Self-taught Robots: Autonomous and Weakly-Supervised Learning for Robotic Manipulation

Minttu Alakuijala

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École Normale Supérieure

Self-taught Robots: Autonomous and Weakly-Supervised Learning for Robotic Manipulation

Soutenue par

Minttu Alakuijala

Le 13 Décembre 2022

École doctorale n°386

Sciences Mathématiques de Paris Centre

Spécialité

Informatique

Composition du jury :

Nicolas Mansard LAAS	*Rapporteur*
Ludovic Righetti NYU	*Rapporteur*
François Chaumette Inria	*Examinateur*
Cordelia Schmid Inria	*Examinatrice*
Jean Ponce Inria	*Directeur de thèse*
Julien Mairal Inria	*Directeur de thèse*

ENS | PSL

List of Abbreviations

ASR Automatic speech recognition. 74, 75

BC Behavioral Cloning. 16, 19, 26, 28–30, 32–35, 38, 39, 42, 43

CNN Convolutional neural network. 29, 39

D4PG Distributed Distributional Deep Deterministic Policy Gradient. 32

DMPO Distributional Maximum a-posteriori Policy Optimization. 32, 33, 35, 39–41

DoF Degrees of freedom. 4, 25, 30, 31

DQN Deep Q-networks. 18

DVD Domain-agnostic Video Discriminator. 54–56

HOLD Human Offline Learned Distances. 45, 47, 48, 52, 53, 56, 57, 59, 60, 62

HOLD-C Human Offline Learned Distances – Contrastive. 51, 53, 55, 57–62

HOLD-R Human Offline Learned Distances – Regression. 51, 53, 55, 56, 58–62

IK Inverse kinematics. 7

IoU Intersection over union. 76, 77

LfD Learning from demonstration. 22, 84

MDP Markov decision process. 16, 23, 30, 50

MPO Maximum a-posteriori Policy Optimization. 32

MSE Mean squared error. 59, 60

R3M Reusable Representations for Robotic Manipulation. 20, 57, 58

RGB Red green blue. 29, 32

RL Reinforcement learning. 1–4, 9–12, 16–18, 20–22, 25–32, 38–41, 43, 47–49, 52, 54, 56, 58, 62, 83, 84

RLV Reinforcement Learning with Videos. 54–58, 60–63

RRLfD Residual Reinforcement Learning from Demonstrations. 26, 29, 38, 41

SAC Soft Actor-Critic. 54

SSv2 Something-Something v2. 53, 57, 60, 61, 63

TCN Time-contrastive Networks. 49, 51, 52, 57

TD Temporal difference. 32

ViT Vision Transformer. 53, 59, 60, 62

ViViT Video Vision Transformer. 53, 59, 60, 62

Abstract

Despite significant advances in machine learning in recent years, robotic control learned from data has yet to show large-scale impact in the real world. One of the main limitations is access to data – especially when coupled with the complexity of high-dimensional and underactuated control problems. Unlike in domains such as image classification or machine translation, explicit training examples cannot be easily sourced and annotated on the internet but data collection is bounded by real-time robot operation.

This thesis presents several ways to leverage external data sources, from task demonstrations to full-length tutorial videos, to address the challenge of slow data collection and thus accelerate learning of robotic manipulation tasks. First, we propose a method to efficiently leverage a small number of demonstrations as a starting point, and autonomously improve this initial policy through residual reinforcement learning. No reward shaping, controller engineering or state estimation is needed as the policy uses image and proprioceptive inputs as well as sparse task completion rewards only.

In our second contribution, we show that robotic agents can acquire inductive biases for manipulation by watching videos of humans using their hands and arms before ever interacting with the world themselves. We demonstrate that our reward functions, though trained exclusively on human data, are able to generalize their predictions of task progress to robot arms and accelerate training of several unseen manipulation tasks.

Finally, we propose that narrated instruction videos can not only help agents gain subtask execution skills, but also teach them which subtasks are needed to accomplish long-horizon goals and in which order, as well as how they map to natural language instructions. Specifically, we present a discriminative clustering based method leveraging the temporal alignment of the narration and visual streams for automatic subtask discovery and segmentation. Both short and long-form instructional videos are especially promising data sources as they are widely available on the internet.

In this thesis, we investigate the performance of purely learning-based algorithms for robotic manipulation, while acknowledging that optimal control as well as hybrid approaches can provide complementary solutions to some of the open challenges. Our key argument is that advances in the related fields of computer vision, signal processing, natural language processing, imitation and deep reinforcement learning can help lead the way towards more adaptive robotic agents. In manipulation domains, in particular, the variety of materials, shapes and tasks present in the real world beyond tightly controlled operating conditions poses great difficulty for fixed control strategies and the precise physical modelling required by classical model-predictive control approaches. Our overarching goal is therefore to enable more capable and versatile robotic manipu-

lation through data-driven methods. Reducing the amount of domain expertise required to train robots by emphasizing example-based learning and autonomous improvement will ultimately support more widespread adoption of adaptive robotic solutions.

Résumé

Malgré des progrès considérables réalisés ces dernières années dans l'apprentissage automatique, son utilisation dans le cadre de la commande de robots n'a pas encore eu d'impact à grande échelle. L'une des principales limitations est l'accès aux données, surtout si l'on tient compte de la complexité des problèmes de commande en haute dimension et pour des systèmes sous-actionnés. Contrairement au cas des domaines tels que la classification d'images ou la traduction automatique, il est difficile de trouver des exemples d'entraînement annotés sur Internet, et la collecte de données dans des environnements physiques est limitée par le fonctionnement du robot.

Cette thèse présente plusieurs façons d'exploiter des sources de données externes, de démonstrations de tâches aux tutoriels vidéo, pour relever le défi de la lenteur de la collecte de données et ainsi accélérer l'apprentissage des tâches de manipulation robotique. Nous proposons d'abord une méthode pour exploiter efficacement un petit nombre de démonstrations comme point de départ, et améliorer de manière automatique la politique initiale par un apprentissage par renforcement résiduel. Des étapes d'affinement de la fonction de récompense, des contrôleurs ou encore d'estimation d'état ne sont pas nécessaires car la politique utilise uniquement des entrées d'image et proprioceptives ainsi que des récompenses binaires obtenues en fonction de la réussite de la tâche désirée.

Dans notre deuxième contribution, nous montrons que les agents robotiques peuvent acquérir des biais inductifs pour la manipulation avant d'interagir eux-mêmes avec le monde physique, en regardant des vidéos de personnes utilisant leurs mains et leurs bras. Nous démontrons que nos fonctions de récompense, bien qu'entraînées exclusivement sur des données humaines, sont capables de généraliser leurs prédictions de progression de tâche aux bras robotiques et d'accélérer l'entraînement de plusieurs tâches de manipulation.

Enfin, nous proposons d'utiliser des tutoriels vidéo pour enseigner aux agents des compétences en matière d'exécution de sous-tâches, quelles sont les sous-tâches nécessaires pour accomplir une tâche à long terme, leur ordre d'exécution, et comment elles correspondent aux instructions en langage naturel. Plus précisément, nous présentons une méthode basée sur le clustering discriminant qui exploite l'alignement temporel de la narration et du flux visuel pour la découverte et la segmentation des sous-tâches. Les tutoriels vidéo, qu'ils soient courts ou longs, sont des sources de données particulièrement prometteuses car ils sont abondants sur Internet.

Dans cette thèse, nous étudions la performance des algorithmes purement basés sur l'apprentissage pour la manipulation robotique, tout en reconnaissant que la commande optimale ainsi que les approches hybrides peuvent fournir des solutions complémen-

taires à certains des défis ouverts. Notre argument central est que les avancées dans les domaines connexes de la vision par ordinateur, du traitement du signal, du traitement du langage naturel, des l'apprentissage par imitation et par renforcement profond peuvent aider à ouvrir la voie à des agents robotiques plus adaptatifs. C'est particulièrement le cas pour le domaine de la manipulation dans le monde réel, en dehors de conditions d'exploitation étroitement contrôlées. En effet, la variété des matériaux, des formes et des tâches pose de grandes difficultés pour les stratégies de contrôle fixes et les approches classiques de commande prédictive qui nécessitent une modélisation physique précise. Notre objectif principal est donc de permettre une manipulation robotique plus performante et polyvalente grâce à des méthodes appris à partir des données. Réduire l'expertise liée au domaine nécessaire pour former les robots, en mettant l'accent sur l'apprentissage à partir d'exemples et l'amélioration autonome, favorisera en fin de compte l'adoption plus large de solutions robotiques adaptatives.

Acknowledgements

I am grateful for the support of many colleagues and friends, who made this journey so enriching and memorable. First, I want to thank my advisors Cordelia Schmid, Julien Mairal and Jean Ponce for giving me the opportunity to pursue this doctorate under their guidance. They granted me a significant amount of freedom to identify and take on interesting problems, which allowed me to grow as a researcher. I am also deeply thankful to Gabriel Dulac-Arnold for lending his expertise and advising in our collaborations.

My thesis was mostly funded by a Google CIFRE PhD Fellowship. Highly appreciative of the unique opportunity to pursue a CIFRE PhD with Google, I want to thank Cordelia Schmid, Bertrand Rondepierre and the many others who worked on making this possible. My fellow CIFRE student Valentin Gabeur greatly helped with my ANRT application and initial doctoral school registration. This thesis was also in part supported by the Inria / NYU collaboration, the Louis Vuitton / ENS chair on artificial intelligence and the French government under management of Agence Nationale de la Recherche as part of the *Investissements d'avenir* program (PRAIRIE 3IA Institute).

I also wish to thank the members of my defense jury for their time, and especially Nicolas Mansard and Ludovic Righetti for accepting to review this thesis. Thanks also to Francis Bach and Nicolas Mansard for their mentorship through my comité de suivi.

Undoubtedly, the work leading up to this thesis was inspired by many stimulating discussions and feedback from several colleagues at the Ganesha and Brain Agents teams at Google, as well as the Thoth and Willow teams at Inria over the years. Special thanks to Olivier Pietquin, Jack Valmadre, Justin Carpentier and Elliot Chane-Sane for reviewing our manuscripts and to Fabian Schramm for reviewing part of this thesis. Robert Dadashi, Matthieu Geist and Damien Vincent at Brain Agents, Pete Florence at Brain Robotics, Jean-Baptiste Alayrac at DeepMind as well as Elliot Chane-Sane, Alexander Pashevich, Robin Strudel and Ricardo Garcia Pinel at Inria also contributed to insightful research discussions. During my time at Google, I had the chance to be mentored by Marco Tagliasacchi. Thanks also to Kevin Zakka, Annie Chen and Karl Schmeckpeper for taking time to answer my questions and helping me rerun their work.

Real robot experiments take up a significant amount of time to set up, so I am also very grateful for Etienne Arlaud and Ricardo Garcia Pinel's support and advice in navigating this domain that was entirely new to me.

Last but not least, I am also deeply thankful to my friends and family, especially my dear Eric, for their patience and support along the way.

Contents

1 Introduction **1**

1.1 Objective . . . 1

1.1.1 The generality of reinforcement learning . . . 1

1.1.2 The role of simulation . . . 3

1.1.3 Limitations of current robotics . . . 4

1.1.4 Towards learned control . . . 6

1.2 Motivation . . . 7

1.2.1 Applications of adaptive manipulation . . . 7

1.3 Challenges . . . 9

1.4 Contributions . . . 10

1.5 Thesis outline . . . 13

2 Related work **15**

2.1 Weakly supervised learning . . . 15

2.2 Imitation learning . . . 15

2.3 Reinforcement learning . . . 16

2.3.1 Deep RL . . . 18

2.3.2 Offline RL . . . 19

2.3.3 Large-scale methods . . . 19

2.3.4 Representation learning . . . 20

2.3.5 Self-supervised objectives . . . 20

2.4 Data-driven robotics . . . 21

2.4.1 Data reuse . . . 21

2.4.2 Reward annotation . . . 22

2.4.3 Learning from demonstration . . . 22

2.4.4 Combined learning and optimal control . . . 23

2.5 Learning from human videos . . . 23

3 Residual RL from Demonstrations **25**

3.1 Introduction . . . 25

3.2 Related work . . . 27
3.3 Residual Policy Learning from Demonstrations . . . 29
3.3.1 Base policy . . . 29
3.3.2 Residual policy . . . 30
3.4 Experiments . . . 30
3.4.1 Environments . . . 30
3.4.2 Demonstrations . . . 31
3.4.3 Observation space . . . 32
3.4.4 RL algorithm . . . 32
3.4.5 Training details . . . 32
3.4.6 Base policy . . . 33
3.4.7 Residual policy . . . 35
3.4.8 Fine-tuning baseline . . . 38
3.4.9 RL baseline . . . 39
3.4.10 Real robot experiments . . . 41
3.5 Discussion . . . 43

4 Learning Reward Functions for Robotic Manipulation by Observing Humans 45
4.1 Introduction . . . 46
4.2 Related work . . . 48
4.3 Human Offline Learned Distances . . . 50
4.3.1 Functional distances from observation-only data . . . 50
4.3.2 Policy learning . . . 52
4.4 Experimental results . . . 53
4.4.1 Distance learning . . . 53
4.4.2 Policy learning . . . 54
4.4.3 Baseline comparisons . . . 57
4.5 Limitations . . . 58
4.6 Training hyperparameters . . . 58
4.7 Distance model evaluation on human data . . . 59
4.8 Distance model ablations . . . 60
4.9 Longer training for sparse reward . . . 62
4.10 Training data coverage of evaluated robot tasks . . . 63
4.11 Conclusion . . . 64

5 Discovering Actions by Jointly Clustering Video and Narration 65
5.1 Introduction . . . 65
5.2 Related work . . . 67

5.2.1 Weakly supervised learning of actions in video 67
5.2.2 Sharing across tasks 68
5.3 Proposed model 68
5.3.1 Objective function 68
5.3.2 Separate constraints 70
5.3.3 Joint constraints 71
5.3.4 Constraint relaxation 71
5.4 Optimization 73
5.5 Experiments 74
5.5.1 Datasets 74
5.5.2 Feature representations 74
5.5.3 Further objective terms 75
5.5.4 Evaluation 75
5.5.5 Baselines 76
5.5.6 Results 77
5.6 One-to-many constraints 79
5.7 Sequence alignment 80
5.8 Computational complexity 81
5.9 Rounding scheme 82
5.10 Conclusion 82

6 Conclusion **83**
6.1 Summary of contributions 83
6.2 Further work 84
6.2.1 Physical priors 84
6.2.2 Predicting affordances 85
6.2.3 Long-horizon and hierarchical models 85
6.2.4 Language conditioning 85

Bibliography **87**

Chapter 1

Introduction

1.1 Objective

GENERAL-PURPOSE ROBOTIC AGENTS face tremendous obstacles when navigating the physical world. Capabilities requiring little thought in animals and humans, such as locomotion and fine motor skills, have proven exceedingly difficult for artificial agents to recreate [Moravec, 1988]. Although machines have reached superhuman performance in many tasks requiring symbolic *System 2* [Kahneman, 2011] thinking, such as arithmetic and strategy games, many behaviors requiring intuitive *System 1* thinking that are effortless for humans remain largely open problems.

However, advances in machine learning, largely driven by deep neural networks in the last decade [Krizhevsky et al., 2012; He et al., 2016; Vaswani et al., 2017], have begun to address this class of fast, approximate System 1 tasks. Considerable progress has been made in methods that classify, detect, segment and reproduce patterns in incoming data streams. Many of these tasks naturally arise as subproblems in robotic systems, and the success of deep learning has inspired an abundance of work in robot learning. Yet, the majority of machine learning methods have exclusively leveraged supervised and self-supervised objectives. One of the central arguments of this thesis is that any autonomous agent expected to perform general-purpose problem solving in novel situations will also benefit from the ability to learn from scalar-valued feedback obtained from task completion signals or from humans – the problem addressed by reinforcement learning (RL).

1.1.1 The generality of reinforcement learning

Many types of sequential decision problems can be formulated as RL problems with few assumptions. Learning from scalar reward signals obtained through interaction presents several advantages over learning solely from static datasets with fixed prediction targets:

1) By defining a reward (or cost) function instead of labels, the learning agent can be tasked with problems for which no solution or only a suboptimal solution is known.

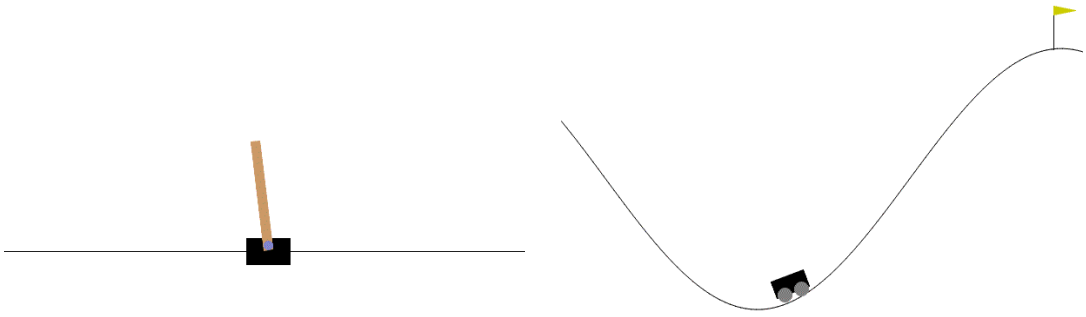
2) Continuing to learn from interaction naturally handles distribution shift between an initial training period and later testing times as well as personalization to an end user or application.

3) Supervised learning tasks often involve state estimation type problems but do not always include the actual decision to take based on a predicted quantity; the decision strategy is in many cases hardcoded on top of estimated features. By taking ultimate task success into account, the full observation-action loop can be closed, supporting end-to-end learning.

4) No assumptions are made of the environment, such as access to dynamics, gradients, or the full cost function. The cost function is also not assumed to be deterministic.

A drawback of the generality and wide applicability of RL is that learning is notoriously slow. Most successes have been achieved in simulation, after millions of interaction steps [Mnih et al., 2015; Silver et al., 2016; Andrychowicz et al., 2020; Siekmann et al., 2021]. The absence of assumptions regarding policy structure, objective function shape or environment dynamics does result in an extended training time, especially for deep learning based approaches (i.e., deep RL). For these methods to have impact in the real world beyond easily simulated tasks, including in robotics, we therefore argue it is crucial to move beyond the *tabula rasa* setting by incorporating priors in the form of demonstrations, other external data, auxiliary objectives, representation learning, or learned rewards.

While RL allows robotic agents to flexibly discover new behaviors, problems in robotics in turn serve as a testing ground for advances in learning algorithms, imposing real-world constraints [Kober et al., 2013]. Robotic control neatly fits the sequential decision problem setting of RL. As a result, several of the earliest and most standard benchmark tasks in RL have been inspired by robotics, such as Cart-pole [Barto et al., 1983] (Fig. 1.1a), where a cart on a one-dimensional track must balance an upright pole on top of it, or Mountain car [Moore, 1990] (Fig. 1.1b), where an under-powered car must gather momentum in order to escape a steep valley. When interfacing with the real world, RL approaches must pay particular attention to sample efficiency, safety of exploration and limited use of scene resets requiring manual intervention [Dulac-Arnold et al., 2021]. Especially in tasks requiring high-frequency closed-loop control,



(a) Cart-pole (b) Mountain car

Figure 1.1: Classical control tasks in RL, from OpenAI Gym [Brockman et al., 2016].

algorithms also need to predict actions in real time and tolerate delays in sensor observations and actuation. Hardly any of these issues arise in simulated or virtual tasks.

1.1.2 The role of simulation

For the majority of real-world phenomena, no accurate simulators exist. Careful modeling of a changing process and the dependencies between its observed and unobserved components as well as external factors is difficult and time-consuming, and many dynamical systems (in human behavior, biology, markets, climate, and traffic, to name a few) are simply too complex to be well understood without prohibitive cost and time investment. Yet, perhaps RL's characteristic reliance on simulation is not an issue in robotics – after all, several aspects of physical simulation of rigid bodies *are* relatively well-understood, and most classical control methods are also first developed, tuned and tested in simulation. When such physical models are available, data collection can be scaled up by running several simulation instances in parallel. Although very data-hungry algorithms will naturally impose some computational cost in dataset creation, simulated data is still significantly less costly than robot time.

However, simulations are only useful to the extent that their observation space and dynamics truly agree with the environment of interest. In order to enable a control policy trained in simulation to work on a real robot, not only must the simulator be carefully designed to align with its real counterpart as closely as possible, but the inevitably remaining differences must also be addressed in the sim2real transfer process. Controllers must deal with the *physical gap* in dynamics between simulation and the real world; methods using camera inputs must additionally be tolerant to a *visual gap* between the systems. Domain randomization [Tobin et al., 2017] is a popular sim2real method in which random variations of the physical parameters (such as mass, size and friction coefficients) as well as visual appearance (color, texture and lighting) are intro-

duced at training time to ensure the resulting policy is robust to such differences.

Nonetheless, significant time and effort goes into developing simulators as well as precisely tuning them to match real world conditions, and the success of policy training, even with sim2real adaptation, strongly relies on fidelity to the real system. Unfortunately, progress towards closing the physical gap between a specific simulation and the corresponding real system only supports better sim2real transfer in that problem setting – improving policies for the task, environment and robot embodiment being modeled – while the process must be repeated nearly all over again for each new setup.

At the same time, several recent approaches have managed to train RL policies directly on real robots in an hour or less by leveraging auxiliary representation learning objectives [Zhan et al., 2020], or advances in model-free RL [Smith et al., 2022] and model-based RL [Wu et al., 2022]. These works demonstrate that with the right algorithmic advances and inductive biases, deep RL approaches need not be limited to training in simulation, and will likely continue making inroads into other unsimulable domains. In the meantime, particularly for development and benchmarking, it is sensible to favour simulated tasks that can be parallelized and easily repeated with identical environmental conditions. In this thesis, we therefore mainly focus on simulated benchmarks, but also present a preliminary simulation-free experiment in which we train residual RL to push objects on a table directly with a UR5 arm in Chapter 3.

1.1.3 Limitations of current robotics

An alternative to data-driven robotics is to reason explicitly about the dynamics of a system and encode these assumptions into a physical model, the approach taken by classical control methods. However, these systems are necessarily limited by how accurately the real world can be modeled. The precise physical properties of tasks involving contacts, friction, as well as deformable objects and substances are often hard to measure and several of the underlying physical principles remain active areas of research, despite the fact that humans navigate these types of interactions with ease on a day-to-day basis.

Today, robots mostly provide value in heavily constrained settings (Fig. 1.2). In applications such as manufacturing and warehousing, entire factories must be designed around robotic controllers that rely on highly structured environments with minimal variation. Most optimal control methods assume perfect knowledge of low-dimensional state such as object and target positions, with their performance degrading under state estimation drift or calibration errors. In many cases, precise state estimation also requires specialized hardware. For now, large scale operations may have been able to afford, install and maintain equipment such as 6-DoF motion capture in all spaces where

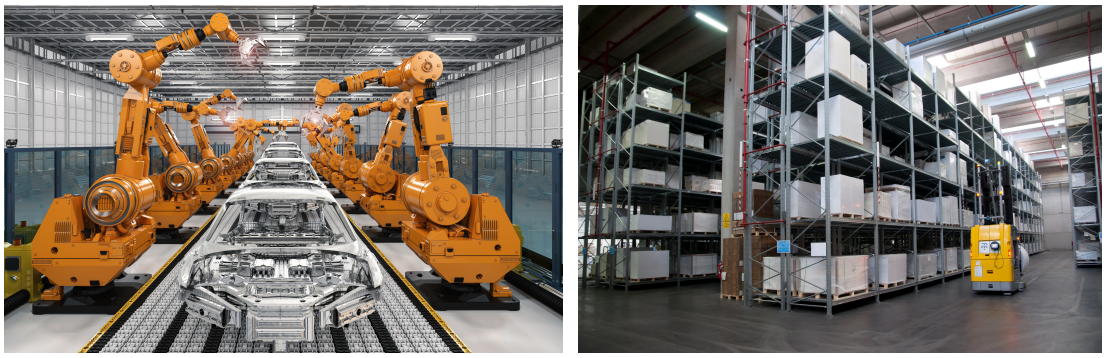


Figure 1.2: Current robotics applications work almost exclusively in highly structured environments such as production lines and warehouses.

robots operate, but in order to move towards environments designed for humans – such as streets, homes, offices, hospitals and retail spaces – we must relax the assumption of perfect state estimation and design approaches replacing predetermined models of the environment with adaptation to incoming data and state estimation from on-board sensors.

In addition to precisely controlled operating conditions, the robotic behaviors themselves are typically hand-engineered. While the most accurate manipulators, such as those found in automotive manufacturing, are capable of repeating hardcoded motions with sub-millimetre accuracy, they are also expensive and heavy, and hence risky to nearby operators, due to their sturdiness and high-power motors. The parameters of each controller on an assembly line have typically been carefully tuned by hand to fit a specific purpose, with precise start and goal configurations. As a result, the assembly steps must remain unchanged. For most types of manual work in everyday life, there is significant variation in incoming materials and situations, which is why automation has yet to show promise in tasks such as sewing, cleaning, cooking, and carpentry, and why there continues to be huge demand for manual labor. Collaborative robots working together with people may be better equipped to handle some of these highly variable tasks compared to robotic agents alone. However, as evidenced by demonstrations of perfectly capable completion of several domestic tasks by *teleoperated* robot arms [Wyrobek et al., 2008], current hardware allows for a much wider variety of manipulation tasks to be solved, and the bottleneck lies in the software.

Another downside of classical control is specialization to a particular problem setting. Work towards designing controllers or modeling physical systems in a particular sector, such as manufacturing, is unlikely to transfer to another, such as healthcare. General algorithms for learning from demonstrations, reward-annotated offline data or online exploration where it can be safely performed, however, have been shown to work in a diverse array of application domains, including playing ping-pong in a video game

[Mnih et al., 2015], quadruped walking [Haarnoja et al., 2018b], bipedal stair climbing [Siekmann et al., 2021], controlling a data center cooling system [Lazic et al., 2018] or magnetic actuator coils in nuclear fusion reactor [Degrave et al., 2022].

1.1.4 Towards learned control

In order for robots to handle real-world variations, they should be able to learn from experience. Optimal control methods must first address the subproblem of estimating low-dimensional input states from available sensor inputs. However, directly incorporating raw sensor data, including depth or RGB camera observations, in the control policy itself presents several advantages. Unlike hand-designed compact representations, using visual inputs enables robustness to miscalibration, drift in proprioceptive state estimation and omissions of relevant state information at design time. Moreover, optimal control usually requires breaking down the desired behaviors into discrete types, such as using separate manually specified controllers for standing up, walking, running, and recovering from perturbation in the case locomotion. Reasoning about and enumerating the full set of possible behavior modalities or environment conditions a-priori is nearly impossible, especially for complex problem domains. When faced with real-world diversity, particularly when additionally having to navigate around unexpected obstacles or other actors, hand-engineered controllers become brittle and often insufficient.

Learning-based methods are also inherently well-suited for low-cost robots, as they do not assume perfect repeatability and deterministic environments. Many learning approaches intentionally apply a small amount of noise to the selected actions at training time to be robust to small variations, and where available, using real data allows policies to directly encounter and adapt to any randomness that may be present in real-world dynamics. Being highly precise and repeatable requires expensive and rigid joints, typically resulting in heavy robots with high power consumption. As part of a shift towards more learning from experience and less controller engineering, more errors in state estimation, calibration, robot configuration parameters and motor repeatability could be tolerated as they could be corrected for at the algorithmic level based on visual inputs rather than estimated intermediate state features. Such manipulators could be designed to be cheaper, more energy-efficient, lighter and therefore safer for human operators.

In this thesis, we present three contributions towards improving general-purpose robotic manipulation by using scalable and readily available data sources. We investigate scripted and teleoperated demonstrations collected in the robot's observation and action space (Chapter 3), short video clips of human demonstrations (Chapter 4), and finally, full-length instruction videos (Chapter 5). As is the case in many related domains

such as computer vision and natural language processing, we argue that advances in data-driven and deep learning approaches may unlock tremendous potential for learning adaptive behavior in robotics. Given sufficient data and compute, generic and scalable models have been shown to outperform hand-crafted solutions time and time again [Sutton, 2019].

