Deep Reinforcement Learning and Transfer Learning of Robot In-hand Dexterous Manipulation

Yingyi Kuang

Doctor of Philosophy

Aston University

Department of Computer Science

June 2023

ASTON UNIVERSITY

Deep Reinforcement Learning and Transfer Learning of Robot In-hand Dexterous Manipulation

Yingyi Kuang

Doctor of Philosophy

March, 2024

In recent years, deep reinforcement learning (RL) and imitation learning (IL) have shown remarkable success in many robotics areas. However, the domain of in-hand dexterous manipulation remains challenging for RL and IL. Achieving proficiency in these tasks often requires millions of attempts or demonstrations before a stable strategy is learnt. Consequently, improving the learning speed and efficiency becomes paramount for RL and IL to be practically used in real-world in-hand dexterous manipulation tasks.

This thesis primarily addressed multi-goal robot in-hand dexterous manipulation tasks, with various methods proposed to improve learning efficiency: For RL, (1) the Goal Density-based Hindsight Experience Prioritisation (GDP) is proposed to improve learning efficiency by prioritising some experiences during the replay stage; Furthermore, (2) another method called Policy-level-based Curriculum Goal Selection (PL-CGS) is proposed to automatically generate goals during the learning process that could form a curriculum learning process; For IL, (3) the Goal-based Self-Adaptive Generative Adversarial Imitation Learning (Goal-SGAIL) incorporates a self-adaptive mechanism into the GAIL framework that applies to multi-goal learning scenarios.

Extensive experiments were conducted in simulation with OpenAI Gym, focusing on robot manipulation tasks, to compare the proposed methods against existing RL and IL approaches. GDP and PL-CGS showed faster learning speed compared with the vanilla DDPG+HER method for some of the tasks in the RL experiments. For experiments in IL that involve sub-optimal demonstrations, especially those with highly sub-optimal demonstrations from human teleoperation, Goal-SGAIL showed its ability to overcome the demonstrations' sub-optimality and outperformed DDPGfD+HER and Goal-GAIL for some challenging in-hand manipulation tasks.

Keywords: Reinforcement learning, HER, Experience prioritisation, Curriculum learning, Learning from demonstration, GAIL.

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to my supervisors: Dr Luis Manso, Dr George Vogiatzis and Dr Diego Faria. Their invaluable advice, continuous support, and enduring patience have been the cornerstone of my PhD journey. I would like to thank them for investing their precious time with their vast knowledge and extensive experience to guide me through each phase of my study.

I am also grateful to Aston University and the CHIST-ERA InDex project for their generous scholarship and sponsorship, which made my PhD research feasible. My heartfelt thanks go to Dr Maria Chli for offering me a research position within her project and funding my extension year, a gesture that has significantly contributed to my academic growth.

My appreciation extends to our collaborators in the CHIST-ERA InDex project, whose shared expertise and resources have been instrumental in advancing my research.

Additionally, I would like to thank my qualifying report examiner, Dr Felipe Campelo, and my lab colleagues (Renato, Deepeka, Tom etc.), for their assistance and insightful advice on my studies and research throughout my PhD. My gratitude also goes to Dr Martin Rudorfer for his help in revising my thesis.

Lastly, but most importantly, I wish to express my profound thanks to my husband, Dr Xiaotian Dai, who supports me unconditionally in my studies and daily life. Without his continuous understanding and encouragement during the last few years, completing this journey would have been an insurmountable task.

Declaration

I declare that this thesis is a presentation of original work, and I am the sole author. Certain parts of the material presented within this thesis have appeared in published papers/reports or have been prepared as papers to be submitted. Specifically, these are:

- Yingyi Kuang, Abraham Itzhak Weinberg, George Vogiatzis and Diego R. Faria. *"Goal density-based hindsight experience prioritization for multi-goal robot manipulation reinforcement learning"* [54]. 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN 2020). IEEE, 2020.
- Yingyi Kuang, George Vogiatzis, Diego R. Faria. *"Improving Exploration Efficiency of Single-goal In-hand Manipulation Reinforcement Learning by Progressive Goal Generation"* [53]. Workshop on Hand-OBject Interaction: From human demonstrations to robot manipulation. 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN 2020). IEEE, 2020.
- Alessandro Carfì, Timothy Patten, Yingyi Kuang, Ali Hammoud, Mohamad Alameh et al. *"Hand-object interaction: From human demonstrations to robot manipulation"* [12]. Frontiers in Robotics and AI Journal. Frontiers, 2021.
- Yingyi Kuang, George Vogiatzis, Diego R. Faria and Luis Manso. *"Policy-level-based Automatic Curriculum Goal Selection for multi-goal orientated robot manipulation reinforcement learning"*. In preparation for submission.
- Yingyi Kuang, George Vogiatzis, Diego R. Faria and Luis Manso. *"Goal-based Self-adaptive Generative Adversarial Imitation Learning for multi-goal robot manipulation tasks"*. In preparation for submission.

This work has not previously been presented for an award at this, or any other university. All sources are acknowledged as references.

Contents

Abbreviations

RL	Reinforcement Learning
IL	Imitation Learning
InDex	In-hand Dexterous Manipulation
LfD	Learning from Demonstration
DNN	Deep Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
GAN	Generative Adversarial Network
MDP	Markov Decision Process
MLP	Multi-Layer Perceptron
DQN	Deep Q-Network
DDPG	Deep Deterministic Policy Gradient
TRPO	Trust Region Policy Gradient
DDPGfD	Deep Deterministic Policy Gradient from Demonstrations
UVFAs	Universal Value Function Approximators
HER	Hindsight Experience Replay
GMM	Gaussian Mixture Model
GOID	Goals of Intermediate Difficulty
EBP	Energy-based Prioritisation

MEP	Maximum Entropy-based Prioritisation
GDP	Goal-Density based Prioritised experience replay
PSES	Prioritisation Switching with Ensembling Strategy
CGM	Curriculum Goal Masking
HGG	Hindsight Goal Generation
PL-CGS	Policy-Level-based automatic Curriculum Goal Selection
BC	Behaviour Cloning
IRL	Inverse Reinforcement Learning
GAIL	Generative Adversarial Imitation Learning
SAIL	Self-adaptive Generative Adversarial Imitation Learning
Goal-GAIL	Goal-conditioned Generative Adversarial Imitation Learning
Goal-SGAIL	Goal-based Self-Adaptive Generative Adversarial Imitation Learning

List of Illustrations

Figures

Tables

Chapter 1

Introduction

In recent decades, robotic manipulators have seen rapid advancements and have been increasingly integrated into sectors like industrial manufacturing, assistance for humans, and medical surgeries. These manipulators, tailored with various joint structures and control techniques, cater to specific needs across applications. As a significant subset of robotics, robot manipulators have been systemically researched, focusing on their kinematic features and dynamic models [101]. The robot hand, acting as the manipulator's end-effector, is pivotal for direct interaction with the environment and, notably, for grasping objects. The development of robotic hands has spanned over forty years [44, 81]. leading to the creation of basic models such as two-finger and three-finger grippers, as well as suction cup grippers, which are commonly utilized in industrial settings. While these simpler designs prove effective for particular applications, these simpler designs fall short in complexity in tasks requiring operation within human-centric environments. Such environments frequently necessitate the manipulation of tools and devices crafted for human hands and often require fine manipulation. This poses a challenge to continuous research and innovation for more sophisticated robotic hand control methods that are capable of navigating the nuances of human-oriented tasks.

The human hand distinguishes itself in dexterity and flexibility compared to those of most other animals. Our brain's exceptional ability to control our hands enables us to manipulate objects and perform a vast array of daily tasks effortlessly. During the lifelong learning process, we enhance our hand manipulation skills through observation, self-discovery, and adaptation to interact with new objects and navigate complex environments. Inspired by the structure of the human hand, numerous anthropomorphic robot hand prototypes have been developed, with some advancing to commercial production [93]. Similar to the human hand, these robot hands usually feature 4 or 5 fingers, each with 2 or 3 joints. Typically, these joints are independently controlled by DC motors [32, 47, 50, 109] or pneumatic actuators [30]. An example of such a commercially designed anthropomorphic robot hand is illustrated in Figure 1.1. Later researchers have explored the tendon-driven actuation system, inspired by the biomechanics of the human hand [118]. This approach allows for more natural and fluid movements, bridging the gap further between robotic and human hands. Additionally, recent work [20, 119] have designed robot hands that mimic the shapes of human bones and soft tissue structures, enhancing the robot hand's ability to perform human-oriented tasks with greater authenticity.

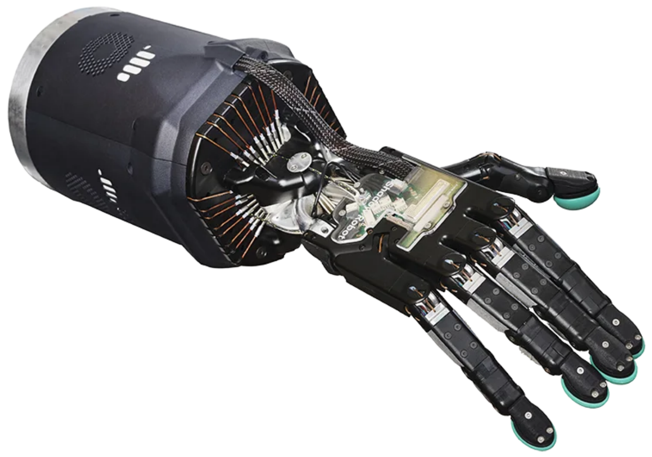


Figure 1.1: A commercialised anthropomorphic robot hand (Shadow Hand [93]). Picture from Shadow Robot company website: `https://www.shadowrobot.com/`

Beyond mechanical designs, sensors have a crucial role in enhancing the functionality of robot hand systems by providing necessary feedback. Sensing based on machine vision, for example, using RGB cameras, is the primary way to estimate the pose or position of objects and to observe finger movements [3]. This technology equips robotic systems with the capability to acquire a visual understanding of their surroundings, facilitating precise manipulation and interaction with objects. Tactile sensors, which gather contact information as part of the feedback for anthropomorphic hands, have gained more attention recently [41]. This expansion aims to mimic the human hand's touch capabilities more closely, allowing for a more nuanced percept and interaction with objects and surfaces. While most designs incorporate tactile sensors at the fingertips [93] to maximise sensitivity and accuracy in object manipulation, some work attempts to extend this coverage across the entire finger and palm areas [50].

Dexterity, as a concept in the context of anthropomorphic manipulation, refers to the capability of utilising multiple manipulators, or fingers from a single manipulator, to alter an object's position and orientation from one state to another [10]. The concept of in-hand dexterous manipulation, however, lacks a universally accepted definition. The distinguishing characteristic of in-hand dexterous manipulation, setting it apart from conventional robotics manipulation problems, is its emphasis on object-centred interaction [82]. In-hand dexterous manipulation encompasses a broad spectrum of tasks, ranging from basic and conceptual tasks, such as re-grasping, rolling, and sliding [70], to practical applications like object reorientation, repositioning, and extending to more intricate activities like twirling pens, opening bottle caps, etc. Despite progress in developing the control algorithms for anthropomorphic robot hands, achieving human-like dexterity and adaptation for these in-hand dexterous manipulation tasks remains elusive. Current systems struggle to handle a variety of objects and adapt to changing environments. The manipulation skills are still not flexible enough to handle most human daily objects in a natural, human-like manner. This limitation hampers the potential to use robot hands as intelligent assistants in real-world settings.

Machine learning offers a possibility for robot controllers to acquire some level of cognitive capability. It has seen significant achievements across various domains such as image processing, voice and text recognition, and notably, robotic control. The subsequent advancements in deep neural networks (DNNs) have been particularly transformative. DNNs, with their deep architectures, are capable of encapsulating complex and non-linear model information. This makes DNNs suitable for modelling and solving the anthropomorphic robot hand control problems, which are high-dimensional and non-linear.

Reinforcement learning (RL), a specialised branch of machine learning, has demonstrated remarkable strength in finding solutions for complex challenges across a variety of domains, including robotics. Its application to learn in-hand dexterous manipulation tasks with anthropomorphic robot hands reveals its potential to significantly advance the robot's capabilities [18]. However, due to the iterative process of policy refinement in RL, it requires the robot to have extensive interaction with the environment for data collection (usually needs millions of time steps). When applied to real-world robots, especially those with delicate structures like anthropomorphic robot hands, this requirement becomes not only costly and time-consuming but also fraught with safety concerns. The risk of damaging the robot hands during data collection is a significant deterrent, which makes RL for complex in-hand dexterous tasks with a real robot hand nearly infeasible with current technology. Until today, sampling inefficiency is still the most critical barrier to the wider adoption of RL in robotic manipulation [79], particularly for tasks requiring the finesse and subtlety of in-hand dexterous manipulation. Addressing this issue requires innovations aimed at improving learning efficiency.

Model-based RL approaches such as Guided Policy Search (GPS) [61] and Probabilistic Inference for Learning COntrol (PILCO) [17] aim to construct an internal model of the environment, which they then use to simulate interactions and generate data for policy training. These methods offer promising strategies to address the data inefficiency problem inherent in traditional model-free RL methods. Despite their advantages in data efficiency, model-based RL approaches come with their own set of challenges. They often require detailed knowledge of specific task domains and extra effort to accurately model the environment, which can be a complex and daunting task in itself.

Some other strategies focus on improving the balance between exploration (trying out new actions to discover effective strategies) and exploitation (leveraging known strategies to gain rewards) during learning. This is especially important for tasks in continuous high-dimensional action space, such as in-hand dexterous manipulation. Techniques such as experience prioritisation [98] and curriculum learning [26] have shown promise in making learning more efficient. However, these improvements have limited effects in high-dimensional space and remain challenging. The development of more sophisticated models, algorithms, and training techniques continues to be a critical area of research in making RL a practical solution for complex robotic manipulation tasks.

Imitation learning (IL), also recognised as learning from demonstration (LfD), is another direction to explore that may help overcome RL sampling inefficiency. Instead of trying to learn the task from scratch in a traditional

RL pipeline, imitation learning allows the agent to leverage examples of expert behaviour directly. By imitating the decisions of experts, the agent significantly reduces the exploration space required for learning. This leads to improvements in both the speed and accuracy of the learning process, making IL particularly appealing for complex tasks. In the robot manipulation field, these examples or demonstrations of expert behaviour can be sourced from robots with established expert control policies or directly from human operators. The success of imitation learning heavily depends on the quality and quantity of the demonstrations provided. However, the challenge lies in gathering a sufficient number of optimal demonstrations, especially for complex tasks such as in-hand dexterous manipulation. Consequently, the issue of how to learn effectively from a limited set of sub-optimal demonstrations emerges as a critical area of research within the field of robotic manipulation.

Research focusing on in-hand dexterous manipulation tasks still has many gaps:

- **Challenges of Reinforcement Learning**: Many works have attempted to address in-hand dexterous manipulation through RL. However, these efforts still encounter obstacles related to learning efficiency. The majority of these studies are confined to simulations. Despite the acknowledgement of this issue within the research community, a comprehensive investigation into enhancing RL's learning efficiency for such tasks remains sparse. This presses the need for more focused research in this area (more discussion is provided in Section 2.2.3).

- **Challenges of Imitation Learning**: The application of IL to in-hand dexterous manipulation, especially with humanoid robot hands, is not as prevalent. The existing IL approaches for these tasks [34, 90] are characterised as naive. This indicates a significant opportunity for advancement, as advanced IL algorithms have not yet been fully explored or adapted for the nuanced challenges presented by in-hand dexterous manipulation tasks.

In summary, while RL and IL have made significant achievements in robotics, their application to in-hand dexterous manipulation tasks is still in its early stages. Bridging this gap requires targeted research efforts to either enhance learning efficiency for RL or investigate more sophisticated IL algorithms.

1.1 Research Aim and Research Questions

This thesis focuses on advancing the reinforcement learning and imitation learning strategies for multi-goal-orientated in-hand dexterous manipulation tasks using five-fingered anthropomorphic robot hands. The thesis addresses the following two research questions:

- **RQ.1**: On the reinforcement learning side, how can the learning efficiency be improved under the multi-goal-orientated condition?
- **RQ.2**: On the imitation learning side, how to utilise the demonstrations to accelerate the learning speed of multi-goal learning by applying the recent advances in imitation learning?

1.2 Thesis Contributions

This section describes the contributions made in response to the research questions posed in the preceding section. The key contributions of this thesis are enumerated below:

- **C.1**: In response to research question **RQ.1**, an experience prioritisation strategy known as the Goal Density-based hindsight experience Prioritisation (GDP) method is introduced to boost the efficiency of the multi-goal reinforcement learning by optimising the exploitation of experiences gathered during the learning process.
- **C.2**: Furthermore, in addressing research question **RQ.1**, this thesis presents a curriculum goal selection strategy named the Policy-level-based Automatic Curriculum Goal Selection (PL-CGS). This method is designed to enhance the learning efficiency of multi-goal reinforcement learning by facilitating more effective exploration of the task space.
- **C.3**: In response to research question **RQ.2**, a Goal-based Self-adaptive Generative Adversarial Imitation Learning (Goal-SGAIL) strategy is tailored for learning from demonstrations for multi-goal in-hand dexterous manipulation tasks.

1.3 Thesis Organisation

This chapter provides an overview of the thesis, beginning with an introduction to the latest advancements in robot manipulators, with a particular focus on the anthropomorphic robot hand. It then delves into the application of machine learning to in-hand dexterous manipulation problems and outlines the associated challenges. The structure of the remaining contents of this thesis is outlined as follows:

- **Chapter 2:** This chapter lays the foundation by introducing the fundamental concepts upon which this thesis builds and provides an overview of prior work in the field. A review and discussion of the related literature are also included in this chapter.
- **Chapter 3:** This chapter begins by introducing the definition of the multi-goal reinforcement learning (RL) problem and examines the existing literature on Hindsight Experience Replay (HER) related works. It then introduces the *Goal Density-based hindsight experience Prioritisation (GDP)* method and the *Policy-level-based Automatic Curriculum Goal Selection (PL-CGS)* method, both aimed at enhancing learning efficiency in multi-goal RL settings.
- **Chapter 4:** This chapter first discusses the existing research on Generative Adversarial Imitation Learning (GAIL). It then presents the *Goal-based Self-adaptive Generative Adversarial Imitation Learning (Goal-SGAIL)* approach. Following that, it provides a practical imitation learning example, employing various methods with data gathered via human teleportation.
- **Chapter 5:** This chapter summarises the work detailed in this thesis, offers conclusions for each method proposed, and highlights potential directions for future research.

For the sake of brevity in this thesis, the term 'in-hand dexterous manipulation' will be abbreviated to 'InDex', 'reinforcement learning' will be denoted as 'RL', 'imitation learning' will be referred to as 'IL', and 'learning from demonstrations' will be abbreviated as 'LfD' in subsequent chapters. Additionally, specific abbreviations relevant to our contributions are defined within each chapter as they are mentioned.

Chapter 2

Background

This chapter presents a detailed review of the literature relevant to this dissertation. It begins by exploring traditional control approaches for robot in-hand dexterous manipulation. Following this, the discussion shifts to various RL algorithms suitable for robot manipulation tasks. This is followed by a review of prior studies focusing on deep reinforcement learning in the context of both general robotic manipulation and specific in-hand dexterous manipulation tasks. Subsequently, the chapter delves into imitation learning approaches as applied to the broader domain of robotic manipulation, and highlights recent advancements in in-hand dexterous manipulation tasks. The chapter concludes with a summary that encapsulates the current advancements and research gaps in deep reinforcement learning and imitation learning within the realm of robotic in-hand manipulation.

