes from a fixed-room reset (i.e, room number 10). The curiosity policy was half sampled and half greedy with the exact split being 40% greedy policy random-room reset, 25% sample policy random-room reset, 25% sample policy fixed-room reset, and 10% greedy policy fixed-room reset.

For each scenario, we collect 5 human demonstrations each and give every 10th frame as input to the agent for the task of visual imitation. For each human path, we evaluate on 50 different seeds where the agent starts with a uniformly sampled orientation. We then get the median across 250 (50x5) total runs for each type of environment and report median of the percentage of the human path reached by the agent and how soon it got to that point relative to the human.

In the main paper, we report median accuracy and the confidence interval for median [1]. Since the initial position of the agent is randomized in orientation compared to the one in visual demonstration, the mean results suffer from high variance due to outliers. Hence, median accuracy results in a more reliable metric. However, we report mean results in Table A.2 for the completion.

Implementation Details All models were trained with batch size 64, Adam Solver with 1e-4 learning rate, and landmark slices uniformly sampled between 5 to 15 action steps for each batch. The observations are 42x42 resolution, grayscale images with only one-time channel both for goal and current state. All models used the same goal recognizer that was trained on the curiosity data. For selecting the hyper-parameters in forward regularizer, pixel-based forward consistency, and feature-based forward consistency models, we selected the best loss coefficient among $\{0.01, 0.05, 0.1\}$ that achieved the highest median completion on our validation environment which consisted of the training maps with novel textures.

[1] Formula for computing median confidence intervals: `http://www.ucl.ac.uk/ich/short-courses-events/about-stats-courses/stats-rm/Chapter_8_Content/confidence_interval_single_median`