Nonetheless, our aim is of course not to entirely replace classical methods in robotics with learning-based approaches, as they are complementary in their strengths and weaknesses. While end-to-end learning can flexibly adapt to unseen situations and correct for inaccuracies in state estimation, first-principles physical modelling and feedback control allow parts of the problem which are well understood to be efficiently solved. Assuming modelling is accurate, incorporating it into hybrid models would likely reduce training data requirements. Furthermore, optimal control is well suited to imposing constraints for robot behavior. One way of combining both families of approaches could be to use classical controllers in a subroutine of an RL policy, which might in turn take actions at a lower frequency to handle longer-term goal setting. In a similar vein, we make use of fixed inverse kinematics (IK) submodules with knowledge of the physical model of the robot, and simplify the learning problem to controlling the end-effector position in task space (with IK handling the conversion to joint parameters) in the methods presented in Chapters 3 and 4. However, even in seemingly simple control problems, there may be many hard-to-model external and internal perturbations such as air resistance, wind, gear backlash and hardware wear and tear.

While this thesis is in large part motivated by robotic manipulation, RL, learning from demonstration and weakly-supervised learning from video – including methods similar to those we present in the following chapters – could equally be applied to learn from sparse rewards in sequential decision making tasks in many other domains, such as website navigation, trading, information retrieval, conversational agents, as well as robotic locomotion and navigation tasks, too.

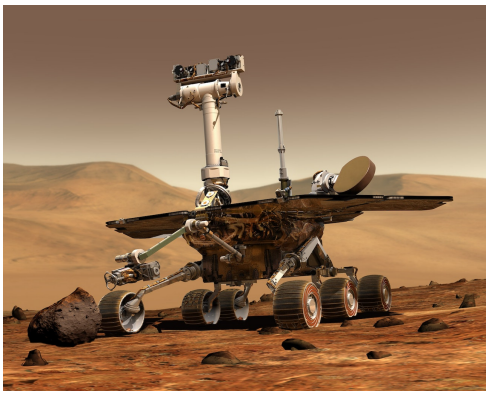
1.2 Motivation

1.2.1 Applications of adaptive manipulation

There remains a significant amount of unrealized potential for increased productivity and economic growth from deploying robotics applications. In this section, we outline just some of the many domains to which general-purpose manipulation skills could be applied.

(a) Many cooking tasks require fine manipulation.

(b) Wheeled delivery robots are unable to get past doors or gates.



(c) Rovers deployed in space must adapt to and learn from unknown environments.

Lot-size-one manufacturing. While manufacturing is one of the oldest domains to be automated, current production lines require significant upfront investment per category of item and hence require demand for vast quantities of one object for production to be economically viable. Factory lines are typically highly specialized and allow for little variation in the end product. On the other hand, 3D printing is dramatically increasing access to individually customizable items such as personalized medical devices. Being able to send a robot manufacturing and assembly instructions on-demand would greatly diversify the types of custom items that could be cheaply produced.

Logistics and retail. Warehousing, packaging, and last mile delivery systems are among the most well-known large-scale applications of robots. Prototype delivery robots are currently being tested out, but typically move on wheels and have limited ability to traverse obstacles and navigate the challenges of built environments, such as stairs, gates and doors (Fig. 1.3b). Early bipedal models are in development.

Hazardous environments. Ideally, autonomous and adaptive robots could allow people to limit their exposure to situations posing serious health or safety risks, in domains such as search-and-rescue following disasters, mining, construction, as well as exploration in extreme conditions such as the ocean floor or outer space (Fig. 1.3c). These agents would need to navigate diverse terrains, but comprehensive manipulation skills would also greatly extend their capabilities. Waste management and sorting can also pose occupational hazards through exposure to dangerous substances. Much of waste management around the world is still done manually.

Healthcare and prosthetics. Artificial limbs, exoskeletons and other assistive technologies have potential to greatly improve quality of life for people with impaired mo-

bility. Robot arms are also routinely used in surgery [Kasina et al., 2017], though usually teleoperated by a surgeon. Increasing autonomy for routine procedures could further increase the availability of highly skilled labor.

Laboratory tasks. Robotic manipulation is also being applied to the natural sciences in the form of scaling up laboratory work. The scope of experiments that can be considered could be vastly increased and the pace of discovery accelerated using robots. Laboratories are typically much more structured than most human environments, making robotic applications feasible in the short to medium term.

Domestic tasks. While robotic vacuum cleaners have been available to consumers for two decades [Jones, 2006], and robotic lawn mowers and pool cleaners for roughly three decades [Hicks II and Hall, 2000; Prassler et al., 2000], robots for other domestic tasks have lagged behind. Although prototype arms have been proposed for cooking (Fig. 1.3a), current systems mainly cater to a desire for novelty rather than practical use cases, and no solution has yet offered sufficient generality at a feasible price point. Due to the highly unconstrained environment and rigorous safety requirements of working in immediate proximity with humans, these applications are some of the hardest to tackle. In the medium term, robotized kitchens at the restaurant or factory scale could serve as a stepping stone in this direction.

1.3 Challenges

Safety. An immediate concern in any robotic deployment is the safety of surrounding people (and property). For learning-based methods, this is also a concern during training, which may require exploration in the form of trial and error. While in some applications this issue may be side-stepped by imposing distance or physical barriers between humans and robots, in others, such as household and healthcare applications it must be addressed before any substantial progress can be made. In this thesis, we mainly consider settings falling under the former category; in particular, fixed-base manipulation arms.

Reward specification. The reward function is a critical component to define correctly in order for RL policies to learn what was intended. Finding the right definition often involves careful balancing of several weighted components with additional tunable hyperparameters. Moreover, these reward terms typically depend on full state information such as target positions, which may not be accessible but must be estimated from readily available signals, such as camera images, in an intermediate step. In this thesis, we

remove the need for manually shaped rewards by inferring them from human data using inverse RL (Chapter 4), or by developing sufficiently robust and data-efficient methods to learn from only sparse rewards (Chapters 3 and 4).

Data reuse. Data collected on a real robot is always subject to the constraint of real-time execution. Moreover, creating large datasets by sharing data across tasks or across robots is not straightforward. In Chapter 4, we propose a method able to learn from mixed-task data from different actors such as other robots, or humans.

Scene resets. While a non-issue in simulation, when training RL on a real robot, re-setting the scene to an initial configuration may impose a significant amount of manual work in an otherwise autonomous learning process. While not the focus of this thesis, potential solutions have been proposed to alleviate this issue [Gupta et al., 2021].

1.4 Contributions

This thesis is based on the following publications:

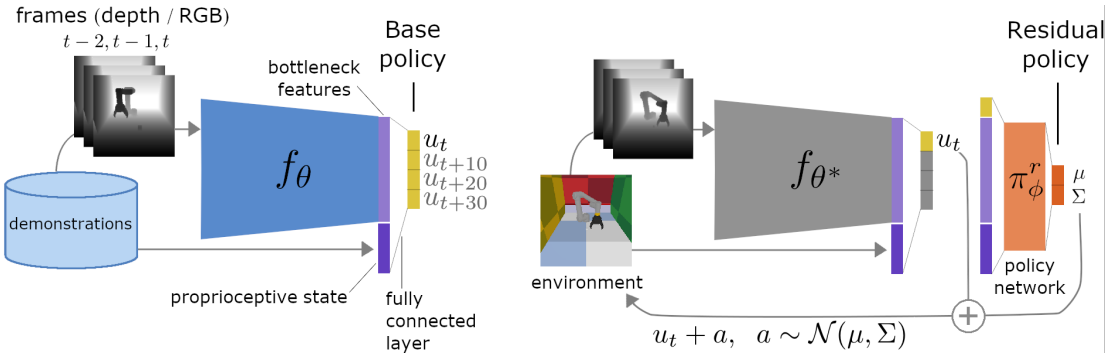
- **Minttu Alakuijala**, Gabriel Dulac-Arnold, Julien Mairal, Jean Ponce, Cordelia Schmid. *Residual Reinforcement Learning from Demonstrations*.
  RSS 2020 Workshop on Advances & Challenges in Imitation Learning for Robotics. [Alakuijala et al., 2020a]
  A longer version is in preparation for a future journal submission.

- **Minttu Alakuijala**, Gabriel Dulac-Arnold, Julien Mairal, Jean Ponce, Cordelia Schmid. *Learning Reward Functions for Robotic Manipulation by Observing Humans*. 2022, Under review. [Alakuijala et al., 2022]

- **Minttu Alakuijala**, Julien Mairal, Jean Ponce, Cordelia Schmid.
  Discovering Actions by Jointly Clustering Video and Narration Streams Across Tasks.
  CVPR 2020 Workshop on Learning from Instructional Videos.
  [Alakuijala et al., 2020b]

In the first contribution, Residual Reinforcement Learning from Demonstrations [Alakuijala et al., 2020a], we extend work in residual RL by incorporating data-driven learning for defining base controllers. Prior residual RL approaches propose to superimpose a hand-designed but improperly tuned classical feedback controller with an RL policy, in order to autonomously learn to correct the base controller's actions by maximizing task reward. Instead of depending on an existing controller being available, our

method additively adapts a base policy learned from demonstrations. The initial imitation learning policy may perform suboptimally, especially with limited training data, for regions of the state space not included in demonstrations and under any changes between data collection and testing conditions. We demonstrate consistent performance improvement after residual training starting from base policies learned from just a handful of demonstrations, which can be easily provided by non-experts. A major advantage of our approach is that it does not require access to full state information, such as the positions of objects and targets, which are typically assumed to be available in optimal control. To accurately estimate these unknown state features from sensor data, input pipelines handling this challenging subproblem must therefore also be designed, adding further engineering effort. Our method is instead entirely learned from data; yet, thanks to priors in the form of demonstrations, the residual policy structure as well as reuse of visual features learned using the imitation objective, we show our method to be sufficiently sample-efficient to be able to learn from only binary task-completion rewards (Fig, 1.4). This is in contrast to many RL methods that must define dense rewards a-priori, which often include several precisely weighted components and impose non-negligible tuning effort.



(a) (b)

Figure 1.4: a) We propose a way to leverage demonstration data to learn a control policy as well as task-relevant visual features through behavioral cloning on image and proprioceptive inputs. b) The policy is then improved through reinforcement learning by a superimposed residual policy, based on the learned visual features, allowing data-efficient learning of control policies in image space from sparse rewards.

Our second contribution, Human Offline Learned Distances [Alakuijala et al., 2022], offers a different perspective on reward specification by adopting an inverse RL approach to learning from human videos. The objective in inverse RL is to infer a reward function that best explains the behavior of an expert, in order to use the learned reward

in subsequent RL policy training. The expert data is usually assumed to come from the action and observation spaces of the target policy, a requirement which we propose to relax by learning from third-person observations of humans, instead. The specific goals of our method are a) to learn priors for robotic manipulation by observing humans performing various object interaction tasks, a cheap and easily scalable data source, and b) to train a single, task-agnostic reward function conditioned on a goal image that can be reused for several robot tasks. Specifically, we train functional distance models to predict how far apart two image observations occur in unlabeled human video demonstrations (Fig. 1.5), either by directly regressing the time remaining to reach a goal state or in the form of distances in an embedding space learned using a time-contrastive objective. Through goal conditioning, our model can be trained even on ungrouped human data without task labels. We propose to use the learned model as a reward function for RL and demonstrate accelerated training in several sparse reward manipulation tasks.

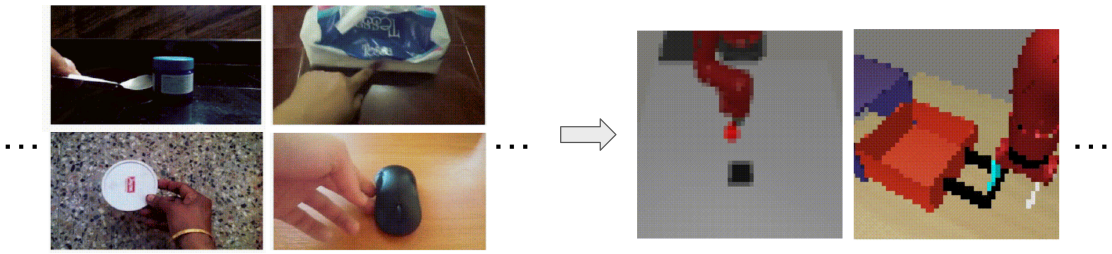


Figure 1.5: Using short video clips of humans performing various manipulation tasks, we propose to learn a model of remaining transition time from the current state to a goal image. Thanks to highly diverse training data from the Something-Something dataset [Goyal et al., 2017] (visualized on the left), the learned functions are able to generalize to observations of robots in various simulated domains (examples shown on the right), despite never having seen a robot arm in training.

Finally, our third contribution Discovering Actions by Jointly Clustering Video and Narration Streams Across Tasks [Alakuijala et al., 2020b] addresses the challenging problem of full-length video tutorial understanding. Beyond the short crowd-sourced demonstrations considered in the second contribution, we now wish to learn from in-the-wild instruction videos, which may be untrimmed, and often include irrelevant digressions, scene transitions, changes in lighting, camera motion, footage of the narrator speaking instead of visually demonstrating a task, as well as unstructured audio narration. Using tutorial videos from YouTube and their corresponding transcripts obtained with automatic speech recognition, we propose a discriminative clustering based method detecting actions (i.e., subtasks) jointly occurring in the narration and visual streams. Unlike most prior work, we do not assume task labels are available for the

videos but propose to learn action classes across tasks (Fig. 1.6). By leveraging only the relative timing of narration and video frames as well as pretrained visual and language representations, our method discovers key steps making up each instructional video without any subtask labels. The clustering objective is augmented with a sequence alignment penalty, encouraging the sequence of actions assigned to the narration to match the sequence assigned to frames, with approximate temporal alignment. Our method produces a full temporal segmentation of mixed-task demonstrations into one of k subtasks or background. The predicted subtask decompositions and extracted demonstrations could be used as training data for robots, for learning both individual skills as well as longer-horizon task planning using visual inputs, language descriptions, or both. Indeed, evaluating the quality of the subtask predictions within the context of robot learning is an interesting direction for further work.

Figure 1.6: Using only weak supervision from narration, we automatically discover actions that might occur across different tasks and contexts—without assuming the task depicted, such as the recipe label, is known.

1.5 Thesis outline

This thesis is organized into six chapters, including this introduction. In Chapter 2, we review the existing literature on data-driven robotics and weakly-supervised learning. In Chapter 3, we present our first contribution, a reinforcement learning method combining demonstrations with online trial-and-error learning [Alakuijala et al., 2020a]. Our second contribution, a task-agnostic reward function for robotic manipulation learned exclusively from human videos [Alakuijala et al., 2022], is introduced in Chapter 4.

Chapter 5 presents our third contribution, a weakly-supervised method for action discovery and segmentation in long-form narrated instruction videos [Alakuijala et al., 2020b]. Finally, conclusions and further work are discussed in Chapter 6.

Chapter 2

Related work

LEARNING-BASED ROBOTIC CONTROL dates back to at least the 1980s [Pomerleau, 1988], and optimal control has an even longer history. This chapter introduces the reader to the technical concepts used throughout this thesis and includes a literature review of recent related work.

2.1 Weakly supervised learning

In supervised learning, the most widely used type of machine learning, the objective is to learn a function $f : \mathbb{R}^m \rightarrow \mathbb{R}^d$ mapping each example in a training dataset $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ to its corresponding label in $\boldsymbol{y} \in \mathbb{R}^{n \times d}$. The setting where $\boldsymbol{y}$ is only partially available is referred to as weakly supervised. Such cases often arise in prediction tasks where a full coverage of accurate labels is costly or even impossible to collect, and learning methods need to be adapted to handle cheaper forms of supervision. This may mean that $\boldsymbol{y}$ may be entirely or partially missing for some examples, or that each label y_i may be associated to a group of examples $\boldsymbol{X}' \subseteq \boldsymbol{X}$, only one of which is assumed to truly match the label (or possibly several, depending on the setting). In some cases, the labels that are available are not the prediction targets of interest but some related quantities or attributes. In computer vision, classical examples include labels collected at the image or video level, without fully defined temporal (interval start and end) or spatial (bounding boxes, segmentation masks) extent.

2.2 Imitation learning

The objective in imitation learning is to reproduce the behavior of an expert demonstrator. Given a dataset of demonstrations D consisting of state-action pairs (s, a), where $s \in \mathcal{S}$, the set of environment states, and $a \in \mathcal{A}$, the set of possible actions, the aim is

to map state observations to actions using a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$. In behavioral cloning (BC) [Pomerleau, 1988], one of the earliest and most standard imitation learning methods, the policy is trained to simply predict the expert actions under the distribution of the data:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \underset{(s,a)\sim D}{\mathbb{E}} ||a - \pi_\theta(s)||_2^2, \tag{2.1}$$

with the policy π parameterized by θ, such as the weights of a neural network. This training objective is identical to supervised learning as introduced in Section 2.1; however, at test time the policy is evaluated for task success rather than its precision in matching expert actions, and in particular, for episodes from the distribution visited by π rather than D. As a result, BC suffers from the problem of compounding errors: as small prediction errors lead the policy to steer away from states that were visited by the expert and hence seen at training time, the quality of the predictions decreases further, leading to yet further compounding.

Dataset Aggregation (DAgger) was proposed by Ross et al. [2011] to alleviate this issue by allowing the expert demonstrator to be queried for newly encountered states, in an interactive loop alternating BC training and data collection with the learned policy. However, DAgger comes at an increased cost in both environment interaction and labeling effort. Several subsequent imitation learning methods have addressed the challenge of compounding errors even in the fixed dataset setting without further access to the expert, typically by matching the full distribution of expert actions rather than only their expectation [Ho and Ermon, 2016; Dadashi et al., 2021; Florence et al., 2021]. Alternatively, Laskey et al. [2017] propose to expand the state coverage of demonstrations by adding noise to the demonstrator actions at data collection time.

2.3 Reinforcement learning

Similar to imitation learning, in reinforcement learning (RL) [Sutton and Barto, 2018] we are also interested in learning a policy π mapping observations (or states) to actions. However, the goal in RL is to take actions in the environment so as to maximize the cumulative output of a scalar-valued reward function over time (Figure 2.1). Unlike in the case of supervised or weakly supervised learning, correct prediction targets are never provided but the agent must learn from just scalar feedback, which may be incomplete or delayed. The environment is typically described using the Markov decision process (MDP) formalism. An MDP consists of a 5-tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ is the set of environment states, $\mathcal{A}$ is the set of possible actions, $p : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the transition function describing forward dynamics, $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, and $\gamma \in [0, 1]$ is a discount factor representing a trade-off between imme-

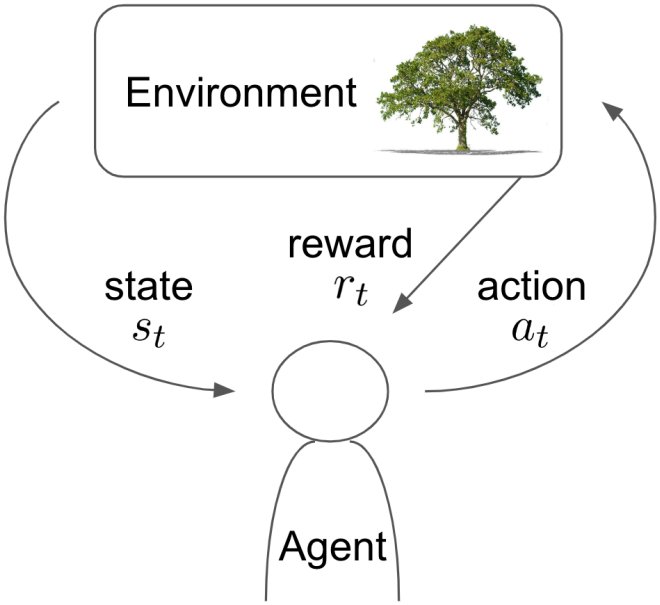


Figure 2.1: The reinforcement learning problem setting. An agent is faced with a sequential decision making problem: at each time step t, it observes the state of the environment s_t and must decide on an action a_t. In response, the environment produces a scalar reward r_t and transitions to a new state s_{t+1}. The goal of the learning agent is to choose actions in order maximize not just the immediate reward r_t, but the cumulative reward from the current time step onward.

diate rewards and long-term cumulative reward. The environment is assumed to be Markovian, meaning that knowing the current state is sufficient to predict future states: $p(s_{t+1}|s_{1:t}, a_{1:t}) = p(s_{t+1}|s_t, a_t) \; \forall \, s_t, s_{t+1} \in \mathcal{S}, a_t \in \mathcal{A}$. The objective for learning π_θ, at time t, is to choose a_t given s_t so as to maximize expected *discounted return*

$$\mathbb{E}[G_t^\pi] := \mathbb{E}\Big[\sum_{k=t}^{T} \gamma^{k-t} r_k \;\Big|\; r_k \sim r(s_k, a_k),\; s_{k+1} \sim p(s_k, a_k)\Big], \tag{2.2}$$

where T is the episode length, which may be infinite in the non-episodic case (for $\gamma < 1$). RL algorithms vary in how actions $a_{k>t}$ are sampled to evaluate this expectation.

In RL problems, p, r, or both may be stochastic and be unknown a priori, in which case they must be estimated from samples only (also called black-box or zeroth order optimization). Specifically, the objective is maximized on batches of incoming interaction data $(s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, ...)$ (with a full episode often called a trajectory) rather than analytical gradients of p and r, which are not assumed to be available. Algorithms using only newly gathered data from the current π_θ are known as on-policy methods, whereas those able to learn from data gathered using a different policy, such as an earlier iteration of π_θ, are referred to as off-policy methods.

In model-based RL, p (and often, r) is estimated explicitly, whereas the objective in model-free RL is to directly maximize Eq. 2.2 without tackling these intermediate prediction problems in the process. Model-free algorithms are further divided into

a) value-based methods, which estimate value functions:
typically, state value functions
$V^\pi(s_t) \coloneqq \mathbb{E}[r_t + \gamma V^\pi(s_{t+1}) \,\big|\, r_t \sim r(s_t, a_t), a_t \sim \pi_\theta(s_t), s_{t+1} \sim p(s_t, a_t)]$,
or state-action value functions
$Q^\pi(s_t, a_t) \coloneqq \mathbb{E}\Big[r_t + \gamma V^\pi(s_{t+1}) \Big| r_t \sim r(s_t, a_t),\, s_{t+1} \sim p(s_t, a_t)\Big]$ [Watkins, 1989],

b) policy-based methods, which directly optimize the parameters of π_θ to maximize the policy objective on incoming data, and

c) actor-critic methods, which combine both.

RL makes few assumptions about the environment beyond the Markov property. Even this requirement can, in fact, often be side-stepped by changing the state representation. If s_t is not rich enough to capture the entire state of a system, further state variables can be added to it, such as any previous states $s_{t'}$, for $t' < t$. Using a short history of stacked image observations is common practice in image-based RL methods [Mnih et al., 2015].

Thanks to the generality of the problem formulation and its few assumptions, very similar algorithms with few changes have been successfully applied in domains as diverse as video games [Mnih et al., 2015], locomotion [Haarnoja et al., 2018b], data center cooling [Lazic et al., 2018] and nuclear fusion [Degrave et al., 2022]. In this thesis, we focus on applications to robotic manipulation.

2.3.1 Deep RL

Deep neural networks may be used in RL to represent the policy and value functions, and additionally to estimate environment dynamics and the reward function in model-based RL. Deep Q-networks (DQN) [Mnih et al., 2015] and Deep Deterministic Policy Gradients (DDPG) [Lillicrap et al., 2016] were the earliest methods to combine RL and deep learning, for discrete and continuous action spaces, respectively. Several algorithmic advances have been proposed since. A popular line of work has been to augment the policy objective from Eq. 2.2 with terms encouraging the updated policy to stay close to the previous iterate for improved learning stability [Schulman et al., 2015, 2017], or with terms encouraging exploration (e.g., the maximum entropy RL framework) [Haarnoja et al., 2018a]. Rather than predicting the expected discounted return, several methods have also proposed to estimate the full distribution of G_t^π [Bellemare et al., 2017; Dabney et al., 2018a,b; Barth-Maron et al., 2018].

2.3.2 Offline RL

Standard RL requires online interaction with the environment to improve π, which may be problematic on real systems, including robots but also user-facing systems such as recommendation engines and even treatment plans in medicine. Offline RL [Kumar et al., 2019; Fujimoto et al., 2019], also called batch RL, instead aims to learn a policy and / or value function from a fixed dataset of trajectories consisting of (s_t, a_t, r_t, s_{t+1}) tuples per time step. Naively applying off-policy RL algorithms to a static dataset suffers from value estimation errors for actions that were not included in the dataset. While online Q-learning also suffers from overestimation bias under conditions of the *deadly triad* [Sutton and Barto, 2018] of bootstrapping, experience replay and function approximation, the problem is aggravated in offline RL where no new data is observed that could correct the bias. To address this problem, offline RL approaches typically aim to balance maximizing reward while staying close to the dataset collection policy [Fujimoto et al., 2019; Kumar et al., 2019, 2020]. Approaches include regressing over the demonstrated actions (as in BC) while weighting samples according to a learned value function [Wang et al., 2020; Peng et al., 2019; Nair et al., 2020]. Nair et al. [2020] study the setting where offline data is available but training can be continued online.

2.3.3 Large-scale methods

While access to data is a bottleneck in both online and offline RL, in applications where abundant data collection is cheap and parallelizable, such as in simulation or video games, the full representational capacity of deep neural networks can be leveraged. This has lead to an active line of work on asynchronous and distributed algorithms [Mnih et al., 2016; Horgan et al., 2018] able to merge environment trajectories [Horgan et al., 2018] or policy parameter updates [Mnih et al., 2016] across copies of actor-environment interaction threads running in parallel. As network architectures have grown to contain as many as hundreds of billions of parameters with attention-based architectures such as Transformers [Vaswani et al., 2017], the amount of training data required has drastically increased. Notable applications of transformers in RL include Decision Transformers, [Chen et al., 2021b], Trajectory Transformers [Janner et al., 2021] and Gato [Reed et al., 2022]. Reed et al. [2022] train a single policy network on over 600 tasks with various embodiments and action spaces, including Atari games, language generation, and block stacking with a real robot arm. We discuss large-scale learning in the context of robotics in Section 2.4.

2.3.4 Representation learning

Several approaches for learning task-relevant representations from high-dimensional inputs have been proposed. They typically involve self-supervised tasks with prediction targets that are readily available or can be automatically sourced. Various supervised objectives can then be leveraged to capture relevant state information from raw observations. This greatly reduces the difficulty of the RL problem compared to learning representations as purely a side effect of mapping inputs to actions given only scalar feedback. Jaderberg et al. [2017] observed that predicting state features that were known but not made visible to the policy, such as estimating depth from RGB as an auxiliary task, improved policy performance more than simply giving the agent access to the depth data directly. Pathak et al. [2017] propose a self-supervised objective for learning state representations ϕ such that the action a_t should be recoverable given $(\phi(s_t), \phi(s_{t+1}))$. Contrastive objectives have been widely embraced for self-supervised pretraining in other domains and have similarly been applied to RL, by using image observations with different crops and augmentations [Laskin et al., 2020], or different views of simultaneous video data [Sermanet et al., 2018] as positive pairs.

State representations are a natural place to incorporate pretraining on external data. Efforts towards learning universal robot representations from external datasets include Reusable Representations for Robotic Manipulation (R3M) [Nair et al., 2022], which could be learned once on a large amount of data and reused for several downstream tasks.

2.3.5 Self-supervised objectives

In the absence of supervision in the form of rewards, there are still several ways to acquire skills in the environment using RL objectives. An agent can define its own objective, such as diversity, coverage or goal reaching. A prominent example is goal-conditioned RL [Schaul et al., 2015; Andrychowicz et al., 2017], where the task is to transition to a goal state, which can be any state in $\mathcal{S}$. Alternatively, the objective could be defined as learning skills (i.e., low-level policies) with maximally different state visitation distributions [Eysenbach et al., 2019]. Pure exploration strategies such as Random Network Distillation [Burda et al., 2019] also fall in this category, although most commonly used together with a task reward. Go-explore [Ecoffet et al., 2019] uses the strategy of eventually returning to all previously visited states to tackle hard exploration problems: tasks with extremely sparse rewards, which are essentially state coverage maximization problems until the first reward is observed. Methods in self-supervised RL are typically evaluated by how quickly they learn a newly specified task once its corresponding reward does become available.