2.1 Traditional research pipeline for dexterous manipulation

The development of controllers for dexterous manipulation, including those involving in-hand dexterous manipulation, represents a significant and dynamic field of study within robotic [56]. The majority of efforts in this domain have focused on understanding and modelling the kinematics and dynamics of the system, with a particular emphasis on the force of contact and friction between the finger and the object [13]. In-hand dexterous manipulation tasks such as rolling [9] and sliding [103,111] have the feature that the contact points between robot fingers and the object surface move during manipulation. The goal of these studies is to build dynamic kinematics models for the points in contact with the object [51] to help motion control. Additionally, some research has extended to include the role of the hand palm within the analysis of operational space dynamics [7].

A group of other works emphasises planning the transitions between different motion phases and identifying the events that initiate these transitions [43, 112]. These works address the problems of changing kinematic and dynamic constraints that are encountered during manipulation tasks, offering a variety of frameworks to solve the problem. The smoothness of phase transition triggered by an event is critical since careless motion changes can cause undesired behaviour, such as dropping the object.

Other studies have investigated optimising grasping strategies. Given the kinematic redundancy typical of manipulators used for in-hand dexterous manipulation, an infinite array of grasping options exists. Research focuses on identifying the 'optimal' grasp, either through quality measurements [106] or by using classifiers trained on human demonstrations [67]. Additionally, there is research aimed at higher-level planning for in-hand manipulation tasks. Efforts are being made to develop a taxonomy or identify features of manipulation tasks that apply to both human and robotic hands [25,59,68]. Then either use the feature extracted from human demonstration to guide robot manipulation or regard different actions as sub-tasks and design planning strategy for more complicated manipulation tasks [85]. The classification method could be hand-centric, motion-centric or object-centric [10].

Traditional model-based approaches have achieved sufficient control ability and accuracy on simple structured manipulators. Nonetheless, commanding multi-finger humanoid hands to execute precise and reliable in-hand manipulation remains challenging. Anthropomorphic hands typically have 20 to 24 degree-of-freedoms. Due to the high degree of freedom involved in system modelling and the complexity of actuation dynamics, it is extremely challenging to execute practical dexterous manipulation applications with traditional methods. Moreover, controllers that are manually designed for specific tasks within highly restrictive conditions lack the versatility needed to adapt to different objects or to operate in new environments.

2.2 Reinforcement learning and in-hand dexterous manipulation related work

In this section, reinforcement learning under the context of robot manipulation is discussed. A variety of RL algorithms are introduced and compared. Following this, the discussion shifts to the latest advancements in RL research pertinent to in-hand dexterous manipulation (InDex).

2.2.1 General reinforcement learning introduction

Reinforcement learning (RL) is one of the three main branches of machine learning, alongside supervised and unsupervised learning [65]. It focuses on solving sequential decision-making problems. In RL, an agent observes its environment, takes actions within that environment, and receives rewards based on those actions. The reward is designed to encourage positive behaviours and discourage negative ones. Through a process of trial and error, the agent's policy converges towards an optimal strategy which selects actions that maximise the expected cumulative reward over time [107]. A typical agent-environment interaction loop of RL is shown in Figure 2.1.

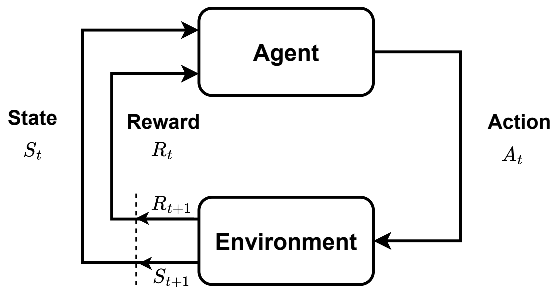


Figure 2.1: The agent-environment interaction loop for RL

Preliminaries on RL

First, some key concepts and terminologies of RL are given here. A classical RL problem is formed by an agent interacting with an environment. At each time step t, the agent receives a state s_t from the environment and takes action a_t according to the policy $\pi_\theta(a_t|s_t)$ parameterised by θ, a mapping from state s_t to action a_t. After that, the agent receives a reward r defined by a reward function $r_t = R(s_t, a_t)$ and the state transits to s_{t+1}. The policy can be a stochastic distribution of actions over the current state $a \sim \pi_\theta(.|s)$, or a deterministic function $a = \mu_\theta(s)$. θ is the denotation of the policy parameters.

RL problems are typically modelled as Markov decision processes (MDPs), a mathematical framework for sequential decision-making. Systems represented by MDP need to satisfy the Markov property: $P(s_{t+1}|s_1, a_1, ..., s_t, a_t) = P(s_{t+1}|s_t, a_t)$, which means transitions only depend on the most recent state and action. A MDP is normally represented as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, p(s_0))$:

- State space $\mathcal{S}$, consist of a finite set of possible states $s \in \mathcal{S}$;
- Action space $\mathcal{A}$, consist of a finite set of possible actions $a \in \mathcal{A}$ that the agent can take;
- A reward function $R(s, a) \in \mathcal{R}$;
- A state transition probability function $P(s'|s_t, a_t)$: the probability of transitioning into state s' if starting in state s and taking action a, it represents the environments transition rules;
- A distribution of initial states $p(s_0)$.

The RL aims to learn an optimal policy π^*, so that following this policy, the expectation of the long-term return from the initial state is maximised. A cumulative return is the sum of rewards obtained in a fixed window of steps. A discount factor $\gamma \in (0, 1]$ is usually applied in cumulative return calculation to discount how far off in the future the reward is obtained. The discounted cumulative return R_t at time t is defined as:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{2.1}$$

Two terms are helpful for many RL algorithms: value function and action-value function(Q-function). The value function determines what is considered as good in the long run from a state s and always acts according to the policy π. The value function $V^\pi(s)$ is defined as the expected total discounted reward from a state s under the policy π.

$$V^\pi(s) = \mathbf{E}_\pi[R_t|s_0 = s] = \mathbf{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k}|s_0 = s] \tag{2.2}$$

Similarly, the action-value function represents the expected total discounted reward from a state s if the agent takes an arbitrary action a (which may not have come from the policy π), then forever after acts according to the policy π.

$$Q^{\pi}(s,a) = \mathbf{E}_{\pi}[R_t|s_t = s, a_t = a] = \mathbf{E}_{\pi}[\sum_{k=0}^{\infty}\gamma^k r_{t+k}|s_t = s, a_t = a] \tag{2.3}$$

The optimal value function and optimal action-value function are the functions with π being the optimal policy.

$$V^*(s) = \max_{\pi} \mathbf{E}_{\pi}[R_t|s_0 = s] \tag{2.4}$$

$$Q^*(s,a) = \max_{\pi} \mathbf{E}_{\pi}[R_t|s_0 = s, a_0 = a] \tag{2.5}$$

By unfolding and iterating the Equations (2.2) to (2.5), the following four self-consistence Bellman functions are presented as follows:

$$V^{\pi}(s) = \mathbf{E}_{a\sim\pi, s'\sim P}[R(s,a) + \gamma V^{\pi}(s')] \tag{2.6}$$

$$Q^{\pi}(s,a) = \mathbf{E}_{s\sim P}[R(s,a) + \mathbf{E}_{a'\sim\pi}[Q^{\pi}(s',a')]] \tag{2.7}$$

$$V^*(s) = \max_{a} \mathbf{E}_{s'\sim\pi}[R(s,a) + \gamma V^*(s')] \tag{2.8}$$

$$Q^*(s,a) = \mathbf{E}_{s'\sim P}[R(s,a) + \gamma \max_{a'} Q^*(s',a')] \tag{2.9}$$

The advantage function describes how much better it is to take a specific action a in state s under the policy π. It is defined as

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s) \tag{2.10}$$

During the research over the past decades, various RL methods have been proposed. The recognised taxonomy of RL algorithms divides the existing algorithms into two classes: model-free RL and model-based RL [79]. It is based on whether the agent has access or can learn a model of the environment.

Model-free RL

A majority number of research in RL is dedicated to model-free approaches. These strategies view the environment as a ‘black box’ and seek to develop a direct policy mapping through iterative interactions, without explicit modelling of the environment’s dynamics.

Recent advancements in model-free RL leveraged deep neural networks (DNNs) [31] to process raw input data and identify complex, high-level features. With the special layered architecture, DNNs can be employed as policy approximators in RL, and have achieved remarkable success across a broad spectrum of machine learning applications, such as pattern and speech recognition, computer vision and natural language processing [73,74]. This includes fields like robot manipulation [60] and in-hand dexterous manipulation [45], where deep learning-enhanced RL methods have demonstrated significant potential and effectiveness.

Current RL algorithms can be further categorised into policy-based, value-based and actor-critic approaches. This classification hinges on the involvement of the Q-function in the learning mechanism:

- **Policy-based**: Approaches such as Vanilla Policy Gradient [108], Deterministic Policy Gradient (DPG) [105], Trust Region Policy Gradient (TRPO) [99] and Proximal Policy Optimisation (PPO) [100] aim to optimise the parameters θ of a stochastic or deterministic policy $\pi(s, a|\theta)$. This is achieved by applying gradient descent on an objective function $J(\pi)$ to adjust the parameters θ of the policy. Policy-based methods offer gradual updates to the policy, ensuring stable convergence throughout the training process. However, because the update process relies on data samples from the current policy iteration, the methods typically operate on-policy and are therefore seen as less sample-efficient. The need for extensive online data collection poses a challenge to their practical application in real-world robot manipulation scenarios.

- **Value-based**: Another category within model-free methods employs the Q-learning [116] framework, known as value-based approaches. These algorithms aim to estimate the action-value function $Q_\pi(s, a)$ using the Bellman equation to derive optimum policy $Q^*(s, a)$. The optimal policy is linked to Q^* such that the agent's actions are determined by $a(s) = \arg\max_a Q_\theta(s, a)$. Deep Q-Network (DQN) [73], a representative algorithm in this group, leverages a deep convolutional neural network (CNN) to model the action-value function $Q(s, a)$, demonstrating its efficacy in playing Atari games. The Double Q-learning (D-DQN) [113] method introduces a dual Q-networks structure to simultaneously estimate the value function $V(s)$ and advantage functions $A(s, a)$ with the action-value function $Q(s, a)$ being the composite of these two. It shows better convergence compared with traditional Q-learning. In value-based methods, policy updates are indirect, with learning experiences stored in a replay buffer. Updates can utilise data from any stage of training, sampled from the buffer, allowing off-policy training. This decoupling of policy training from data generation enhances sample efficiency, making these methods appealing for some robot manipulation studies [124]. However, the indirect policy optimisation through training Q_θ introduces instability and potential slow convergence, making this approach generally less stable than policy-based methods.

- **Actor-Critic**: Strengths and weaknesses are revealed in both two categories of RL methods mentioned previously. Researchers have since discovered that policy optimisation and Q-learning can indeed be harmoniously integrated. In policy-based methods, the policy function, or actor, is optimised using the accumulated return R_t. By substituting this with a critic: either the action-value function $Q(s, a)$ or the advantage function $A(s, a)$, it becomes possible to synergize the two approaches, jointly updating the actor and critic through gradient descent. This integration allows the critic to bootstrap the actor, enhancing the optimisation process. Actor-critic algorithms, which employ both an actor network to determine policy functions and a critic network to evaluate value functions, exemplify this efficient optimization. Notable examples include the Advantage Actor-Critic (A2C) and its asynchronous variant, Asynchronous Advantage Actor-Critic(A3C) [72], Deep Deterministic Policy Gradient (DDPG) [66] and its enhanced iteration,Twin Delayed DDPG (TD3) [29], and Soft Actor-Critic (SAC) [35,36], etc. Thanks to the combined advantages of actor-critic methods, they have become a popular choice for tackling robot manipulation challenges [33,90].

Model-based RL

Model-free RL operates under scenarios where the dynamics of the environment are too intricate or outright unfeasible to model accurately. Nonetheless, there are instances, particularly within the realm of robotics, where the transition dynamics of the environment $P(s'|s_t, a_t)$ can either be explicitly defined or inferred through experiential learning [18,69]. Algorithms that leverage such environmental models for RL are categorised under model-based RL [17,76]. Coincidentally, the principles of model-based RL align closely with the concept of optimal adaptive control in the automotive control systems [56,58]. These approaches focus on learning the parameters of a predefined model structure, thereby facilitating the control process. The advantage of having an accurate model in model-based RL is significantly improved sample efficiency when compared to model-free approaches. However, model-based RL suffers from the issue of model identification in which the model can not be estimated accurately. Furthermore, the effectiveness of model-based RL is constrained by the fidelity of the estimated model, often limiting its overall performance [65].

2.2.2 Existing reinforcement learning works on robot in-hand dexterous manipulation

Research in RL that focuses solely on learning object manipulation at a symbolic level has been documented [49]. Yet, it was not until nearly a decade ago that significant advancements in RL research of InDex began to emerge.

The first demonstration of applying RL on dexterous manipulation skill [114] learns a policy to direct a two-fingered robotic gripper to adeptly rotate a cylinder between its fingertips. This study introduced a model-free approach known as non-parametric relative entropy policy search, designed for smooth learning for non-linear control policy. Tactile sensors positioned on each finger provided feedback on the interaction between the robot and the object. This work successfully demonstrated the potential of utilising model-free RL to teach dexterous manipulation skills to real-world robots. Despite its simplicity in terms of the robot hand design and the manipulation task used, compared to subsequent studies, it marked a significant step forward in the field.

Over the next few years after the initial explorations into dexterous manipulation using model-free RL, this topic has witnessed significant advancement, largely due to the integration of deep learning techniques and the use of deep neural networks for policy representation. A notable milestone was achieved by Katyal [48] in 2016, who demonstrated the first trial using a purely vision-based approach to performing in-hand manipulation RL. Their work was conducted on the ROS (Gazabo) simulation platform and involved a typical InDex task: controlling a five-finger humanoid robot hand to continuously reorient a cricket ball within its grasp. The policy was approximated using a 4-layer deep convolutional neural network, which processes 4 frames of 84×84 pixel grey-scale images to produce 18-dimensional action signals to control the joint movement of the robot hand. The DQN method was employed to facilitate the RL process. Although the author did not declare the condition of terminology for each episode and the definition of the reward, it is mentioned that the training had converged to a maximum average reward after approximately 11 million steps. This work presented an end-to-end deep RL solution to the InDex problem.

Subsequently, research efforts focus on exploring various model-free RL approaches to solve InDex tasks with a multi-finger anthropomorphic robot hand. These tasks are notably difficult due to the high degree of freedom and the extensive potential contact points the robot hand presents. The majority of the related works were conducted on MuJoCo (Multi-Joint Dynamics with Contact) [110] platform, a physical simulation engine that has gained widespread popularity in robotics studies. Prototypes of several robot hands, such as the Shadow Hand [93] and its variant, the ADROIT hand [55], were simulated within MuJoCo. A variety of in-hand manipulation tasks and sophisticated grasping activities as been simulated in Mujoco. Figure 2.2 illustrates several InDex and grasping tasks performed using the ADROIT hand developed within MuJoCo.

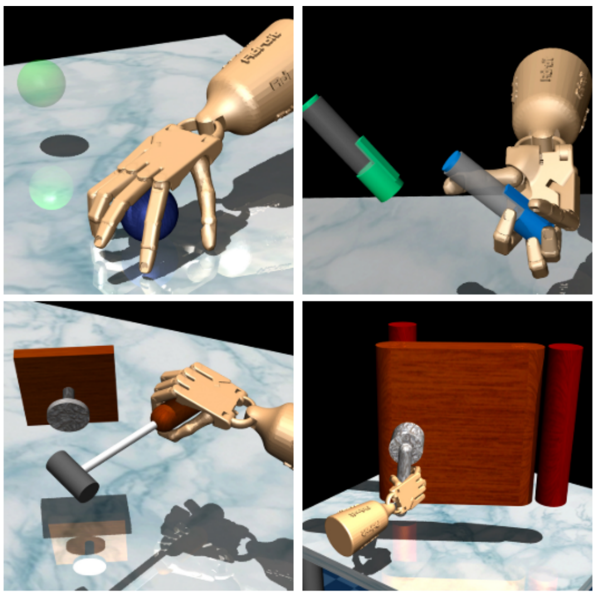


Figure 2.2: InDex tasks developed for ADROIT robot hand with MuJoCo physical engine, extracted from the original paper [90].

Mudigonda [75] presented an instance of model-free RL applied to an anthropomorphic hand in Mujoco physics engine, focusing on the fundamental task of grasping and lifting objects. The policy-based TRPO method was chosen as the learning strategy, employing a multi-layer perceptron (MLP) with two hidden layers to represent the Gaussian distribution for action sampling. Given the simulated setting, the agent had full observation of the joint position/velocity and the object position information. To encourage the robot's attempts at reaching for the object, even if not fully successful, the reward was designed to penalise the distance from the palm to the object. While the outcomes were promising, the training process was found to be particularly sensitive to various parameters like batch size, the initial standard deviation of the Gaussian policy, and the observation normalising.

Rajeswaran and Kumar [90] evaluated two types of model-free RL methods: A policy-based Natural Policy Gradient (NPG) [46] method, and an actor-critic-based DDPG method, across four different robot dexterous manipulation tasks in the MuJoCo simulation environment. These tasks included object grasping and relocating, re-orientating an object (such as a pen) within the hand, opening a door and using tools (specifically picking up a hammer and then driving the nail into a board). The findings underscored the importance of reward shaping, informed by human prior knowledge, for effective learning. NPG demonstrated better convergence rates over DDPG when the rewards were carefully crafted. Nonetheless, both methods struggled to learn most tasks when only sparse task completion rewards were provided, with the exception of in-hand reorientation. Further investigation revealed challenges with using pure RL approaches: the learned policies were not robust to changes in the environment (e.g., variations in object mass and size) and tended to produce peculiar and unnatural movements.

This research demonstrates the feasibility of addressing the InDex problem end-to-end using a model-free RL approach combined with deep neural networks, particularly in the context of high-dimensional, non-linear anthropomorphic hand robotic systems. However, due to the sample inefficiency of model-free RL, it is still not possible to directly conduct learning with a real robot hand. The learning processes mentioned previously are mostly done in the simulation environment. Furthermore, a common limitation across nearly all related studies is the necessity for task-specific reward engineering to achieve successful training or to expedite convergence. This requirement not only adds complexity but also lacks generalizability across different tasks. Additionally, the learning process is notably sensitive to the choice of hyper parameters, and the policies developed often exhibit limited robustness to changes in the environment.

Besides the InDex reinforcement learning (RL) examples previously discussed, which leveraged simulations, the research by Zhu and Gupta [128] aims to showcase the potential for directly acquiring complex multi-finger manipulation skills using model-free deep RL algorithms on actual robotic platforms. For their experiments, they utilised two robotic manipulators: a three-fingered D-claw and a four-fingered Allegro hand. The range of tasks undertaken in this study encompasses valve rotation, box flipping, and door opening. Figure 2.3 demonstrates the two robot manipulators performing these tasks.

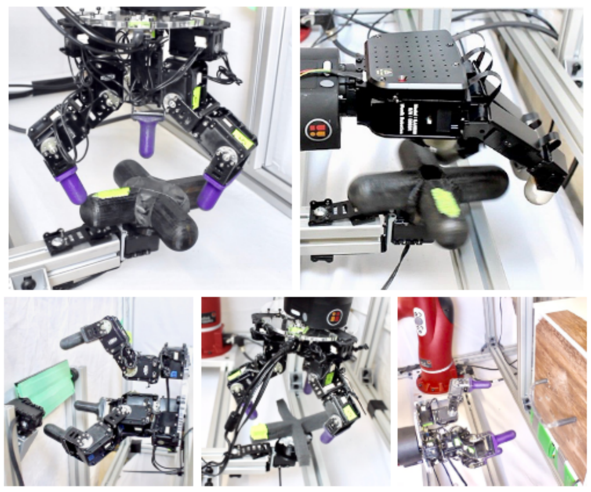


Figure 2.3: D-Claw gripper and Allegro hand used in work [128], duplicated from the original paper

This research employed the Truncated Natural Policy Gradient (TNPG) method, a streamlined variant of the NPG introduced by Rajeswaran et al. [91]. The learning phase was deemed complete, and the policy was considered successfully learned, upon achieving a 100% success rate across ten evaluations The findings revealed that model-free deep RL can learn these coherent manipulation skills in a time scale of a few hours. These outcomes suggest that direct training in real-world settings might offer more efficiency for certain tasks, and potentially lead to better policies. However, it was noted that while such learning processes could be conducted with limited data, this feasibility does not extend to more complex tasks.