2.4 Data-driven robotics

Any approach using real or simulated interaction data to train a behavior or to fit a model can be considered data-driven, without limiting this definition specifically to imitation and reinforcement learning. However, methods in this category vary in the extent to which they are conditioned on data: e.g. system identification does use data to set the parameters of a physical model, but the parametrization and structure of the model itself is strongly predefined, whereas end-to-end learned systems are almost entirely determined by data. While the line can be blurry, we mainly use data-driven robotics to refer to methods that employ neural networks, without assumptions such as linearity of system dynamics or a quadratic cost function. Recent successes of this line of work include applications to throwing [Zeng et al., 2019], stacking [Lee et al., 2021], quadruped walking [Wu et al., 2022], bipedal stair climbing [Siekmann et al., 2021], in-hand manipulation [Andrychowicz et al., 2020], shaping dough [Qi et al., 2022] and various kitchen tasks [Ahn et al., 2022].

While the imitation learning and RL methods covered in the previous sections are also relevant and applicable to most robot control tasks, several constraints arise specifically in robotics applications. When interfacing with physical hardware in the real world, issues that can be largely overlooked in simulation become significant challenges. Once learning algorithms move to real robots, emloying parallelization is much less straightforward. Kalashnikov et al. [2018] used a farm of 7 identical robot arms collecting over 800 hours (33+ days) of execution data to learn picking of diverse objects from images.

Much like the ImageNet dataset [Deng et al., 2009] enabled a significant leap ahead in image classification and serves now as one of the most standard benchmarking and pretraining datasets, several efforts have set out to release a similar large dataset for robotics, including RoboNet [Dasari et al., 2020], RoboTurk [Mandlekar et al., 2018], and Bridge Data [Ebert et al., 2021].

2.4.1 Data reuse

Sharing data is essential in order to avoid starting data collection from scratch for each new experiment. In order to reuse data across agents, robot instances and experiments, we must employ off-policy algorithms as they are able to learn from data collected with a different policy, unlike on-policy methods. In addition to distributed training where data is shared across several actor-environment threads, we may want to also share data across tasks, environments, robots and even between a human and a robot (discussed in Section 2.5). One way to collect a single dataset relevant for learning several down-

stream tasks is collecting human teleoperated play data in a robot environment [Lynch et al., 2020; Lynch and Sermanet, 2021]. Lynch and Sermanet [2021] make use of a common weakly supervised learning pattern of labeling a small subset of data (language descriptions collected for <1% of sequences) and using readily available proxy labels (goal images) for the rest. The robot's own data collected while learning one task can also be reused for learning other tasks [Cabi et al., 2019; Kalashnikov et al., 2021]. In Chapter 4, we propose a novel way to learn robotic reward functions from mixed-task demonstration data.

2.4.2 Reward annotation

Many results in RL depend on a significant time investment in designing and tuning the parameters of reward functions. While a globally optimal reward function is highly non-trivial to determine for complex behaviors such as front flips, Christiano et al. [2017] simplify the task to rating one of two policy executions as being closer to the intended behavior. In a similar vein, Cabi et al. [2019] ask annotators to provide sketched curves of perceived progress to the goal, which are significantly faster to provide than per-timestep manual reward annotation. Learned success classifiers or reward models [Sermanet et al., 2017] are another way to reduce annotation workload.

2.4.3 Learning from demonstration

While sometimes considered synonymous with imitation learning, learning from demonstration (LfD) refers to the family of approaches making use of expert data together with RL objectives (also called apprenticeship learning). Unlike the case of pure imitation, a task reward is typically assumed to be available, and unlike in offline or mixed offline-online RL, the actions in the external data are assumed to be optimal or at least of high quality (thus combining supervised learning and RL). LfD approaches aim to maximize task reward while staying close to the expert data [Vecerik et al., 2017; Rajeswaran et al., 2017]. This may present advantages over either RL or imitation learning alone: if the exploration problem is difficult or the reward is poorly specified, demonstrations allow to bootstrap and regularize learning to a desirable subspace of policies, yet learning from task reward, if it is available, allows to potentially outperform the demonstrations through trial and error.

Demonstrations can also be used to learn state representations, infer rewards (in what is called inverse RL) or to learn an action representation to simplify a complicated action space. Dadashi et al. [2022] propose to use demonstrations to learn a discretization of a continuous action space.

2.4.4 Combined learning and optimal control

Several hybrid methods taking inductive biases from classical control have been proposed. The original residual RL methods Johannink et al. [2019]; Silver et al. [2018], which we build on in Chapter 3, used a hand-engineered controller for a block insertion task [Johannink et al., 2019] or pushing and fetching with a hook [Silver et al., 2018]. Dynamic movement primitives [Ijspeert et al., 2013] have also been additively adapted with residual RL [Davchev et al., 2020]. In the same vein, [Zeng et al., 2019] additively correct a trajectory model proposing release velocities for throwing with autonomous trial and error.

2.5 Learning from human videos

Whereas the methods above exclusively use data from the same MDP as the target task (with the possible exception of the reward function, which may not be available in imitation and inverse RL tasks), more recently, a line of work has proposed to share experience across actors operating in different but related state and action spaces. An instance of this problem which is particularly relevant in the real world is robots learning from observing humans. Schmeckpeper et al. [2020] propose to learn a domain-invariant embedding space for robot and human observations from the same task as well as an inverse model for predicting robot actions corresponding to the embeddings for human transitions. Zakka et al. [2022] use a temporal cycle consistency constraint to align trajectories accomplishing the same task but possibly demonstrated by different embodiments – such as a robotic gripper, a human hand, two human hands or a human using various tools – to learn an embodiment-invariant representation of task progress. Sermanet et al. [2018] use multiple views of the same human demonstrations to define a time-contrastive objective for also learning task progression aware image representations. While Schmeckpeper et al. [2020], Zakka et al. [2022] and Sermanet et al. [2018] use data for a single task at a time, Chen et al. [2021a] and Shao et al. [2021] propose to use a much larger, multi-task human video dataset [Goyal et al., 2017] for learning a reward function conditioned on a human video [Chen et al., 2021a] or a language description of the task [Shao et al., 2021].

Outside the context of robotics, classic video understanding tasks include action recognition [Wang and Schmid, 2013; Carreira and Zisserman, 2017], segmentation (both spatial [Tsai et al., 2016] and temporal [Koprinska and Carrato, 2001]), object tracking and prediction / generation. Video data is well suited for weakly supervised and self-supervised objectives as it provides significantly more information than standalone

images: audio, dynamics, length of time, direction of time, and temporal continuity in contiguous frames. Alayrac et al. [2016] propose to use narrated instruction videos to automatically discover key subtasks commonly occurring in videos of the same task. Zhukov et al. [2019] extend upon this work by learning component models, such as ones modelling verbs and objects of action descriptions, that may occur across several tasks. Moreover, Elhamifar and Zaing [2019] propose a Hidden Markov Model whose latent states correspond to different action classes and which supports repeated, omitted and out-of-order actions. Progress in general video understanding tasks is likely to lead towards more capable robotic agents in the long term.

Chapter 3

Residual RL from Demonstrations

RESIDUAL REINFORCEMENT LEARNING (RL) has been proposed as a way to solve challenging robotic tasks by adapting control actions from a conventional feedback controller to maximize a reward signal. We extend the residual formulation to learn from visual inputs and sparse rewards using demonstrations. Learning from images, proprioceptive inputs and a sparse task-completion reward relaxes the requirement of accessing full state features, such as object and target positions. In addition, replacing the base controller with a policy learned from demonstrations removes the dependency on a hand-engineered controller in favour of a set of demonstrations, which can be provided by non-experts. Our experimental evaluation on simulated manipulation tasks on a 6-DoF UR5 arm and a 28-DoF dexterous hand demonstrates that residual RL from demonstrations is able to generalize to unseen environment conditions more flexibly than either behavioral cloning or RL fine-tuning, and is capable of solving high-dimensional, sparse-reward tasks out of reach for RL from scratch.

3.1 Introduction

Reinforcement learning has the potential to enable autonomous learning of hard-to-specify behaviors under varying environment conditions. However, learning control entirely from data requires a significant amount of robot experience to be collected for most tasks of interest. To improve sample efficiency, residual RL [Johannink et al., 2019; Silver et al., 2018] takes advantage of a conventional controller to address the part of the task that can be solved efficiently by feedback control, but adapts the control outputs through a superimposed residual policy, trained with RL.

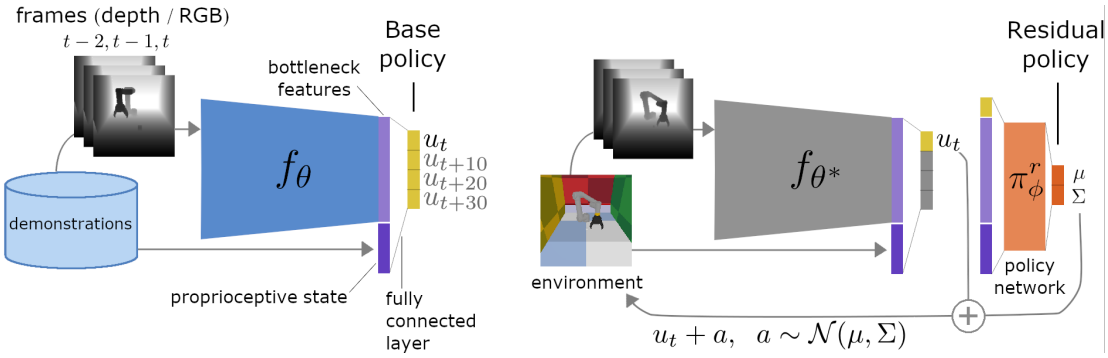
Videos of policy executions are available on the project website:
https://sites.google.com/view/rrlfd/.

However, relying on hand-engineered base controllers imposes certain limitations on the kind of tasks that can be addressed. Providing prior information in the form of demonstrations instead of a manually specified feedback controller would allow residual RL to be used in a wider set of applications for which first-principles physical modeling and accurate state estimation are not feasible. Drawing from the extensive field of imitation learning, we train residual RL to complement a policy learned with behavioral cloning (BC) [Pomerleau, 1988].

Replacing the base controller with a policy learned from demonstrations presents a number of advantages. First, the resulting policy can be trained from data alone, in the form of demonstrations for the base controller and rollouts in the environment for the residual policy. We argue that in many settings, demonstrations are easier to provide than custom controllers for new tasks, objects or robot configurations. Demonstrations may, for example, be obtained by teleoperation, or any logged task execution data. Second, feedback controllers typically rely on full state estimation, such as the positions of objects, targets and obstacles. We instead propose to learn control entirely from visual and robot proprioceptive inputs, and learn task-specific state features through BC on demonstrated trajectories. Whereas the residual policies considered in prior work require hand-crafted state features [Johannink et al., 2019; Silver et al., 2018; Barekatain et al., 2020; Schoettler et al., 2020; Rana et al., 2020; Davchev et al., 2020], our method, Residual Reinforcement Learning from Demonstrations (RRLfD), learns from images and requires minimal feature engineering.

Moreover, learning a residual on top of a base controller enables a significant decrease in sample complexity compared to training RL from scratch [Johannink et al., 2019; Silver et al., 2018; Schoettler et al., 2020; Rana et al., 2020; Davchev et al., 2020] as the controller provides a good prior for the RL policy. This reduces the importance of exploration and thus of reward shaping, and allows for more difficult and longer-horizon tasks to be learned from sparse rewards compared to a randomly exploring agent that might never observe a task completion in practice. The residual formulation also simplifies the learning task, allowing the RL policy to take smaller actions and to reduce the magnitude of exploration required—an important consideration for robotics applications. Moreover, the residual form has potential to alleviate catastrophic forgetting [McCloskey and Cohen, 1989] compared to a policy that is simply initialized from demonstrations and fine-tuned with RL.

Our contribution is therefore twofold: we propose a novel way to leverage data gathered from the system to accelerate the training of residual RL policies in image space through visual feature sharing, and present a fully data-driven method, RRLfD, leveraging both imitation learning and residual RL to allow for efficient training of sparse image-based tasks from demonstrations. We evaluate our method on robotic



(a) (b)

Figure 3.1: a) We propose a way to leverage demonstration data to learn a control policy as well as task-relevant visual features through behavioral cloning on image and proprioceptive inputs. b) The policy is then improved through reinforcement learning by a superimposed residual policy, based on the learned visual features, allowing data-efficient learning of control policies in image space from sparse rewards.

manipulation tasks; however, the formulation is general and readily applicable to any control task in a continuous action space. We begin by comparing the proposed method to prior work in Section 3.2, then describe the residual policy's structure and training in Section 3.3. We evaluate our method in seven experimental settings from two simulated task suites and present the results in Section 3.4.

3.2 Related work

Residual RL for robot control, concurrently proposed by Johannink et al. [2019] and Silver et al. [2018], has been studied for manipulation tasks [Johannink et al., 2019; Silver et al., 2018; Schoettler et al., 2020] and navigation [Rana et al., 2020] to complement a conventional feedback controller. Most related to our method is the work of Davchev et al. [2020], where the base policy is structured as a dynamic movement primitive whose coefficients are learned using demonstrations but which operates on full state features, including a goal position for insertion. We instead make no assumptions about the structure of the base policy but train a convolutional neural network using direct behavioural cloning in image space, without requiring access to accurate state estimation.

Another line of work has generalized the residual formulation to consider adaptively re-weighting multiple base policies, each trained with RL to solve the same task under different robot dynamics [Barekatain et al., 2020]. Related to residual control is also a recent work on robotic throwing [Zeng et al., 2019], where a scalar release velocity

given by projectile physics is additively corrected using self-supervised learning. In addition to policies, residual formulations have also been proposed for correcting the predictions of approximate environment models [Ajay et al., 2018; Allevato et al., 2019; Saveriano et al., 2017] and adapting features in transfer learning [Zhang et al., 2020].

Alternatively, an agent's policy could be initialized using demonstrations and its training continued using RL [Kober and Peters, 2008; Rajeswaran et al., 2018]. We instead fix the base policy after training and always add it to the continuous control action in order to alleviate the risk of catastrophic forgetting for parts of the state space.

Also related to our work are off-policy RL methods which learn jointly from demonstrations and new environment rollouts, by merging them in a single replay buffer [Hester et al., 2018; Vecerik et al., 2017] or with an auxiliary objective that aims to stay close to the demonstrations while also maximizing observed reward [Nair et al., 2018; Rajeswaran et al., 2018; Zhu et al., 2018]. These approaches either pretrain a policy offline and then continue training on the real system, or learn from scratch using offline data as a prior. There is also work on fully offline RL that can be used to train a policy using only demonstrations without access to the system [Gülçehre et al., 2020; Fu et al., 2020; Wang et al., 2020; Argenson and Dulac-Arnold, 2021]. As our approach is rather orthogonal to existing learning from demonstration methods, it could be readily combined with other training protocols (e.g., a pre-populated replay buffer) and objectives (e.g., staying close to demonstrations). An interesting direction for future work could be to extend fully offline methods to further online training, for example by using offline RL in lieu of BC in our method.

Prior work on picking and insertion from images has leveraged both demonstrations and black-box policies. Schoettler et al. [2020] train connector plug insertion using both residual RL and RL with demonstrations from images. Although the task is executed on a real robot, the target is fixed and known by the base controller, and the final control policy is not shown to generalize to unseen target positions, unlike our method. The QT-Opt algorithm [Kalashnikov et al., 2018] learns grasping from images using a parallel set of robots (similar to Levine et al. [2018]) and is initialized with a black-box controller to help with the initial exploration phase. However, it does not use residual control, and the initial controller is hand-coded. QT-Opt is able to learn a good grasping policy both in simulation and on real robots, but comparing data efficiency is not straightforward as performance is often reported for the number of episodes or gradient updates rather than environment time steps. Our method can instead be initialized from demonstrations without a hand-engineered controller.

3.3 Residual Policy Learning from Demonstrations

RRLfD is trained in two stages (Fig. 3.1). A convolutional neural network (CNN) is first trained to predict demonstrated control actions given a short history of depth or RGB images as well as current robot proprioceptive state (Section 3.3.1). Using demonstrated trajectories, the network learns to capture visual features relevant to solving the control task.

However, the base policy can only learn behaviour that was present in the demonstrations. Depending on the coverage of the dataset, behaviors involving recovery from failure or states near the edges of the workspace may not be included. Moreover, pure BC is known to suffer from compounding errors [Ross and Bagnell, 2010]. To improve the policy with autonomous environment interaction, a light-weight policy network on top of the learned CNN features is trained with RL to additively correct the base policy's actions (Section 3.3.2).

3.3.1 Base policy

We learn a base policy using behavioral cloning (BC). First, N demonstrations are gathered for a task of interest to create a dataset consisting of the demonstrated trajectories' states $S^i = [s_1^i, ..., s_{T_i}^i]$ and actions $A^i = [a_1^i, ..., a_{T_i}^i]$ taken at each time step $t = 1, ...T_i, i = 1, ..., N$, drawn from a behavior policy $\pi^e(s_t^i)$. Each state s includes a history of stacked camera frames, as either depth or RGB, as well the robot's proprioceptive state. We consider a history of three previous frames to allow policies to capture velocity and acceleration of objects in the scene. Where applicable, the inverse kinematics of the robot are used to reduce the effective dimensionality of the action space from joint space control to end-effector control.

A convolutional neural network f_θ mapping a state s to a sequence of actions $[\hat{a}_t, \hat{a}_{t+10}, \hat{a}_{t+20}, \hat{a}_{t+30}]$, and parametrized by θ, is then trained with BC on this dataset:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \underset{(s_t,a_t)\sim(S^{1:N},A^{1:N})}{\mathbb{E}} \sum_{\delta=0,10,20,30} L(\hat{a}_{t+\delta}, a_{t+\delta}), \tag{3.1}$$

to obtain a policy $\pi_\theta^b(s) = \hat{a}_t$. In addition to predicting a_t, we include actions 10, 20, and 30 time steps ahead and minimize loss over each of these targets, as done by Strudel et al. [2020]. This serves as an auxiliary prediction task leading to better data efficiency [Jaderberg et al., 2017] and encourages the policy to capture longer-term task structure (up to three seconds in the future given control actions at 10Hz).

Depending on the task, the action space may have discrete and continuous components, which we may denote by g and v, respectively, such that $a = [v, g]$ and $\hat{a} = [\hat{v}, \hat{g}]$.

For tasks requiring discrete components such as gripper state control, the loss L is defined by weighting L2 and cross-entropy terms, as in Strudel et al. [2020]:

$$L((\hat{v}, \hat{g}), (v, g)) = \lambda ||\hat{v} - v||_2^2 - (1 - \lambda) \sum_{c=1}^{|g|} g_c \log \hat{g}_c, \tag{3.2}$$

where λ is a hyperparameter. For tasks with fully continuous action spaces, only the regression loss is considered ($\lambda = 1$).

3.3.2 Residual policy

After BC training, the base policy π_θ^b is fixed. Given $u = \pi_\theta^b(s)$, a residual policy $\pi_\phi^r(s, u) = a$ is trained using RL to complement the base action u, and the resulting control action $u + a$ is executed in the environment. π_ϕ^r is trained to maximize the expected discounted return G^π in the residual Markov Decision Process (MDP):

$$\begin{aligned} &Q^\pi(s, a) = \mathbb{E}\left[G^\pi(s, a)\right] = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right], \\ &s_t \sim p(\cdot \mid s_{t-1}, u_{t-1} + a_{t-1}), a_t \sim \pi_\phi^r(\cdot \mid s_t, u_t), \\ &u_t = \pi_\theta^b(s_t), s_0 = s \end{aligned} \tag{3.3}$$

where p is the state transition dynamics, γ is a discount factor in $[0, 1)$ and r is a reward function whose value is 1 if the task has been completed and 0 otherwise. π_ϕ^r takes as input the base action u, the robot's proprioceptive state, and the features of the bottleneck layer (i.e., the final hidden layer before the fully connected output layer) of the BC policy network f_θ. The bottleneck layer features provide the residual policy with visual information learned during BC. π_ϕ^r then outputs a corrective action a by predicting the parameters of a Gaussian distribution with diagonal covariance:

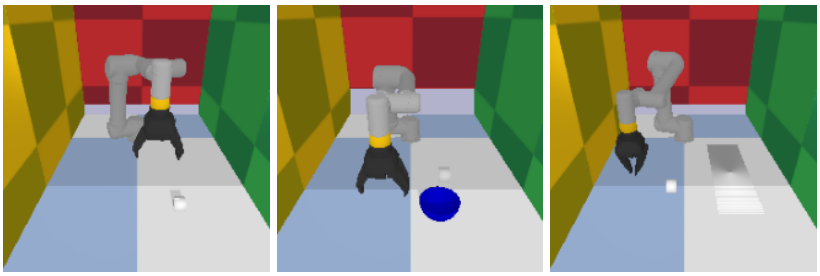
$$a \sim \mathcal{N}(\mu, \Sigma), \quad \text{with } (\mu, \Sigma) = \pi_\phi^r(s, u). \tag{3.4}$$

At evaluation time, the policy is made deterministic: $u + \mu$ is executed in the environment instead of drawing a sample from the predicted Gaussian.

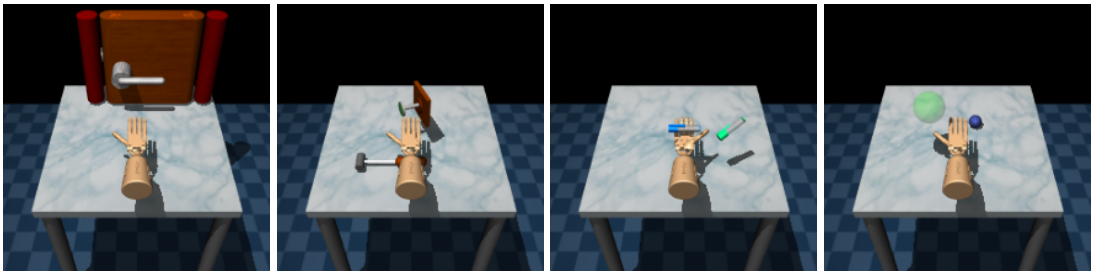
3.4 Experiments

3.4.1 Environments

To demonstrate the generality of our approach, we consider control tasks from two distinct manipulation suites of different complexity and dimensionality. The Mime environments [Strudel et al., 2020] define of a 6-DoF UR5 robotic arm with a 3-finger Robotiq gripper using the PyBullet physics simulator [Coumans and Bai, 2016–2021]. An



(a) Pick (b) Bowl (c) Push



(d) Door (e) Hammer (f) Pen (g) Relocate

Figure 3.2: We evaluate RRLfD on seven manipulation tasks on two different robotic simulation platforms: a 6-DoF UR5 arm (a–c) and a 28-DoF ShadowHand model (d–g).

inverse kinematics submodule allows control in task space by converting end-effector velocities to joint velocities at 10Hz, exposing to the higher-level policies π_θ^b and π_ϕ^r an action space consisting of the translational velocity v in $\mathbb{R}^3$ and a binary gripper state g in $\{0, 1\}$. We consider two standard tasks included in Strudel et al. [2020]: picking up a cube (Fig. 3.2a) as well as picking up a cube and placing it in a bowl (3.2b), and additionally define a third task of pushing a cube to a fixed target location (3.2c). All object positions and sizes, starting robot pose as well as the positions of the surrounding walls are drawn at random for each episode.

Our second task suite of interest, Adroit [Rajeswaran et al., 2018], defines a five-fingered arm controlled at 10Hz in a continuous action space consisting of joint values for a ShadowHand-inspired dexterous hand (24 DoF) and the position of the arm (4 DoF). The suite consists of four MuJoCo [Todorov et al., 2012] environments, defining tasks of opening a door, hammering a nail, in-hand orientation of a pen, and object relocation (Fig. 3.2d–3.2g).

3.4.2 Demonstrations

We follow convention set by prior work and define the expert policy π^e by either a script (for Mime environments [Strudel et al., 2020]) or a previously trained RL agent (for Adroit environments [Jain et al., 2019]), but it could equally come from a human teleoperator or any black-box controller.

3.4.3 Observation space

We consider two kinds of inputs to the base policy: images only, and images with robot proprioceptive state. To give an indication of an upper bound reachable with flawless state estimation pipelines, we also consider BC trained on images with full state, including both proprioception and the positions of relevant objects. For the Mime environments, the proprioceptive state consists of tool position in $\mathbb{R}^3$, tool velocity in $\mathbb{R}^3$, gripper state in $\mathbb{R}$ and gripper opening (if positive) or closing (if negative) velocity in $\mathbb{R}$. Full state additionally includes cube position in $\mathbb{R}^3$ (and bowl position in $\mathbb{R}^3$, if applicable). For Adroit, proprioception includes joint positions, joint velocities, palm position and the fingertips' tactile sensor readings as defined by Jain et al. [2019]. As full state, we use the environments' original observation space. We keep each task suite's default setting and use depth cameras in Mime and RGB cameras in Adroit, but use a consistent resolution of 240×240px in each.

3.4.4 RL algorithm

We apply Distributional Maximum a-posteriori Policy Optimization (DMPO), a variant of Maximum a-posteriori Policy Optimization (MPO) [Abdolmaleki et al., 2018] with a distributional critic on the residual task, as it outperformed pure MPO and another recent off-policy algorithm, D4PG [Barth-Maron et al., 2018], in our experiments. The implementation is publicly available as part of the Acme framework [Hoffman et al., 2020]. MPO is an off-policy actor-critic algorithm based on maximum entropy RL. It treats RL as an inference problem: at each update step, a distribution of policy parameters is updated using a single-step temporal difference (TD) update, with a Gaussian prior around the current policy. Distributional MPO (DMPO) additionally adopts a distributional Q-network parametrized as in C51 [Bellemare et al., 2017]: instead of predicting the expectation of $G^\pi(s, a)$, the critic learns to model its full distribution with a categorical distribution of 51 atoms.

3.4.5 Training details

For data efficiency, we use image augmentation in BC training as presented by Strudel et al. [2020], namely random crops and rotations. For Mime tasks, we additionally use the environments' built-in viewpoint augmentation and record the demonstrations from five camera positions which are sampled from a section of a sphere centered on the robot. For details, see Strudel et al. [2020]. Image inputs are normalized to be in $[-1, 1]$ and the demonstrated actions as well as proprioceptive features are normalized per dimension to have zero mean and unit variance in demonstration data. For tasks

requiring the gripper, we empirically set λ to 0.9 as done by Strudel et al. [2020].

For f_θ, we use a ResNet-18 [He et al., 2016] with all layer sizes halved, such that the bottleneck layer features have dimensionality $d = 256$, and concatenate the scalar inputs, where used, to the bottleneck features. Given demonstration datasets of different sizes, we use 95% of the data for training and validate on the remaining 5%, with the f_θ training epoch with the lowest validation loss evaluated in the environment on a fixed set of 100 unseen initial states. To represent the residual policy π^r_ϕ, we use a three-layer fully connected neural network with layer normalization and layer sizes $[256, 256, 256]$, and another with sizes $[512, 512, 512]$ for the critic (defaults of the DMPO implementation [Hoffman et al., 2020]).

Given the normalization scheme applied to demonstrated actions in BC training, actions drawn from π^b_θ must again be *denormalized* to match the original action space of the environment:

$$u = \sigma(A) \circ \pi^b_\theta(s) + \mu(A), \tag{3.5}$$

where A is the set of all actions in the BC dataset and $\circ$ denotes element-wise multiplication. Like base policy actions, we also denormalize residual actions sampled from π^r_ϕ using statistics from the demonstrations, but with the output distribution centered around zero rather than the demonstrations' mean:

$$a \sim c\sigma(A) \circ \mathcal{N}(\pi^r_\phi(s, \pi^b_\theta(s))), \tag{3.6}$$

where c is an optional scalar. This allows both π^b_θ and π^r_ϕ to better handle a potentially very heterogeneous action space and efficiently learn tasks that may require significantly different distributions of actions per action dimension. The residual denormalized actions may additionally be scaled using $c < 1$ to encourage small corrective actions relative to the variance of demonstrator actions. In addition to c, we find the size of exploration (the diagonal elements of Σ) at initialization to be an important hyperparameter: a larger Σ means larger exploratory residual actions will be taken early on in training. We set both c and Σ's diagonal (a single value for all elements) by grid search over $\{0.01, 0.033, 0.1, 0.33, 1\}$. The range for the critic's value distribution is set to $[0, 1]$ to match the range of possible sparse returns, and the learning rate is set empirically to 3.3×10^{-4} for all tasks (by considering a small grid around the implementation default, 10^{-4}). All other algorithm hyperparameters were kept at default values of the Acme implementation.