Active research in the field suggests that using a deep neural network to control a multi-finger humanoid robot hand for dexterous in-hand manipulation is becoming increasingly feasible. In 2018, OpenAI showcased their research [3] to illustrate a systematic and extensive application of state-of-the-art deep RL methods for deriving a neural network policy solution to this complex control challenge. The task is to rotate an object (either a block or an octagonal prism) to a randomly chosen target configuration (goal) in hand. A new goal was set upon achieving the current one, continuing until the object was dropped or successfully handled 50 times. The control policy was designed as a recurrent neural network (RNN) with memory capabilities, specifically using the LSTM architecture [40]. The state-of-art policy-based RL strategy, PPO [100], was employed for training. To enhance training efficiency, the reward is augmented to include an additional reward of 5 for each goal achieved and a penalty of -20 for dropping the object. Training occurred on a simulated Shadow Hand model within the MuJoCo platform, with experience generation carried out by 384 worker machines, each equipped with 16 CPU cores, while network optimization was conducted on a single machine boasting 8 GPUs. Additionally, a separate vision-based pose estimator was trained via supervised learning to deduce the object's pose using images from three cameras. This estimated pose, along with the fingertips' positions provided by a 3D motion capture system, were input into the control policy. The optimally trained control policy from the simulation was then directly applied to a physical platform for testing. Figure 2.4 illustrates the system overview of their work.

The training results in the simulated environment are promising, showing that the robot hand can consistently perform approximately 30 consecutive reorientations without dropping the objects, applicable to both object types. By incorporating a diverse array of manually selected randomisation during the simulation training, when applied to a real platform, the robot hand managed to achieve 11 consecutive reorientations with a block and five with an octagonal prism. The performance approaches perfection when the ground truth object state is directly supplied by the simulation rather than estimated by vision. In such cases, the robot hand successfully completes all 50 reorientations in each trial.

Shortly after their initial work, OpenAI tackled a more challenging problem: solving a Rubik's cube with a humanoid robot hand [2]. To this end, they integrated an existing Rubik's cube-solving algorithm, the Kociemba solver, into their system to generate solution sequences. By employing a more accurately calibrated MuJoCo model and introducing a novel automated domain

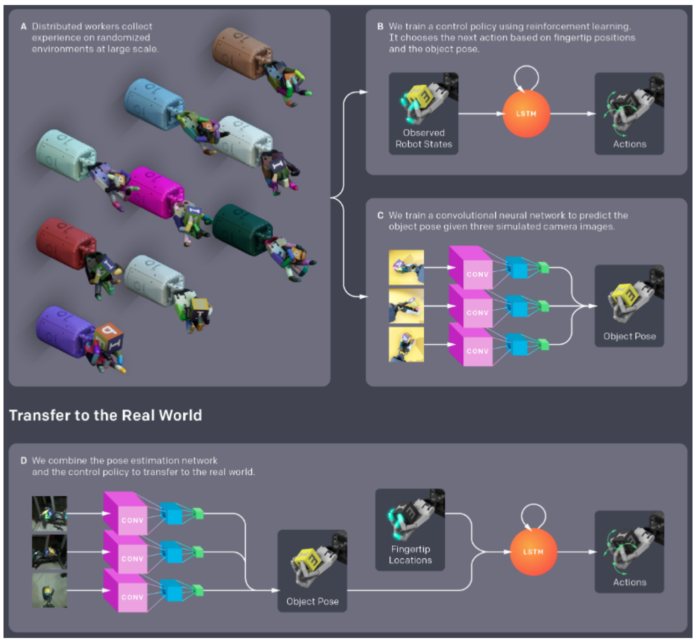

A Distributed workers collect experience on randomized environments at large scale.
B We train a control policy using reinforcement learning. It chooses the next action based on fingertip positions and the object pose.
Observed Robot States
LSTM
Actions
C We train a convolutional neural network to predict the object pose given three simulated camera images.
CONV
CONV
CONV
Object Pose
Transfer to the Real World
D We combine the pose estimation network and the control policy to transfer to the real world.
CONV
CONV
CONV
Object Pose
Fingertip Locations
LSTM
Actions

Figure 2.4: An system overview of [3], extracted from the original paper

randomization (ADR) method to facilitate curriculum learning in the training process, the resulting trained policy was able to solve the Rubik's cube through manipulation most of the time. Surprisingly, this policy demonstrated high robustness, effectively managing various perturbations. These two OpenAI demonstrations represent significant milestones in applying RL to InDex tasks using real humanoid robot hands.

Due to the nature of model-free RL, previous work on robot manipulation usually requires extensive reward design and engineering to guide the model toward proper convergence. This can be exceedingly challenging and requires expertise and domain-specific knowledge. Yet, in most practical RL scenarios, rewards are typically sparse and binary and is usually only given upon the completion of a task. Dealing with such sparse and delayed reward signals is challenging for continuous robot manipulation RL. Sample efficiency remains a primary hurdle in the application of RL to robotics, particularly for tasks characterised by sparse rewards where the agent frequently fails, thereby receiving predominantly negative reinforcement signals at the early stage of the training.

Hindsight Experience Replay (HER) [4] proposes a method for sample-efficient learning from sparse and binary rewards, avoiding the complexity of intricate reward engineering. HER is particularly suitable for multi-goal RL challenges (formally described in [97]), where goals are derivable from the state representation, and each goal g can be categorised within a set G of potential goals. HER draws inspiration from the human capacity to learn from failures. When replaying the experience that falls short of achieving a specific goal g, identifying instances where an alternative goal g' was inadvertently accomplished. This approach not only aids in learning what actions to avoid when aiming for the original goal g but also in discerning the appropriate actions to achieve the alternative goal g'. Under the context of HER, Training an agent to perform tasks in a continuous space becomes easier, thanks to the generalisation ability of goal space. As a cornerstone of our work, HER is explained in more detail in chapter 3.

In the foundational work introducing HER, it was combined with one of the state-of-art actor-critic off-policy RL method, DDPG [66], and applied to the simulation of a 7-DOF Fetch Robotics arm. Plappert and Andrychowicz [83] extended this approach to simulate a Shadow Hand robotic hand performing a series of in-hand reorientation and repositioning tasks. Compared to utilising DDPG, the combination of DDPG and HER generally results in significantly faster learning. Later, researchers [96] thoroughly benchmarked and compared the simple DDPG, a policy-based RL method PPO and DDPG+HER on several robotic manipulation tasks with both dense and sparse rewards. Results reveal that DDPG+HER achieves the best success rate over all InDex tasks with the sparse rewards applied. Interestingly, it was observed in [83] that the convergence rate is quicker with sparse rewards rather than with the dense rewards crafted. In another work [64], The DDPG+HER strategy has also been successfully employed to derive an operator policy for solving a Rubiks cube with a dexterous hand and has demonstrated effective performance.

Model-based RL approaches have also been deployed in InDex tasks in some studies, mostly through the combination of local trajectory optimisation and RL. In the work proposed by Kumar and Gupta [56], the task is to rotate a long tube placed on top of the hand click-wisely using the ADROIT robot hand. A trajectory-centric RL method is designed to learn a series of local time-varying linear-Gaussian controllers with different predefined initial states, via the linear-quadratic regulator (LQR) method. Sequentially, they train a deep neural network to generalise across a wider range of initial conditions with training data produced by the trained local controllers and compare the result with the nearest neighbour switching strategy. Their evaluation results show that the neural-network-based trained agent cannot compete with the nearest neighbour strategy regarding success rate.

Nagabandi and Konoglie [77] also utilise model-based RL to learn dexterous manipulation skills with multi-fingered hands and combine the advantage of deep neural networks in their work. They designed an online plan strategy with deep dynamics models (PDDM) to solve a range of InDex problems such as valve rotation, in-hand reorientation, handwriting, and manipulating Baoding balls. In their method, a deep neural network model is learned to represent the underlying system dynamics, then employ an online planning process with model-predictive control (MPC) to select actions. Their method is evaluated both in simulation and on the real robot hand platform (Baoding Ball rotation task). The results show that this method achieves substantially better results than other prior deep model-based RL methods. Compared with model-free RL, it requires much less training data thus the training process can be directly deployed on the real robots.

2.2.3 Challenges on reinforcement learning for robot in-hand dexterous manipulation

Previous research on RL for InDex tasks has explored a range of activities, with many studies demonstrating that these tasks can indeed be learned through RL. However, these investigations also uncovered several underlying issues and challenges that hinder the application of RL in addressing real-world InDex problems:

- The issue of **sample efficiency** remains outstanding and unresolved for model-free RL on InDex tasks. Even the most advanced current RL algorithms, mostly off-policy based, require substantial amounts of data for learning. For instance, in OpenAI's demonstrations, the Model-free training process required the involvement of hundreds of machines, and despite the massive computational resources, days or even months are needed to achieve the observed performance levels. The approach is still far from being practically applicable. Model-based algorithms, such as Guided Policy Search [61] and Probabilistic Inference for Learning Control (PILCO) [17] method, are generally more data-efficient. This is because they construct or train a derived model for the environment and then use this model to train the policy. Currently, model-based approaches are considered one of the most effective strategies for improving sample efficiency in RL for most robotics tasks [79]. However, for InDex tasks, due to the high complexity of the task itself which results in low model accuracy, there is not much positive progress. Alternatively, some approaches aim to generate augmented synthetic data for training, such as Hindsight Experience Replay (HER) [4]), to circumvent the need for directly collecting large volumes of training data.

- RL algorithms must engage in the cyclic process of generating new data through random actions and then training with the collected dataset. For multi-goal-based tasks [83], it is crucial for the agent to not only explore new goals but also to revisit previously learned ones. The trade-off between **exploration and exploitation** always exists for this process [120]. The policy can easily be trapped into local optima if this is not considered carefully. Each RL algorithm applies a different strategy to tackle this issue. However, discovering an efficient exploration strategy within the vast action or goal spaces typical of robotics tasks, which are both continuous and high-dimensional, continues to pose a significant challenge.

- **Generalisation** remains a challenge across all RL methods. Most trained policies are too fitted into the specific task that even minor changes in the environment or task can lead to failure. Additionally, reproducibility in RL is an underappreciated issue, influenced by factors such as network structure and random seeds used for initialisation [38], making it difficult to replicate results consistently.

- In terms of application, most prior studies utilise precise data on the objects' and robot joints' position/orientation as inputs. While such features are easily accessed in simulations, they pose considerable difficulties in real-world settings. The accurate **detection of an object's position and orientation** in real-time has become a notable hurdle. Vision-based solutions, which require setting up cameras around the robot, are commonly employed in RL experiments for robotic manipulation applications. However, this approach remains constrained to laboratory conditions, and the accuracy of the estimations significantly impacts manipulation performance. Investigating the integration of additional perception methods, such as tactile sensors, could be beneficial.

2.3 Imitation learning and in-hand dexterous manipulation related work

This section presents an overview of imitation learning (IL) and delves into various IL algorithms pertinent to robotic manipulation. Subsequently, it examines the current state-of-the-art in IL specifically for InDex tasks. The discussion concludes by summarising some of the prevailing challenges faced in applying IL to InDex tasks.

2.3.1 Introduction to imitation learning

The RL approach necessitates continuous interaction between the agent and its environment. In the initial stages of the learning process, the agent primarily engages in random exploration. As a result, the effort and time to collect sufficient data for the agent to learn the optimal policy is usually massive. Additionally, RL's effectiveness exceedingly relies on the reward signal. However, designing or obtaining suitable rewards that provide frequent and precise feedback is challenging in many learning environments. In some instances, rewards can be sparse, such as receiving a signal only upon successful task completion, or in more extreme cases, a direct reward function may be absent, such as in the context of teaching a self-driving vehicle. An inefficient reward system can lead to slow training speed in RL or prevent the learning process from converging to an acceptable solution.

In IL, the agent tries to learn the optimal policy by mimicking the behaviour of an expert, typically a human, who provides a set of demonstrations. These demonstrations are often represented as trajectories $\tau = (s_0, a_0, s_1, a_1, s_2, a_2, ...)$, showcasing the expert's decisions across different states. Similarly to RL, IL operates within an environment modelled by Markov Decision Process (MDP), characterised by a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition mode $P(s'|s_t, a_t)$ and an unknown reward function $R(s, a)$. The agent generates its action based on its policy π, striving to align it with the expert's 'optimal' policy π^*, from which the demonstrations derive. In some cases, the expert may provide additional demonstrations online, further aiding the agent's learning process. To facilitate this learning, a supervised learning algorithm equipped with a proper loss function is employed, enabling the agent's policy π to converge towards the expert policy π^* by learning from the state and action pairs extracted from τ.

The efficiency of IL in reducing exploration efforts and minimising trial-and-error is a significant advantage, offering a high-efficiency learning process [42]. IL is particularly beneficial when it is easier to collect desired behaviour demonstration from the expert than to specify an accurate and detailed reward function to learn the policy directly. Also, it is often merged with RL techniques to boost learning speed and accuracy. The IL process typically encompasses three stages: demonstration, representation and imitation learning algorithm itself [24].

Demonstration collection methods for robotic manipulation IL can be mainly categorised into two types: Direct and Indirect [56]. Direct demonstrations involve acquiring teaching examples directly from the robot, ensuring that the demonstrated state-action samples are highly compatible with the targeted robot system and can be readily transformed into learning data. Robots are typically controlled through kinesthetic teaching [89] or teleoperation [57] in this approach. Conversely, indirect demonstrations are collected in a separate environment where the robot itself is not involved. This method normally captures human motion through visual system [102] or wearable devices equipped with tactile sensors. Robots learn by observing images of human motion or contact signals. The indirect approach facilitates the convenient collection of a large volume of demonstrations. However, the transformation of state-action samples to the targeted robot system is less precise due to kinematic differences between humans and robots. Consequently, indirect demonstrations tend to be less optimal compared with direct demonstrations.

During the representation stage, characterised information is extracted from the demonstration for use in supervisor learning algorithms. Lower-level representations, such as time series of system states and actions, can be abstracted to form the trajectory τ. Additionally, some high-level symbolic representations can be derived to help identify the millstones of the task. The symbolic characteristics are particularly useful for dividing the task into several segments or stages, facilitating the learning process for complex and multi-step tasks.

The algorithm is pivotal in Imitation Learning (IL), enabling robots to replicate expert behaviour and generalise this behaviour across unfamiliar scenarios. The mainstream imitation algorithms applied in the field of robotic manipulation can be broadly categorised into the following types:

- **Behaviour Cloning (BC)**: this method represents the simplest form of IL, where the policy is derived through supervised learning from demonstration data. The agent is trained to precisely mimic the state-action mapping observed in the demonstrations. BC is notably efficient and excels in tasks that are simple and focused on a single target. However, the limited number of demonstrations cannot encompass the state-action distribution. Cumulative errors during operation add up and can put the policy into a state the demonstration has not covered. The behaviour in these unseen states can be unreachable, or not executable, and can lead to catastrophic failure. Consequently, a BC-learned policy often lacks the capacity to generalise to unseen states, making it less suitable for complex tasks that require long-term planning.

- **Inverse Reinforcement Learning (IRL)**: this method seeks to decipher the underlying reward function characterising expert behaviour from demonstrations, subsequently leveraging this reward function to identify the optimal policy via forward reinforcement learning [28]. The process of learning the reward function presupposes the expert's optimality, aiming to distil the intentions behind the expert's actions [52]. The implementation of the reward function can adopt a structured format, such as a linear combination of features, or a model-free approach, utilising neural networks, for instance. The strength of IRL lies in its ability to generalise the learned strategy from a limited set of demonstrations. A primary challenge in IRL is the potential for an infinite array of reward functions to correspond to the same expert policy. Approaches like maximum entropy IRL [1] have been developed to address this issue.

- **Generative Adversarial Imitation Learning (GAIL) [39]**: Inspired by the structure of IRL and the principles of Generative Adversarial Networks (GANs), GAIL employs a discriminator network to discriminate how much the data generated by the training agent is different from the expert demonstrations. The discriminators feedback then serves as the reward signal for training the agent (generator) using standard RL methods, encouraging the agent to produce data indistinguishable from the demonstrations. Eventually, it will converge to the expert as much as possible. GAIL can handle complex, high-degree-of-freedom robot manipulation tasks and demonstrates strong generalisation capabilities in unseen scenarios, Further developments have enabled GAIL to sometimes surpass the performance of expert demonstrations [130]. A more detailed discussion on GAIL is presented in Chapter 4.

2.3.2 Existing imitation learning works on robot in-hand dexterous manipulation

For InDex, the dimension of both state space and action space is notably higher compared with other robot manipulation challenges, yet the margin for achieving a successful solution is exceedingly slim. During the initial learning phases, a significant portion of the experiences tend to result in failure, underscoring the heightened challenge of exploration. This intense need for exploration substantially slows down the training process. Moreover, facilitating adequate exploration becomes particularly problematic when training involves a physical robot, due to the practical limitations and potential risks associated with extensive trial-and-error in the real world.

In the model-based RL study by Kumar and Gupta [56], a grasp and pick-up strategy is learned using IL. Expert demonstrations are captured via MuJoCo Haptix system [57], which employs a cyber-glove for teleoperating a robotic hand in a simulated environment, complemented by stereoscopic visualisation through OpenGL projection, simulating the user's viewpoint. Following a successful demonstration, a trajectory encompassing hand joint angles, object position and rotation, cylinder pressure vectors, and valve command signals are utilised for Behavior Cloning (BC). However, due to the distribution discrepancy between the simulated and real robot hands, the initial learned policy fails to pick up the object. To address this, the authors introduce a hybrid approach, merging Learning from Demonstration (LfD) with RL, leveraging the BC-derived policy as a foundation for RL. By integrating an additional shaping cost in the RL phase, they ensure the learning trajectory remains aligned with the expert demonstration, culminating in the acquisition of a successful policy.

In another study by Jain [45], BC is applied to acquire a deep visuomotor policy for various manipulation tasks, extending the work of [90]. They present an end-to-end framework in simulation that combines three continuous frames of camera images, robot joint sensors and tactile sensors that provide on-off contact signals. The demonstrations are acquired from a pre-trained simulation-based policy from previous work [90]. BC and a more advanced IL approach called Dataset Aggregation (DAgger) [94] are tried to learn the policy. The finding indicates that both IL methods can efficiently train a policy within a few hundred trajectories, with DAgger slightly outperforming BC in most scenarios. Notably, tactile feedback proves beneficial, especially in situations of object occlusion, enhancing learning efficiency.

The idea of object-centric demonstration which only demonstrates the motion of the object was adopted in InDex tasks in the work of Gupta and Eppner [34]. A soft robotic hand, where each finger is powered by an individual air valve, is used in this work. The affordability and adaptivity of this type of robotic hand make it suitable for a wide range of grasping tasks. However, it is extremely hard to model such a hand structure. The object-centric demonstrations were collected from humans. The authors formulated a methodology to train multiple controllers under diverse initial conditions, aiming to replicate the demonstration trajectories that the robot could most accurately mimic. An innovative algorithm was developed to determine the most suitable demonstration for imitation based on each initial state, excluding those unattainable by the robot. Following this, a neural network is trained to generalise across the assorted controllers, utilising the Guided Policy Search (GPS) framework [60].

In the previously mentioned model-free RL works [90, 128], the concept of demonstration-initialised RL has been explored. These two works utilise a technique known as Demo Augmented Policy Gradient (DAPG) for IL. DAPG melds demonstration data with policy gradient methods, starting with an initial policy crafted through Behavior Cloning (BC) and further refined by RL fine-tuning, incorporating an augmented loss function. Demonstrations were gathered via teleoperation using Virtual Reality (VR) technologies. Their teleportation system includes motion tracking and visual monitoring with Cyber-glove&HTC Vive, alongside the MuJoCo Haptix system for work in [90], while [128] employed kinesthetic teaching for data acquisition. The findings from these investigations indicate that DAPG performs slightly better convergence performance in training relative to BC and BC combined with Natural Policy Gradient (NPG), and outperforms other model-free IL methods like DDPG from Demonstrations (DDPGfD) [115].

Teleoperation via virtual reality and cyber gloves are considered expensive and complex to deploy. Later works [86] [5] have proposed vision-based teleoperation techniques to collect demonstrations. These methods only require a single camera to detect and estimate the human hand geometry, then perform motion retargeting to map the joint skeleton of the human hand to various robot hands. In these two works, DAPG is also used to conduct IL to learn several robot manipulation tasks including in-hand object rotation tasks.

Ruppel and Zhang [95] proposed a pioneering approach to learning control frameworks directly from human demonstrations, bypassing the need for teleoperation. They employed an instrumented glove to capture human finger and object motion trajectories along with tactile information. A trajectory-based reconstruction method is designed to regularise and reconstruct the collected demonstrations for the learning process. Subsequently, they trained two types of trajectory-based policy networks, a feed-forward structure and a recurrent structure, using the reconstructed demonstration data. Models for tactile feedback and object state were also developed as part of the learning framework. The trajectory commands generated by the policy network are converted into robot joint angles via online trajectory optimisation, facilitating the robot's execution of various manipulation tasks. This approach was tested on tasks ranging from simple object pick-and-place to more complex actions like whipping and opening a bottle lid. IL directly from human demonstration largely reduces the cost of data collection and allows humans to perform diverse tasks more naturally.

A similar idea applies to work presented by Qin and Wu [87]. They introduced a vision-based learning from demonstration approach called Dexterous Manipulation from Videos (DexMV) that applies IL directly from human demonstrations. Their key contribution lies in devising an optimisation-based method to convert human hand trajectories into robot hand demonstrations, employing pose estimation and motion retargeting. An overview of the DexMV learning pipeline is shown in Figure 2.5. For the learning aspect, they utilise and compare both state-action Imitation Learning with Generative Adversarial Imitation Learning (GAIL) [39] and Demo Augmented Policy Gradient (DAPG) [90], as well as a state-only Imitation Learning method, State-Only Imitation Learning (SOIL) [88]. The learning experiments focused on a variety of manipulation tasks in simulation, including object relocation, pouring from a mug, and placing an object within a container. The findings indicate that all IL methods, utilising the translated demonstrations, significantly surpass the baseline RL performance achieved with Trust Region Policy Optimisation (TRPO) [99], showcasing the effectiveness of leveraging direct human demonstrations for dexterous manipulation tasks.

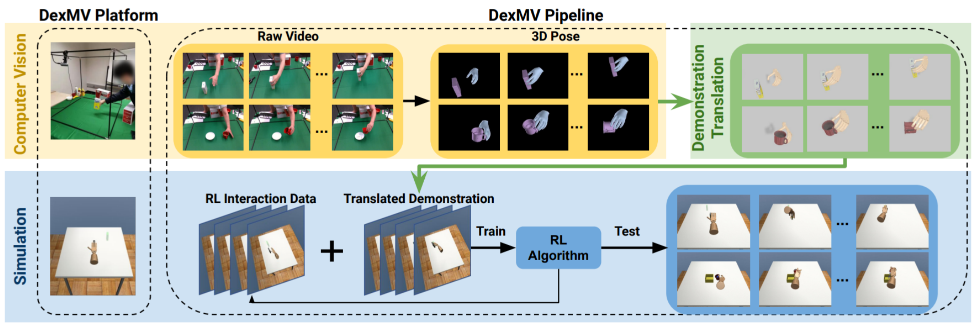


Figure 2.5: An overview of DexMV, duplicated from the original paper [87]

2.3.3 Challenges on imitation learning for robot in-hand dexterous manipulation tasks

Previous research in Imitation Learning (IL) for InDex tasks has explored various strategies for gathering and leveraging demonstrations. A range of algorithms has been developed to implement IL using these demonstrations. Despite these advancements, several challenges remain in the field:

- The traditional methods for **demonstration collection** in robot manipulation primarily involve kinesthetic teaching or teleoperation. Kinesthetic teaching, while straightforward, becomes challenging with complex robotic hands, such as those with five fingers, especially for tasks necessitating responsive and smooth movements. Teleoperation allows for direct use of collected demonstrations in learning, bypassing the need for data transfer. Glove-based teleoperation systems for humanoid hands tend to be expensive and require careful calibration for each operator. On the other hand, vision-based teleoperation systems offer a more cost-effective and user-friendly alternative, gaining popularity in recent research. Despite these advantages, teleoperation systems generally lack tactile feedback, making it difficult for operators to perform precise manipulations.

- The scope of demonstrations that can be collected is inherently limited. When it comes to multi-task learning or tasks with varying initial conditions, gathering a sufficient number of demonstrations to cover the entire exploration space presents a significant challenge. Consequently, **data augmentation** becomes necessary to increase the volume of available demonstrations for IL, enabling a more comprehensive understanding and replication of desired behaviours.

- In the realm of **IL algorithms**, a significant trend involves designing model-based IL approaches tailored to specific problems, enhancing their feasibility for application on physical robots. However, the intricacy of creating models for diverse tasks and robotic structures limits their adaptability for general use. On the model-free front, Behavior Cloning (BC) is seen as rudimentary and struggles with efficient learning in complex tasks requiring long-term planning, primarily due to mismatches in data distribution. The most utilised IL algorithms for InDex tasks include DDPG from Demonstrations (DDPGfD) and its variant, Demo Augmented Policy Gradient (DAPG), which have demonstrated commendable performance across various studies. Yet, other IL algorithms, such as Inverse Reinforcement Learning (IRL) and Generative Adversarial Imitation Learning (GAIL), have seen limited exploration within the context of multi-goal-oriented robot manipulation, suggesting potential avenues for advancing the field.

2.4 Summary

In this chapter, the fundamental concepts of RL and IL are explained, followed by a concise overview of some elementary algorithms within these domains. Additionally, the current state-of-the-art in robotic InDex was examined from three perspectives: traditional control approaches, RL, and IL. A summary and comparative analysis of the key aspects of recent RL and IL research, as outlined in the core papers discussed in this chapter, are presented in Table 2.1 and Table 2.2

Previous research on InDex with RL has illuminated novel pathways distinct from traditional non-learning methods. The fusion of deep neural networks with model-free RL approaches has yielded promising outcomes in tackling InDex challenges. Yet, learning efficiency remains a significant barrier, hindering the application of anthropomorphic robotic hands in practical settings. Among the various tasks explored, multi-goal-oriented robot manipulation tasks [83] have attracted considerable interest due to the intricate continuous goal space they present, posing substantial challenges to most existing RL methodologies. Hindsight Experience Replay (HER) [4] has emerged as a notably effective strategy for enhancing learning efficiency in these complex scenarios. The concept of re-labelling experiences in hindsight introduces a broadly applicable strategy for refining RL techniques in robot manipulation. Combining HER with recent advancements in RL to improve RL learning efficiency further is a promising research direction.

In Chapter 3, we delve into our contributions toward enhancing Hindsight Experience Replay (HER) for tackling multi-goal-oriented InDex tasks.We introduce two novel methods aimed at accelerating the learning process and addressing specific challenges identified in section 2.2.3 Our first innovation, Goal Density-based Hindsight Experience Prioritisation (GDP), introduces a prioritised experience replay mechanism designed to boost learning speed, which seeks to enhance sample efficiency, directly addressing the first challenge outlined in section 2.2.3. The second method, Policy-level-based Curriculum Goal Selection (PL-CGS), incorporates a curriculum-based goal selection strategy, tailored to optimise the learning pace further. PL-CGS tries to balance between exploration and exploitation during the learning process. which addresses the second challenge highlighted in the RL challenges in section 2.2.3.

Simultaneously, some work has investigated the IL approaches for InDex tasks. In terms of algorithms, a majority of the work has predominantly applied fundamental techniques like Behavior Cloning (BC) and Deep Deterministic Policy Gradient from Demonstrations (DDPGfD), along with their variations. However, more sophisticated and recent algorithms such as Inverse Reinforcement Learning (IRL) and Generative Adversarial Imitation Learning (GAIL) have seen limited exploration within the domain of multi-goal-oriented robot manipulation. This indicates a potential area for further research, suggesting that these advanced IL algorithms could offer novel insights and improvements for tackling the complexities associated with multi-goal-oriented tasks in robotic manipulation.

In Chapter 4, we present our innovative approach, Goal-based Self-Adaptive Generative Adversarial Imitation Learning (Goal-SGAIL), which tailors GAIL to multi-goal-orientated InDex tasks. Goal-SGAIL employs a strategy of applying hindsight to demonstration trajectories, thereby augmenting the data available for demonstration. This approach is further enhanced by incorporating several recent advancements in GAIL, aiming to significantly boost the efficiency of IL. Our methodology seeks to address the second and third challenges associated with IL, as detailed in section 2.3.3, offering a comprehensive strategy to overcome the hurdles faced in IL for multi-goal-oriented robotic manipulation.

Table 2.1: A summary of the core papers of RL and IL on InDex

Paper	Year	Platform	Robot	Tasks	RL Method	Inputs	Sim-to-real transfer	Imitation method	Demo data collection
van2015learning [114]	2015	Real	2-fingers	Roll object between fingertips	Model-free, Policy search	Tactile sensor, Joint angle, Distal link angle	–	–	
katyal2016hand [48]	2016	Sim(gazabo)	Shadowhand	In-hand object rotation	Model-free, DQN	Vision	–	–	
kumar2016learning [56]	2016	Sim(mujoco), Real	Shadowhand	Rotate long thick tube, Pick up long tube	Model-based and Adaptive optimal control	Joint angle, Object pos info	–	BC+RL	Teleoperation (CyberGlove)
gupta2016learning [34]	2016	Real	5-finger soft hand	Grasp bottle, Push beads on abacu, Turning valve	–	–	–	GPS	Kinesthetic teaching (Object only)
rajeswaran2017learning [90]	2017	Sim(mujoco)	Shadowhand	Grasp and relocate object, Re-orientate thick pen, Open door, Use hammer	Model-free, NPG & DDPG	Joint angle, Object info	–	DAPG, DDPGfD, BC+RL	Teleoperation (Haptix+CyberGlove)
mudigonda2018investigating [75]	2018	Sim(mujoco)	Shadowhand	Grasp and lift object	Model-free, TRPO	Joint angle&pos, Object info, Touch sensor	–	–	
andrychowicz2018learning [3]	2018	Sim(mujoco, unity), Real	Shadowhand	In-hand object rotation	Model-free, PPO	Vision, Joint angle, Object info, Fingertips location	Domain randamization	–	
plappert2018multi [83]	2018	Sim(mujoco)	Shadowhand	In-hand object re-orientate/re-position	Model-free, DDPG+HER	Joint angle and pos, Object info		–	
zhu2018dexterous [128]	2018	Real	D-Claw, Allegro	Open door, Rotate cross-shape valve, Box flipping	Model-free, TNPG	Joint angle, Object info	Domain randomisation	DAPG	Kinesthetic teaching (Robot&Object)
falco2018policy [23]	2018	Real	5-finger hand (3 finger used)	open cup, rotate bottle	Model-based, PILCO	Vision for object pos, Tendon position, Tactile on fingertips	–	–	
lowrey2018plan [69]	2018	Sim	Shadowhand	In-hand object re-orientate/re-position	Model-based, POLO (plan online learn offline)	Joint angle, Object info	–	–	
akkaya2019solving [2]	2019	Sim(mujoco, unity),Real	Shadowhand	Solving 3-dimension rubik's cube	Kociemba solver, Model-free operator(PPO)	Vision, Joint angle, Object info	Automatic Domain randomisation	–	
li2019learning [64]	2019	Sim(mujoco)	Shadowhand	Solving 3-dimension rubik's cube	IDA* solver, Model-free operator (DDPG+HER)	Joint angle, Object info	–	–	

Table 2.2: A summary of the core papers of RL and IL on InDex (Continued)

Paper	Year	Platform	Robot	Tasks	RL Method	Inputs	Sim-to-real transfer	Imitation method	Demo data collection
jain2019learning [45]	2019	Sim(mujoco)	Shadowhand	Grasp and relocate object, Re-orientate thick pen, Open door, Use hammer	–	Vision, Tectile Sonsor, Joint angle and pos	–	BC, DAGGER	Pre-trained expert policy
nagabandi2020deep	2020	Sim(mujoco) Real	Shadowhand	Baoding Ball rotation, Valve turning, In-hand object rotation, Hand writing	PDDM (Model-based)	Joint angle, Object info	–	–	
ruppel2020learning	2020	Real	Shadowhand	Object pick-and-place, Whipping, Opening bootle lids	–	Human hand trajectories, Tactile info	–	Policy network + object/tactile models + trajectory optimiser	Instrumented glove
gupta2021reset	2021	Sim & Real	D'Hand	Object insertation, Pick up and reorient	MTRF	Joint angle, Object info	–	–	
huang2021generalization	2021	Sim only	Shadowhand	in-hand object rotation	DDPG+HER	Joint angle, Object info	–	–	
qin2022one	2022	Sim & Real	Schunk Allegro Adroid	Object relocation Object flipping	–	Joint position (estimated from video stream)	Domain randomization	DAPG	Teleoperation (vision-based, singel-camera)
qin2022dexmv	2022	Sim only	Adroit	Object relocate Puring Place inside	–	Raw video	–	DexMV	Record human videos
pitz2023dextrous	2023	Sim & Real	DLR-Hand II	Re-orientate block	Modular RL(SAC-based)	Tactile(torque and position sensors)	Domain randomization	–	
arunachalam2023dexterous	2023	Sim & Real	Allegro	Box flipping, Knob spinning, In-hand object ratotion	–	Joint position (estimated from video stream)	–	BCRL, DAPG	Teleoperation (vision-based, singel-camera)

Chapter 3

Reinforcement Learning for In-Hand Dexterous Manipulation

In this chapter, we will explore multi-goal reinforcement learning (RL) with sparse reward, focusing on InDex tasks. We begin by introducing key concepts related to multi-goal RL, followed by a review of the most relevant work to our research. After highlighting the limitations of existing approaches, we propose two novel methods designed to enhance learning efficiency from distinct perspectives:

- Goal Density-based Hindsight Experience Prioritisation (GDP), which aims to improve learning efficiency by prioritizing certain experiences during the experience replay phase;
- Policy-level-based Curriculum Goal Selection (PL-CGS), aimed at enhancing learning efficiency through the automatic generation of goals that facilitate a curriculum learning process during training.

Both proposed methods undergo thorough evaluation in a simulated environment across a range of robotic manipulation tasks. The chapter concludes with a summary and discussion of our findings.

3.1 Related work

3.1.1 Multi-goal RL

In many robotic RL problems, agents learn to solve a multitude of related tasks, each characterised by seemingly different reward functions. For example, pick a load and place it in a specified location, where the target location varies for each task instance. In most cases, these 'meta-tasks' can be conceptualised as 'goals': a subset of the observation state space that defines the task's objective. For multi-goal tasks, where the goals occupy a continuous space, Universal Value Function Approximators (UVFAs) [97] have been proposed.UVFAs utilise a single value function approximator to learn across the goal space, demonstrating a strong generalisation ability to unseen goals.

Here is a formal description of a multi-goal RL problem: considering an RL problem modelled as a Markov decision process (MDP): $(\mathcal{S}, \mathcal{A}, R, P, p(s_0))$, with state space $\mathcal{S}$ and action space $\mathcal{A}$. In this context, goals are used here to describe the desired outcome of a task. A goal g is sampled from a possible goal space $\mathcal{G}$ and can be mapped from the observation states s by $g = f(s)$, which makes the goal space a subset of the state space $\mathcal{G} \subset \mathcal{S}$. Each episode starts with an original goal $g_0 \in \mathcal{G}$ initialised alongside the initial states s_0 and stays fixed throughout the whole episode. The reward function is not only dependent on states and actions but also is goal-conditional. RL aims to learn a universal goal-conditioned policy $\pi(s_t, g_0)$ that can maximise the cumulative reward with a reward function R_g across the whole goal space $\mathcal{G}$.

3.1.2 Hindsight Experience Replay (HER)

RL needs the environment to provide rewards as feedback for training. However, in the real world, the reward signal is usually sparse, meaning rewards are only given when successful. For multi-goal scenarios with sparse rewards, a positive reward is only given when the desired goal is considered achieved within a certain tolerance. Dealing with such sparse rewards is especially challenging for most existing RL algorithms. On the other hand, most experiences are unsuccessful during the earlier stage of learning and, therefore, cannot provide enough positive reinforcement when a sparse reward is applied. For continuous multi-goal RL problems, researchers have proposed Hindsight Experience Replay (HER) [4] to conquer the difficulties of sparse reward. It allows the agent to learn from its 'failure' by using achieved goals on their past trajectories as substitute goals to replace the original goals as the targets. The rewards are recalculated based on these substitute goals. This technique greatly increases the percentage of positive rewards, thus speeding up the learning process. When combined with the off-policy RL algorithm Deep Deterministic Policy Gradient (DDPG) [66], it shows a much better learning efficiency on OpenAI multi-goal robotics tasks [83].

In a multi-goal RL problem, at each time step of an episode, an achieved goal g' can be extracted from the states s. By comparing the achieved goal g'_{t+1} of the next time-step $t+1$ and the desired goal g_d, the reward of each time-step t can be represented as $r_{g_d}(t) = -[f(g'_{t+1}) = 0], f(g') = |g' - g_d| < \epsilon$, where ϵ is a fixed tolerance. The reward is -1 for unsuccessful and 0 for successful time steps. As an off-policy algorithm, HER stores the experience in a replay buffer. Observation states, actions, the desired goal (do not change during the episode), and the current achieved goal are stored as one data tuple (s_t, a_t, g'_t, g_d) in the replay buffer. During the replay stage, some experiences are modified to use the achieved goals from the future time steps as substitute goals g_s to replace the original desired goals g_d. an achieved goal from the future time-step g'_{future} of the same episode is used as substitute goal g_s to replace the original desired goal g_d. The reward of these experiences will be calculated based on the substitute goals. $r_{g_s}(t) = -[f(g'_{t+1}) = 0], f(g') = |g' - g'_{future}| < \epsilon$. By using this approach, the RL algorithm can learn to achieve a goal even if that goal is not the original.

Although HER can be considered the current state-of-the-art method for multi-goal sparse reward RL, it can still be improved in several ways. There are two directions to improve learning efficiency: setting new goals to optimise the experiences gained (i.e. goal generation [92]) and efficiently exploiting existing experiences in the buffer.

3.1.3 Experience Prioritisation for HER

Off-policy RL algorithms leverage previous experiences stored in a replay buffer for training. Typically, these experiences are uniformly randomly sampled from the replay buffer and utilised to update the policy and value approximators.

The importance of experience prioritisation was highlighted by Schaul [98], who proposed a method for prioritizing experiences based on their higher temporal-difference (TD) error. TD error quantifies the discrepancy between the ultimately correct reward and the current prediction for a sampled experience. Experiences with higher TD errors are indicative of potentially greater losses for the current policy, thus holding more value for training. While this method effectively identifies valuable experiences, it is computationally demanding. It necessitates recalculating the TD error for all experiences each time the networks are updated, making it resource-intensive.

Another method, Energy-based Prioritisation (EBP), as proposed by Zhao et al. [126] utilises the trajectory energy to indicate the difficulty level of the achieved trajectory. The trajectory energy is calculated as the sum of three types of transition energies (potential, kinetic, and rotational) of the target object along the movement trajectory. The trajectories with higher trajectory energy are deemed more challenging but still achievable for the agent, and therefore, are prioritised for training.