3.4.6 Base policy

Success rates for the BC policy are given in Fig. 3.3. Although there is significant variance across random seeds for most tasks, overall the performance increases as more

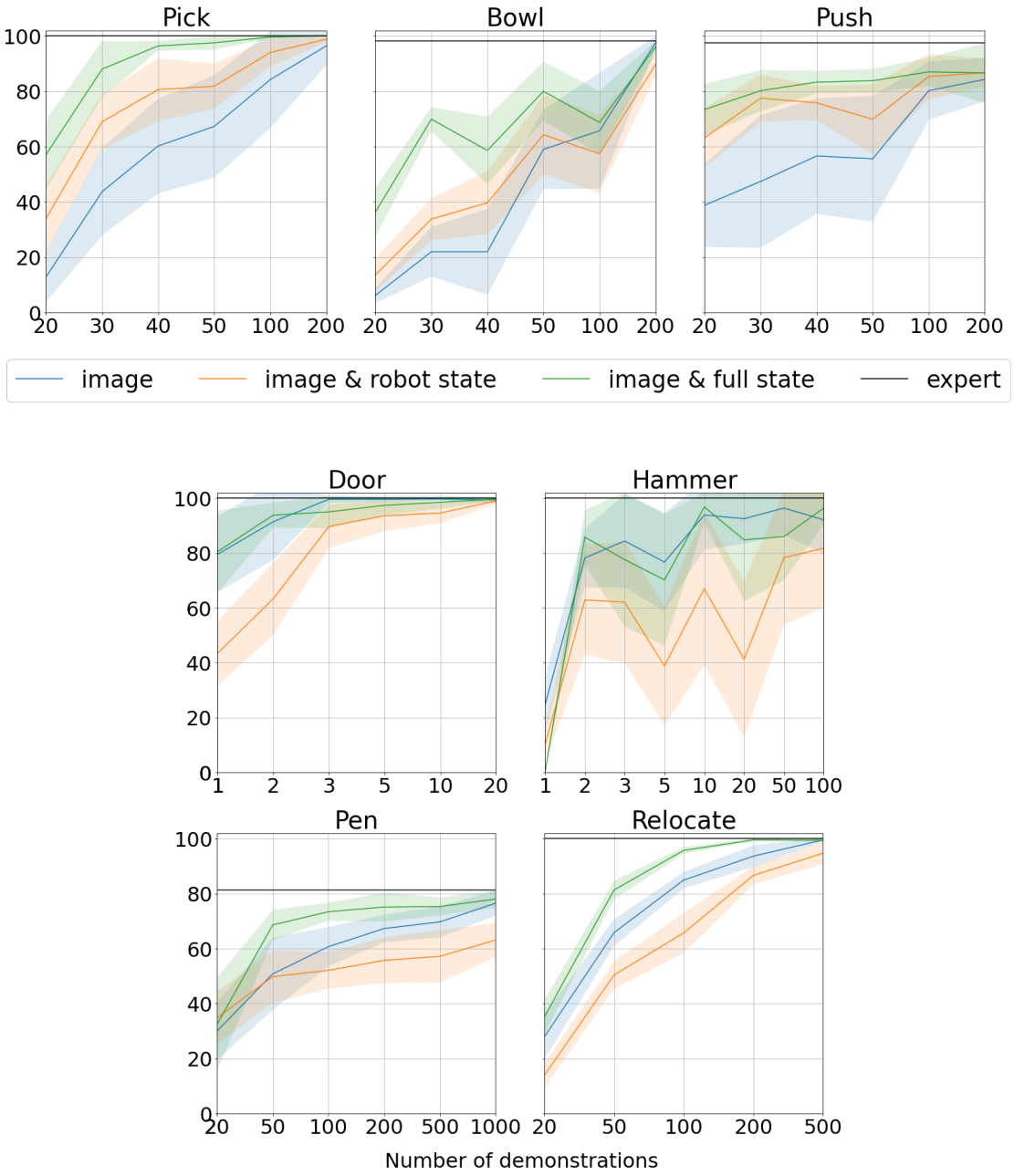


Figure 3.3: BC success rates (as %) evaluated on 100 unseen initial states. (10 seeds, 95% confidence intervals)

demonstrations are added and eventually approaches the expert's performance. We observe that conditioning BC on the robot proprioceptive state in addition to images improves performance on Mime but not on Adroit. We believe this is due to 1) the tasks being solvable from images alone, and 2) overfitting enabled by the high dimensionality of proprioceptive state in Adroit (54–100, depending on the task). Exploring architectures and training procedures that would enable successful fusion of these modalities, leading to a performance increase on Adroit relative to image-only inputs, would be an

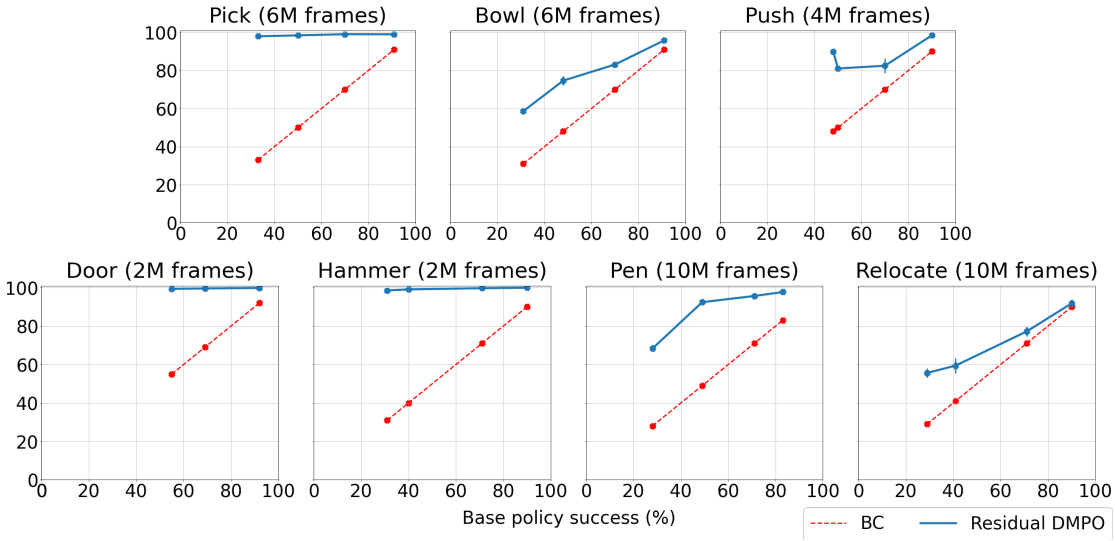


Figure 3.4: Success rates of residual policies as a function of base policy success rate (mean of 5 seeds, 95% confidence intervals).

interesting topic for further work.

A further increase in success rate can be obtained when full environment state is included; however, we assume it is not known in general. We therefore only consider base policies without access to full state in the residual experiments.

3.4.7 Residual policy

We train residual DMPO on top of BC policies of various strengths to investigate the effect of base success rate on final performance and the required training time. Out of the BC policies included in Fig. 3.3, we choose policies with success rates close to 30%, 50%, 70% and 90% for each task (with the exception of Door, for which no BC policy achieved less than 55%) to cover a wide range of possible base policy performances. The residual success rate at convergence is shown in Fig. 3.4 as a function of base success. However, significantly shorter training times are sufficient to outperform the base policies: Fig. 3.5 shows the improvement in success rate over training. Notably, we observe only a minor drop, if any, in success rate relative to the BC policy at the start of training, making residual training viable on real-world systems where performance should not drastically fall. The policies are evaluated every 100,000 training frames (i.e., steps in the environment); in reporting final numbers, we consider a moving average of 5 evaluations.

Although Adroit environments are originally defined with full state observations and shaped rewards [Rajeswaran et al., 2018], we make use of neither and instead use camera observations, proprioceptive state (joint positions, joint velocities, palm position

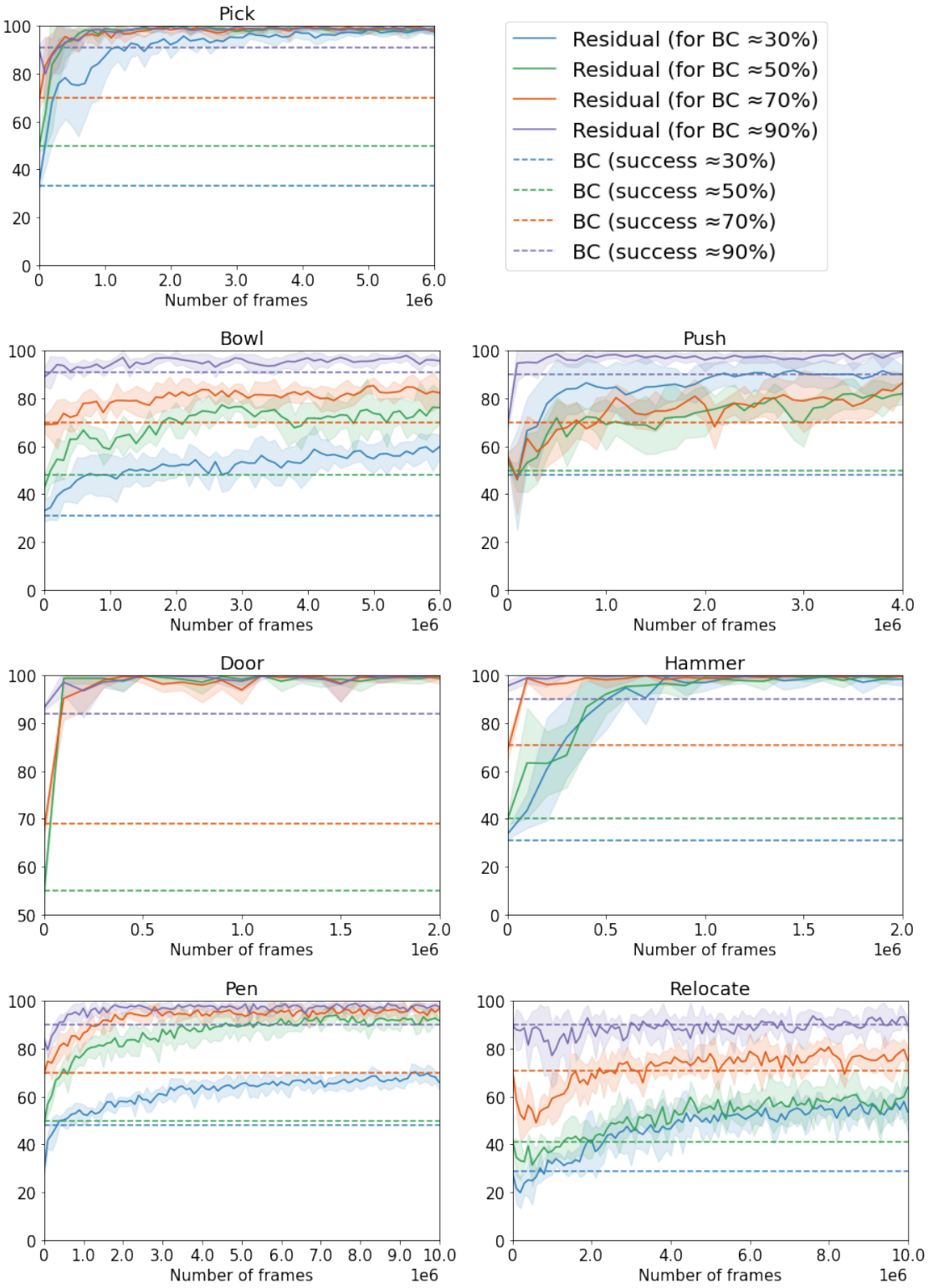


Figure 3.5: Success rates for the residual agent over training (5 seeds, 95% confidence intervals).

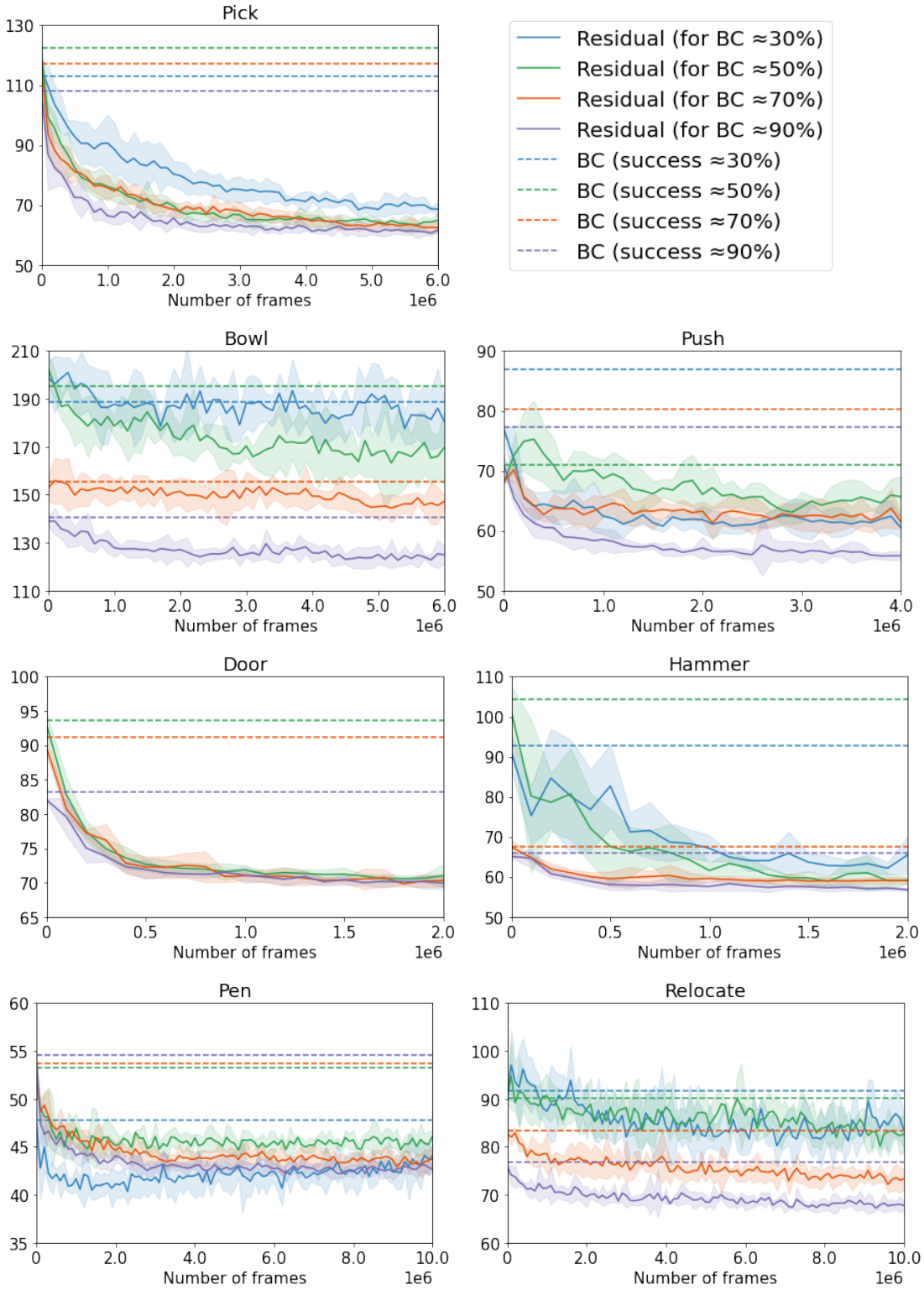


Figure 3.6: Length (in number of time steps) of successful episodes for the residual agent over training. A lower number means the policy is able to solve the task faster.

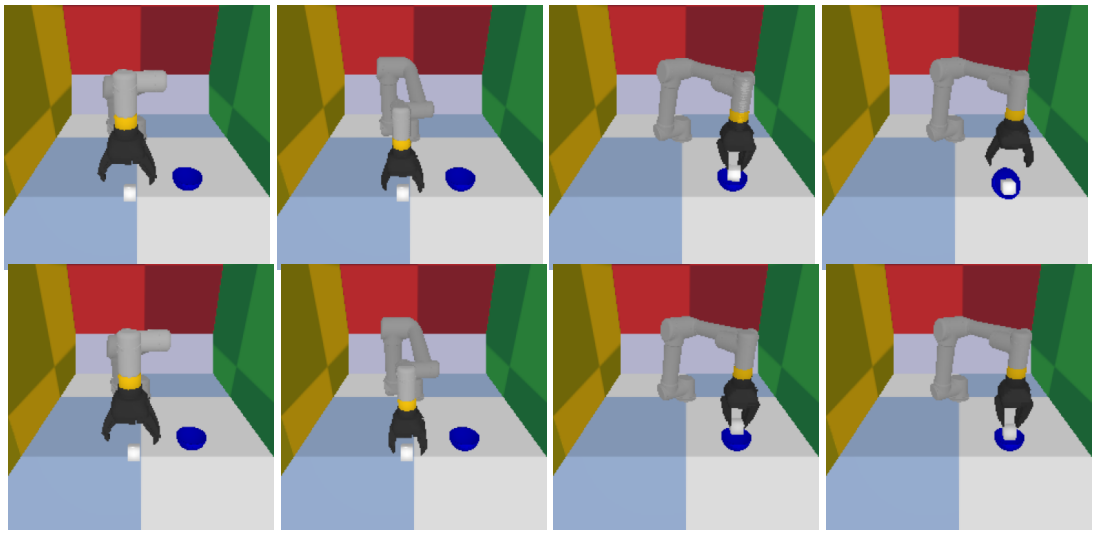


Figure 3.7: An example episode from the test set for Bowl where BC fails but the residual policy succeeds. The instance is difficult as the cube is large relative to the bowl. Residual RL has learned to adjust the position from which the cube is released to be more precisely centered on the bowl. Top: BC policy trained on 30 demonstrations. Bottom: Residual policy.

as well as tactile sensor readings – the observation space previously defined in Jain et al. [2019]), and sparse rewards only. Accordingly, we measure success rate of the resulting policies in our evaluation, not episodic return as defined by the shaped reward functions. We observed that the RL policy, unlike BC, is able to successfully combine robot state information with visual inputs in Mime as well as in Adroit, and omitting robot state consistently hurt residual performance.

The average length of successful episodes over the course of training in shown in Fig. 3.6 for each task. The residual policies learn to solve the tasks not only more reliably, but also more efficiently than BC for all tasks and base policies while respecting the environments' velocity limits, with an average 21% reduction in execution time, corresponding to 2 seconds of robot time per episode. An example side-by-side comparison of BC and the residual policy is shown in Fig. 3.7.

3.4.8 Fine-tuning baseline

RRLfD's residual shape with a fixed base policy protects against catastrophic forgetting, which can be a significant source of instability in deep RL. One alternative way to incorporate demonstrations as a prior in RL training is to initialize the policy network using an imitation learning method, such as BC, and then continue training using a non-residual RL objective. We considered two settings: we either froze the ResNet

weights as in residual training, or included them in fine-tuning. The network architecture (as shown in Fig. 3.1b) is kept constant, except for the omission of the base action u_t; the network π_ϕ^r is included at BC training stage. Moreover, a continuous action space is used, including velocity control of the gripper, as DMPO does not support a discrete-continuous action space. Appropriately handling a hybrid action space could be considered in further work, as was done by Neunert et al. [2019].

We found that fine-tuning the full network leads to immediate catastrophic interference in the ResNet features given that they are shared between the actor and the critic. As the critic network cannot be directly initialized from demonstrations without rewards, updating the full network using the randomly initialized critic's objective destroys the pretrained features – we found this to be the case even for very small learning rates. As the pretrained policy in turn depends on these features, its performance falls to near 0%, after which no further useful training data can be collected.

In an attempt to alleviate catastrophic forgetting, we also run experiments with the CNN weights fixed and fine-tune the policy layers π_ϕ^r only. Although the fine-tuned policy is able to improve upon BC for some of the easier tasks, it requires longer training and larger exploratory actions than residual RL, both of which would be undesirable on a real robot. For Door, success rates of 97.9% and 99.4% rates are achieved given initializations with 75% and 89% success, respectively, after 3 million frames and only with the diagonal elements of Σ initialized to 1.0 (0.33 is sufficient in residual training). For Pick and Pen, fine-tuning was able to slightly improve over BC, but only for low-performing initializations: at best, it scored 50.8% for Pick after 3 million frames, and 52.2% for Pen after 7 million frames, starting from 30% in both cases. For all other tasks and BC initializations (policies with ≈30 and ≈90% success were considered for each task), success never exceeded BC performance. A direction for further work could be to consider also initializing the value function from demonstrations (see e.g. [Wang et al., 2020]) by labeling demonstrations with a sparse reward, assuming successful completion. However, a perfect demonstrator is not always available: our expert scores just 81% for Pen, for example.

3.4.9 RL baseline

We have also run experiments to find out the number of frames required to train DMPO from scratch on each task. As input, we consider either images with robot proprioceptive state (denoted robot state) or the full environment state including robot state (denoted full state), in which case the CNN is omitted. Each action dimension is normalized to the unit interval, as demonstration statistics are not assumed to be available.

Although our setting of interest is that of sparse task completion rewards and read-

	image + robot		full state			
	dense reward		sparse reward		dense reward	
task	success	steps	success	steps	success	steps
Pick	0.0	8M	59.2	9M	98.8	10M
Bowl	0.0	8M	0.0	20M	52.2	9M
Push	1.0	3M	99.0	2M	98.0	2M
Door	0.0	6M	20.0	5M	99.5	7M
Hammer	6.3	6M	20.0	7M	98.9	6M
Pen	9.4	4M	93.1	13M	91.2	12M
Relocate	0.0	6M	0.0	20M	97.4	12M

Table 3.1: Sample efficiency and success rate at convergence of DMPO trained from scratch. Sparse reward policies from images and robot state scored nearly 0.0 for all tasks (not shown), which is not surprising given that RL without any priors is extremely difficult in settings with sparse rewards and very high-dimensional inputs.

ily available state features, we also consider settings with dense reward signals and full state features for each task to highlight the trade-offs between providing demonstrations and designing shaped rewards or higher-level state features. For Pick, we use the negative Euclidean distance between the gripper and the cube; for Bowl, the sum of the negative distance between the gripper and the cube, and between the cube and the bowl; and for Push, the sum of negative distance between the gripper and the cube as well as the cube and the goal region. While further reward shaping beyond our choice of reward functions could accelerate learning, defining a fully dense reward, i.e., one that guides the policy towards solving the task from any state—without creating unintentional local maxima—can in general be as difficult as designing a controller to solve the task, and usually requires balancing multiple reward components, such as approaching the cube vs. bringing the cube closer to the bowl. For Adroit environments, we use the carefully shaped multi-component rewards defined by Rajeswaran et al. [2018]. We also add sparse task completion to each dense reward, appropriately scaled to prioritize task success, as we found this to increase success rates.

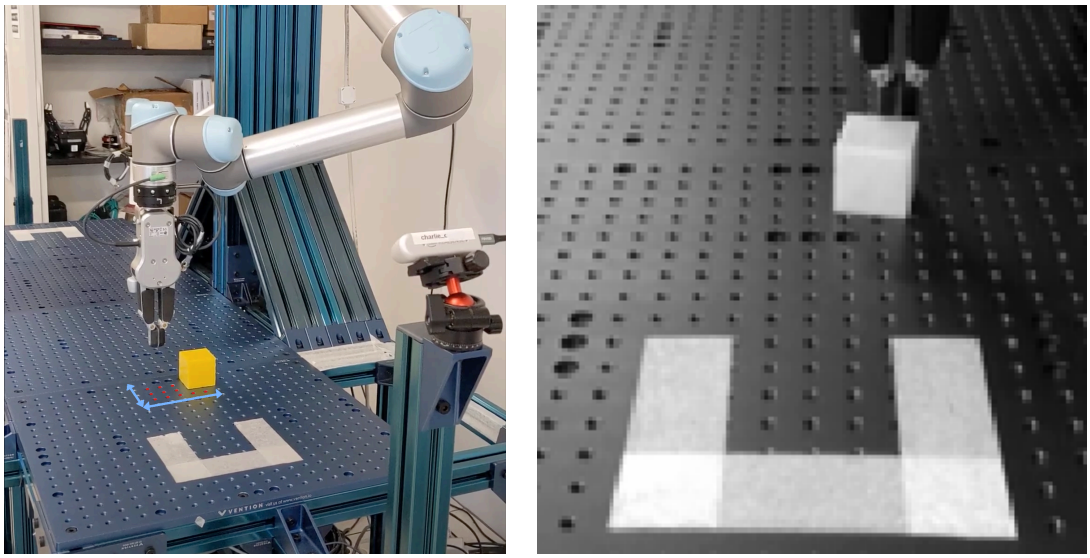
As shown in Table 3.1, learning from images does not lead to any successful policies within the training budgets: although the agent occasionally completes an episode successfully, this is not sufficient to learn useful visual representations, even with a dense reward. It is difficult and time-consuming to learn control from scratch without any priors in this setting due to the high dimensionality of both the input and the action space; many RL tasks consider a smaller dimensionality and use full state based inputs

or low-resolution images instead. Replacing the depth frames with the full environment state simplifies the task and regularly leads to successful policies in the sparse reward setting for Push and Pen – as well as occasionally for Pick, Door and Hammer – and in the dense reward setting for all tasks. Learning Bowl from scratch is particularly tricky as adept gripper control is of central importance, whereas DMPO considers a continuous action space.

Comparing the empirical sample efficiencies of the RL baselines and the results presented in Fig. 3.4, we have shown demonstrations to be a viable alternative to manual reward shaping and full state estimation. Using a residual formulation, learning from demonstration can be combined with any RL method.

3.4.10 Real robot experiments

To evaluate the viability of RRLfD in learning manipulation tasks on real robots, we also propose a preliminary experiment on a real UR5 arm. Using a single Intel RealSense camera and a joystick for 2D teleoperation, we record 50 demonstrations of a pushing task (Fig. 3.8). The end-effector (a closed OnRobot RG6 gripper) is controlled in the xy-plane at a fixed height. The starting position of a 5cm cube is randomized within a 10cm x 14cm region of the table, and the task is considered successful if the cube is pushed fully behind the front end of the tape within a maximum of 15 seconds without it hitting the tape at any point. Evaluation for all policies is performed on a fixed set of



(a) The cube's initial position is randomized within the region shown in blue. The evaluation positions (15 in total) are shown in red.

(b) Image from the robot's observation space.

Figure 3.8: The pushing task setup on a UR5 arm.

Number of demonstrations	π_θ^b(image, gripper position)	π_θ^b(image, gripper position, gripper velocity)
20	7/15	**10/15**
50	10/15	**13/15**

Table 3.2: Success rates for BC as a function of the number of demonstrations and policy inputs on Real UR5 Pushing.

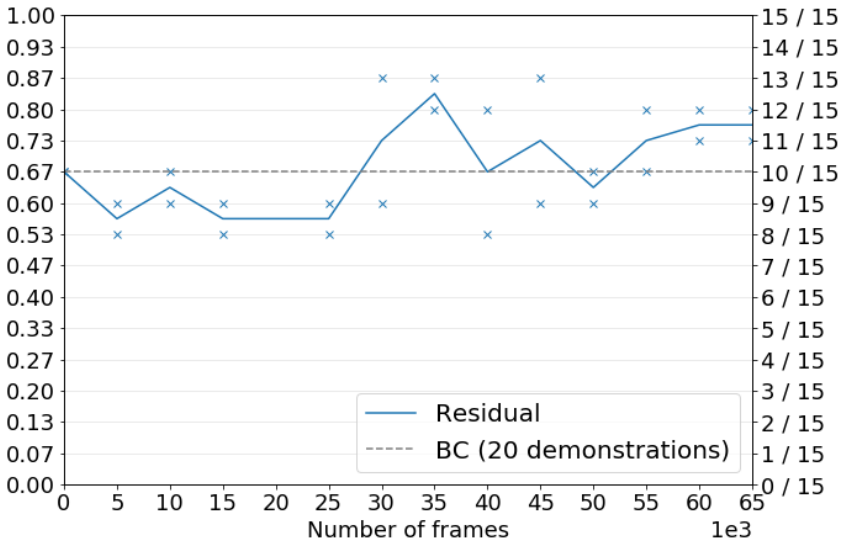


Figure 3.9: Success rates (out of 15) over initial training on Real UR5 Pushing. We plot the average performance over two repeated evaluations with the same starting positions (to the extent manually repeatable in practice).

15 initializations, shown in Figure 3.8a.

As the task does not require color vision, we convert the RGB camera images to grayscale, crop and downscale them to 128x128px for computational efficiency (Fig. 3.8b). All other training details and network architecture are unchanged from Sections 3.4.4 and 3.4.5.

Results for the BC stage are shown in Table 3.2. Including the gripper's xy-velocity in the proprioceptive state was found to improve pushing success.