HER primarily learns from the hindsight experience, where the substitute goal distribution is biased from the original goal distribution [125]. Most achieved goals cluster around the starting states and are overrepresented as substitute goals, risking the agent's overlearning from a homogeneous set of easily achieved goals. To mitigate this 'hindsight bias' in HER, various methods have been proposed. Naive approaches include filtering out experiences that do not cause the object to move [71] or whose final achieved points are not sufficiently close to the original targets [80]. Other strategies focus on prioritizing experiences deemed more instructional [71] or more interesting [127] to the agent.

A universal approach is to apply entropy-regularisation over goal states. The Maximum Entropy-based Prioritisation (MEP) method [125] estimates the entropy of the trajectory and prioritises those with higher entropy, determined by the probability distribution of trajectory goals. The rarer a trajectory is, the more likely it is to be sampled for replay. This approach introduces the concept of trajectory entropy, highlighting the value of rare experiences for learning. However, it applies goal density fitting for each trajectory, and experience prioritisation remains confined to the trajectory level, with experience sampling within a trajectory still being random.

3.1.4 Curriculum Goal Generation for Multi-goal RL

Curriculum learning was first introduced and explored within the realm of supervised learning [8]. The core idea involves optimising the sequence in which experiences are utilised for learning by the agent [78]. In the RL domain, curriculum learning modifies the distribution of training data by adjusting the learning situation according to the capabilities of the RL agent. This can be implemented at the data collection stage, employing a series of auxiliary tasks to guide policy optimisation [27], or at the data exploitation stage, organising a sequence of experiences for replay from the data stored in the buffer [14]. Our discussion will primarily focus on the former scenario.

In multi-goal RL, goals are typically generated randomly within the goal space $\mathcal{G}$, which is naive and cannot provide progressive exploration for the overall task. In the initial stages of learning, some goals may be too challenging for the agent to achieve, offering little to no positive reinforcement. Conversely, in later stages, it becomes crucial for the agent to explore more novel and challenging goals while revisiting previously learned goals to prevent forgetting past learnings. To accommodate this, it's advisable to enable the agent to set its own goals, strategically selecting those of intermediate difficulty that are most appropriate for its current stage of learning [84].

This objective is realised in GoalGAN [26], where a generator, based on a Generative Adversarial Network (GAN), is trained to propose goals of intermediate difficulty, referred to as GOID. In this framework, a 'policy-level' measure is introduced to assess the difficulty of goals relative to the current capabilities of the agent. A generator and a discriminator are trained in tandem with the learning agent: the discriminator learns to identify goals that are suitably challenging for the agent at its current level, while the generator is trained to produce these optimally challenging goals. Although this method has been applied to a series of simple 2D navigation problems, it has yet to be evaluated on more complex 3D robot manipulation tasks. Furthermore, the authors have not yet explored integrating their algorithm with hindsight experience replay (HER).

Rather than training a GAN to generate suitable goals, the Curriculum Goal Masking (CGM) method proposed by Eppe et al. [22] employs masks on goals (sub-goal combinations) to generate various combinations of sub-goals and to associate different levels of difficulty with distinct goal masks. The authors posit that sub-goals sharing the same goal mask have equivalent difficulty levels. By evaluating the success rate of sub-goals generated by each goal mask based on the most recent rollouts, the method selects the goal mask that achieves an optimal 'Goldilocks' difficulty level, thereby maximising learning efficiency. However, this method faces scalability issues, as the number of goal masks increases exponentially with the dimensionality of the goal space, making it impractical for tasks involving high-dimensional robotic manipulation, such as those in the InDex context. Furthermore, CGM is better suited to tasks that can be decomposed into discrete sub-tasks, like reach-and-push or reach-grasp-and-move.

A more straightforward method for assessing the difficulty of robotic manipulation tasks involves using distance as a metric. Hindsight Goal Generation (HGG), proposed by Ren et al. [92], aims to identify achieved goals from previously visited trajectories that are closest to the target goal distribution. These selected achieved goals are then utilised as new objectives for guided exploration. Initially, the method randomly selects a set of goals within the target goal region and employs a bipartite matching technique to find a subset of achieved goals from the replay buffer that closely matches these selected goals. This strategy has been shown to accelerate learning speed across a variety of custom OpenAI goal-oriented robotic manipulation tasks, particularly in scenarios where the initial starting region does not overlap with the target region. However, it has not yet been evaluated in a more generalised environment setup. Additionally, while the method traditionally calculates the distance between two goals using Euclidean distance, this metric may not accurately reflect the complexity of in-hand object rotation tasks, where goal distances should ideally be measured by orientation differences.

3.2 Goal-density based Prioritised Experience Replay (GDP)

In this section, we concentrate on the question of *how to efficiently exploit the experience collected during the learning process.* To address this, we analyse the density distribution of achieved points, giving priority to those that are infrequently encountered in the replay buffer for use as substitute goals in Hindsight Experience Replay (HER). Furthermore, we introduce the Prioritisation Switching with Ensembling Strategy (PSES) method. This approach involves alternating among different experience prioritisation algorithms throughout the learning process, thereby enabling the selection of the most effective strategy at each stage of learning.

3.2.1 Methodology

Definition of goal-pair

During the replay stage, HER first samples trajectories and then samples an achieved goal from each trajectory to be the substitute goal and synthesis a new experience. This essentially means that for each trajectory, the number of possible experiences for HER to sample equals the number of trajectory time steps. The total number of possible experiences that could be sampled for replay buffer is the number of trajectories m times the trajectory length n in time steps (assuming that all trajectories have the same length).

Our algorithm is inspired by the Maximum Entropy-based Prioritisation (MEP) [125]. Original MEP estimates the probability distribution of the goal-state trajectories $p(\tau^{g_d})$ and prioritises the rare trajectories in the overall trajectory density distribution for replay. However, we seek to estimate the density distribution of the possible experiences that could be sampled for replay instead.

Assuming that only one specific route exists from each initial achieved point p_0 to each substitute goal point p_s, each experience could be represented by the pair of an initial point and a substitute goal point (p_0, p_s). In HER, the initial point p_0^τ of an episode trajectory τ is essentially the achieved goal of the initial time-step, and here we define it as initial goal g_0^τ. Since a substitute goal point p_s^τ of trajectory τ can only be sampled from the future achieved goals, an achieved goal g'^τ at each time-step (except for the initial time-step of τ) can be used to express its possible substitute goal point. For each trajectory τ in the replay buffer, there exists n achieved goals, thus n sets of such goal pair (g_0^τ, g'^τ) that can represent all possible experience samples exist. In our algorithm, all pairs of initial and achieved goals (g_0, g') of all trajectories are used to estimate the goal density. The total number of elements in the density distribution of m trajectories is $m \times n$.

The Algorithm

The complete training algorithm for HER with GDP is provided in Algorithm 1. In short, GDP constructs goal-pairs with initial goals and achieved goals during the data collection stage and uses them to fit a goal-density model. The density values are used to prioritise those achieved goals with lower goal-density for HER to use as substitute goals. Using this method, the substitute goal distribution for HER is less biased, and rare experiences are utilised more often during the learning phase. Each step of the algorithm is explained below:

Data collection and storage: The data-collecting process of the proposed method is the same as the original HER. At the beginning of each episode, an initial state s_0 and a desired goal g_d are randomly sampled. At each time step, the agent takes an action a_t according to the current policy with randomisation, and a new state is reached afterwards. Each episode has a fixed number of time-step n.

For each episode trajectory τ, the original HER records the observation states s, the actions a, all achieved goals g' including the initial goal g_0 and the desired goal of the episode g_d. In GDP, the goal-pair of each time-step's initial goal and achieved goal t as $pair_t = (g_0, g'_t)$ are also recorded and stored. Each trajectory records n sets of goal-pairs. These goal-pairs are used for goal-density estimation.

Goal density estimation: After some episodes of experience are collected, the density model ρ is fitted using the goal-pairs stored in the buffer. A variational Bayesian Gaussian Mixture Model (GMM) is used to represent the density distribution of the goal-pairs. A finite mixture model with Dirichlet distribution is used as the type of weight concentration prior. The Expectation Maximisation (EM) algorithm is used to estimate the parameters of the GMM. The density model is fitted every epoch for all goal-pairs saved in the replay buffer:

$$\rho_i = GMM(pair_i). \tag{3.1}$$

Then, the density is normalised as follows:

$$\bar{\rho_i} = \frac{\rho_i - \rho_{min}}{\sum_{n=1}^{N}(\rho_n - \rho_{min})}, \tag{3.2}$$

where ρ_{min} is the minimum value of all density values, N represents the current number of data points in the replay buffer. After normalisation, all density values are within the range $0 < \bar{\rho} < 1$ and stored in the replay buffer for prioritisation afterwards.

Experience replay with goal prioritisation: During the HER sampling stage, the normalised density values of goal-pairs are used to prioritise the achieved goals as HER substitute goals. To ensure that rare experiences with smaller density have higher prioritisation, the complementary density values are ranked and the ranking number $rank(1 - \bar{\rho})$ is directly used to calculate

Algorithm 1 HER with Goal Density-based hindsight experience Prioritisation (GDP)

```
Require:
    - An off-policy algorithm $\mathbb{A}$    ▷ e.g. DDPG, DQN
    - A goal-based reward function $\mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$    ▷ e.g. $r(s, a, g_d) = -1$ if fails, 0 if success
1:  Initialise actor and critic networks for $\mathbb{A}$
2:  Initialise replay buffer $\mathbb{R}$
3:  for $epoch = (1, L)$ do
4:      for $episode = (1, M)$ do    ▷ Step 1: data collection and storage
5:          Sample an desired goal $g_d$ and initial state $s_0$
6:          Extract the initial goal $g_0$ from the initial state $s_0$
7:          for $t = (0, T-1)$ do
8:              Sample an action $a_t$ using the behaviour policy $\pi$ from $\mathbb{A}$
9:              Execute the action at and observe a reward $r_t$ and a new state $s_{t+1}$
10:         end for
11:         Store the trajectory $\tau = (s_t, g_d, g'_t, a_t, r_t)_{t=0}^{T}$ in the replay buffer $\mathbb{R}$
12:         Store the goal-pairs $pair_t = (g_0, g'_t)_{t=1}^{T}$ of trajectory $\tau$ in $\mathbb{R}$
13:     end for
14:     Fit the goal-density model using the goal-pairs in $\mathbb{R}$    ▷ Step 2: goal density estimation
15:     Update the densities $\rho$ and $\bar{\rho}$ using Eq. (1) and (2) for all samples in $\mathbb{R}$
16:     for $batch = (1, N)$ do    ▷ Step 3: experience replay
17:         Sample a set of achieved goals $g'_i$ based on the probability $p(g'_i)$ using Eq. (3)    ▷ goal prioritisation
18:         Sample transitions $(s_t, g_d, a_t, g'_t, s_{t+1}, g'_{t+1})$ based on $g'_i$
19:         $r'_t = r(s_t, a_t, g_s)$ using the achieved goals as substitute goals $g_s = g'_i$    ▷ recalculate reward (HER)
20:         Store resembled transitions $(s_t, a_t, s_{t+1}, r'_t)$ in the mini-batch $\mathbb{B}$
21:         Perform one step optimisation using $\mathbb{A}$ and $\mathbb{B}$
22:     end for
23: end for
```

the probability for sampling. For an achieved goal g_i' in trajectory τ with goal-pair represented as $pair_i = (g_0^\tau, g_i')$, where g_0^τ is the initial goal of trajectory τ, its goal density is $\rho_i = GMM(pair_i)$. Its probability is then sampled for replay as follows:

$$p(g_i') = \frac{rank(1-\bar{\rho_i})}{\sum_{n=1}^{N}[rank(1-\bar{\rho_n})]}. \tag{3.3}$$

If an achieved goal g_i' from time-step i in trajectory τ is sampled for replay, a transition (s_t, a_t, g_t', g_d) is then uniformly sampled from the same trajectory τ, but before the achieved goal time-step i, which means $t \in (0, i)$. The achieved goal g_i' is used as substitute goal $g_s = g_i'$ to replace the original goal g_d for this transition to calculate the reward using the HER algorithm.

Algorithm 2 Prioritisation Switching with Ensembling Strategy (PSES)

```
Input:
    - RL algorithms $\mathbb{A}_1$, $\mathbb{A}_2$, $\mathbb{A}_3$ ... with different prioritisation method.  ▷ e.g.
    GDP, MEP, PER etc.
    - A time window $T$                                                  ▷ e.g. 5 epochs
Output:
    - The outcome algorithm $\mathbb{A}_o$
1: Initialise the selector $S$ with one of the algorithms.
2: for $epoch = 1, L$ do
3:     Switch outcome algorithm to the selector $\mathbb{A}_o \to S$
4:     if $epoch == T$ then
5:         Calculate the average performance for all algo-
           rithms $\mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_3$...
6:         Point the selector to the best algorithm with the
           highest average performance $S \to \mathbb{A}_b$, where
           $\mathbb{A}_b \in \{\mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_3 ...\}$
7:     end if
8: end for
```

Prioritisation Switching with Ensembling Strategy (PSES)

There exists a number of experience prioritisation methods that can be combined with HER [98,125,126] in addition to GDP. Each method benefits different types of tasks and improves learning efficiency at different training stages. For an unknown task, it is hard to decide which approach to be used that can provide the best performance. In our work, an ensembling strategy is proposed that utilises several different prioritisation methods simultaneously and achieves the overall best performance at all learning stages. This strategy is inspired by ensembling learning, in which multiple algorithms are trained to solve the same problem. The prediction of all algorithms is combined to get a better prediction.

During the learning process, several algorithms with different prioritisation methods run in parallel. The ensembling selects algorithms based on a time window strategy. A time window is used to analyse the average performance of each method by stages. The average performance within a time window is defined as the average test success rate during this time window. The method selection of each time window is based on the method with the best average performance of the previous time window. The complete algorithm for *Prioritisation Switching with Ensembling Strategy* (PSES) is summarised in Algorithm 2.

When combining different prioritisation methods with PSES, the method used as the output is adjusted on the fly. This ensures the best-performed method is always chosen at each stage, and a more monotonic learning curve is achieved. One can reasonably expect that the finest time window leads to the best outcome performance. However, larger time windows sometimes still achieve relatively good results and require less computing power. In the next section, we present experiments that optimise the time window.

3.2.2 Experimental Results

Environments

The proposed GDP method is tested on various OpenAI Gym simulated robotic manipulation tasks, see Figure 3.1. These tasks are all multi-goal scenarios that are challenging for evaluation:

- *FetchPush-v1*: A block is placed on a table in this task. A random goal position is chosen on the table, and the fetch robot arm needs to manipulate its end-effector to push the block to reach the goal position.
- *FetchSlide-v1*: A block is placed on a long slippery table. A random goal position outside of the robot's reach is chosen. The fetch robot needs to use its end-effector to hit the block with a force so that it slides towards the goal position and stops at the goal position due to friction.

- *FetchPickAndPlace-v1*: A random 3D position is chosen in the air above the table. The fetch robot needs to grasp a box originally placed on the table to reach the goal position.

- *HandManipulateEggRotate-v0*: An egg-shaped object is placed in the palm of a shadow robot hand. The robot hand needs to manipulate its fingers to rotate the egg to reach a goal orientation, which is different from the original orientation.

- *HandManipulateBlockRotateXYZ-v0*: Similar to the egg rotation task, the shadow robot hand needs to manipulate its finger to rotate a block-shaped object in its palm to reach a new orientation goal.

- *HandManipulatePenRotate-v0*: Similar task for the shadow robot hand needs to rotate a long thick pen-shaped object placed in its palm to reach a new orientation.

The length of each episode is fixed for these environments: 50 time steps for fetch environments and 100 time steps for hand environments. The reinforcement learning setup for these environments is described below:

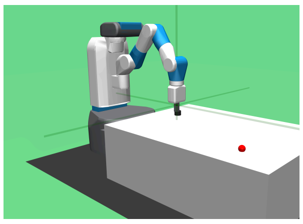
Observation: the position and velocity of the robot's gripper/joints and the object's position, rotation, and velocity. For fetch environments, it also consists of the object's relative position and velocity to the gripper.

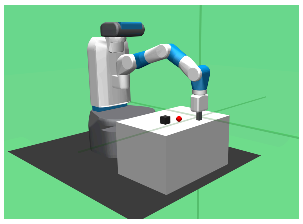
Desired goal g_d: a vector containing the object's target position and rotation (only for hand environments). The environment randomly generates the desired goal and keeps its value throughout the episode.

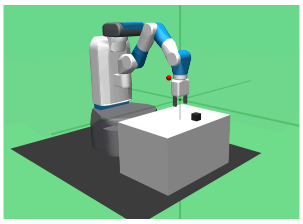
Achieved goal g': extracted from the observation and provided as a separate vector. It is constructed by the object's current position and rotation (only for hand environments). The achieved goal has the same dimension as the desired goal.

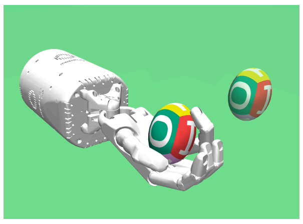
Reward: binary and sparse. It is 0 when the desired goal is considered achieved, which means the difference between the desired goal and the achieved goal after the robot takes action is within a threshold ϵ, otherwise -1.

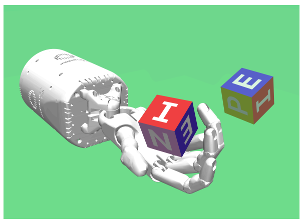
Action: For fetch environments, it includes the positions to control the robot gripper's movement and the binary signal to control the opening and closing of the gripper. For hand environments, it consists of the absolute positions to control the non-coupled hand joints for hand environments.


(a) FetchSlide


(b) FetchPush


(c) FetchPickPlace


(d) HandEgg


(e) HandBlock

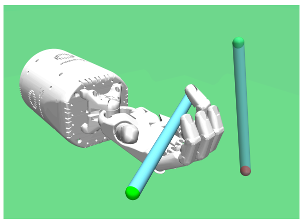

(f) HandPen

Figure 3.1: OpenAI Gym robotics multi-goal RL environments: These environments are created with the MuJoCo [110] physical simulation platform and wrapped with API functions for RL training. The first three environments are to slide, push or pick and place a small block to reach the goal position in 2D/3D space with the 7-DoF fetch robotics arm. The latter three environments fine-manipulate an object (egg/block/pen) within hand to reach the goal position and orientation in 3D space with the 24-DoF Shadow robotics hand. More details about the setup for these environments can be found in [83].

Implementation detail

We use the vanilla HER method as the baseline and compare it against MEP and our GDP methods. In our experiments, we consider each cycle to be a combination of 40 episode data collection loops and 40 batches of network updates with mini-batch size 5120. Every 50 cycles are regarded as one epoch. The hyper-parameters used for DDPG and HER are the same for all comparison experiments. After each epoch, we update the density model for MEP and GDP and do the evaluation. Each task's experiments are carried out using five randomly selected seeds, with the seeds being consistent across the different learning methods.

Performance comparison

The learning performance of DDPG is combined with three different types of experience utilisation and prioritisation methods in 6 robotics environments.

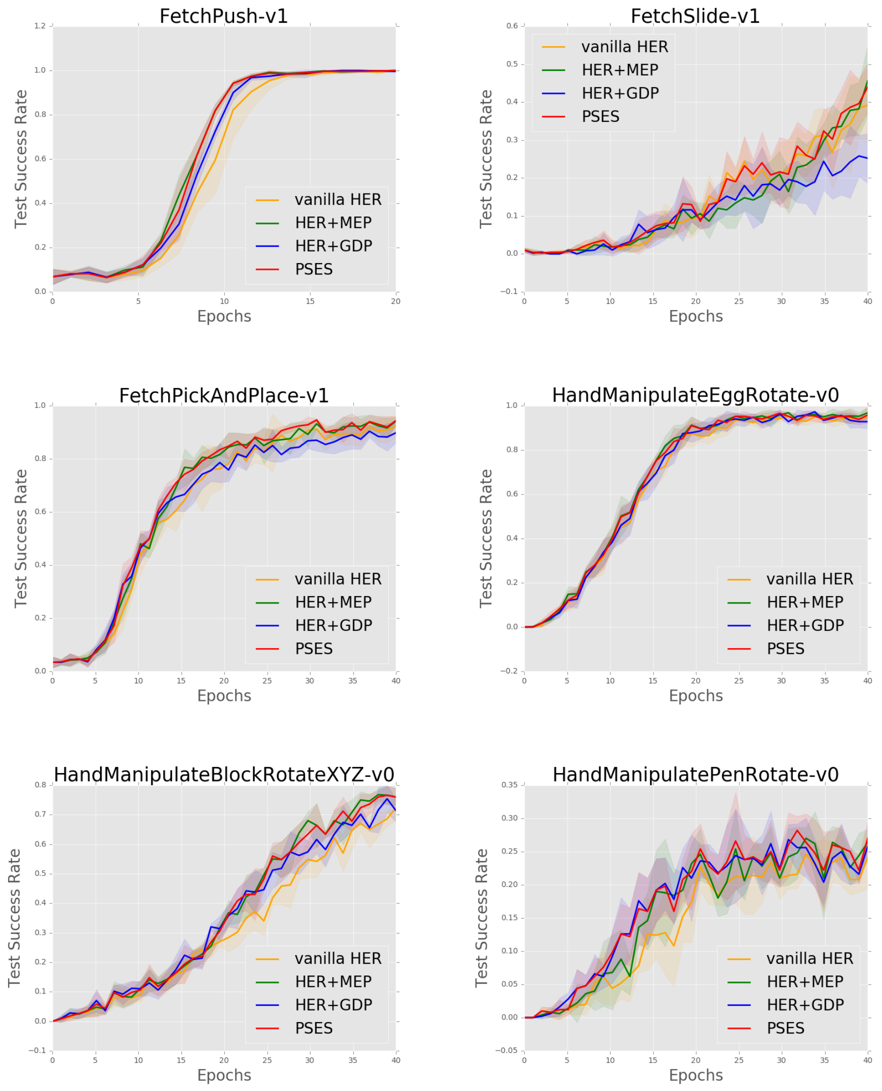


Figure 3.2: The learning curve for different methods in all six multi-goal robotics environments. The orange, green, and blue lines represent the learning curve for vanilla HER, HER+MEP and HER+GDP, respectively. The red lines are the performance of the PSES method with all three approaches combined. The time window used for PSES is two epochs for all tasks.

FetchPush					**FetchSlide**				
stage	1	2	3	4	stage	1	2	3	4
seed1	0	0.05	0.08	0.01	seed1	0	-0.01	0.04	0.05
seed2	0	0.04	0.06	0	seed2	-0.01	-0.01	0.06	-0.10
seed3	0.01	0.01	0.05	0	seed3	0	0	-0.02	0.04
seed4	-0.01	0.10	0.07	0	seed4	0	-0.02	-0.03	-0.11
seed5	0	-0.07	0.01	0	seed5	-0.01	-0.04	-0.11	0.09
mean	0	0.03	**0.05**	0	mean	0	-0.01	-0.04	-0.01
std	0.01	0.06	**0.02**	0	std	0	0.02	0.05	0.08
p-value	1.000	0.139	**0**	0.083	p-value	0.168	0.067	0.002	0.650
FetchPickAndPlace					**HandManipulateEggRotate**				
stage	1	2	3	4	stage	1	2	3	4
seed1	0	-0.02	-0.02	0	seed1	-0.03	-0.04	0.01	0.01
seed2	0	0	-0.05	0.02	seed2	-0.01	0.03	-0.01	0.01
seed3	0.04	-0.04	-0.05	-0.08	seed3	0.01	0.03	0	0
seed4	0.03	0.07	0	0	seed4	0.02	0.03	0	-0.03
seed5	0.01	0.06	0.07	0.04	seed5	0.01	0	-0.03	-0.02
mean	**0.01**	0.02	0.01	-0.01	mean	0	0.01	-0.01	-0.01
std	**0.02**	0.04	0.04	0.04	std	0.02	0.03	0.01	0.02
p-value	**0.014**	0.089	0.186	0.073	p-value	0.806	0.255	0.215	0.185
HandManipulateBlockRotateXYZ					**HandManipulatePenRotate**				
stage	1	2	3	4	stage	1	2	3	4
seed1	0	0.08	0.05	0.03	seed1	0.02	0.07	0.05	0.02
seed2	0.01	0.04	0.09	-0.03	seed2	-0.01	0.05	0.01	0.03
seed3	0	0.01	0.05	0.02	seed3	0.03	0.06	-0.01	0.03
seed4	0.01	0.05	0.05	0.01	seed4	0.02	0.11	0.03	0.01
seed5	0	0.02	0.07	0.02	seed5	0.02	0.07	0.02	-0.01
mean	0	**0.04**	**0.06**	0.01	mean	**0.02**	**0.07**	**0.02**	**0.02**
std	0	**0.02**	**0.02**	0.02	std	**0.01**	**0.02**	**0.02**	**0.01**
p-value	0.098	**0**	**0**	0.184	p-value	**0.001**	**0**	**0.002**	**0.011**

Table 3.1: The Statistical analysis for stage improvements for GDP: The stage improvement during each learning stage is calculated for each run with a different seed. The mean and standard deviation across five seeds are also given. The p-value of a one-sample t-test is given to indicate whether the improvement is considered statistically significant and not random ($p < 0.05$). We specifically highlighted stages showing significant positive improvements, where the mean is greater than 0 and the p-value is less than 0.05

The average learning curves across five seeds are shown in Figure 3.2, with a shaded area representing the standard deviation for each method.

Statistical analysis is also conducted on the learning results. We utilise *improvement*, defined as the difference in success rate between HER+GDP and the vanilla HER baseline, to assess how much GDP enhances the learning speed. Each set of 10 epochs (or 5 for FechPush) is considered a learning stage, and we calculate the *stage improvement* by averaging the *improvement* during this stage. The *stage improvements* from the five runs of each experiment, along with their mean and standard deviation, are presented in Table 3.1. To assess whether the improvement is consistent and not occurring by chance, we conduct a one-sample t-test with the collection of *improvements* across five runs for each learning stage. The p-values from the t-test for each learning stage are also provided in Table 3.1. We consider the improvement to be statistically significant and not random if $p < 0.05$.

Analysis of the learning curves and statistical data reveals that, for most Fetch robot tasks (with the exception of the third stage of FetchPush) and the hand egg rotation task, GDP does not yield significant improvement. However, for more challenging hand rotation tasks, such as HandBlockRotation and HandPenRotation, the enhancement is noteworthy. This is particularly evident in the HandManipulatePenRotate task, where improvement is observed consistently throughout the entire learning period. Nonetheless, this pattern does not extend to all difficult tasks. For instance, FetchSlide, despite being considered challenging due to the learning process not converging within the allotted simulation time, experiences a decrease in learning speed during the third and fourth stages upon the introduction of GDP.

Here, we explore several potential reasons why GDP may not lead to improvements in certain tasks. The HandManipulateEggRotate task is relatively simple compared to other hand rotation tasks. After initial learning, all goals seem to contribute evenly to the learning process, making the prioritization of goals less impactful. This is also observed with MEP, which also does not enhance learning speed for HandManipulateEggRotate, as depicted in Figure 3.2. In the FetchPickAndPlace task, a critical bottleneck occurs when the robot successfully reaches and grasps the object before lifting it. Here, prioritisation should ideally target trajectories involving successful grasps. FetchSlide presents a unique challenge, as the robot makes only a single contact with the object. In such cases, learning to control the force exerted by the robot

Window (in epoch)	1	2	3	4	5	10	15	20
seed 1	58.0	46.0	50.0	40.0	36.0	40.0	52.0	50.0
seed 2	50.0	68.0	70.0	64.0	58.0	44.0	42.0	42.0
seed 3	66.0	74.0	72.0	60.0	64.0	34.0	36.0	34.0
seed 4	46.0	56.0	50.0	56.0	52.0	54.0	44.0	54.0
seed 5	72.0	74.0	70.0	70.0	60.0	44.0	50.0	30.0
average	**58.4**	**63.6**	**62.4**	**58.0**	**54.0**	**43.2**	**44.8**	**42.0**
std	9.7	10.9	10.2	10.1	9.8	6.5	5.7	9.1

(a) FetchSlide-v1

Window (in epoch)	1	2	3	4	5	10	15	20
seed 1	42.5	52.5	50.0	47.5	42.5	55.0	40.0	35.0
seed 2	55.0	72.5	75.0	70.0	65.0	65.0	52.5	45.0
seed 3	57.5	60.0	57.5	62.5	65.0	50.0	57.5	50.0
seed 4	70.0	65.0	67.5	57.5	55.0	45.0	50.0	27.5
seed 5	57.5	65.0	37.5	35.0	47.5	50.0	60.0	40.0
average	**56.5**	**63.0**	**57.5**	**54.5**	**55.0**	**53.0**	**52.0**	**39.5**
std	8.8	6.6	13.1	12.2	9.0	6.8	7.0	7.8

(b) HandManipulatePenRotate-v0

Table 3.2: The optimality (in percentage) for PSES. Experiments are done with two environments and repeated with five random seeds. Different time windows are compared. The average and the standard deviation across five seeds are also calculated for each time window.

to achieve closer positions may merit prioritisation in the early stages. This could explain the negative effects observed when prioritising trajectories with novel reaching positions.

In addition to assessing the performance of GDP, we also implemented PSES alongside all three prioritization methods. For this strategy, an optimal time window of 2 epochs was chosen.The mean test success rate is depicted by red lines in Figure 3.2. The results indicate that PSES enhances overall performance by dynamically selecting the most effective method during the learning process across all tasks.

Ideally, we would expect the method to select the one with the best performance at all times. However, in real cases, the algorithm has no knowledge of the current step's performance and selects the method based on the most recent history. To analyse how accurately our strategy selects the best method, we calculate the optimality of each PSES learning process. Optimality is defined as the percentage of time steps (epochs) among the whole learning process that PSES selects the method with the best performance. Different time windows are applied and compared in Table 3.2.

It can be seen that with a relatively small time window of fewer than ten epochs, the performance is robust and better than random selection in general. On the other hand, when the time window is too large (i.e. larger than ten epochs for FetchSlide-v1 task, or 20 epochs for HandManipulatePenRotate-v0 task), it is not always that the performance outperforms the individual components, thus making the strategy less useful in this specific case.

3.3 Policy-level-based Automatic Curriculum Goal Selection (PL-CGS)

The research question we focus on in this section is: *How to efficiently explore the goal space by generating goals in a curriculum form during the learning process.*

We consider the design of a curriculum learning approach that selects goals for the agent, which allows a continuous multi-goal task to be learned more efficiently. This method considerably improves the sample efficiency of the learning process to reach all feasible goals. We were enlightened by the concept of policy-level [6, 26] to label collected experiences. We extend this idea to a hindsight experience replay (HER) setup and adapt it to complex robot manipulation and InDex scenarios. Our proposed method selects new goals for each learning stage based on analysing the goal-pairs of the previously collected experiences that satisfy a certain restriction based on the policy-level. Our method selects goals at an appropriate difficulty level for the current agent. The agent first learns how to reach easier goals, then gradually approaches more complex goals, and finally achieves all target goals.

3.3.1 Methodology

In our work, we focus on solving multi-goal robotic manipulation problems. The ultimate objective for these tasks is to learn to reach all possible goals in a continuous goal space. In each episode E, the object starts its pos with the initial goal g_{init}, and its original final goal is to reach g_o. In the t_{th} step, the object reaches an achieved goal g^t. All episodes have the same time step length T. The reward of these tasks is sparse and goal conditioned: $R_{g_d}(t) = -[d(g^{t+1}) = 0]$, where g_d is the goal that the reward is checked upon, normally is the original final g_o, and can be any future selected goal g'_{future} in HER algorithm. ϵ is a fixed tolerance. The goal distance function $d(g, g_d)$ is task-specific: usually Euclidean distance for position orient tasks, and object orientation difference for object rotation tasks. The overall learning objective is to find a policy $\pi(a_t|s_t, g_o)$ that can produce trajectories with higher episode returns.

Our method can be divided into three parts: First, we label the goal-pairs from the most recent experience by their policy-level to identify whether they are at the appropriate difficulty level for the current agent. The goal-pairs with the policy-level that are neither too easy nor too hard for the current agent are saved in a buffer $\mathbb{R}_{gp}$. Then, we analyse the distribution of these goal-pairs saved in the buffer by either fitting a Gaussian Mixture Model (GMM) or training a discriminator that can determine if a given goal-pair is at an appropriate level of difficulty. Finally, we select new goal-pairs based on the GMM/discriminator and apply these selected goal-pairs to the next training iteration.

Policy-level and goal-pair labelling

The episodes selected by the curriculum need to be at an intermediate level of difficulty for the current agent. We use goal-pair $gp = [g_{init}, g_o]$, which combines the initial goal g_{init} and the original final goal g_o, to represent the target of an episode. We inherited the concept of policy-level from [6] to define the level of difficulty of different goal-pairs for the current agent. The policy-level $C(i)$ describe the level of policy $(\pi(i))$ that produces episode $E(i)$. There are many ways to describe the policy-level in a sparse reward setup:

- $C^i_1 = 1|d(g^i(T-1), g^i_o) < \epsilon$ indicates whether the last element of the episode $g^i(T-1$ reaches the original final goal g^i_o, where $C^i_1 = 1$ if the episode is successful. The function $d(.)$ is the Euclidean distance for position-based tasks and quaternion difference for orientation-based tasks, and ϵ is a small tolerant;

- $C^i_2 = (\sum_j 1|d(g^i_j, g^i_o) < \epsilon)/T$, named the **goal-stay ratio** [26], indicates the ratio of the reached states in the whole episode E^i counting from the end of the episode.

- $C^i_3 = 1|d(g^i(T-1), g^i_{init}) > \epsilon$ indicates the object starts to move away from the initial goal. This value could be especially important for some object non-contact manipulation cases like FetchPush and FetchPickAndPlace;

In our proposed method, all three ways shown above are combined to describe the policy-level:

$$C(i) = \prod_{i=1,2,3} C^i \tag{3.4}$$

We store the most recent experiences' goal-pairs and their corresponding policy-level in a replay buffer $\mathbb{R}_{gp}$ for further analysis. In order to extend these experiences saved in the buffer $\mathbb{R}_{gp}$, which will be used for analysing and selecting the proper goal-pairs for the next iteration, we apply hindsight experience replay (HER) when calculating the policy-level. For each episode $E(i)$, the original final goal g_o can be replaced with the achieved goal g_t at a given time-step t. The goal-pair constructed in this case is called the currently achieved goal-pair $gp_t = g_{init}, g_t$. Here we assume that the achieved goal stays at g_t after time-step t when calculating the policy-level for gp_t. After applying HER, one episode $E(i)$ will be transformed into T episodes, and the number of currently achieved goal-pairs and associated policy-levels that are saved in $\mathbb{R}_{gp}$ is T.

Goal-pairs distribution analysing

We utilise the goal-pairs associated with their policy-level stored in the buffer $\mathbb{R}_{gp}$ to analyse what goal-pair is at an intermediate level of difficulty to be proposed for the next training iteration. Buffer $\mathbb{R}_{gp}$ uses the FIFO method to store data, thus the information saved in $\mathbb{R}_{gp}$ always represents the experiences collected with the current policy or the most recent policies.

To define which goal-pair is appropriate to learn for the current policy, we refer to the idea of Goals of Intermediate Difficulty (GOID) in [26]. First, we aim to train the policy with goal-pairs with which the policy can reach the final goal with some minimum return, thus $C \geqslant C_{min}$. We also would not like to train the policy from the goal-pairs that the policy mastered. In this case, the policy-level is restricted to $C \leqslant C_{max}$ so that the policy focuses on training those goal-pairs that still need improvement. The overall GOID restriction can be represented as:

$$C \in (C_{min}, C_{max}) \tag{3.5}$$

We regard those goal-pairs with policy-levels that satisfy the GOID restriction as **positive goal-pair examples** and treat others as **negative goal-pair examples**. The next step is to analyse the distribution of these goal-pairs to identify if a new goal-pair belongs to the 'positive' or 'negative' category. There are two methods in our experiments that are used to model the goal-pair distribution:

- **Gaussian Mixture Model (GMM)**: A Bayesian Gaussian Mixture Model is used to fit the goal-pair values of the positive examples saved in the buffer $\mathbb{R}_{gp}$. After each model fit finishes, the highest and lowest scores (log-likelihood) of all positive examples under the current model are recorded, and a tolerance score is calculated as

  $$s_{tol} = \gamma(s_{high} - s_{low}) + s_{low} \tag{3.6}$$

  If the score of a given new goal-pair gp is higher than the tolerance score $s_{gp} > s_{tol}$, it will be considered an appropriate goal-pair for the current policy to learn;

- **Neural network Discriminator**: A neural network, multi-layer perceptron (MLP), in our experiments, is trained as a discriminator to distinguish the positive and negative examples. After each training batch finishes, the positive examples will have a high predicted score approaching one and the negative examples have a low predicted score approaching zero. If the predicted score D_{gp} of a random new goal-pair gp is higher than a certain tolerance D_{tol} (for example, 0.6 is used in our experiments), it will be considered a positive example and used for learning.

Curriculum goal selection

During the data-collecting stage, random initial and final goals are generated after resetting the environment for a new episode. The goal-pair of this episode is extracted and forwarded to the GMM/Discriminator to check whether this goal-pair is appropriate for the current agent to learn. If the goal-pair is selected, the agent will continue this episode until finished to collect new data. If the goal-pair is not selected, this episode will be discarded, and the environment will be restarted again with a new goal-pair generated.