After a relatively short training time (compared to Section 3.4.7) of 65,000 environment steps, we report an improved success rate of 77% for the residual policy starting from a 20-demonstration base policy with 67% success (Figure 3.9). 65,000 steps at a command frequency of 5Hz corresponds to 3h 37min of robot time, with an additional 3h 15min taken by scene resets. At this training length, the policy loss had not yet converged and it is likely that further training would lead to a higher performance increase,

but verifying this is left for future work.

3.5 Discussion

We have presented a novel method for combining demonstrations and RL to learn continuous control through a residual formulation. Our approach does not require base controller engineering, reward shaping or feature engineering as it is able to learn from readily available image and proprioceptive inputs and sparse rewards. We have empirically demonstrated sample-efficient training on robotic manipulation tasks in simulation, in a setting out of reach for RL from scratch with current algorithms.

As the base policy is treated as a black box during residual training, the proposed method is equally compatible with hand-engineered base controllers as with the BC policy considered in this work. Provided a sufficient quantity of demonstration data is available, using a BC policy as the base controller represents a more flexible choice that may be improved as more data becomes available. Data gathered on the system, which BC directly models, better represents true environment dynamics than is possible with first-order modelling in hand-engineered controllers without extensive manual tuning, especially in tasks with rich contacts and friction and in high-dimensional action spaces. In a low-data regime, however, and when accurate state estimation is feasible, using a classical controller remains a good alternative. In this setting as in the case of BC base policies, a residual in image space is a valuable addition as it can learn to correct for state estimation noise or omission of relevant state features. Task-specific visual features learned through BC on collected trajectories can equally be used in the classical residual RL setting to accelerate learning from images on the residual control task.

Chapter 4

Learning Reward Functions for Robotic Manipulation by Observing Humans

OBSERVING A HUMAN DEMONSTRATOR manipulate objects provides a rich, scalable and inexpensive source of data for learning robotic policies. However, transferring skills from human videos to a robotic manipulator poses several challenges, not least a difference in action and observation spaces. In this work, we use unlabeled videos of humans solving a wide range of manipulation tasks to learn a task-agnostic reward function for robotic manipulation policies. Thanks to the diversity of this training data, the learned reward function sufficiently generalizes to image observations from a previously unseen robot embodiment and environment to provide a meaningful prior for directed exploration in reinforcement learning. The learned rewards are based on distances to a goal in an embedding space learned using a time-contrastive objective. By conditioning the function on a goal image, we are able to reuse one model across a variety of tasks. Unlike prior work on leveraging human videos to teach robots, our method, Human Offline Learned Distances (HOLD) requires neither a priori data from the robot environment, nor a set of task-specific human demonstrations, nor a predefined notion of correspondence across morphologies, yet it is able to accelerate training of several manipulation tasks on a simulated robot arm compared to using only a sparse reward obtained from task completion. Videos of predicted rewards and the trained policies are included on the project website[1].

[1] https://sites.google.com/view/hold-rewards

4.1 Introduction

Deep learning has greatly advanced the state of the art in applications ranging from computer vision [He et al., 2016; Dosovitskiy et al., 2020] to natural language processing [Brown et al., 2020; Chowdhery et al., 2022] to speech recognition [He et al., 2019], but its significance in robotics has been blunted by limited access to large-scale data. Although previous efforts have attempted to sufficiently cover a specific embodiment and task [Levine et al., 2018; Fang et al., 2020], collecting a massive dataset for each robot and environment of interest is simply not feasible due to the cost of maintenance and human oversight, hardware wear and tear as well as the bottleneck imposed by real-time execution. For these reasons, creative reuse of data is of central importance for unlocking the benefits of powerful function approximation and data-driven learning in robotics.

One potential source of external data is videos of humans performing arbitrary tasks, widely available on the internet and inexpensive to produce. We focus on manipulation tasks in this work, with the aim of learning from crowd-sourced videos of human arms and hands. However, replicating the demonstrated actions and object interactions with a robot is a challenging open problem. On the perception side, there is a significant visual domain gap between observations of a person and of a robot. Human and robot arms usually have very different morphologies and dynamics, particularly in the end-effector, creating a physical domain gap and making a 1:1 mapping between poses ill-defined in general. Moreover, the actions taken by humans are not observed unless explicitly recorded with specialized equipment, and hence conventional imitation learning [Pomerleau, 1991; Ho and Ermon, 2016] or offline reinforcement learning [Kumar et al., 2019; Fujimoto et al., 2019] methods are not applicable.

To overcome these challenges, we investigate the use of videos of people solving manipulation tasks to learn a notion of distance between images from the observation space of a task. We leverage this learned distance as a reward signal on tasks with similar structure but very different visual appearance on a set of robotic manipulation domains that the model has never observed.

By training on diverse human demonstrations, we employ a strategy analogous to domain randomization [Tobin et al., 2017] used for sim-to-real transfer in robotics, which applies variations to visual and physical simulation parameters at training time so that a real-world robotic task with unknown physical properties is more likely to fall in the training distribution. Similarly, when trained with different demonstrators, backgrounds, viewpoints, lightings, objects and tasks, our distance model learns to generalize to a variety of manipulator appearances. Furthermore, several aspects of the task as

solved by a human are preserved in the robot workspace. For example, object displacements must respect the laws of physics regardless of the actor.

The learned distance function captures roughly how long it takes for an expert policy to transition from one state to another, and is therefore closely related to a dense reward function representing task progress that can be optimized with reinforcement learning (RL). Learning dense rewards is especially useful in hard exploration tasks where it is straightforward to define a sparse task-completion reward, but laborious and error-prone to specify a well-shaped dense reward.

In addition to model-free RL, reward functions estimating task progress can also be optimized with model predictive control [Chen et al., 2021a; Tian et al., 2021], in which case both a predictive forward model of the environment dynamics and a state-action value function need to be learned, typically from undirected exploration data in the target environment. However, extensive a priori data collection with sufficient coverage on a target robot environment and its action space are required for these methods to be applicable, and learning accurate video prediction models remains a challenging open problem in itself. We instead propose to learn a state-value function from observation-only data which allows for the reuse of data from different embodiments, and train a policy for the target embodiment with online RL. We empirically show better sample efficiency per task in online training than was required to learn a model in prior work [Chen et al., 2021a].

Our contributions are as follows:

i) We train HOLD, a global goal-conditioned distance model, which removes the need for demonstration task labels and exact alignment between robot tasks and demonstrated tasks required by prior work [Chen et al., 2021a; Schmeckpeper et al., 2020; Zakka et al., 2022; Shao et al., 2021].

ii) We show that time-contrastive embeddings [Sermanet et al., 2018] can successfully represent distances for multiple tasks at once despite a high degree of multi-modality in mixed-task training data.

iii) We show generalization of reward functions trained from unconstrained human videos to robot arms of various morphologies and environments.

iv) We demonstrate up to 18x accelerated training of model-free RL on 5 simulated manipulation tasks by either providing shaped rewards in sparse-reward tasks, or even entirely replacing the reward signal in some tasks.

v) We show our method to significantly outperform existing cross-domain imitation [Sermanet et al., 2018] and representation learning [Nair et al., 2022] approaches.

4.2 Related work

Intermediate representations Several prior works have addressed learning robotic policies from human videos via intermediate representations such as pose estimation or keypoint tracking [Qin et al., 2021; Petrík et al., 2020; Das et al., 2020]. In this work, our aim is to advance the capabilities of learning from raw video data, without depending on hand-crafted intermediate representations of human hands or an object database.

Imitation learning Our work is related to imitation learning from observation, although this line of work has mostly addressed the case of demonstrations from the same observation space [Ho and Ermon, 2016; Torabi et al., 2018; Aytar et al., 2018; Kostrikov et al., 2019]. We instead tackle the more difficult problem of inverse reinforcement learning from observation under significant observational and dynamical domain shift.

Offline RL Similarly to HOLD, Offline RL [Fujimoto et al., 2019; Wu et al., 2019; Wang et al., 2020; Peng et al., 2019] also aims to learn a value function from a dataset of existing trajectories. However, our setting is significantly different from the offline RL problem as we do not have access to either the actions or the rewards of the demonstrator in our dataset, nor do we have a forward model of which states are reachable from a given state, making temporal difference based methods not applicable.

Mapping methods Many methods for learning from videos seek to learn a direct mapping between demonstration videos and robot states and/or actions, such as an inverse model labeling each human transition with an action from the robot action space [Schmeckpeper et al., 2020], or an image-to-image translation of a human demonstration to a corresponding robot demonstration [Xiong et al., 2021; Li et al., 2021]. By contrast, our method does not assume a precise 1:1 mapping between the observation and action spaces of the human and the robot and can therefore leverage arbitrarily large amounts of human demonstration videos without any manual supervision cost.

Consistency methods A line of prior work has proposed to learn domain-invariant features capturing task progress regardless of whether the actor is a human or a robot arm [Schmeckpeper et al., 2020; Zakka et al., 2022; Sermanet et al., 2018] with reward defined as distance to a human demonstration [Sermanet et al., 2018] or to a goal state [Zakka et al., 2022] in the feature space. One issue with using geometrical distances is that transition times between states are not symmetrical if the environment includes

unidirectional transitions, such as dropping an object or knocking something down. To account for this, we also propose an alternative which predicts distances as a function of two ordered states.

Among learned embedding methods, sequence-based objectives such as temporal cycle consistency [Zakka et al., 2022] are well suited for single-task learning where all trajectories can be aligned along a global task progression, but it is unclear whether these approaches would work on data from several tasks. Our task-invariant distance function is instead able to fully take advantage of the diversity of hands, objects, backgrounds, lighting variations etc. across tasks, even when a grouping of demonstrations into distinct tasks is not available.

Most existing approaches to learning robotic manipulation from human videos also require exact overlap between tasks demonstrated by humans and the robot tasks [Chen et al., 2021a; Schmeckpeper et al., 2020; Zakka et al., 2022; Shao et al., 2021], and some require robot demonstrations for some of the same tasks [Chen et al., 2021a]. As our model is not specialized for any single task and learns from human data only, no robot demonstrations are needed and the target robot task does not need to be strictly included in the training data as long as a goal image is available to specify the new task.

Time-contrastive embeddings Sermanet et al. [2018] propose to use distances in an embedding space learned with a time-contrastive objective, but only consider reward learning for a single task, whereas we learn a single multi-task reward model. Moreover, while Sermanet et al. [2018] propose to directly imitate a human demonstration at 1:1 speed, we instead define the task with a goal image from the robot's observation space. As we show experimentally, the Time-Contrastive Network (TCN) of Sermanet et al. [2018] needs a nearly identical alignment in the initial states, execution speed and cropping between the video and the robot observations, which is a significant limitation. By contrast, our inverse RL approach requires less supervision and allows the robot to potentially outperform the demonstrator, either by executing the task faster or by finding a more optimal trajectory.

Functional distance Our work is also related to estimating functional (also called dynamical) distance between states from online [Hartikainen et al., 2020] or offline robot data [Tian et al., 2021]. Most related to our method, Tian et al. [2021] use an offline dataset and use estimated time-to-goal as a value function in a model predictive control loop. However, both works use only robot data from the same environment, without transfer of the action or observation spaces. Our approach is instead based on estimating the state-value function of the demonstrated behavior drawn from an unknown action space.

4.3 Human Offline Learned Distances

4.3.1 Functional distances from observation-only data

We propose to learn about distances in state space by observing humans and using this prior knowledge of environment dynamics to accelerate training of reinforcement learning policies on a robot manipulator. Specifically, our goal is to estimate *functional distance* $d(s, g)$, as defined by Tian et al. [2021], between an image s of the current state and a goal image g, where $s, g \in \mathcal{S}_r$, the set of camera observations from the robot's observation space. This metric should correlate with $\delta(s, g)$, the number of time steps it takes for an expert policy π^* to reach the goal g from the state s:

$$\delta(s, g) = \mathbb{E}[\, T \,|\, s_T = g,\ s_0 = s,\ a_t \sim \pi^*(s_t, g),\ s_{t+1} \sim p(s_t, a_t)\,], \tag{4.1}$$

where p are the transition dynamics of the environment, modeled as a Markov decision process (MDP). The negated time difference $-\delta(s, g)$ is equal to the value function V^* for an optimal policy π^* for the reward function

$$r(s, g) = \begin{cases} 0 & s = g \\ -1 & \text{otherwise.} \end{cases} \tag{4.2}$$

However, this is not the only reward function that can be optimized to recover π^*. In order to serve as a useful reward function for the task defined by g, the value of d does not need to perfectly correlate with δ; instead, pairwise rankings should be preserved for all states:

$$\delta(s, g) > \delta(s', g) \implies d(s, g) > d(s', g) \text{ for any } s, s', g \in \mathcal{S}_r. \tag{4.3}$$

Although defined in terms of an expert policy π^*, $\delta(s, g)$, and consequently the functions $d(s, g)$ that preserve its rankings, can be estimated from observation-only data, without access to actions a, the expert π^*, or even its action space, by obtaining self-supervised time deltas without manual annotation. While Tian et al. [2021] learn the Q-function corresponding to Eq. (4.2) from offline trajectories from the robot, our choice of a state-value function, agnostic to a specific action space, allows reuse of data gathered with different but related morphologies, such as other robots or humans. Strictly speaking, the ability to share the function δ between human and robot MDPs relies on them being isomorphic [Schmeckpeper et al., 2020], requiring a 1:1 mapping between the action and observation spaces that preserves dynamics p. While this may not fully hold in practice, and the distribution of δ in human data may not necessarily match the robot's dynamics in absolute terms due to embodiment differences, the

rankings produced by d can be transferred under fewer assumptions. For example, one embodiment may be twice as fast as the other while still preserving all pairwise rankings of states.

We assume access to a dataset of N video demonstrations of humans executing a variety of manipulation tasks using approximately shortest paths. In practice, the precise length of time may vary significantly across trials and human demonstrators, and depend on the optimality of the demonstration. Although the absolute length of such time intervals may not be consistent across demonstrators, their relative durations provide a useful learning signal; in order to push an object to the right, one must first approach its current position from the left before starting the pushing maneuver, and not the other way around. We present two methods for learning d on this data.

Direct regression (HOLD-R) We assume the demonstrations are optimal and pose the functional distance learning problem as a supervised regression task:

$$\theta^* = \text{argmin} \sum_{i=1}^{N} \sum_{t=1}^{T_i} \sum_{\delta=1}^{T_i - t} ||d_\theta(s_t^i, s_{t+\delta}^i) - \delta||_2^2 \tag{4.4}$$

where s_t^i is the tth frame of the ith video, T_i is the length of the ith video, and d_θ is a function parameterized by θ trained to predict δ from Eq. (4.1). The third summation corresponds to data augmentation allowing any future time step in the video to be considered the goal rather than only the last.

Time-contrastive embeddings (HOLD-C) Since directly predicting time intervals is difficult and sensitive to noise, we may also consider learning an embedding space where distances can be defined. We propose to use a time-contrastive objective as in TCN [Sermanet et al., 2018], adapted for single-view video. Frames within a small temporal window are encouraged to lie close together in embedding space, whereas embeddings for frames outside some temporal neighborhood are pushed apart. Specifically, if s_p is a positive instance for anchor s, and s_n is a negative instance, for all triplets, we want:

$$||f(s) - f(s_p)||_2^2 + m < ||f(s) - f(s_n)||_2^2 \tag{4.5}$$

where the margin m is a hyperparameter. However, unlike the single-task setup proposed in Sermanet et al. [2018], we train f on multi-task data and show it to accelerate robot learning across tasks. Moreover, our method improves upon TCN in several ways at the policy training stage:

i) HOLD enables the robot to outperform the demonstrations by learning relevant shortcuts through interaction, or by simply moving faster, whereas TCN aims to

imitate the human. TCN defines the task using a human video, and minimizes distance to each of its states at 1:1 speed – although the distances are minimized with RL, the best possible reward is defined as matching the human performance.

ii) HOLD requires less supervision: TCN needs one human trajectory of the full task whereas we use distance to a goal image only and require no task demonstrations.

iii) We use a simpler Euclidean distance to define the metric $d(s, g)$ in the space f, whereas Sermanet et al. [2018] apply a weighted mixture of squared Euclidean and a Huber-style loss

$$d(s_t, g_t) = \alpha||f(s_t) - f(g_t)||_2^2 + \beta\sqrt{\gamma + ||f(s_t) - f(g_t)||_2^2}, \tag{4.6}$$

requiring two additional hyperparameters (β and γ) to be tuned in an already computationally expensive RL training setup.

4.3.2 Policy learning

We propose to use the learned functional distance to define a dense reward function for an RL policy. Although our reward function is goal-conditioned and shared across tasks, we train one policy per robot task. As we want to minimize distance to the goal frame, we define reward as follows:

$$r(s_t, a_t, s_{t+1}, g) = -\max(0, d(s_{t+1}, g) - d(g, g))/T \tag{4.7}$$

where $s_t, s_{t+1}, g \in \mathcal{S}_r$, a_t is an action from the robot's action space, and T is an optional normalizer. We subtract the baseline $d(g, g)$, the prediction from the goal image to itself, from the distance estimates to ensure arriving at the goal has reward 0, and upper bound to zero to ensure no other state has higher reward; $d(g, g)$ may be positive for the regression models due to untrimmed training videos where the demonstrator idles after solving the task instead of terminating the demonstration immediately.

This definition of reward corresponds to minimizing the sum of distances until the end of the episode, as done by Tian et al. [2021] and Hartikainen et al. [2020]. Alternatively, r could be defined based on the difference $d(s_{t+1}, g) - d(s_t, g)$, such that only the reduction is maximized for each time step. However, we found the cumulative form to perform better empirically (details in Section 4.8), possibly due to being less sensitive to noise.

4.4 Experimental results

4.4.1 Distance learning

Dataset We train HOLD on Something-Something v2 (SSv2) [Goyal et al., 2017], a crowd-sourced dataset of 220,847 video clips of 174 action classes (with examples in Fig. 4.1). Each action is demonstrated with arbitrary objects, matching templates such as *Moving [something] closer to [something]*. The clips last 4 seconds on average and are mostly filmed using handheld devices, with non-negligible camera motion. Although SSv2 videos are grouped into discrete action classes, we do not make use of these labels[2], making our method applicable on any large-scale goal-oriented manipulation data. As we train a single goal image-conditioned distance function, there also does not need to be exact overlap between the demonstrated tasks and the target tasks on the robot, unlike in prior works Chen et al. [2021a], Schmeckpeper et al. [2020], Zakka et al. [2022], and Shao et al. [2021].

Training details We consider two sizes of network architecture: a ResNet-50 [He et al., 2016] and a Video Vision Transformer (ViViT) [Arnab et al., 2021] pretrained on SSv2 classification. As the single-view time-contrastive objective only supports embedding single images (as their immediate temporal neighbors are used as positive anchors), for HOLD-C we instead use either a ResNet or a Vision Transformer (ViT) [Dosovitskiy et al., 2020] pretrained on ImageNet-21K [Deng et al., 2009]. We train the ResNet models from scratch, and fine-tune the pretrained models on SSv2 without labels after replacing their classification heads. To adapt the pretrained ViViT model for regression, we also reinitialize its temporal position embeddings and shorten the temporal window to 4, including the 3 most recent frames and one goal frame. We also reduce the temporal filter dimension to 1 as there is no longer a computational benefit to shortening the sequence length. For time-contrastive training, we sample batches of 32 subsequent frames per video and use a positive window of 0.2 seconds and a negative window of 0.4s, as done by Sermanet et al. [2018]. For both objectives, we apply the same data augmentation procedure as Arnab et al. [2021], but leave out MixUp [Zhang et al., 2018]. For further training details, see Section 4.6.

We observed better policy training performance for the ResNet model for HOLD-C, and for ViViT for HOLD-R, so we report results using these backbones in Section

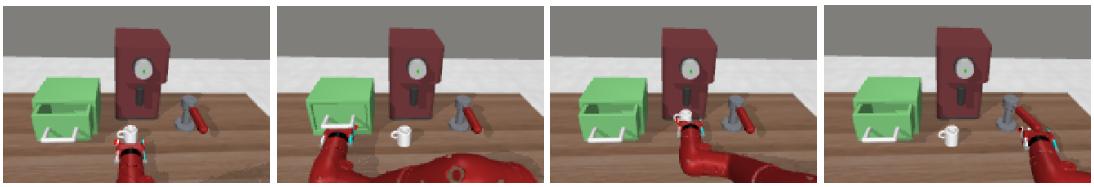
[2]Only HOLD-R with ViViT architecture made use of labels in pretraining, whereas HOLD-R with ResNet backbones as well as all HOLD-C models did not. However, the pretraining could have potentially been done on a different labeled dataset such as Kinetics [Kay et al., 2017] or skipped altogether, and no labels are needed for the regression task. We are working on evaluating a fully label-free ViViT regression model.

(a) Pushing mouse from left to right (b) Putting paint brush underneath magazine (c) Moving book up

Figure 4.1: Example human videos from Something-Something v2 used to train the distance models.

(a) Pushing start (b) Pushing goal (c) Drawer start (d) Drawer subtask (e) Drawer goal

Figure 4.2: The RLV tasks.



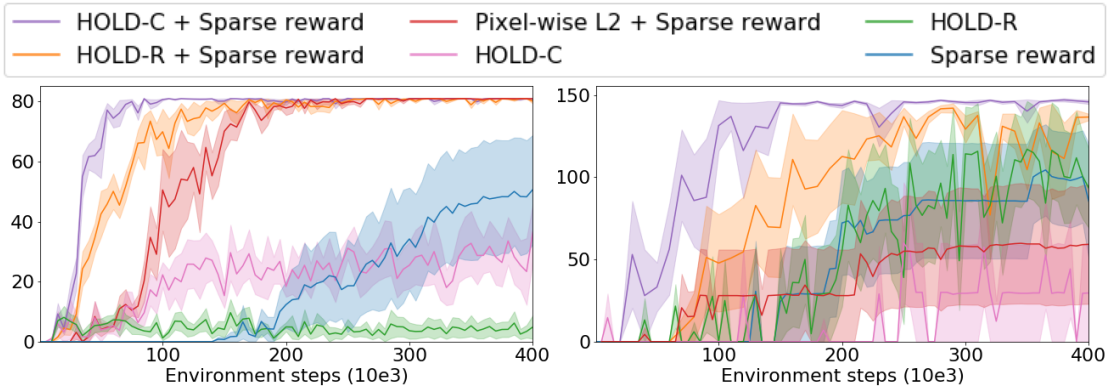
(a) Start state for all tasks (b) Close Drawer (c) Push Cup Forward (d) Turn Faucet Right

Figure 4.3: The DVD tasks: a Sawyer arm in a tabletop environment adapted from Meta-World [Yu et al., 2020].

4.4.2. Ablations using the other architectures in included in Section 4.8, and strategies for evaluating the distance models on human data before testing them in robot policy training are discussed in Section 4.7.

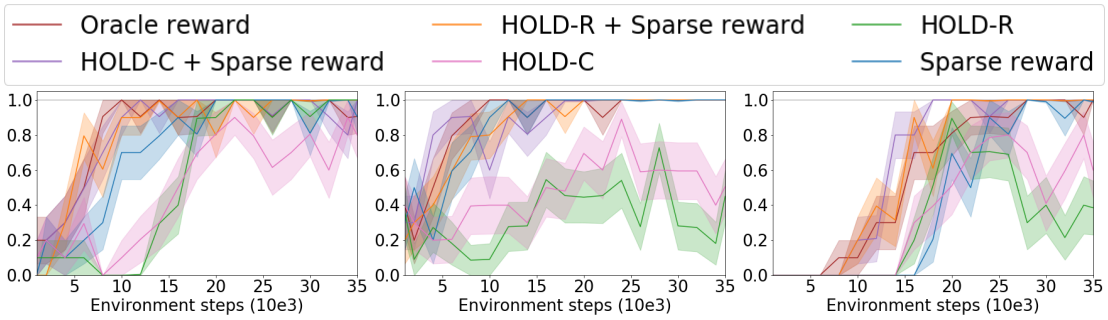
4.4.2 Policy learning

To demonstrate the utility of our method as a reward function for training RL policies, we evaluate it on the Pushing and Drawer Opening tasks defined by RLV Schmeckpeper et al. [2020] (Fig. 4.2) and on the Close Drawer, Push Cup Forward and Turn Faucet Right tasks defined by DVD [Chen et al., 2021a] (Fig. 4.3). We follow prior work [Schmeckpeper et al., 2020; Zakka et al., 2022] in using Soft Actor-Critic (SAC) [Haarnoja et al., 2018a] as the underlying RL algorithm and evaluate it on 20 episodes for all tasks. All policies use images as input, and we reuse the policy and critic architectures as well as algorithm hyperparameters from Schmeckpeper et al. [2020].



(a) Pushing (b) Drawer Opening (2 subtasks)

Figure 4.4: Return on the RLV tasks (5 random seeds, with standard error).



(a) Close Drawer (b) Push Cup Forward (c) Turn Faucet Right

Figure 4.5: Success rates on the DVD tasks (10 random seeds, with standard error). Our reward functions improve over sparse reward, and learn the Close Drawer task without using sparse reward.

Like Schmeckpeper et al. [2020], we augment our learned reward from Eq. (4.7) with a sparse task reward: +1 for success, and 0 otherwise, defined by each environment based on distance to the target configuration. Since the predicted distances can be significantly larger than 1 but should not override the sparse reward, we scale the predicted rewards by $1/T$, where T is set such that the scale of initial state distances is $\approx 1/3$. Ablations for other values are included in Section 4.8.

RLV tasks

As shown in Fig. 4.4, the sum of both reward functions, appropriately balanced, significantly accelerates training compared to using the sparse reward alone. In our experiments, using only sparse reward required 10x more samples for Pushing and >18x more for Drawer to reach the return of HOLD-C. We find that HOLD-C outperforms HOLD-R overall for Pushing, both with and without sparse reward, and in the sparse

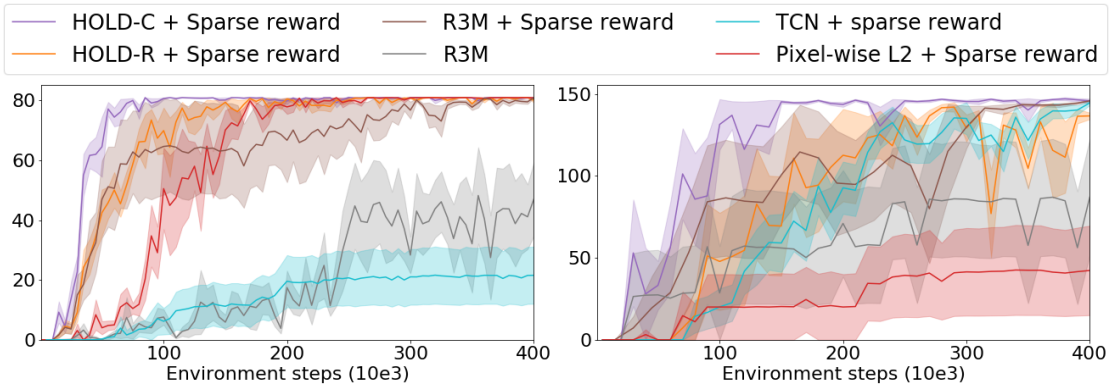
reward setting for Drawer Opening.

Pushing Without added sparse reward, a single failure case is prominent: while the policy quickly learns to match the end-effector position in the goal frame, it fails to pay attention to the puck position. As observed by Tian et al. [2021], it is easy for the distance function to excessively focus on fully actuated components in the scene as these are highly predictive of temporal offset. Although HOLD is able to generalize from human arms to a robot arm, for tasks with variable object positions, it may be better suited as an exploration strategy used together with an otherwise rarely-observed sparse reward than a standalone multi-task reward. Note that although Zakka et al. [2022] also evaluate on Pushing, their results are not comparable as their method is trained on the easier RLV Pushing dataset [Schmeckpeper et al., 2020] collected to match the appearance of the robot task, and report on the simpler State Pusher task where the policy directly observes the 2D puck position and the 3D end-effector position.

Drawer The Drawer Opening task has double the episode length (200 steps) of Pushing, and consists of two distinct motions: approaching and inserting the gripper into the handle, and pulling the drawer open once there. We find that applying the HOLD models on the full task suffers from the local minimum of only matching the arm position in the goal image. However, if we instead define the task in two parts using an intermediate goal image (in Fig. 4.2d), our rewards significantly improve sample efficiency compared to the sparse task-completion reward provided by the environment alone, as shown in Fig. 4.4b. Moreover, HOLD-R alone without any environment reward performs on par with sparse reward in this setting. We train a single policy for both subtasks, which is conditioned on the active goal image by concatenating it to the observation s. For all distance functions, we switch to the next subtask when $d < 1$ for at least 3 consecutive time steps.

DVD tasks

We report success rate for the DVD tasks in Fig. 4.5. These tasks are significantly easier than the RLV tasks and quickly learned even using only sparse reward. To estimate the upper bound in learning speed achievable by improving the reward alone, we define an oracle reward using knowledge of robot and object positions. Since we observe only a narrow performance gap between the oracle and the sparse reward, the learning speed in these tasks is limited mostly by the RL algorithm. Although it is therefore difficult to show much improvement over the sparse reward, both HOLD models outperform it, particularly for Close Drawer and Turn Faucet Right. For Close Drawer, HOLD also solves the task without sparse reward. Unlike Chen et al. [2021a], we do not first collect



(a) Pushing

(b) Drawer Opening (2 subtasks)

Figure 4.6: HOLD-C outperforms TCN and distances in R3M representation space on both RLV tasks.

a dataset of 10,000 trajectories, or 600,000 steps, of random exploration on the robot to learn a model of the environment, but instead focus on the model-free setting. We show adaptation to a new robot, set of objects and environment in just 12,000–18,000 steps, or 200 to 300 episodes, when sparse environment reward is available, or 22,000 without sparse reward for Close Drawer.

4.4.3 Baseline comparisons

We compare HOLD to rewards defined by two prior methods: TCN [Sermanet et al., 2018] and R3M [Nair et al., 2022]. TCN proposes to transfer a policy given a human video demonstration by minimizing distance to the embeddings of each of the visited states g_t in turn. We empirically set the hyperparameters of $d(s_t, g_t)$ to $\alpha = 0.005, \beta = 0.02$, and $\gamma = 0.2$. As performance may vary based on the exact demonstration video used, we evaluate 3 demonstrations per task from the RLV dataset, which is collected to closely match the RLV robot tasks, and report the average performance across demonstrations (trained with 5 seeds each) in Figure 4.6. Even the closely aligned demonstrations transfer poorly to policy learning, especially for Pushing, due to slight differences in initial state, cropping or execution speed, highlighting the brittleness of the trajectory-following objective of TCN.

Although R3M is proposed as a general feature representation, we also compare against using Euclidean distance in the representation space for defining dense rewards. We used the ResNet-50 model checkpoint from Nair et al. [2022], trained on the much larger Ego4D [Grauman et al., 2022] (3,500 hours) rather than SSv2 (200 hours). As shown in Figure 4.6, HOLD-C outperforms R3M in both RLV tasks despite having been trained on less data and requiring no language descriptions. Like our method,

R3M also requires sparse rewards to fully solve the tasks, and an intermediate goal for Drawer opening.

We also include a simple baseline of using the negative pixel-wise distance in image space between the current observation and the goal image as a reward. While using distances in image space with sparse reward also learns the Pushing task faster than sparse reward alone, as shown in Figure 4.6, this still requires many more training samples than either HOLD-R or HOLD-C, and fails to reliably learn the Drawer task.

4.5 Limitations

Long-horizon and non-Markovian tasks In the robot learning literature, several ways of defining tasks have been proposed. Some task may be best described in natural language, while for others showing how the task should be performed, either as a still frame or a full demonstration video, may better resolve any ambiguity in which objects should be manipulated and how. While a complete demonstration is more laborious to provide, especially in robot observation space, goal images may not be able to represent certain tasks. One limitation is that task progress must be fully observable, i.e. the task must be Markovian, to be describable with a {start image, goal image} pair.

As our distance function in its general definition is not specialized to any specific tasks or objects, it is very difficult for it to predict what should happen in long-horizon tasks, or in order to produce particular object deformations as demonstrated by the RLV Drawer Opening task. However, this limitation could be overcome either by defining a hierarchical model capable of producing its own subgoals, or by reintroducing some task specialization, such as human data focused on, say, maneuvers involving handles of drawers and doors.

4.6 Training hyperparameters

Our distance models are implemented in JAX [Bradbury et al., 2018] using the Scenic library [Dehghani et al., 2021]. Hyperparameter settings are shown in Table 4.1 for the regression models and in Table 4.2 for time-contrastive training.

For policy training, we reuse the implementation of SAC from Schmeckpeper et al. [2020] based on Softlearning Haarnoja et al. [2018b]. All RL hyperparameter settings are unchanged (included in Table 4.3 for reference).

Parameter	ViViT	ResNet-50
Epochs	20	100
Base learning rate	0.1	3e-4
Optimizer	Momentum	Adam
Batch size	64	32

Table 4.1: Training hyperparameters for HOLD-R.

Parameter	ViT	ResNet-50
Epochs	5	100
Sequence length	32	32
Base learning rate	1e-4	1e-4
Optimizer	Adam	Adam
Batch size	8	8

Table 4.2: Training hyperparameters for HOLD-C.

Parameter	Value
Initial exploration steps	1000
Learning rate	3e-4
Batch size	256
Optimizer	Adam
Gradient steps per environment step	1

Table 4.3: Training hyperparameters for policy training.

4.7 Distance model evaluation on human data

To avoid evaluating every variation of HOLD in the target robot environment, it would be preferable to be able to rank and pre-select models based on their performance on held-out human data, and test only the most promising ones in robot policy training. However, it is difficult to evaluate generalization without access to robot data, and it is not straightforward to design a suitable test metric that captures both smoothness and correct ranking of states. In this section, we propose several relevant metrics.

For the regression models, in addition to the training objective mean squared error (MSE), we can also evaluate mean absolute error in time steps and in seconds. However, these metrics assume uniform progress at each time step toward task completion, and require high-scoring models to match the scale of ground truth time intervals. Using a hinge loss instead allows non-uniform progress and only penalizes out-of-order

Loss	Network; frames	Spearman	Misclassif.	MSE	Mean error	Hinge loss
R	ViViT; 3	0.6709	0.4539	499.2	18.0 (1.54 s)	0.0663
R	ResNet-50; 1	0.7136	**0.3976**	**482.1**	**17.3 (1.48 s)**	**0.0233**
R	ResNet-50; 3	**0.7139**	0.4385	514.1	18.1 (1.55 s)	0.0611
C	ResNet-50; 1	0.6246	0.4015			
C	ViT; 1	0.6559	0.4006			

Table 4.4: Evaluation scores on the Something-Something v2 validation set. Time-based metrics are only defined for HOLD-R as HOLD-C models do not predict time.

predictions:

$$\mathcal{L}_h = \sum_{i=1}^{N} \sum_{j,k=1}^{T_i-1} \max(0, d(s_k^i, s_{T_i}^i) - d(s_j^i, s_{T_i}^i))\, \mathbb{I}[j < k] \,/\, \sum_{i=1}^{N} (T_i - 1). \tag{4.8}$$

Another option is to not use time-based metrics at all. As explained in Section 4.3.1, it is ultimately more important for the distance models to preserve the ranking of states with respect to a goal frame than to reproduce δ in absolute terms. With the aim of maximally preserving pairwise rankings as defined in Eq. (4.3), we propose two further metrics, namely misclassification rate:

$$\mathcal{L}_{miscl} = \sum_{i=1}^{N} \sum_{j,k=1}^{T_i-1} \mathbb{I}[d(s_k^i, s_{T_i}^i) > d(s_j^i, s_{T_i}^i)] \mathbb{I}[j < k] \,/\, \sum_{i=1}^{N} (T_i - 1), \tag{4.9}$$

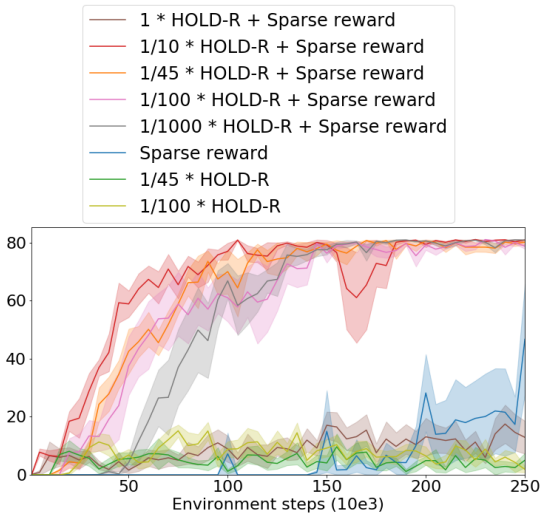
and Spearman correlation, i.e., the correlation between rankings assigned to each frame in the full sequence $s_{1:T_i-1}$, and the ground truth order.

The scores of each of the models we present are shown in Table 4.4. As we assume no access to robot data at distance training time, we use the SSv2 validation set as a proxy for model performance, and use Spearman correlation as an early stopping criterion. However, we observe that the scores on human data are not predictive of the downstream robot performance these models obtain, highlighting the difficulty of the domain transfer.

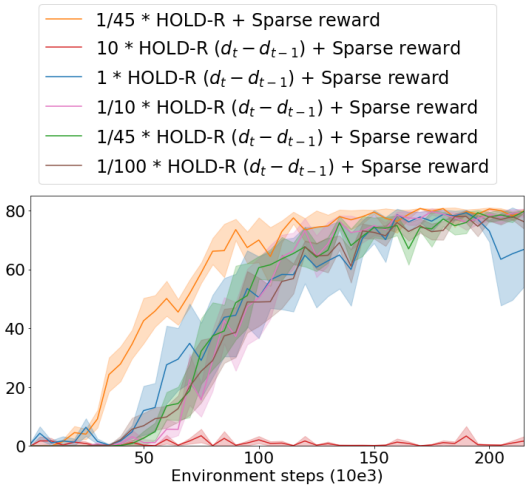
4.8 Distance model ablations

We evaluate a variety of design choices in HOLD models on the RLV Pushing task. Specifically, we compare several values for the reward normalizer T introduced in Eq. (4.7), as well as variants of the network architecture and the form of the reward (cumulative vs. instantaneous, as described in Section 4.3.2).

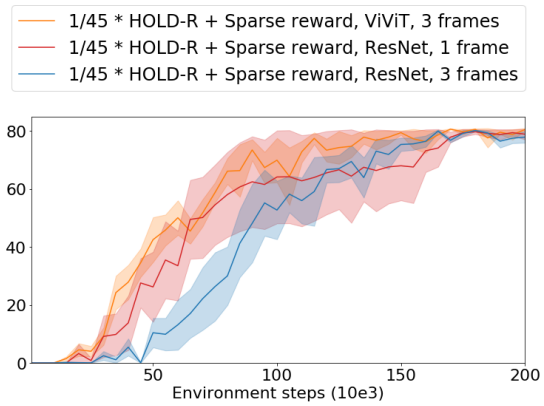
The effect of reward scale for HOLD-R is shown in Fig. 4.7a. While $T = 10$ increases return the fastest, its performance is less stable towards the end of training



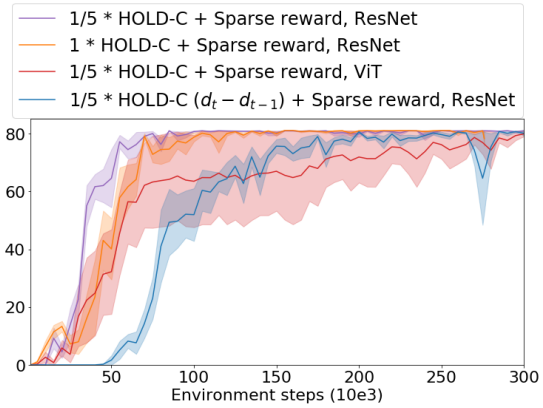
(a) Effect of reward scale for HOLD-R.



(b) Cumulative distance vs. subtracting the previous distance for HOLD-R.



(c) Effect of architecture choice for HOLD-R.



(d) Effect of reward definition, scaling and architecture choice for HOLD-C.

Figure 4.7: HOLD model ablations on RLV Pushing.

and it suffers a momentary drop in performance when the policy appears to overfit to the distance reward over the sparse task reward. The normalizer $T = 45$, equal to the average length of a training video in SSv2, provides the best trade-off in sample efficiency and stability and we therefore report results using this setting in Section 4.4.2. This value is used for all tasks except RLV Open Drawer, where the task horizon is twice as long and we found $T = 100$ to work significantly better (see Fig. 4.8). To avoid further extensive tuning of the reward scale for HOLD-C and for each baseline, we simply set T such that the scale of initial predictions is approximately $1/3$, the same scale as HOLD-R with $T = 45$. Using this strategy, we obtain $T = 5$ for HOLD-C and $T = 30$ for the L2 baseline (or $T = 10$ and $T = 100$ for RLV Drawer, respectively),

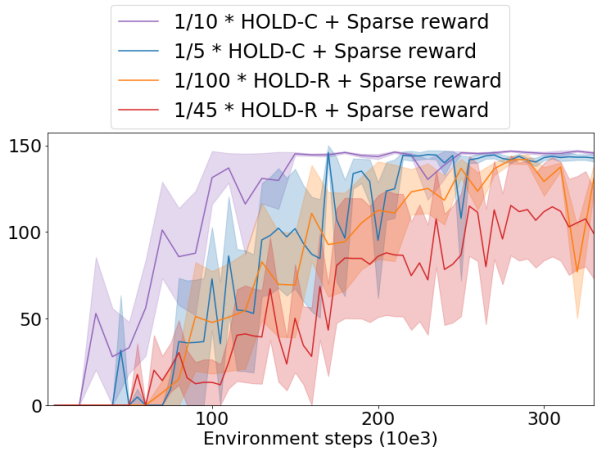


Figure 4.8: Effect of reward scale on RLV Open Drawer.

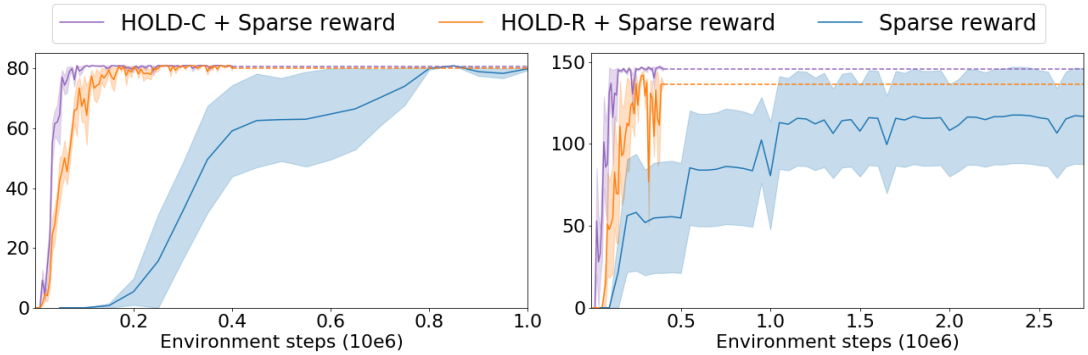
which we indeed found to perform better than respective alternatives $T = 1$ and $T = 45$.

As for reward definition, the cumulative distance reward clearly outperforms instantaneous distance reward (i.e. subtracting the distance at the previous time step) for both HOLD-R (Fig. 4.7b) and HOLD-C (Fig. 4.7d). Although the scale of $d_t - d_{t-1}$ is expected to be different from the scale of d_t and hence the normalizer T may need to be set differently, we found the HOLD-R instantaneous distance form to perform consistently worse for a wide range of values of T: $\{0.1, 1, 10, 45, 100\}$. Moreover, the choice of T seemed to have very little effect on the learning performance for $T \geq 1$.

Finally, in Fig. 4.7c, we compare the HOLD-R ViViT model against ResNet-50 conditioned on either 1 or 3 frames, but found that these smaller models had slightly worse sample efficiency in RL training than ViViT. For HOLD-C models, we found ResNet to outperform ViT, however, ViT may have benefited from longer training. In Section 4.4.2, we therefore report HOLD-C results using the ResNet architecture.

4.9 Longer training for sparse reward

HOLD-C converges to a return of 80 for Pushing after 80,000 environment steps of training and a return of 145 for Drawer after 150,000 steps. In order to estimate how much training time is accelerated compared to using only the sparse reward, we also run experiments with considerably longer training for the sparse reward baseline. The return of HOLD is eventually reached after 800,000 samples for Pushing, whereas for Drawer, it is not reached even after 2.75 million steps. We therefore obtain a speedup of 10x for Pushing (Fig. 4.9a) and at least 18x for Drawer (Fig. 4.9b).



(a) Pushing (b) Drawer Opening

Figure 4.9: Eventual return for the sparse reward on the RLV tasks after 1–2.75 million environment steps (2.5–7x increase from Fig. 4.4).

4.10 Training data coverage of evaluated robot tasks

Our distance models are not specialized for any specific task and can therefore be applied on previously unseen manipulation tasks, or tasks with very few human demonstrations. In this section, we investigate to what extent the robot tasks we evaluate on are covered in SSv2 training data. Note that our models never observe the task labels and are only trained on ungrouped SSv2 videos.

The tasks included in SSv2 are intentionally very diverse. As task templates include generic movements such as *Moving something up*, *Moving something down*, *Pushing something from left to right*, *Pushing something from right to left*, it is genuinely difficult to find manipulation tasks unrelated to any of the 174 templates. However, several tasks include drastically different manipulations depending on the objects considered: e.g., opening the screw cap of a bottle and opening a book use the same template *Opening something* but very different motions. As shown in Table 4.5, the action-object pairs we evaluate in the robot tasks have never been demonstrated for Turn Faucet Right, and have been demonstrated <5 times for Push Cup Forward and RLV Pushing (*Moving a cap towards the camera*). The objects *puck* or *disk* do not appear in SSv2, so we replace *puck* by *cap*.

Moreover, on closer inspection of the three videos labeled *Moving cap towards the camera*, we observe significant variation in the interpretation of the action labels themselves. Instead of pushing an object along a surface like in our robot task, one out of three videos in fact shows pulling a (baseball) cap along a surface, and the remaining two show a person holding a bottle cap and moving it directly towards the camera lens without using a surface at all. The same two motions are demonstrated for related objects such as *lid* (2 videos) and *bottle cap* (1 video). It seems unlikely that the same

Robot task	Closest action *a*; Closest object *o*	*#a*	*#o*	*#(a, o)*
Pushing	Moving sth towards the camera; cap	927	611	3
Open Drawer	Opening sth; drawer	1585	619	67
Close Drawer	Closing sth; drawer	1296	619	62
Push Cup Forward	Moving sth away from the camera; mug	937	951	4
Turn Faucet Right	Pushing sth from left to right; faucet	3199	28	0

Table 4.5: SSv2 training examples most closely matching the evaluated robot tasks. *a* corresponds to the most similar action template to each robot task, whereas *o* is the most similar object among the objects manipulated across all tasks templates. In columns 3–5, we give the number of videos in the training and validation sets labeled with either *a*, *o*, or both, respectively. The full dataset consists of 220,847 videos: 168,913 in the train set, 24,777 in the validation set and the remaining 27,157 in the test set.

task, *pushing* a puck towards the camera is demonstrated at all (the pushing templates featuring horizontal directions only). Similarly, only 1/4 videos labeled *Moving mug away from the camera* and 1/4 videos labeled *Moving cup away from the camera* push an object along a surface at all, and the rest perform the maneuver in the air.

We conclude that we have shown generalization to at least one and possibly multiple novel tasks which were not included in the training dataset.

4.11 Conclusion

We have presented a method for learning goal image conditioned reward functions for robotic manipulation from unlabeled human videos, in a challenging setting which no prior work has addressed to our knowledge. Learning a prior for robot behavior from a dataset of human demonstrations without task labels requires generalization both across tasks and across a significant domain shift. While most accurate for short-horizon tasks with single-step movements, the distance functions we train produce useful rewards for visually different robot environments that are able to accelerate training over using sparse reward alone, and can be composed to perform more general multi-step manipulation tasks using subgoals. Finally, we have shown that for some tasks, the predicted rewards alone are sufficient to learn the task without any additional success signals.

Chapter 5

Discovering Actions by Jointly Clustering Video and Narration

TUTORIAL VIDEOS ON THE INTERNET are a rich source of supervision for teaching robots to accomplish goals and to interact with objects. In this chapter, we address the problem of discovering actions in narrated instruction videos by jointly clustering visual features and text, without any external supervision. Eventually, the predicted action classes and extracted intervals could be used as training data for robotic policies, for learning both individual skills and their composition for longer tasks.

Unlike previous work, our method does not assume any prior grouping of the videos into distinct tasks, or that videos depicting the same task share an identical script, i.e., the same sequence of actions. In this work, only the narration and the visual stream of each individual video are assumed to be mutually consistent and depict the same sequence of actions with approximate temporal alignment. Our method is based on a discriminative clustering objective, with a penalty term corresponding to the distance between the sequence of actions assigned to frames and that assigned to words in each video. This encourages the order and timing of actions in each assignment to become consistent over the course of optimization. Our experimental evaluation on the Inria Instruction Videos and CrossTask datasets shows that our method achieves comparable performance to existing task-specific methods while greatly improving on their generality by relaxing the assumption of a shared script, and by requiring less supervision.

5.1 Introduction

Tutorial videos are a varied and abundant source of data for machine learning systems [Alayrac et al., 2016; Elhamifar and Zaing, 2019; Miech et al., 2019; Tang et al., 2019; Zhukov et al., 2019]. However, full temporal annotation of video datasets sufficiently

large for supervised action recognition comes at a significant labeling cost, which limits the amount of progress to be made by fully supervised methods. This motivates efforts to learn from instruction videos using weaker forms of supervision, or no external supervision at all. One widely available source of weak supervision is video narration, the usefulness of which has grown in recent years thanks to notable improvements in automatic speech recognition quality.

In this chapter, we address the problem of extracting from video tutorials the corresponding procedure steps, using only weak supervision from narration. This problem is commonly addressed in the case where many tutorials for the same activity (e.g., changing a tire) are available [Alayrac et al., 2016; Elhamifar and Zaing, 2019; Sener and Yao, 2018]. We consider here instead the case where many tutorials for different activities that may contain shared components are available, and cast our problem as one of joint clustering of textual and video information into a finite set of actions. Contrary to the single-task setting, temporal ordering constraints are not available for the text data alone, since we may only have a single tutorial instance for each activity of interest. As opposed to prior work, our method does not require any prior grouping of the tutorials, such as which task is demonstrated. Since the narration can also be transcribed without human input, using automatic speech recognition and parsing, our method can be applied on large, automatically sourced datasets, without any labeling cost.

Our approach to the action discovery problem is more general than existing methods, as we do not assume all videos depicting a given activity, such as *build a desk*, consist of a shared sequence of steps, always performed in an identical order. Moreover, our method returns a full segmentation of the video, unlike most prior work, where exactly one time step per action per video is labeled—even if a step may be missing. This also allows our method to identify repetitions of the same action, which most approaches focused on shared sequences do not permit.

The rest of this chapter is structured as follows. In Section 5.2, we discuss related work on weakly supervised action discovery in videos. The problem of learning label assignments on frames and words is defined in Section 5.3.1. Sections 5.3.2 and 5.3.3 outline the constraints connecting the two assignments, and how they are relaxed during optimization. We describe in Section 5.4 our optimization procedure, where each intermediate assignment of frames to actions is informed by, but not fully dictated by, its counterpart word-to-action assignment, and vice versa. Finally, our experimental evaluation is described in Section 5.5, with conclusions in Section 5.10.

Figure 5.1: Using only weak supervision from narration, we automatically discover actions that might occur across different tasks and contexts—without assuming the task depicted, such as the recipe label, is known.

5.2 Related work

5.2.1 Weakly supervised learning of actions in video

Learning to recognize actions in videos is an active area of research. However, temporally labeling a video in the weakly supervised setting is a difficult problem due to its combinatorial nature. To constrain the space of label assignments considered, many examples of prior work assume an ordered list of actions to be available, obtained for instance from a transcript or a movie script [Bojanowski et al., 2014]. Using unordered meta-tags, Richard et al. [2018] consider the slightly more difficult problem where the set of actions is known, but their order is not.

Instructional videos, on the other hand, exhibit common structure that allows the requirement of an ordered sequence of actions to be known a priori to be relaxed. When videos are split into distinct tasks, and it is assumed that each task is described by a global script, the shared order can be learned from data [Alayrac et al., 2016; Sener et al., 2015]. As most video tutorials also have one or more narrators that describe the steps being performed, the transcribed narration provides further constraints on the action timings.

However, the prior works on instruction videos mentioned above have focused on learning from a single activity at a time. This requires a grouping of the videos of interest into known activities, such as which recipe is being depicted. Moreover, it is not always the case that all videos depicting the same activity follow a shared sequence of actions: steps may be skipped, shuffled or repeated. Out of the prior work in weakly supervised action classification, Richard et al. [2018] and Elhamifar and Zaing [2019] are among the few to allow repetitions and shuffles, and Elhamifar and Zaing [2019] additionally allows for missing steps.

Also closely related to our work are methods that segment videos into a set of actions [Elhamifar and Zaing, 2019; Kuehne et al., 2017; Sener and Yao, 2018], instead of simply identifying the single most salient time steps. However, unlike our approach, they model each task separately of others, and do not use narration, as we do in this work. Kukleva et al. [2019] additionally consider the problem of separating an ungrouped dataset into tasks, but also learn task-specific action classifiers.

5.2.2 Sharing across tasks

Learning from a single activity at a time also overlooks the opportunity to jointly learn action models across tasks. A single cooking step may appear in many recipes, or a single assembly step might appear in many home DIY tutorials, and learning from all available data is likely to lead to better classifiers. Sharing across activity categories has been approached in the form of a component model [Zhukov et al., 2019]: two steps such as *pour coffee* and *pour mixture* share the verb *pour*, for which a classifier is trained. Similarly, a classifier is learned for each word in any step description. The sharing is directly determined by a common sequence of steps for each activity, known a priori, and is dependent on the exact phrasing of these steps. In this work, we take advantage of sharing on a general set of videos without assuming that an ordered sequence of steps is known.

5.3 Proposed model

5.3.1 Objective function

We consider a dataset of N narrated videos. Each video i is made up of m_i frames, each represented by some feature in $\mathbb{R}^p$, such that the video itself can be represented by a matrix X^i in $\mathbb{R}^{m_i \times p}$. We assume that a narration made of n_i words, each represented by a feature in $\mathbb{R}^q$, is attached to the video, and similarly represent it by Y^i in $\mathbb{R}^{n_i \times q}$. Here, the terms *frame* and *word* correspond to any representation of visual and textual

content of a video: a *frame* may actually consist of a block of frames, and include some motion or even sound information. Likewise, a *word* may also designate a direct object relation (verb + object) or even a sentence, and a *script* may designate an actual script, subtitles, or a text transcription of an audio commentary. Where available, we also take advantage of an approximate alignment of the frames that overlap with a given word and vice versa, represented by a binary matrix A^i of size $m_i \times n_i$. In the case of transcripts obtained from automatic speech recognition, the timings of captions are obtained without any labelling cost.

We wish to automatically discover k action classes occurring across the dataset, using only weak supervision in the form of the transcribed narration. Each word and each frame may be associated to one of k actions or to a background class. The corresponding assignments can respectively be represented by $m_i \times K$ and $n_i \times K$ binary matrices P^i and Q^i for each video i (see Figure 5.2), where $K = k + 1$ in order to include a column for background. The label assignments must respect certain constraints (described in Sections 5.3.2 and 5.3.3), which we denote here by the fact that the ordered pair (P, Q) must belong to some constraint set $\mathcal{C}$ in $\{0,1\}^{m\times K} \times \{0,1\}^{n\times K}$, where $P = [P^1; ...; P^N]$ and $Q = [Q^1; ...; Q^N]$ are the stacked assignments, $m = \sum_i m_i$ and $n = \sum_i n_i$.