Algorithm 3 HER with Policy-Level-based Curriculum Goal Selection (PL-CGS), with GMM

```
Require:
  - An off-policy algorithm $\mathbb{A}$                                  ▷ e.g. DDPG, DQN
  - A goal-based reward function $\mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$    ▷ e.g. $r(s, a, g_d) = -1$ if fails, 0 if success
  Initialise actor and critic networks for $\mathbb{A}$
  Initialise replay buffer $\mathbb{R}$ and goal-pair replay buffer $\mathbb{R}_{gp}$
  for $epoch = (1, L)$ do
    for $episode = (1, M)$ do                    ▷ Step 1: data collection and storage
      Sample original desired goal $g_o$ and initial goal $g_{init}$
      Construct goal-pair for this episode $gp = [g_{init}, g_o]$
      Calculate the GMM score $s_{gp}$ of $gp$
      if $\mathbb{R}_{gp}$ is not half filled or $s_{gp} > s_{tol}$ then
        for $t = (0, T-1)$ do
          Sample action $a_t$ using the behaviour policy $\pi$ from $\mathbb{A}$
          Execute the action and observe reward $r_t$ and new state $s_{t+1}$
          Construct the currently achieved goal-pair $gp_t = [g_{init}, g_t]$
          Calculate policy-level $C_t$ for $gp_t$ using Equation (3.4)
          Save $gp_t$ and $C_t$ in $\mathbb{R}_{gp}$ if $C_t$ satisfies Equation (3.5)    ▷ Goal-pair storage
        end for
        Store the trajectory $\tau = (s_t, g_d, g'_t, a_t, r_t)_{t=0}^{T}$ in $\mathbb{R}$
      end if
    end for
    Fit GMM using goal-pairs in $\mathbb{R}_{gp}$          ▷ Step 2: goal-pair distribution analysis
    Update tolerance score $s_{tol}$ using Equation (3.6)
    for $batch = (1, N)$ do                      ▷ Step 3: policy update with HER
      Sample transitions $(s_t, g_o, a_t, g_t, s_{t+1}, g_{t+1})$ in $\mathbb{R}$, store them in $\mathbb{B}$
      Apply HER on transitions in $\mathbb{B}$, recalculate the reward
      Perform one step optimisation for $\mathbb{A}$ using $\mathbb{B}$
    end for
  end for
```

Algorithm 4 HER with Policy-Level-based Curriculum Goal Selection (PL-CGS), with Discriminator

```
Require:
  - An off-policy algorithm $\mathbb{A}$                                ▷ e.g. DDPG, DQN
  - An discriminator $\mathbb{D}$                                       ▷ usually MLP
  - A goal-based reward function $\mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$      ▷ e.g. $r(s, a, g_d) = -1$ if fails, 0 if success
  Initialise actor and critic networks for $\mathbb{A}$ and discriminator network for $\mathbb{D}$
  Initialise replay buffer $\mathbb{R}$
  Initialise positive goal-pair buffer $\mathbb{R}^p_{gp}$ and negative goal-pair buffer $\mathbb{R}^n_{gp}$
  for $epoch = (1, L)$ do
      for $episode = (1, M)$ do                  ▷ Step 1: data collection and storage
          Sample original desired goal $g_o$ and initial goal $g_{init}$
          Construct goal-pair for this episode $gp = [g_{init}, g_o]$
          Calculate the discriminator predicted score $D_{gp}$ of $gp$ using $\mathbb{D}$
          if $\mathbb{R}_{gp}$ is not half filled or $D_{gp} > D_{tol}$ then
              for $t = (0, T-1)$ do
                  Sample action $a_t$ using the behaviour policy $\pi$ from $\mathbb{A}$
                  Execute the action and observe reward $r_t$ and new state $s_{t+1}$
                  Construct the currently achieved goal-pair $gp_t = [g_{init}, g_t]$
                  Calculate policy-level $C_t$ for $gp_t$ using Equation (3.4)
                  Save $gp_t$ and $C_t$ in $\mathbb{R}^p_{gp}$ if $C_t$ satisfies Equation (3.5), otherwise save them in $\mathbb{R}^n_{gp}$.     ▷ Goal-pair storage
              end for
              Store the trajectory $\tau = (s_t, g_d, g'_t, a_t, r_t)^T_{t=0}$ in $\mathbb{R}$
          end if
      end for
      for $batch = (1, N_d)$ do                    ▷ Step 2: Train discriminator
          Sample positive transitions in $\mathbb{R}^p_{gp}$, sample negative transitions in $\mathbb{R}^n_{gp}$
          Train $\mathbb{D}$
      end for
      for $batch = (1, N)$ do                      ▷ Step 3: policy update with HER
          Sample transitions $(s_t, g_o, a_t, g_t, s_{t+1}, g_{t+1})$ in $\mathbb{R}$, store them in $\mathbb{B}$
          Apply HER on transitions in $\mathbb{B}$, recalculate the reward
          Perform one step optimisation for $\mathbb{A}$ using $\mathbb{B}$
      end for
  end for
```

The Algorithm

The complete training method for Policy-Level-based Curriculum Goal Selection (PL-CGS) is provided in Algorithm 3 (using a GMM) and Algorithm 4 (using a Discriminator). The algorithm first learns freely with all episodes generated by the environment (no goal selection) to collect enough initial data for goal-pair analysis (Step 1). At each time step of an episode, HER is applied to calculate the policy-level of the currently achieved goal-pair gp_t. The goal-pair with the policy-level that satisfies the GOID restriction is saved in the goal-pair buffer with its associate policy-level. In the discriminator version, they are saved in a 'positive' goal-pair buffer, and other goal-pairs that do not satisfy the restriction are saved in a 'negative' goal-pair buffer. The goal-pair analysis process is not started until the goal-pair buffers are filled enough (usually half of the full goal-pair buffer size).

In the goal-pair analysis stage (Step 2), the distribution of the goal-pairs saved in the goal-pair buffer is analysed. In the GMM version, a GMM is fit for all goal-pairs saved in the goal-pair buffer. In the discriminator version, a neural network discriminator is trained to distinguish the goal-pairs saved in the 'positive' and 'negative' buffer.

After the goal-pair buffer is initialised and the first batch of analysis is finished, the data collection stage (Step 1) uses the GMM/Discriminator to select goal-pairs generated by the environment. Only the selected goal-pairs are used for further data collection. Others are discarded, and the environment is restarted to generate new goal-pairs.

The difference between the GMM and the discriminator versions is in the goal-pair storage part in Steps 1 and 2. The off-policy learning strategy (Step 3) is not altered from the original algorithm (DDPG+HER in our experiments).

3.3.2 Experimental Results

Environments

The proposed PL-CGS method is tested on various OpenAI Gym simulated multi-goal robotic manipulation tasks, see Figure 3.1. Here are the environments we have used in our experiments (a detailed introduction of these environments is provided in Section 3.2.2):

- *FetchPush-v1*: Push a block on a table to reach a target position with the fetch robot;

- *FetchSlide-v1*: Hit a block on a slippery table to slide to a target position with the fetch robot;
- *FetchPickAndPlace-v1*: Pick a block on the table with the fetch robot and reach a target position in the air;
- *HandManipulateEggRotate-v0*: Rotate an egg-shaped object within hand to reach a target orientation using the Shadow hand robot;
- *HandManipulateBlockRotateXYZ-v0*: Rotate a block-shaped object within hand to reach a target orientation using the Shadow hand robot;
- *HandManipulatePenRotate-v0*: Rotate a pen-shaped object within hand to reach a target orientation using the Shadow hand robot;

Furthermore, we have customised some of the environments to use a fixed initial goal so that we can analyse how the selected target goals progress through different learning stages:

- *FetchPushFixInitPos-v1*: The initial position of the block is always in the centre of the table;
- *HandEggRotateOnlyFixInitPos-v0*: The initial position and orientation of the egg-shaped object in hand are always the same;

Implementation and Experiment setup

We implemented our method using both GMM and discriminator versions and compared it to the baseline vanilla HER method. In our experiments, each epoch consists of 50 training cycles, where each cycle first collects 40 episodes of data and then trains the policy networks using 40 batches with a mini-batch size of 5120. After each cycle, we update either the GMM model or the discriminator. The discriminator network is trained with the same batches and batch size as the policy networks. Following the completion of each epoch, we conduct an evaluation to assess performance. Both the policy and discriminator networks utilize MLP as their neural network structure, featuring three fully connected hidden layers with 256 nodes each.

The goal-pair data collected in the goal-pair buffer includes the two most recent cycles for Fetch environments and the five most recent cycles for Hand environments. The GOID restriction employs $C_{min} = 0.1$ and $C_{min} = 0.9$ for all tasks. To prevent the method from falling into a local minimum, we collect only 75% of the data via PL-CGS, with the remainder of the data being selected goals randomly.

Performance Comparison

The experiments were conducted across two environments with fixed initial goals and six environments with completely random initial and target goals. For each environment, experiments were run repeatedly using five different random seeds. We compared the HER+PL-CGS method, employing either the GMM or the MLP discriminator, against the baseline vanilla HER method.

Figure 3.3 presents the learning curves averaged across five seeds for each environment, with the shaded areas indicating the standard deviation. From the learning curves, we observe that for PL-CGS with GMM, the learning rate is faster than that of vanilla HER in certain environments (FetchPush, FetchSlide, and HandPenRotate). It is slightly faster in FetchPushFixInitPos, and similar to HER in other environments (FetchPickAndPlace, HandEggRotate). In the HandBlockRotate task, the learning rate starts slightly faster but becomes slower than HER later, continuing this trend until convergence. For PL-GCS with the discriminator, the learning speed is not significantly faster than the HER baseline in all environments.

Since the improvement is not necessarily significant across all stages for PL-CGS with GMM, we aim to finely evaluate how much our method outperforms the baseline throughout different stages of training. We define *improvement* as the difference in success rate between our method and the baseline. We consider every 10 epochs (5 for FechPush and FetchPushFixInitPos) as a learning stage and calculate the average *improvement* during this period as *stage improvement*, denoted as δ. The *stage improvements* from the five runs of each experiment, along with their mean and standard deviation, are presented in Table 3.3. To assess whether the improvement is consistent and not occurring by chance, we conduct a one-sample t-test with the collection of improvements across five runs for each learning stage. The p-values from the t-test for each learning stage are also provided in Table 3.3. We consider the improvement to be statistically significant and not random if $p < 0.05$. From the statistical analysis results, we observe that in most environments, PL-CGS with GMM demonstrates a stable, albeit minor, improvement during

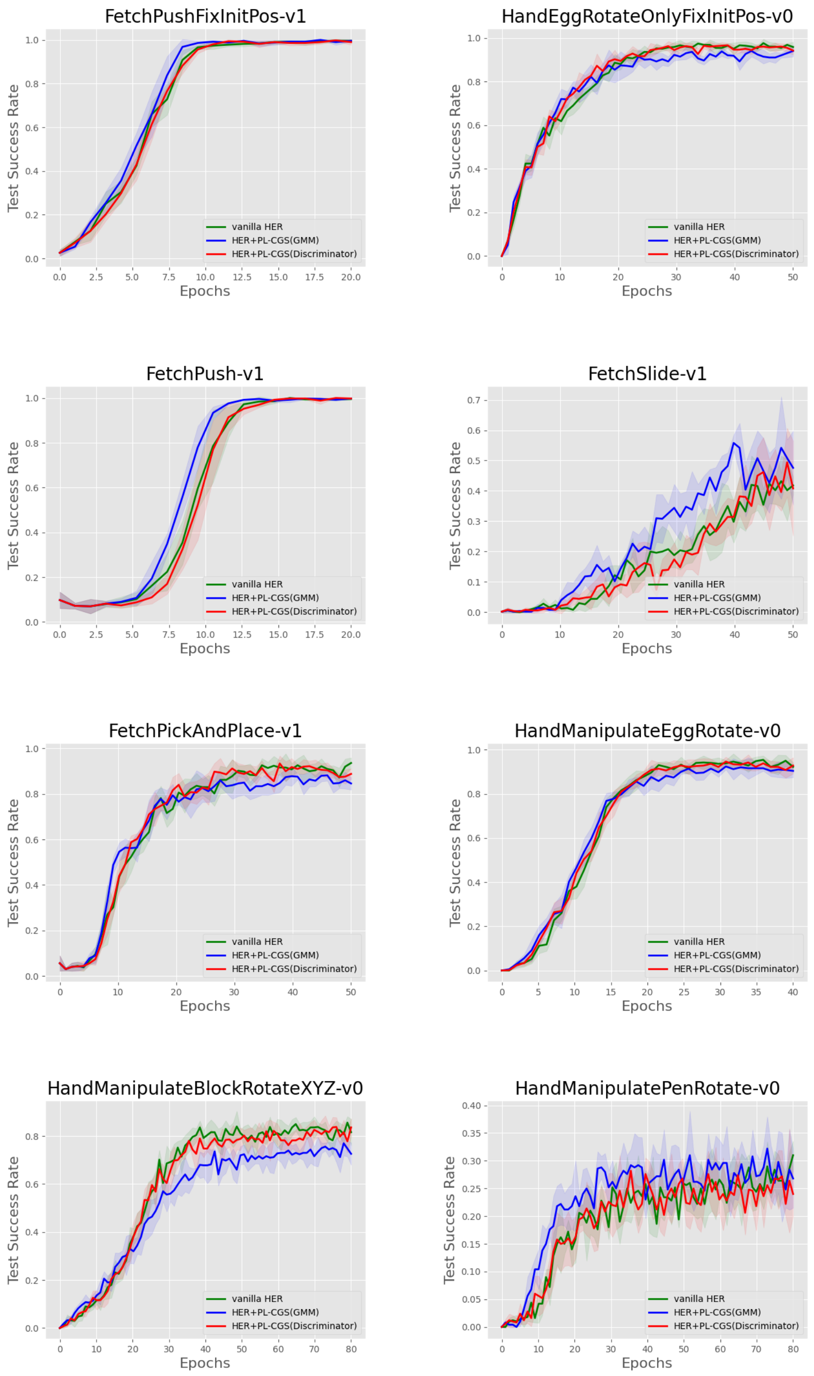


Figure 3.3: The learning curve for different methods in all eight multi-goal robotics environments. The green, blue, and red lines represent the learning curve for vanilla HER, HER+PL-CGS (with GMM), and HER+PL-CGS (with discriminator), respectively.

FetchPushFixInitPos					**HandEggRotateOnlyFixInitPos**				
stage	1	2	3	4	stage	1	2	3	4
seed1	0.05	0.1	0	0	seed1	0.02	0.01	-0.06	-0.06
seed2	0.01	0.07	0.01	0	seed2	-0.04	0.04	-0.03	-0.03
seed3	-0.01	0.02	0.02	0	seed3	0.02	0	-0.04	-0.02
seed4	0.04	0.11	0.01	0	seed4	0.03	0.04	-0.05	-0.05
seed5	-0.01	-0.02	0	0	seed5	0.02	0.03	-0.06	-0.06
mean	0.01	**0.06**	**0.01**	0	mean	0.01	**0.03**	-0.05	-0.04
std	0.03	**0.05**	**0.01**	0	std	0.02	**0.02**	0.01	0.01
p-value	0.257	**0.006**	**0.032**	0.753	p-value	0.329	**0.011**	0	0
FetchPush					**FetchSlide**				
stage	1	2	3	4	stage	1	2	3	4
seed1	0	0.15	0.02	0.01	seed1	-0.01	0.11	0.09	0.18
seed2	0	0.11	0.09	-0.01	seed2	-0.01	0.05	0.08	0.22
seed3	0	0.06	0.02	0	seed3	0	0.09	0.06	0.08
seed4	0.01	0.12	0.01	0	seed4	0	0.05	0.05	0.10
seed5	0	0.11	0.13	0	seed5	0	0	0.10	0.17
mean	0	**0.11**	0.05	0	mean	0	**0.06**	**0.08**	**0.15**
std	0	**0.03**	0.05	0	std	0	**0.04**	**0.02**	**0.05**
p-value	0.679	**0**	0.080	0.832	p-value	0.014	**0**	**0**	**0**
FetchPickAndPlace					**HandEggRotate**				
stage	1	2	3	4	stage	1	2	3	4
seed1	0.02	0.08	-0.05	-0.09	seed1	0.02	0.01	-0.02	-0.04
seed2	0.06	0.07	-0.03	-0.09	seed2	0.01	0.01	-0.04	-0.02
seed3	0.03	0.05	0.08	-0.01	seed3	0.02	0.04	-0.06	-0.04
seed4	0.01	-0.02	-0.05	-0.06	seed4	0.04	0.07	-0.04	-0.02
seed5	0.01	0.02	-0.04	-0.06	seed5	0.05	-0.01	-0.02	-0.01
mean	**0.03**	**0.04**	-0.02	-0.06	mean	**0.03**	**0.02**	-0.04	-0.02
std	**0.02**	**0.03**	0.05	0.03	std	**0.02**	**0.03**	0.01	0.01
p-value	**0.010**	**0.002**	0.149	0	p-value	**0**	**0.024**	0	0.001
HandBlockRotate					**HandPenRotate**				
stage	1	2	3	4	stage	1	2	3	4
seed1	0.02	0.07	-0.01	-0.12	seed1	0.02	0.06	0.08	0.05
seed2	0.02	-0.01	-0.10	-0.09	seed2	0	0.05	0.04	0.02
seed3	0.01	0.02	-0.11	-0.2	seed3	0.03	0.09	0.05	0.02
seed4	0.03	0.03	-0.11	-0.17	seed4	0.02	0.08	0.04	0.04
seed5	0.01	0.03	-0.1	-0.1	seed5	0.01	0.06	0.06	0.06
mean	**0.02**	**0.03**	-0.09	-0.14	mean	**0.01**	**0.07**	**0.05**	**0.04**
std	**0.01**	**0.03**	0.04	0.04	std	**0.01**	**0.02**	**0.01**	**0.02**
p-value	**0**	**0.016**	0	0	p-value	**0.007**	**0**	**0**	**0**

Table 3.3: The Statistical analysis for stage improvements for PL-CGS with GMM: The stage improvement during each learning stage is calculated for each run with a different seed. The mean and standard deviation across five seeds are also given. The p-value of a one-sample t-test is given to indicate whether the improvement is considered statistically significant and not random ($p < 0.05$). We specifically highlighted stages showing significant positive improvements, where the mean is greater than 0 and the p-value is less than 0.05

the early and mid-stages of the learning process.

However, negative effects are observed in many environments (FetchPickAndPlace, two HandEggRotate tasks, and HandBlockRotate) during the later stages when learning has converged, leading to a lower success rate compared to the HER baseline. This could be because, in the final stages of learning, as it approaches convergence, the goals used for learning should increasingly represent scenarios that have never been reached. However, due to the nature of the GOID restriction, PL-CGS tends to select goals that have already been achieved. To address this, we have introduced a portion of goals selected randomly during learning, though this percentage is fixed at 25%. Introducing a mechanism to gradually anneal PL-CGS and shift it towards standard HER practices as the learning approaches converge could be beneficial. In other environments (FetchSlide and HandPenRotate) where improvement is consistent across all stages, these tasks are considered the most challenging on Fetch and Hand robots. The baseline learning strategy struggles to learn all goals across the entire goal space within the applied simulation window in our experiments. PL-CGS provides a good curriculum mechanism to select proper goals.

In the FetchPushFixInitPos environments, where the initial goal is always fixed, we can examine and plot the final goals stored in the 'good' goal-pair buffer, as well as the GMM-selected final goals for different learning stages, as depicted in Figure 3.4. The figures illustrate that, during the earlier stages of learning (epochs 1 and 2), the goals in the 'good' goal-pair buffer and those selected by the GMM are predominantly clustered around the initial goal point. Starting from epoch 3, the distribution of goals begins to spread more evenly across the goal space.

For PL-CGS with the discriminator version, the learning rate is similar to HER baseline across most environments. We have noticed that, in many cases, the discriminator does not converge sufficiently, meaning that the goals selected by the discriminator do not accurately represent the distribution of 'good' goal-pairs. As illustrated in Figure 3.5, we have analysed the goals stored in the good and bad goal-pair buffers throughout the learning process. We found that the distributions of good and bad goals overlap significantly, indicating that the discriminator struggles to distinguish between these two categories. Moreover, the bad goal-pair buffer only includes goals from achieved trajectories, omitting unachieved goals that could also be considered 'bad'. As a result, the distribution of bad goals in this buffer fails to accurately reflect

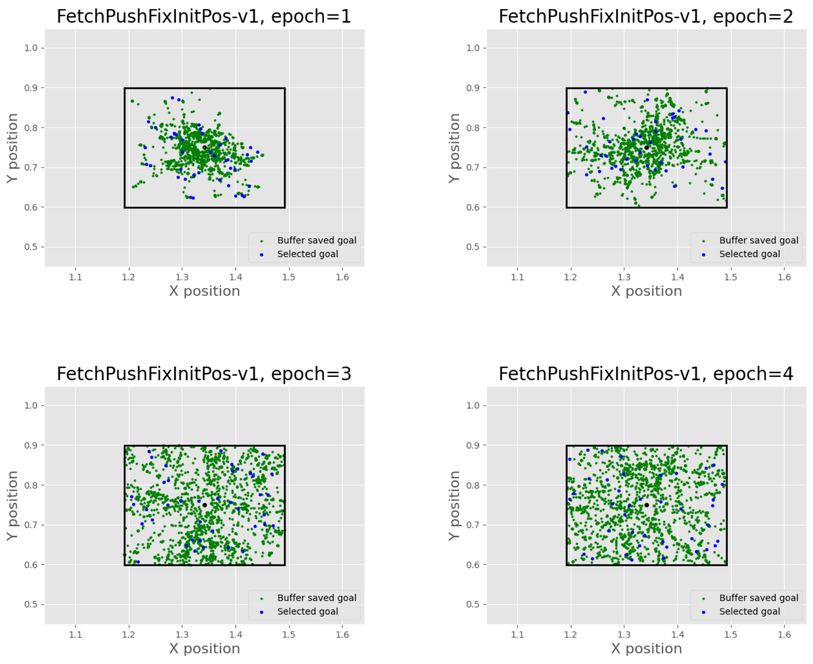


Figure 3.4: The goal distribution of different learning stages (cycle 10 for epochs 1 to 4) for FetchPushFixInitPos task for PL-CGS (GMM version). The green dots represent the goals saved in the good goal buffer. The GMM will use these goals to analyse the distribution. The blue dots represent the goals selected for learning for the next learning cycle. The black square represents the 2D target goal space.