Given N videos and their associated scripts, we can now pose the problem of learning the assignments P and Q as the minimization of a discriminative clustering [Bach and Harchaoui, 2008] objective:

$$\begin{aligned} h(P,Q) = \min_{\substack{U\in\mathbb{R}^{p\times k},\\ V\in\mathbb{R}^{q\times k},\\ a,b\in\mathbb{R}^k}} & \frac{1}{2m}\left\|P - XU - \mathbb{1}_m a^T\right\|_F^2 + \frac{\alpha}{2}\|U\|_F^2 \\ & + \frac{1}{2n}\left\|Q - YV - \mathbb{1}_n b^T\right\|_F^2 + \frac{\beta}{2}\|V\|_F^2 . \end{aligned} \tag{5.1}$$

where X in $\mathbb{R}^{m\times p}$ and Y in $\mathbb{R}^{n\times q}$ are the features X^i and Y^i stacked for each video i, (U, a) and (V, b) are linear (or rather affine) classifiers, the minimization over which can be done in closed form. This gives a quadratic program in P and Q:

$$\min_{(P,Q)\in\mathcal{C}} h(P,Q) = \min_{(P,Q)\in\mathcal{C}} f(P) + g(Q), \tag{5.2}$$

$$\text{where } f(P) = \frac{1}{2m}\operatorname{Tr}(PP^TB), \tag{5.3}$$

$$g(Q) = \frac{1}{2n}\operatorname{Tr}(QQ^TC), \tag{5.4}$$

$$B = I_m - X(X^TX + m\alpha I_p)^{-1}X^T, \text{ and} \tag{5.5}$$

$$C = I_n - Y(Y^TY + n\beta I_q)^{-1}Y^T. \tag{5.6}$$

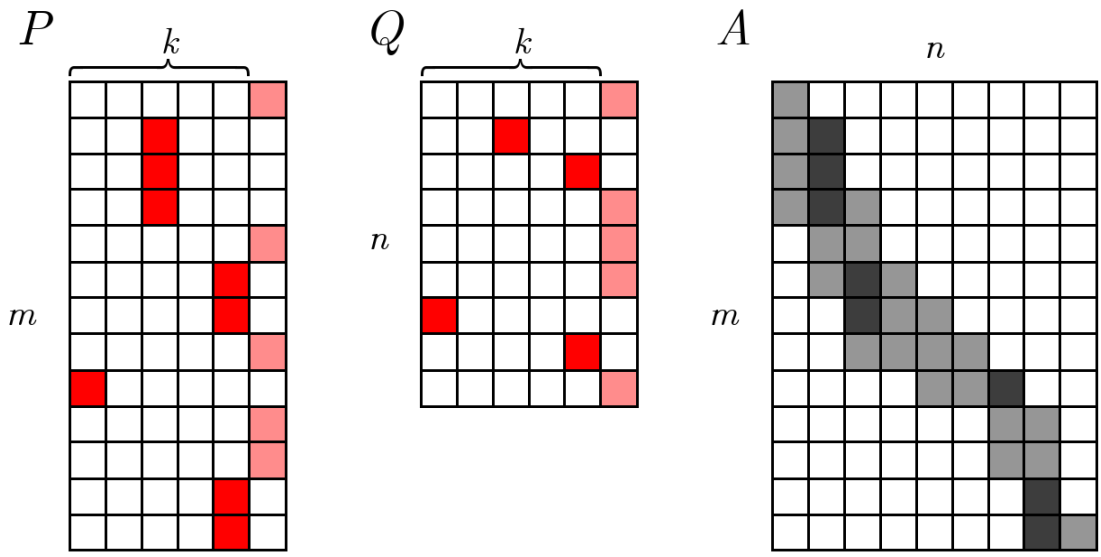


Figure 5.2: Illustration of the indicator matrices considered, shown here for a single video. P (resp. Q) assigns frames (resp. words) in the video to one of k classes, or to no class at all. A encodes an approximate temporal alignment between P and Q (in gray). The true (and a priori unknown) aligned frame-word pairs depicting the same action instance are shown in black for illustration only.

5.3.2 Separate constraints

Let us consider for the remainder of this section a single video and omit the corresponding superscripts i.

Non-overlapping actions Each frame and each word may be assigned to a single action, or no action (background):

$$P\mathbb{1}_K = \mathbb{1}_m; \quad Q\mathbb{1}_K = \mathbb{1}_n. \tag{5.7}$$

Number of actions per video A common problem with discriminative clustering is that it suffers from the trivial optima corresponding to a single cluster, as well as the constant assignment matrix $\frac{1}{K}\mathbb{1}\mathbb{1}_K^T$. This is known to happen when the optimization domain is symmetric over permutations of the label classes [Guo and Schuurmans, 2008], as is the case in our setting. To rule out the solution of a single cluster, we would like to set minimum and maximum proportions for any cluster. However, since we apply the Frank-Wolfe algorithm to our optimization problem and use a dynamic program to solve the linear minimization oracle (more details in Section 5.4), and the proportion constraint cannot be incorporated into a dynamic program, we must relax this constraint. We instead add an entropy regularizer (as done in Joulin et al. [2012]) on the distribu-

tion of action labels summed over the time dimension (frames or words, respectively) to our objective:

$$h(P, Q) = f(P) + g(Q) - \rho_P H(P) - \rho_Q H(Q), \tag{5.8}$$

$$\text{where } H(P) = -\sum_{c=1}^{K} \frac{\sum_j P_{jc}}{\sum_{j,c} P_{jc}} \log \frac{\sum_j P_{jc}}{\sum_{j,c} P_{jc}}. \tag{5.9}$$

High entropy is encouraged to balance the sizes of clusters. The entropy terms of P and of Q are scaled respectively by terms ρ_P and ρ_Q, set by validation.

5.3.3 Joint constraints

Temporal constraints In this work, an action is deemed to appear in a video if it is associated with a frame and a word whose timings overlap. For simplicity, here we consider the case where for each frame labeled with an action class, there must be exactly one word with the same label, and vice versa, but the extension to one-to-many mappings is included in Section 5.6.

For the fth frame of the video to be assigned to an action a in $1, ..., k$, it must be paired with a word, with index w, also assigned to a that is aligned with the fth frame (i.e., $A_{fw} = 1$):

$$P \circ (AQ - 1) \geq 0. \tag{5.10}$$

where $\circ$ is elementwise multiplication. The converse must also hold:

$$Q \circ (A^T P - 1) \geq 0, \tag{5.11}$$

If no temporal information is available for the script, A can be replaced by $\mathbb{1}_m \mathbb{1}_n^T$.

Ordering constraints In addition to overlapping alignment constraints, the order (and number) of actions appearing in the video and its script should also be consistent. Specifically, if a and a' are two actions appearing in P at frame indices f and f', respectively, such that $f < f'$, then there should exist word indices w, w' such that $Q_{wa} = Q_{w'a'} = 1$ where $w < w'$. Again, the inverse must also hold of P given Q. An equal number of actions is implied by this bidirectional constraint. While desirable in a final label assignment, this constraint makes things challenging from an optimization perspective.

5.3.4 Constraint relaxation

The joint constraints linking P and Q are not convex. We propose to use block-coordinate descent and alternately optimize over one assignment matrix while keeping

the other fixed. When either P or Q is fixed, the problem resembles the form considered in the second stage of Alayrac et al. [2016], a discriminative clustering problem in one modality under ordering and temporal alignment constraints. In our case, however, the order and number of actions given in the constraints (P or Q, whichever is currently fixed) is not assumed to be correct at any given iteration: due to shared ordering constraints, treating the initialization of one variate as the correct order would indeed not allow a better order to be discovered, i.e., whichever order and number of actions the initially fixed variate were to encode would necessarily be propagated into the other assignment. Therefore, the alignment and ordering constraints must be relaxed. We replace the hard constraints by a penalty based on the Needleman-Wunsch sequence alignment algorithm [Needleman and Wunsch, 1970], as described below.

Sequence alignment cost Let us consider the labels of each row in P as a sequence p, and the labels of each row in Q as a sequence q, where each item in p and in q is in $0, 1, ...k$ (0 corresponds to background). We wish to measure the distance between p and q in order to penalize an assignment pair (P, Q) in proportion to how inconsistent the sequences are with respect to each other, given the approximate alignment matrix A.

We define this measure of inconsistency as the number of actions in p that are not matched by q, and vice versa. For a fixed pair (p, q), this distance can be calculated using the Needleman-Wunsch sequence alignment algorithm [Needleman and Wunsch, 1970], commonly used on DNA sequences, with the modification that the subset of alignments considered must fall within A. Similar to dynamic time warping, Needleman-Wunsch discovers an alignment of the two sequences so as to maximize an alignment score. We define a pairwise scoring function $s : \{0, 1, ..., k\} \times \{0, 1, ...k\} \rightarrow \mathbb{Z}$ (more details in Section 5.7) such that the negative alignment score corresponds to the number of actions present in Q and skipped in P, plus the number of actions inserted by P. We add this conflict penalty $l(P, Q)$ to our objective function, giving a final objective of the form:

$$\begin{aligned} h(P, Q) =& f(P) + \mu g(Q) - \rho_P H(P) - \rho_Q H(Q) \\ &+ \lambda_t l(P, Q). \end{aligned} \tag{5.12}$$

The conflict l is scaled by λ, which is increased according to some schedule at each iteration t to encourage P and Q to become increasingly consistent over the course of optimization. We also scale $g(Q)$ by μ, which is set by validation, in order to give l different relative weight when optimizing for Q as opposed to P.

5.4 Optimization

Having relaxed the joint alignment and ordering constraints, we are left with the individual constraints P in $\mathcal{D}$ and Q in $\mathcal{E}$ only, corresponding to Eq. (5.7). The sets $\mathcal{D}$ and $\mathcal{E}$ are discrete and hence not convex. To optimize $h(P, Q)$ over them, we use Frank-Wolfe [Frank and Wolfe, 1956], an iterative first-order optimization algorithm, which considers their convex hulls conv($\mathcal{D}$) and conv($\mathcal{E}$). We use block-coordinate descent and alternate between optimizing for P with Q fixed, and optimizing for Q with P fixed, between each Frank-Wolfe step. At each iteration, a first-order Taylor approximation of h around P_{t-1} and Q_{t-1} is minimized subject to P in conv($\mathcal{D}$) to find a corner P_t^c (similarly, Q_t^c is found by minimizing $Q^T \frac{dh}{dQ}$ around P_t and Q_{t-1}). Since optimizing a linear function over our convex constraint set necessarily gives an integer solution, we have P_t^c in $\mathcal{D}$ and Q_t^c in $\mathcal{E}$.

As conflict $l(P, Q)$ is defined for integer P and Q only, a rounded $\hat{Q}_{t-1}$ in $\mathcal{E}$ is considered when optimizing for P_t, and a rounded $\hat{P}_t$ in $\mathcal{D}$ when optimizing for Q_t. However, since h is not differentiable due to its l-term defined for integer P and Q only, we instead minimize:

$$P_t^c = \underset{P \in \mathcal{D}}{\operatorname{argmin}}\, P^T \frac{d(f - \rho H)}{dP}(P_{t-1}) + \lambda_t l(P, \hat{Q}_{t-1}), \tag{5.13}$$

$$Q_t^c = \underset{Q \in \mathcal{E}}{\operatorname{argmin}}\, Q^T \frac{d(g - \rho H)}{dQ}(Q_{t-1}) + \lambda_t l(\hat{P}_t, Q). \tag{5.14}$$

The objective as well as the constraints are block-wise separable by video, so we can solve for P^c and Q^c individually for each video. This is done using a dynamic program.

We update P and Q by taking a step towards P^c and Q^c as follows:

$$\theta_P = \underset{\theta}{\operatorname{argmin}}\, h(((1-\theta)P_{t-1} + \theta P_t^c), \hat{Q}_{t-1}), \tag{5.15}$$

$$P_t = (1 - \theta_P)P_{t-1} + \theta_P P_t^c, \tag{5.16}$$

$$\theta_Q = \underset{\theta}{\operatorname{argmin}}\, h(\hat{P}_t, (1-\theta)Q_{t-1} + \theta Q_t^c), \tag{5.17}$$

$$Q_t = (1 - \theta_Q)Q_{t-1} + \theta_Q Q_t^c, \tag{5.18}$$

where P_0 (resp. Q_0) is obtained by starting from any point in conv($\mathcal{D}$) (resp. conv($\mathcal{E}$)). Other initialization strategies are discussed in Section 5.5. The optimization is continued until both θ_P and θ_Q fall below a threshold. The final solution $(\hat{P}_T, \hat{Q}_T)$ is obtained by rounding P_T and Q_T after the last iteration T.

5.5 Experiments

In this section, we demonstrate the validity of our method through its experimental evaluation on Inria Instruction Videos [Alayrac et al., 2016] and CrossTask [Zhukov et al., 2019].

5.5.1 Datasets

Inria Instruction Videos consists of 150 narrated tutorial videos, depicting 5 tasks. CrossTask, on the other hand, consists of an annotated primary set of 2763 videos (depicting 18 tasks) and an unannotated related set of 1950 videos (depicting 65 tasks). Both datasets are sourced from YouTube, and provide full temporal annotation of ground-truth steps. The tasks included comprise activities such as cooking, interior decoration and car maintenance, with 3 to 12 annotated steps per video. We only consider the primary subset of CrossTask in our evaluation, as ground truth data is not provided for the related set. In the Inria dataset, any off-topic introduction or ending has been trimmed from each video, whereas CrossTask videos are untrimmed. For Inria Instruction Videos, manually corrected transcribed narration is provided, whereas for CrossTask we use captions from automatic speech recognition (ASR), or user-generated captions where available.

Although each dataset groups the videos into categories, in the interest of generality, we *do not make use of these labels*, but treat the order of actions in each video as independent of other videos. Our method only relies on actions occurring often enough across the dataset to allow for enough data to learn from. On Inria Instruction Videos, the included tasks are different enough to not have any overlapping actions; we therefore do not expect to see an improvement on a particular task by combining it with other tasks. However, on CrossTask we can observe the effect of sharing, as 10% of the action steps occur in more than one task.

5.5.2 Feature representations

For a fair comparison with existing methods, we use the original features provided for each dataset as our frame representation. As a word representation, we use Elmo word vectors [Peters et al., 2018]. The narration is processed as follows: the original narration is transcribed to text with ASR. We follow Alayrac et al. [2016] in converting the free-form narration into a sequence of direct object relations (verb-object pairs) using the Stanford parser [Klein and Manning, 2003]. Each direct object relation is lemmatized (verbs are stripped of tense and conjugation, nouns are made singular) and represented by the concatenation of the word vectors of its verb and its object parts.

We also define a time window for each direct object relation as the shortest interval covering the individual timings of both the verb and the object, given the approximate timings of each word returned by ASR. From this, we obtain the alignment matrix A. The raw alignment can optionally be widened to cover delays between when some activity, say a recipe step, is discussed and when it is shown—the motivation being that narrators may describe what they are about to do slightly before doing it—-or set to $\mathbb{1}_m \mathbb{1}_n^T$ if no alignment information is available. In our experiments, the alignment is widened by 10 seconds from its end points as done by Alayrac et al. [2016].

5.5.3 Further objective terms

Making independent predicting for individual time steps can sometimes result in flickering between classes. However, for frames, there is a strong prior that consecutive time steps share the same label. To encourage longer predicted segments, we also add a penalty δ for each time step that is assigned a label different to that of the previous time step in P. Alternatively, shorter segments could be penalized at the rounding stage, but incorporating the penalty in the objective allows the clustering method to take advantage of this prior for learning.

Moreover, we apply a special treatment for the background class: in addition to a discriminative objective, we use a constant penalty γ_P for each frame assigned to background, and γ_Q for each word assigned to background. This is to account for the fact that background will never introduce a conflict penalty as defined by $l(P, Q)$. As the cost of conlicts increases over the course of optimization, this additional penalty allows to balance the size of the background class without necessarily increasing the entropy penalty.

5.5.4 Evaluation

Our method produces a full segmentation of the videos into k classes or background. We evaluate the discovered frame intervals according to how much they overlap with the annotated intervals in each video. Note that our method does not have access to ground-truth labels in training, and ground truth is only used for evaluation. However, none of the datasets we consider come with a predefined mapping of the spoken words to class labels, which is necessary to evaluate the word clusters. We therefore provide only a qualitative evaluation for words.

Label permutations Since there is no inherent ordering in the labels returned by our method (as the search space is symmetric with respect to permutations of the classes), we evaluate our output using the Hungarian algorithm. It finds the most favourable 1:1

mapping between predicted and true classes by maximizing the number of intersecting time steps between the two sets (as was done in Alayrac et al. [2016]; Elhamifar and Zaing [2019]; Sener et al. [2015]).

Evaluation metrics The output of our algorithm is a complete segmentation of the video at each time step into one of k classes or background. This is in contrast to Alayrac et al. [2016] and Zhukov et al. [2019], where only one time step is predicted per action and per video. Defining an evaluation metric to directly compare both outputs is not straightforward.

In Alayrac et al. [2016], a *segment-based F1* metric is used: a prediction is considered a true positive if it falls within an annotated interval of the same label, and a false positive otherwise. Each ground-truth segment that is missed by the predictions is counted as a single false negative, regardless of its length. The counts of true positive, false positive and false negative instances are then added up across the classes to define a single F1 score, i.e., the harmonic mean of precision and recall. In Zhukov et al. [2019], however, the number (and order) of action classes is known a priori, such that there is a one-to-one mapping between predicted and ground truth classes. In this setting, recall is used to measure how many of the ground truth time steps are correctly identified by having the predicted time step fall within the interval.

These evaluation metrics no longer give a useful indication of performance in our setting, as we want a segmentation to score in proportion to how extensively it covers the ground-truth intervals while introducing as few false positive time steps as possible. More relevant for our task is Jaccard index, i.e., intersection over union (IoU), on the number of time steps in the predicted intervals compared to ground truth intervals. To compare our method against prior work that outputs a single time step per prediction using this metric, we convert these single predictions into intervals by considering a fixed size window around each of them. This interval length is set to the average annotated interval duration in the ground truth across all action classes.

5.5.5 Baselines

As baselines, we use a random assignment as well as k-means clustering applied on the frames separately. To highlight the difficulty of the segmentation task, we also report IoU performance in a supervised setting, i.e., the performance obtained by training a linear classifier (which has a closed form solution) on the ground truth labels. The l2-regularization weight for the supervised classifier is fit to maximize performance across 4-fold crossvalidation on 80% of the dataset, after which it is retrained on all 80% and evaluated on the held out 20%.

	Change tire	Make coffee	Perform CPR	Jump-start car	Repot plant	All 5
random	0.037	0.027	0.036	0.016	0.030	0.008
k-means	0.120	0.093	0.127	0.034	0.070	0.088
Ours, init. from k-means	0.125	**0.097**	0.127	0.038	0.070	0.088
Alayrac et al. [2016], fixed length intervals	0.126	0.065	0.076	**0.042**	0.064	n/a
Ours, from Alayrac et al. [2016]'s init.	**0.162**	0.092	**0.238**	0.036	**0.121**	n/a
Supervised	0.193	0.074	0.331	0.033	0.090	0.115

Table 5.1: IoU for each task of Inria Instruction Videos, and for all tasks jointly. We set $k = 10$ for each separate task and $k = 50$ for the joint dataset.

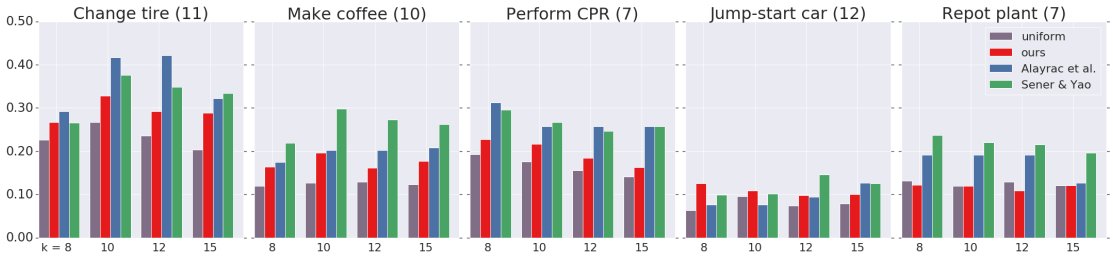


Figure 5.3: F1 scores on Inria Instruction Videos. The number of ground-truth actions is given in parentheses for each task.

5.5.6 Results

Frames Table 5.1 shows IoU for each task of Inria Instruction Videos, as well as for all included tasks combined into a single dataset, where the task labels are hidden. We report results for our method initialized from k-means, and from the initialization of Alayrac et al. [2016], in order to encode some prior knowledge without fully constraining the order. The initialization is obtained by first clustering the words using multiple sequence alignment (see Alayrac et al. [2016] for details), sampling 10 frames from each of the aligned time intervals and setting the initial P to the average of these 10 integer solutions.

Figure 5.3 shows F1 score for frames, for each task of Inria Instruction Videos, and Table 5.2 shows recall results for each individual task of CrossTask. We fit our hyperparameters on the validation set of 20 videos per primary task that is provided with the dataset, and evaluate on the remaining primary task videos. To fairly compare our

method to prior work in these contexts, we additionally constrain our label assignments to predict the same order of actions in each video, as this is a strong inductive bias for the datasets. We consider the middle time step of the predicted intervals when reporting F1 and recall. Although Elhamifar and Zaing [2019] also report F1 results for Inria Instruction Videos, their evaluation is different from ours and hence not comparable—an annotated segment is considered as correctly predicted if at least one time step of a prediction overlaps with it.

	Make Kimchi Rice	Pickle Cucumber	Make Banana Ice Cream	Grill Steak	Jack Up Car	Make Jello Shots	Change Tire	Make Lemonade	Add Oil to Car
Uniform	4.2	7.1	6.4	7.3	17.4	7.1	14.2	9.8	3.1
Ours	8.6	9.1	10.5	8.4	11.8	10.4	20.4	10.4	1.4
Alayrac et al. [2016]	15.6	10.6	7.5	14.2	9.3	11.8	17.3	13.1	6.4
Zhukov et al. [2019]	13.3	18.0	23.4	23.1	16.9	16.5	30.7	21.6	4.6

	Make Latte	Build Shelves	Make Taco Salad	Make French Toast	Make Irish Coffee	Make Strawberry Cake	Make Pancakes	Make Meringue	Make Fish Curry
Uniform	10.7	22.1	5.5	9.5	7.5	9.2	9.2	19.5	5.1
Ours	15.7	21.8	6.5	20.4	10.2	7.8	17.2	22.8	8.7
Alayrac et al. [2016]	12.9	27.2	9.2	15.7	8.6	16.3	13.0	23.2	7.4
Zhukov et al. [2019]	19.5	35.3	10.0	32.3	13.8	29.5	37.6	43.0	13.3

Table 5.2: Recall scores (%) on CrossTask. In order to compare with existing work, we assume the true k is known. Note that the method of Zhukov et al. [2019] has access to the additional data set of related tasks, as well as additional supervision in the form of known recipe steps.

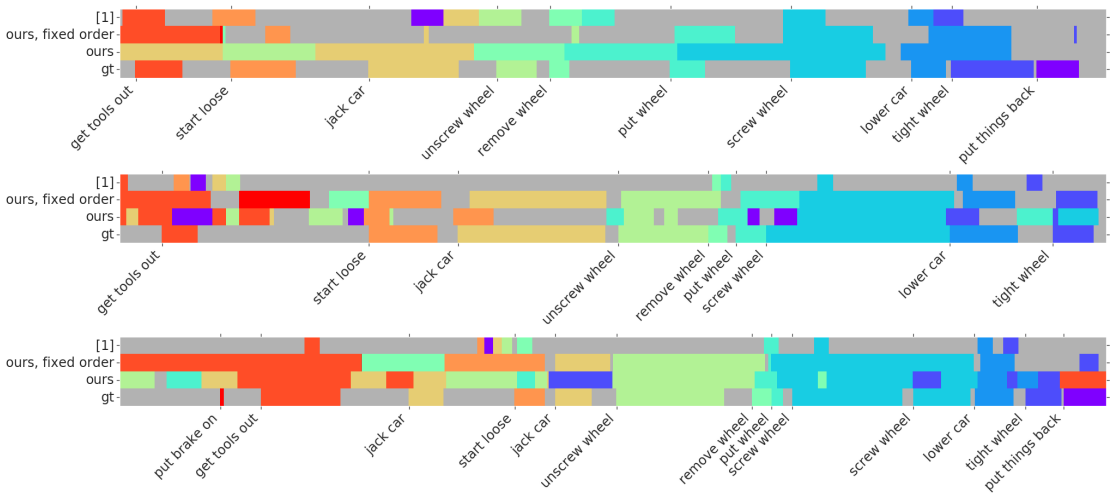


Figure 5.4: The predicted intervals compared with Alayrac et al. [2016] and ground-truth annotations for a few example videos of the *change tire* task of Inria Instruction Videos.

Words We include a visualization of our word to action assignments in Figure 5.5 for the *perform CPR* task. Qualitatively, our word assignment improves over Alayrac et al. [2016], as we are able to identify synonyms and related terminology, such as *tilt head* and *grasp chin*, or *wake casualty* and *tap shoulder*. In contrast, the word clustering stage in Alayrac et al. [2016] supports only basic synonym matching based on the WordNet tree [Oram, 2001]. Using this external corpus, direct object relations such as *perform compression* and *do compression* are correctly identified as describing the same action. However, the majority of the word clusters identified consist of a single direct object relation, such as *open airway*, *start compression*, or *pinch nose*.

5.6 One-to-many constraints

Typically, an action appears in multiple consecutive frames, but is described by only one or a few words. It is therefore desirable to relax the assumption of an equal number of labeled time steps between the two modalities; however, the number of labeled action *segments* should be equal. The temporal constraints in Equations (5.10) and (5.11) already allow one-to-many mappings between frames and words, but in this section we additionally adapt the ordering constraints for this setting. Note that both the temporal and the ordering constraints only apply for the non-background columns, i.e., for $P' = P_{1:m,1:k}$ and $Q' = Q_{1:n,1:k}$.

As defined in Section 5.3.3, for a consistent order between P and Q in the one-to-one setting, we want there to exist two distinct words w and w', $w < w'$, for each

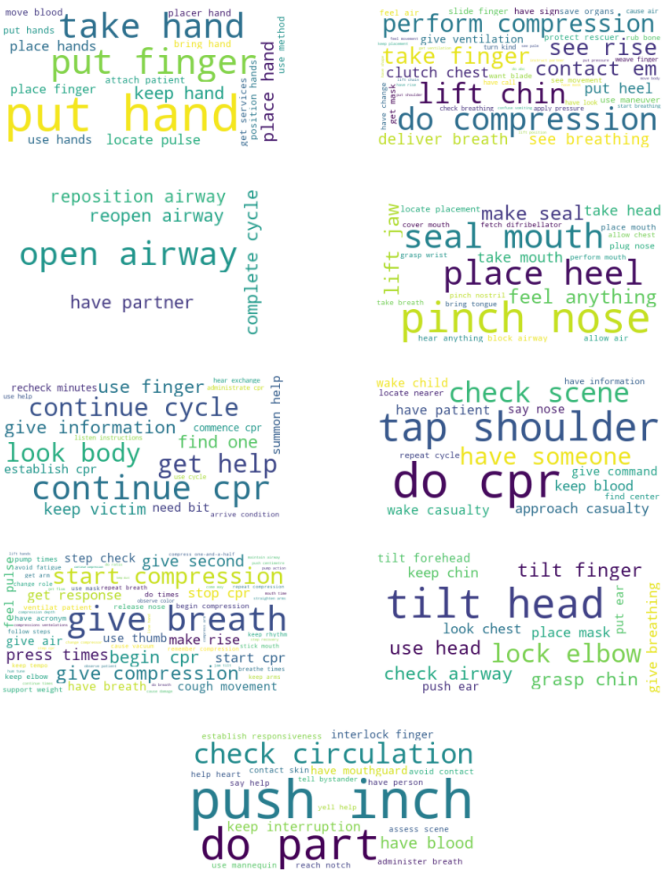


Figure 5.5: Nine of the word clusters discovered on the *perform CPR* task of Inria Instruction Videos. The ground-truth actions for the task are *open airway*, *check response*, *call 911*, *check breathing*, *check pulse*, *give breath*, and *give compression*.

ordered pair of frames f and f', $f < f'$, such that f and w are assigned to the same action a, and f' and w' are assigned to the same action a'. The converse should hold for P given Q. To allow a single word to map to multiple frames and vice versa, we instead have $w \leq w'$ and $f \leq f'$.

One-to-many mappings are enabled in all experiments included in Section 5.5.

5.7 Sequence alignment

As in Section 5.3.4, let us again consider the labels of each row in P as a sequence p, and the labels of each row in Q as a sequence q, where each item in p and in q is in $0, 1, ...k$ (0 corresponds to background). We will now present how the distance between p and q is computed.

We apply the Needleman-Wunsch sequence alignment algorithm [Needleman and Wunsch, 1970] to find the best alignment between P and Q, within the bounds of the

approximate alignment A. Needleman-Wunsch uses dynamic programming to fill in a score table G of size $(m+1) \times (n+1)$, where $G_{y+1,x+1}$ corresponds to the best possible score obtained by aligning sequences $p_{1:y}$ and $q_{1:x}$. Similarly, $G_{1,x}$ corresponds to the score of skipping the first x elements of q, and $G_{y,1}$ of skipping the first y elements of p. Each $G_{y>1,x>1}$ is calculated as the best outcome among three options: moving diagonally from $G_{y-1,x-1}$ (aligning p_y with q_x), moving vertically from $G_{y-1,x}$ (skipping p_y), or moving horizontally from $G_{y,x-1}$ (skipping q_x).

We define a symmetric scoring function $s : \{0, 1, ..., k\} \times \{0, 1, ...k\} \to \mathbb{Z}$ which assigns the following scores when aligning p_y with q_x:

$$s(p_y, q_x) = \begin{cases} 0 & \text{if } p_y = q_x \text{ and } A_{y,x} = 1, \\ -1 & \text{if } p_y = 0, q_x > 0, \\ -1 & \text{if } p_y > 0, q_x = 0, \\ -2 & \text{otherwise.} \end{cases} \tag{5.19}$$

In addition, skipping a sequence element (moving vertically or horizontally) is scored 0 if it is labeled background, and -1 otherwise. An exception to this are one-to-many mappings. A *skip move*, say from (p_{y-1}, q_x) to (p_y, q_x) does not incur a penalty if the previous element q_x matches the skipped element: $p_y = q_x$. However, the elements of p matched by any single q_x (and vice versa) need to be consecutive.

The negative score $-G_{m+1,n+1}$, which we call the conflict penalty $l(P, Q)$, corresponds to the number of actions present in Q and skipped in P, plus the number of actions inserted by P.

5.8 Computational complexity

Taking the gradient $\frac{df}{dP}$ with respect to all videos requires multiplying matrices of size $k \times m$, $m \times p$ and $p \times m$ (similarly, $k \times n$, $n \times q$ and $q \times n$ for $\frac{dg}{dQ}$, although this is likely a cheaper operation due to videos generally having fewer words than frames).

As the constraints are separable by video, the linear oracle can be solved separately for each video. To solve the linear oracle is $O(Nm_i n_i k^2)$ where m_i and n_i are respectively the number of frames and words in any single video, and N is the number of videos considered.

5.9 Rounding scheme

To obtain integer $\hat{P}_t$ from P_t and $\hat{Q}_t$ from Q_t, a number of rounding schemes can be used. The simplest strategy is geometric rounding:

$$\hat{P}_t = \operatorname*{argmin}_{\hat{P} \in \mathcal{D}} \left\| \hat{P} - P_t \right\|_F^2, \; \hat{Q}_t = \operatorname*{argmin}_{\hat{Q} \in \mathcal{E}} \left\| \hat{Q} - Q_t \right\|_F^2. \tag{5.20}$$

However, alternative schemes have been successfully applied with Frank-Wolfe. In Alayrac et al. [2016], the previous corner (P_t^c or Q_t^c in our case) is used when rounding in the word clustering stage (i.e., *frank-wolfe rounding*). On the other hand, *cost classifier rounding* is applied when clustering frames (the second stage):

$$\hat{P}_t = \operatorname*{argmin}_{\hat{P} \in \mathcal{D}} \left\| \hat{P} - XU - \mathbb{1}_m a^T \right\|_F^2, \; \hat{Q}_t = \operatorname*{argmin}_{\hat{Q} \in \mathcal{E}} \left\| \hat{Q} - YV - \mathbb{1}_n b^T \right\|_F^2. \tag{5.21}$$

Section 5.5 introduced two experimental settings. For the fully unconstrained case, we found geometric rounding to perform best, as it is better able to account for the size of the background class. Cost classifier rounding easily overestimates the size of the background class, since learning a linear classifier for background is hard. However, when the order of actions is fixed, the problem of learning label assignments, including the subproblem of balancing background and foreground time steps, is easier. For this setting, we found cost classifier rounding to outperform geometric rounding.

5.10 Conclusion

We have presented a method for automatically discovering actions in narrated tutorial videos, based on discriminative clustering and a sequence alignment penalty. We addressed the difficult problem of complex action segmentation in video, using no external supervision. Through experimental evaluation on two instruction video datasets, we have shown comparable performance to prior work, while requiring less supervision. At no labeling cost, our method can be applied on large video datasets to obtain action classifiers, demonstrations, or verbal descriptions of actions. Evaluating the quality of the subtask predictions in the context of training robots is a clear direction for follow-up work.

Chapter 6

Conclusion

REINFORCEMENT LEARNING AND WEAKLY supervised learning from video have many applications in robotic manipulation tasks, as seen in the previous chapters. In particular, we have focused on studying problem settings with realistic constraints, in the form of sparse or poorly defined reward functions, limited online interaction, and learning directly from sensor inputs such as camera images, rather than privileged information from a simulator's internal state. Taking advantage of external data is critically important in bringing RL methods closer to practical applications in robotics while limiting interaction time on the physical system. Ultimately, the goal is to seamlessly integrate learning from task demonstrations, from undirected or unsupervised large offline datasets, and new data collection on a real or simulated robot.

Advanced computer vision, imitation learning and deep RL have great potential to enable adaptive robotic agents and to fill the gaps in conventional optimal control. Especially in manipulation, the real-world diversity of objects, materials, shapes and tasks encountered outside of strictly controlled settings has proven challenging for classical control approaches. Particularly in tasks involving contracts, friction or air resistance such as pushing, sliding and throwing, or non-rigid deformations such as folding fabric, tying a rope, or working with viscous materials such as dough, the intermediate problem of precise physical modelling required by conventional model-predictive control approaches may well be more difficult than directly learning the relevant manipulation maneuver, the path taken by data-driven methods.

6.1 Summary of contributions

The methods presented in the preceding Chapters 3–5 propose solutions for several challenges in robot learning. A core thread throughout this thesis is an effort to reduce the amount of domain expertise required to train robots. The aim is to decrease the

importance of precise first-principles physical modelling, simulation and controller parameter tuning, as well as specialized solution methods per task, and to focus instead on pretraining on data available on the internet, task demonstrations (which may be provided by annotators without a robotics background), and online autonomous learning. We have equipped robotic agents with vision, proprioception, behavior priors from demonstrations, learned rewards as well as sparse rewards. The goal is to make use of an increasing variety of input data, with an emphasis on reusable and scalable sources:

- online robot interaction in the case of on-policy RL
- online interaction as well as prior interaction data in off-policy RL, including externally generated robot demonstrations in LfD (Ch. 3, 4)
- prior robot data only in offline RL
- unlabeled robot or human data without rewards in self-supervised objectives such as reconstruction, future prediction, or temporal offset estimation (Ch. 4)
- data collected from different robots, environments or tasks in multi-task offline RL (straight-forward extension of the method presented in Ch. 4)
- videos of humans executing manipulation tasks (Ch. 4)
- instruction videos of humans demonstrating activities with multiple subtasks (Ch. 5)

The most readily available sources lower down this list should be prioritized, with the top-most sources used sparingly.

6.2 Further work

The problem of movement is far from solved. In this section, we outline some promising directions for follow-up research. Beyond the avenues we have included below, which are most related to the topics of this thesis, there is no shortage of further open problems in the field of robot learning, such as meta- and multi-task learning, socially aware agents, and multi-agent learning.

6.2.1 Physical priors

Ideally, we should be able to equip our agents with an intuitive understanding of physics and geometry before expecting them to discover useful behaviors unsupervised. Examples could include using structured representations as state observations, such as

depth images, segmentation maps, point clouds or 3D reconstructions. Furthermore, symmetries and translational invariances could be imposed in the observation and action spaces, where relevant. However, such priors should in many cases be soft, and malleable by sufficient contradicting evidence rather than hard and unchangeable. Otherwise, agents are doomed to fail as soon as the assumptions underlying the physical model do not hold in a particular situation.

6.2.2 Predicting affordances

In addition to reward functions as explored in Chapter 4, behavior priors in the form of object affordances [Khetarpal et al., 2020] readily lend themselves to pretraining from offline data, including observations of humans. However, mapping affordances as used by a human demonstrator to robot equivalents remains challenging, as a robot with a significantly different end-effector morphology would likely need to adapt how and where exactly to hold objects, compared to a human hand. Nonetheless, gathering a basic understanding of objects and their relationships to each other would greatly simplify the search space for learning manipulation policies.

6.2.3 Long-horizon and hierarchical models

Skill hierarchies are a natural way to represent long-horizon tasks. Rather than predicting actions at the granularity of individual time steps, a *high-level policy* may perform long-term task planning by repeatedly handing control to one of a set of *low-level policies*, which represent individual subtasks and operate at a higher frequency. However, the task of jointly learning both low-level and high-level policies is difficult due to their direct coupling. Alternatively, hierarchies can be learned from offline data [Ajay et al., 2021; Pertsch et al., 2020]. In the vein of the method presented in Chapter 5, long-form tasks executed by a different actor, such as a human demonstrator, could equally serve as a source for learning hierarchies.

6.2.4 Language conditioning

Natural language is an essential component of human communication. To simplify the processes of specifying tasks and of giving feedback to artificial agents, too, it should a supported in human-computer and human-robot interaction. People routinely learn new tasks through language descriptions, through reading and listening to instructions as well as asking questions. In particular, natural language allows for tasks to be and parametrized and modularized ("place the *red* cup in the *small* box, *then* clear the table").

Recent work in language conditioning has proposed to take advantage of large language models [Ahn et al., 2022; Shridhar et al., 2021; Nair et al., 2021]. However, much of language-conditioned robotic manipulation work has focused on pure imitation [Shridhar et al., 2021; Lynch and Sermanet, 2021] without supporting autonomous improvement beyond demonstrations. Nonetheless, these early results paint an encouraging picture of task generalization in the latent space of natural language models.

Bibliography

A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a posteriori policy optimisation. In *ICLR*, 2018.

M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

A. Ajay, J. Wu, N. Fazeli, M. Bauza, L. P. Kaelbling, J. B. Tenenbaum, and A. Rodriguez. Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing. In *IROS*, pages 3066–3073, 2018.

A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. In *ICLR*, 2021.

M. Alakuijala, G. Dulac-Arnold, J. Mairal, J. Ponce, and C. Schmid. Residual reinforcement learning from demonstrations. In *RSS Workshop on Advances & Challenges in Imitation Learning for Robotics*, 2020a.

M. Alakuijala, J. Mairal, J. Ponce, and C. Schmid. Discovering actions by jointly clustering video and narration streams across tasks. In *CVPR Workshop on Learning from Instructional Videos*, 2020b.

M. Alakuijala, G. Dulac-Arnold, J. Mairal, J. Ponce, and C. Schmid. Learning reward functions for robotic manipulation by observing humans. *Under review*, 2022.

J.-B. Alayrac, P. Bojanowski, N. Agrawal, J. Sivic, I. Laptev, and S. Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4575–4583, 2016.

A. Allevato, E. S. Short, M. Pryor, and A. L. Thomaz. Tunenet: One-shot residual tuning for system identification and sim-to-real robot task transfer. *CoRL*, 2019.

M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

A. Argenson and G. Dulac-Arnold. Model-based offline planning. *ICLR*, 2021.

A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6836–6846, 2021.

Y. Aytar, T. Pfaff, D. Budden, T. Paine, Z. Wang, and N. De Freitas. Playing hard exploration games by watching youtube. *Advances in neural information processing systems*, 31, 2018.

F. R. Bach and Z. Harchaoui. Diffrac: a discriminative and flexible framework for clustering. In *Advances in Neural Information Processing Systems*, pages 49–56, 2008.

M. Barekatain, R. Yonetani, and M. Hamaya. Multipolar: Multi-source policy aggregation for transfer reinforcement learning between diverse environmental dynamics. *IJCAI*, 2020.

G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. Tb, A. Muldal, N. Heess, and T. Lillicrap. Distributed distributional deterministic policy gradients. *ICLR*, 2018.

A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *ICML*, 2017.

P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic. Weakly supervised action labeling in videos under ordering constraints. In *European Conference on Computer Vision*, pages 628–643. Springer, 2014.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. In *ICLR*, 2019.

S. Cabi, S. G. Colmenarejo, A. Novikov, K. Konyushkova, S. Reed, R. Jeong, K. Zolna, Y. Aytar, D. Budden, M. Vecerik, et al. Scaling data-driven robotics with reward sketching and batch reinforcement learning. *arXiv preprint arXiv:1909.12200*, 2019.

J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

A. S. Chen, S. Nair, and C. Finn. Learning generalizable robotic reward functions from "in-the-wild" human videos. In *RSS*, 2021a.

L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021b.

A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

W. Dabney, G. Ostrovski, D. Silver, and R. Munos. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, pages 1096–1105. PMLR, 2018a.

W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos. Distributional reinforcement learning with quantile regression. In *AAAI*, pages 2892–2901, 2018b.

R. Dadashi, L. Hussenot, M. Geist, and O. Pietquin. Primal wasserstein imitation learning. In *ICLR*, 2021.

R. Dadashi, L. Hussenot, D. Vincent, S. Girgin, A. Raichuk, M. Geist, and O. Pietquin. Continuous control with action quantization from demonstrations. In *International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 4537–4557. PMLR, 2022.

N. Das, S. Bechtle, T. Davchev, D. Jayaraman, A. Rai, and F. Meier. Model-based inverse reinforcement learning from visual demonstrations. *arXiv preprint arXiv:2010.09034*, 2020.

S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn. Robonet: Large-scale multi-robot learning. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 885–897. PMLR, 2020.

T. Davchev, K. S. Luck, M. Burke, F. Meier, S. Schaal, and S. Ramamoorthy. Residual learning from demonstration. *arXiv preprint arXiv:2008.07682*, 2020.

J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

M. Dehghani, A. Gritsenko, A. Arnab, M. Minderer, and Y. Tay. Scenic: A JAX library for computer vision research and beyond. *arXiv preprint arXiv:2110.11403*, 2021.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. IEEE, 2009.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

E. Elhamifar and Z. Zaing. Unsupervised procedure learning via joint dynamic summarization. In *International Conference on Computer Vision*, 2019.

B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. In *ICLR*, 2019.

H.-S. Fang, C. Wang, M. Gou, and C. Lu. Graspnet-1billion: A large-scale benchmark for general object grasping. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11444–11453, 2020.

P. Florence, C. Lynch, A. Zeng, O. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. *Conference on Robot Learning (CoRL)*, 2021.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062, 2019.

R. Goyal, S. Ebrahimi Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850, 2017.

K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022.

Ç. Gülçehre, Z. Wang, A. Novikov, T. L. Paine, S. G. Colmenarejo, K. Zolna, R. Agarwal, J. Merel, D. J. Mankowitz, C. Paduraru, G. Dulac-Arnold, J. Li, M. Norouzi, M. Hoffman, O. Nachum, G. Tucker, N. Heess, and N. de Freitas. RL unplugged: A suite of benchmarks for offline reinforcement learning. *NeurIPS*, 2020.

Y. Guo and D. Schuurmans. Convex relaxations of latent variable training. In *Advances in Neural Information Processing Systems*, pages 601–608, 2008.

A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6664–6671. IEEE, 2021.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018a.

T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft actor-critic algorithms and applications. Technical report, 2018b.

K. Hartikainen, X. Geng, T. Haarnoja, and S. Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. *ICLR*, 2020.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, et al. Streaming end-to-end speech recognition for mobile devices. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6381–6385. IEEE, 2019.

T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, G. Dulac-Arnold, J. Agapiou, J. Z. Leibo, and A. Gruslys. Deep q-learning from demonstrations. In *AAAI*, 2018.

R. W. Hicks II and E. L. Hall. Survey of robot lawn mowers. In *Intelligent Robots and Computer Vision XIX: Algorithms, Techniques, and Active Vision*, volume 4197, pages 262–269. SPIE, 2000.

J. Ho and S. Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29:4565–4573, 2016.

M. Hoffman, B. Shahriari, J. Aslanides, G. Barth-Maron, F. Behbahani, T. Norman, A. Abdolmaleki, A. Cassirer, F. Yang, K. Baumli, S. Henderson, A. Novikov, S. G. Colmenarejo, S. Cabi, C. Gulcehre, T. L. Paine, A. Cowie, Z. Wang, B. Piot, and N. de Freitas. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020.

D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver. Distributed prioritized experience replay. In *ICLR*, 2018.

A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *ICLR*, 2017.

D. Jain, A. Li, S. Singhal, A. Rajeswaran, V. Kumar, and E. Todorov. Learning deep visuomotor policies for dexterous hand manipulation. In *ICRA*, pages 3636–3643, 2019.

M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, 2021.

T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine. Residual reinforcement learning for robot control. In *ICRA*, pages 6023–6029, 2019.

J. L. Jones. Robots at the tipping point: the road to irobot roomba. *IEEE Robotics & Automation Magazine*, 13(1):76–78, 2006.

A. Joulin, F. Bach, and J. Ponce. Multi-class cosegmentation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 542–549. IEEE, 2012.

D. Kahneman. *Thinking, fast and slow*. Farrar, Straus and Giroux, 2011.

D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. *CoRL*, 2018.

D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman. Scaling up multi-task robotic reinforcement learning. In *5th Annual Conference on Robot Learning*, 2021.

H. Kasina, M. R. Bahubalendruni, and R. Botcha. Robots in medicine: past, present and future. *International Journal of Manufacturing, Materials, and Mechanical Engineering (IJMMME)*, 7(4):44–64, 2017.

W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

K. Khetarpal, Z. Ahmed, G. Comanici, D. Abel, and D. Precup. What can i do here? a theory of affordances in reinforcement learning. In *ICML*, pages 5243–5253. PMLR, 2020.

D. Klein and C. D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

J. Kober and J. R. Peters. Policy search for motor primitives in robotics. In *NeurIPS*, 2008.

J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

I. Koprinska and S. Carrato. Temporal video segmentation: A survey. *Signal processing: Image communication*, 16(5):477–500, 2001.

I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, and J. Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *arXiv preprint arXiv:1809.02925*, 2019.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

H. Kuehne, A. Richard, and J. Gall. Weakly supervised learning of actions from transcripts. *Computer Vision and Image Understanding*, 163:78–89, 2017.

A. Kukleva, H. Kuehne, F. Sener, and J. Gall. Unsupervised learning of action classes with continuous temporal embedding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on robot learning*, pages 143–156. PMLR, 2017.

M. Laskin, A. Srinivas, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *ICML*, pages 5639–5650. PMLR, 2020.

N. Lazic, C. Boutilier, T. Lu, E. Wong, B. Roy, M. Ryu, and G. Imwalle. Data center cooling using model-predictive control. *Advances in Neural Information Processing Systems*, 31, 2018.

A. X. Lee, C. Devin, Y. Zhou, T. Lampe, K. Bousmalis, J. T. Springenberg, A. Byravan, A. Abdolmaleki, N. Gileadi, D. Khosid, C. Fantacci, J. E. Chen, A. Raju, R. Jeong, M. Neunert, A. Laurens, S. Saliceti, F. Casarini, M. Riedmiller, R. Hadsell, and F. Nori. Beyond pick-and-place: Tackling robotic stacking of diverse shapes. In *Conference on Robot Learning (CoRL)*, 2021.

S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

J. Li, T. Lu, X. Cao, Y. Cai, and S. Wang. Meta-imitation learning by watching video demonstrations. In *ICLR*, 2021.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *ICLR*, 2016.

C. Lynch and P. Sermanet. Language conditioned imitation learning over unstructured data. In *RSS*, 2021.

C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans from play. In *Conference on Robot Learning*, pages 1113–1132. PMLR, 2020.

A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018.

M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. 1989.

A. Miech, D. Zhukov, J.-B. Alayrac, M. Tapaswi, I. Laptev, and J. Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. *arXiv preprint arXiv:1906.03327*, 2019.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, pages 1928–1937. PMLR, 2016.

A. W. Moore. Efficient memory-based learning for robot control. Technical report, University of Cambridge, 1990.

H. Moravec. *Mind children: The future of robot and human intelligence*. Harvard University Press, 1988.

A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *ICRA*, pages 6292–6299, 2018.

A. Nair, A. Gupta, M. Dalal, and S. Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

S. Nair, E. Mitchell, K. Chen, S. Savarese, C. Finn, et al. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *Conference on Robot Learning*, pages 1303–1315. PMLR, 2021.

S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

M. Neunert, A. Abdolmaleki, M. Wulfmeier, T. Lampe, T. Springenberg, R. Hafner, F. Romano, J. Buchli, N. Heess, and M. Riedmiller. Continuous-discrete reinforcement learning for hybrid control in robotics. In *CoRL*, 2019.

P. Oram. Wordnet: An electronic lexical database. christiane fellbaum (ed.). cambridge, ma: Mit press, 1998. pp. 423. *Applied Psycholinguistics*, 22(1):131–134, 2001.

D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

X. B. Peng, A. Kumar, G. Zhang, and S. Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

K. Pertsch, Y. Lee, and J. J. Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning (CoRL)*, 2020.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

V. Petrík, M. Tapaswi, I. Laptev, and J. Sivic. Learning object manipulation skills via approximate state estimation from real videos. *arXiv preprint arXiv:2011.06813*, 2020.

D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *NeurIPS*, 1988.

D. A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

E. Prassler, A. Ritter, C. Schaeffer, and P. Fiorini. A short history of cleaning robots. *Autonomous Robots*, 9(3):211–226, 2000.

C. Qi, X. Lin, and D. Held. Learning closed-loop dough manipulation using a differentiable reset module. *IEEE Robotics and Automation Letters*, 7(4):9857–9864, 2022.

Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. *arXiv preprint arXiv:2108.05877*, 2021.

A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *RSS*, 2018.

K. Rana, B. Talbot, V. Dasagi, M. Milford, and N. Sünderhauf. Residual reactive navigation: Combining classical and learned navigation strategies for deployment in unknown environments. *ICRA*, 2020.

S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

A. Richard, H. Kuehne, and J. Gall. Action sets: Weakly supervised action segmentation without ordering constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5987–5996, 2018.

S. Ross and D. Bagnell. Efficient reductions for imitation learning. In *AISTATS*, pages 661–668, 2010.

S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

M. Saveriano, Y. Yin, P. Falco, and D. Lee. Data-efficient control policy search using residual dynamics learning. In *IROS*, pages 4709–4715, 2017.

T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015.

K. Schmeckpeper, O. Rybkin, K. Daniilidis, S. Levine, and C. Finn. Reinforcement learning with videos: Combining offline observations with interaction. *Conference on Robot Learning*, 2020.

G. Schoettler, A. Nair, J. Luo, S. Bahl, J. A. Ojea, E. Solowjow, and S. Levine. Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards. *IROS*, pages 5548–5555, 2020.

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

F. Sener and A. Yao. Unsupervised learning and segmentation of complex activities from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8368–8376, 2018.

O. Sener, A. R. Zamir, S. Savarese, and A. Saxena. Unsupervised semantic parsing of video collections. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4480–4488, 2015.

P. Sermanet, K. Xu, and S. Levine. Unsupervised perceptual rewards for imitation learning. In *RSS*, 2017.

P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE, 2018.

L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. *The International Journal of Robotics Research*, 40(12-14):1419–1434, 2021.

M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021.

J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst. Blind bipedal stair traversal via sim-to-real reinforcement learning. *RSS*, 2021.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

L. Smith, I. Kostrikov, and S. Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *arXiv preprint arXiv:2208.07860*, 2022.

R. Strudel, A. Pashevich, I. Kalevatykh, I. Laptev, J. Sivic, and C. Schmid. Learning to combine primitive skills: A step towards versatile robotic manipulation. In *ICRA*, 2020.

R. Sutton. The bitter lesson. *Incomplete Ideas (blog)*, 13:12, 2019.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Y. Tang, D. Ding, Y. Rao, Y. Zheng, D. Zhang, L. Zhao, J. Lu, and J. Zhou. Coin: A large-scale dataset for comprehensive instructional video analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1207–1216, 2019.

S. Tian, S. Nair, F. Ebert, S. Dasari, B. Eysenbach, C. Finn, and S. Levine. Model-based visual planning with self-supervised functional distances. In *ICLR*, 2021.

J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *IROS*, pages 5026–5033, 2012.

F. Torabi, G. Warnell, and P. Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

Y.-H. Tsai, M.-H. Yang, and M. J. Black. Video segmentation via object flow. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3899–3908, 2016.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. 2017.

H. Wang and C. Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE international conference on computer vision*, pages 3551–3558, 2013.

Z. Wang, A. Novikov, K. Żołna, J. T. Springenberg, S. Reed, B. Shahriari, N. Siegel, J. Merel, C. Gulcehre, N. Heess, and N. de Freitas. Critic regularized regression. *NeurIPS*, 2020.

C. J. C. H. Watkins. Learning from delayed rewards. 1989.

P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg. Daydreamer: World models for physical robot learning. In *Conference on Robot Learning (CoRL)*, 2022.

Y. Wu, G. Tucker, and O. Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

K. A. Wyrobek, E. H. Berger, H. M. Van der Loos, and J. K. Salisbury. Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot. In *2008 IEEE International Conference on Robotics and Automation*, pages 2165–2170. IEEE, 2008.

H. Xiong, Q. Li, Y.-C. Chen, H. Bharadhwaj, S. Sinha, and A. Garg. Learning by watching: Physical imitation of manipulation skills from human videos. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7827–7834. IEEE, 2021.

T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

K. Zakka, A. Zeng, P. Florence, J. Tompson, J. Bohg, and D. Dwibedi. Xirl: Cross-embodiment inverse reinforcement learning. In *Conference on Robot Learning*, pages 537–546. PMLR, 2022.

A. Zeng, S. Song, J. Lee, A. Rodriquez, and T. A. Funkouser. Tossingbot: Learning to throw arbitrary objects with residual physics. In *RSS*, 2019.

A. Zhan, P. Zhao, L. Pinto, P. Abbeel, and M. Laskin. A framework for efficient robotic manipulation. *arXiv preprint arXiv:2012.07975*, 2020.

H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

J. O. Zhang, A. Sax, A. Zamir, L. Guibas, and J. Malik. Side-tuning: Network adaptation via additive side networks. *ECCV*, 2020.

Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. In *RSS*, 2018.

D. Zhukov, J.-B. Alayrac, R. G. Cinbis, D. Fouhey, I. Laptev, and J. Sivic. Cross-task weakly supervised learning from instructional videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3537–3545, 2019.

RÉSUMÉ

La commande robotique appris à partir de données n'a pas encore eu d'impact à grande échelle dans le monde réel. L'une des principales limitations est l'accès aux données: il est difficile de trouver des exemples d'entraînement annotés sur Internet, et la collecte de données dans des environnements physiques est limitée par le fonctionnement du robot en temps réel. Cette thèse présente plusieurs façons d'exploiter des sources de données externes, de démonstrations de tâches aux tutoriels vidéo, pour relever le défi de la lenteur de la collecte de données et ainsi accélérer l'apprentissage des tâches de manipulation robotique. Notre argument central est que les avancées dans les domaines connexes de la vision par ordinateur, du traitement du signal, du traitement du langage naturel, des l'apprentissage par imitation et par renforcement profond peuvent aider à ouvrir la voie à des agents robotiques plus adaptatifs. C'est particulièrement le cas pour le domaine de la manipulation dans le monde réel, en dehors de conditions d'exploitation étroitement contrôlées. En effet, la variété des matériaux, des formes et des tâches pose de grandes difficultés pour les stratégies de contrôle fixes et les approches classiques de commande prédictive qui nécessitent une modélisation physique précise. Notre objectif principal est donc de permettre une manipulation robotique plus performante et polyvalente grâce à des méthodes appris à partir des données.

MOTS CLÉS

apprentissage par renforcement, prise robotique, compréhension video, apprentissage par démonstration.

ABSTRACT

Robotic control learned from data has yet to show large-scale impact in the real world. One of the main limitations is access to data: explicit training examples cannot be easily sourced and annotated on the internet but data collection is bounded by real-time robot operation. This thesis proposes several ways to leverage external data sources, from task demonstrations to full-length tutorial videos, to address the challenge of slow data collection and thus accelerate learning of robotic manipulation tasks. Our key argument is that advances in the related fields of computer vision, signal processing, natural language processing, imitation and deep reinforcement learning can help lead the way towards more adaptive robotic agents. In manipulation domains, in particular, the variety of materials, shapes and tasks present in the real world beyond tightly controlled operating conditions poses great difficulty for fixed control strategies and the precise physical modelling required by classical model-predictive control approaches. Our overarching goal is therefore to enable more capable and versatile robotic manipulation through data-driven methods.

KEYWORDS

reinforcement learning, robotic grasping, video understanding, learning from demonstration.