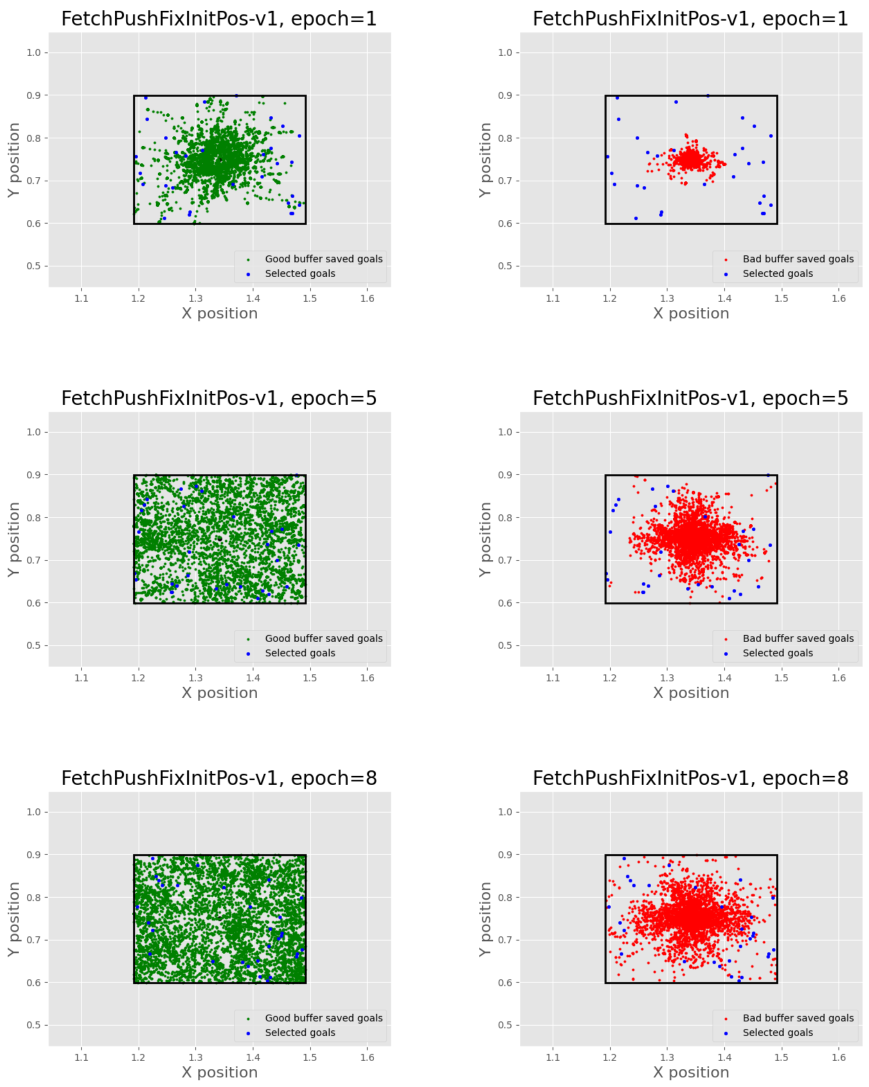


Figure 3.5: The good and bad goal distribution of different learning stages (cycle 10 for epochs 1 and 5) for FetchPushFixInitPos task for PL-CGS (discriminator version). The green and red dots represent the goals saved in the good and bad goal-pair buffer. The blue dots represent the goals selected for learning for the next learning cycle. The black square represents the 2D target goal space.

the true distribution of undesirable goals.

3.4 Summary

In this chapter, we discussed the multi-goal RL problem and the related methods that can be applied to improve learning efficiency. The OpenAI robotic manipulation environments are used in the experiments for evaluation.

Firstly, we introduce the **Goal density-based hindsight experience prioritisation**, which prioritises novel achieved experiences based on the density of the initial and final goal pairs. We implemented the GDP algorithm and evaluated its performance against vanilla HER and other density-based prioritisation methods, such as Maximum Entropy-based Prioritisation (MEP). Furthermore, we propose the Prioritisation Switching with Ensemble Strategy (PSES), a novel approach that amalgamates various experience prioritisation methods. This strategy optimises overall performance by dynamically selecting the best-performing method at each stage of the learning process. Experimental results indicate that while GDP enhances learning efficiency in certain tasks, it does not significantly impact or may even worsen outcomes in others. However, the application of PSES, which integrates GDP with other prioritisation methods, consistently outperforms the vanilla HER method across all evaluated scenarios.

The **contribution of GDP** lies in offering a new perspective on analysing experience density, where leveraging goal-pair density achieves a reduction in computational effort compared to prior density-based methods like MEP. The **limitation of GDP**, however, is its lack of generalisability across diverse manipulation tasks, where only prioritising novel seen goals is not enough.

Next, we introduced the **Policy-level-based curriculum goal selection** (PL-CGS). This approach analyzes the most recently collected experiences to determine the distribution of goals that possess the optimal difficulty level for the current agent's learning phase, thereby using this distribution to select new goals for subsequent learning iterations. We implemented PL-CGS with two different distribution analysis strategies (GMM and MLP discriminator training) and benchmarked it against the vanilla HER method. Our findings reveal that the PL-CGS (GMM version) accelerates the learning rate for several challenging robot manipulation tasks. Nonetheless, it fails to significantly enhance the learning speed in tasks where all goals equally contribute to the learning process, such as HandEggRotation. Meanwhile, the PL-CGS (discriminator version) struggles due to difficulties in training an effective discriminator to identify suitable goals, offering limited curriculum guidance for most tasks.

The **contribution of PL-CGS** is its innovative approach to analysing Goals of Intermediate Difficulty (GOID) for goal selection/generation within the context of multi-goal learning. The **limitation of PL-CGS** becomes evident as we observe a decrease in the success rate when learning nears convergence across many tasks. The reliance on GOID leads to a preference for achievable goals, overlooking the potential learning benefits of unexplored goals in the later stages. Addressing this issue, through a methodological transition, presents an avenue for future exploration.

Chapter 4

Imitation learning for In-Hand Dexterous Manipulation

In this chapter, we delve into the realm of imitation learning for model-free RL problems,with a particular emphasis on its application to manipulation tasks. We specifically explore Learning from Demonstration (LfD) methods tailored for multi-goal learning challenges. The LfD is based on demonstrations of manipulation tasks performed with a Fetch robot arm (in simulation) and a Shadow robot hand (in both simulation and real).

Our discussion begins with a discussion of the foundational works that inform our research, including Generative Adversarial Imitation Learning [39] (GAIL), Self-adaptive GAIL [129] and Goal-conditioned GAIL [21].

Following this, we introduce our innovative approach, Goal-based Self-Adaptive Generative Adversarial Imitation Learning (Goal-SGAIL), which is expressly designed for tackling multi-goal robot manipulation learning problems. We assess the efficacy of our methods on robot manipulation learning tasks within the MuJoCo simulator, setting them in contrast with conventional LfD techniques and goal-GAIL.

Additionally, this chapter covers how various LfD methods, including Goal-SGAIL, can be adapted to leverage demonstrations obtained from human teleoperation. This adaptation is exemplified through a block rotation task executed by the Shadow robot hand on the PyBullet platform. We detail the teleoperation technique employed to gather demonstration data and subsequently analyse the learning outcomes.

To conclude, we summarize and reflect on the contributions and implications of the Goal-SGAIL method.

4.1 Related work

4.1.1 Traditional learning from demonstration (LfD) methods

Unlike some LfD scenarios where an expert agent provides examples online, our research utilises pre-collected expert trajectories. During training, the trainer has access only to these demonstration data as examples.

There are two primary methods for leveraging these demonstrations: Behavior Cloning (BC) and Inverse Reinforcement Learning (IRL). BC involves supervised learning using state-action pairs from the expert trajectories to directly maximise the likelihood of replicating the expert actions. In contrast, IRL seeks to infer the underlying reward function that guided the expert's behavior based on the demonstrations, then performs forward reinforcement learning to maximize this inferred reward. We have previously discussed the limitations of both approaches in Chapter 2.

4.1.2 Generative Adversarial Imitation Learning (GAIL)

IRL involves a computationally intensive cycle of deriving the reward function from demonstrations and then executing forward RL to determine the optimal policy under the current reward function. This process, aimed at enabling the learner to replicate the actions of the expert, can be resource-intensive. Drawing inspiration from IRL and the breakthroughs in Generative Adversarial Networks (GAN), a more streamlined method known as Generative Adversarial Imitation Learning (GAIL) [39] has been introduced. GAIL adopts a more direct strategy compared to IRL by implementing a specific RL procedure with a cost function designed to steer the policy towards expert behaviour, thus circumventing the intermediate steps required by IRL. In essence, GAIL seeks a cost function that assigns low cost to the expert policy while high cost on other divergent policies.

In GAIL, A discriminator $D(s, a)$ is trained to fit the expert state-action distribution and distinguish the expert transitions $(s, a) \sim \tau_{expert}$, from the agent transitions$(s, a) \sim \tau_E$. D is trained to minimise:

$$L_{D(s,a)} = E(s, a) \sim \tau[logD(s, a)] + E(s, a) \sim \tau_{expert}[log(1 - D(s, a))] \quad (4.1)$$

The agent τ is treated as a generative model G to be trained to confuse the discriminator so that, eventually, the agent is good enough that the discriminator cannot differentiate it from the expert. The link between the discriminator model D and the generative model$G$ (the agent τ) is to use the output of $logD(s, a)$ as the reward to train the agent policy to maximise:

$$E(s, a) \sim \tau[logD(s, a)] \quad (4.2)$$

GAIL can be integrated with both on-policy and off-policy RL methods. In the foundational GAIL study [39], it is combined with the Trust Region Policy Optimisation (TRPO) method and applied across nine physics-based robotic control tasks. These experiments demonstrate that GAIL's learning outcomes surpass those of the basic BC method in all tasks, leveraging expert demonstrations acquired through RL-trained expert agents.

4.1.3 Self-adaptive Generative Adversarial Imitation Learning

Most LfD algorithms significantly depend on both the quality and quantity of the demonstrations. However, in practical scenarios, collecting a substantial number of expert demonstrations can be challenging and costly. Moreover, there are instances where the teacher may be sub-optimal, leading to demonstrations of limited quality. These challenges can cap the learned performance to that of the provided demonstrations, preventing the agent from reaching its optimal performance.

To address the limitation posed by sub-optimal demonstrations, Self-adaptive Generative Adversarial Imitation Learning (SAIL) [129] has been proposed. Unlike traditional approaches that rely solely on expert-generated trajectories for demonstrations, SAIL also considers high-quality trajectories produced by the actor during training as viable demonstrations. A trajectory is deemed high-quality if its episode cumulative return surpasses a specific threshold $r_e(\tau) > C_{dT}$. This threshold C_{dT}, is dynamically updated with $C_{dT} = min[r_e(\tau_i)|r_e(\tau_i) \in W_k]$, where W_k is a window to track the top K trajectory rewards of all trajectories in R_T.

SAIL represents an exploration-driven LfD method that maintains an optimal balance between exploration and exploitation. More crucially, SAIL enables the agent to surpass the experts performance level when the agent becomes more proficient than the demonstrations provided. In their research, SAIL was applied to the same robotic control tasks as GAIL, demonstrating superior final learning outcomes compared to both GAIL and the expert demonstrations.

4.1.4 Goal-conditioned Generative Adversarial Imitation Learning (Goal-GAIL) for multi-goal oriented learning

GAIL has demonstrated a robust capability for learning from demonstrations across various decision-making problems and straightforward robotic control tasks. However, its application to multi-goal conditioned tasks, which are inherently more complex, is not straightforward. Moreover, the challenge of obtaining a sufficient number of high-quality demonstrations to comprehensively cover the extensive range of goals in continuous spaces further complicates matters. The expert policies represented by these samples often fall short in adequately covering the goal space.

Goal-conditioned Generative Adversarial Imitation Learning (Goal-GAIL) [21] emerges as the pioneering method to adapt GAIL to multi-goal robotic RL problems, specifically those with binary rewards. Goal-GAIL integrates the advantages of GAIL with Hindsight Experience Replay (HER), leveraging demonstrations to enhance learning efficiency while employing HER's experience relabelling to augment goal space coverage within the demonstrations.

Within Goal-GAIL, GAIL is synergies with the off-policy method DDPG and HER. The amalgamation of DDPG and HER has already shown promising learning capabilities in a variety of multi-goal robotic tasks. HER is implemented during the experience replay phase to ensure a robust positive reinforcement signal. Furthermore, Goal-GAIL introduces the novel concept of applying HER experience relabelling to demonstration sampling, effectively extending the expert demonstration dataset. This can be considered a type of data augmentation and proves especially beneficial in scenarios with limited available demonstrations.

On the GAIL side, the loss function to train the discriminator D has the same structure as GAIL. However, it is also goal-conditioned. The discrimi-

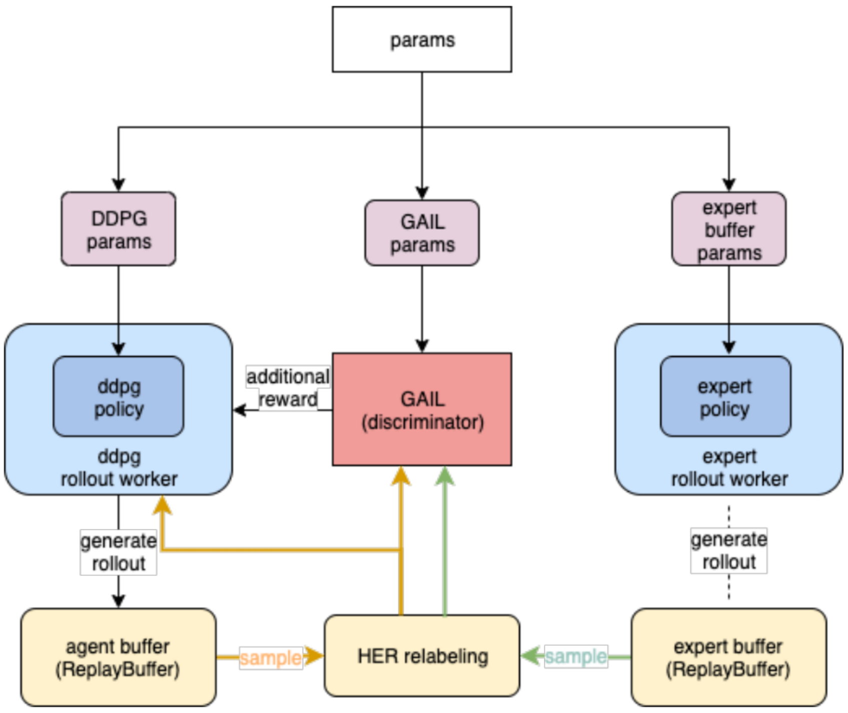


Figure 4.1: The implementation structure of Goal-GAIL [11], copied from the original paper: `https://arxiv.org/pdf/1906.05838.pdf`

nator $D(s, a)$ is trained to minimise:

$$\begin{aligned} L_{D(s,g,a)} &= E(s, g, a) \sim \tau[logD(s, g, a)] \\ &+ E(s, g, a) \sim \tau_{expert}[log(1 - D(s, g, a))] \end{aligned} \tag{4.3}$$

Rather than using the discriminator's output D directly as the reward for RL, Goal-GAIL integrates it with the normal binary reward received from the agent to balance self-learning by the agent and learning from demonstrations. The implementation structure of Goal-GAIL is depicted in Figure 4.1.

4.1.5 Demonstration data collection via teleoperation for InDex tasks

Collecting demonstration data is an integral part of LfD. For robot manipulation tasks, it is essential to gather this data directly on the robot (either in simulation or on an actual robot platform) to maintain kinematic consistency. The robot is typically operated by an expert agent or a human instructor [104]. For complex, multi-fingered InDex tasks, teleoperation techniques, which allow humans to remotely control a robot by hand, are widely used.

Traditional methods for dexterous robot hand teleoperation often employ CyberGloves, available in both custom-made [117] and commercial versions [90]. These gloves can accurately capture the joint movements of the human hand, which are then translated to the robot hand using various kinematic transformation methods [62]. Despite their precision, CyberGloves necessitate user-specific calibration, making them somewhat inconvenient for widespread use.

An alternative teleoperation method involves vision-based hand tracking systems [63]. rimary vision sensors like RGB cameras [37, 123] and the Leap Motion Controller (LMC) [15] are utilized in these systems. For simulated robot hands, integrating a virtual reality system [57] is typically necessary, providing feedback to the operator for more accurate and fluid control.

4.2 Goal-based Self-adaptive Generative Adversarial Imitation Learning (Goal-SGAIL)

4.2.1 Motivation

Although Goal-GAIL extends the GAIL framework to multi-goal task learning, showing enhanced learning efficiency over GAIL, there remains room for improvement. Often, the available demonstrations are too limited to adequately cover the state and action space of multi-goal problems, or the quality of the demonstrations may be sub-optimal, making learning from such demonstrations a challenge.

SAIL offers a promising approach by incorporating high-quality, self-generated trajectories into the expert demonstration buffer, treating these experiences on par with expert demonstrations. However, the original SAIL formulation is not directly applicable to multi-goal learning problems due to the varying difficulty levels of different goals. The episode rewards from goals of differing difficulties cannot be straightforwardly compared, as such comparison must be goal-conditioned. Thus, the straightforward strategy of selecting high-quality trajectories falls short in multi-goal scenarios.

4.2.2 Methodology

Our proposed approach, named *Goal-based Self-adaptive Generative Adversarial Imitation Learning* (Goal-SGAIL), draws inspiration from SAIL and seeks to extend self-adaptive learning principles to Goal-conditioned GAIL. The principal contribution of our method lies in introducing a goal-conditioned strategy for incorporating high-quality, self-generated trajectories into the expert buffer.

Goal-conditioned high-quality self-generated trajectory selection

In Goal-SGAIL, we exclusively incorporate new expert demonstrations from successful self-generated trajectories. Upon collecting a new successful trajectory, our objective is to identify a trajectory within the expert buffer that matches its difficulty level for a comparative analysis of their episode rewards. Drawing inspiration from our previous RL work, PL-GCS, detailed in Chapter 3, we employ the initial and final goal pairs, denoted as $gp = [g_{init}, g_o]$, to succinctly indicate a trajectorys difficulty level. To assess the similarity in difficulty levels between two successful trajectories, we compare their goal pairs.

For each newly collected success trajectory τ^i, its goal pair is represented as $gp^i = [g^i_{init}, g^i_o]$. The combined goal pair distance between τ^i and an expert trajectory τ^e with goal pair $gp^e = [g^e_{init}, g^e_o]$ is calculated as:

$$d_{comb}(\tau_i, \tau_e) = d(g^i_{init}, g^e_{init}) + d(g^i_o, g^e_o) \tag{4.4}$$

For each successful self-generated trajectory, we calculate the combined goal pair distance to every expert trajectory in the expert buffer. The expert trajectory that has the minimum combined goal pair distance, denoted as $\tau_{e(min)}$, is deemed to have the most similar difficulty level to the successful self-generated trajectory τ_i:

$$\tau_{e(min)} = \tau_e \sim min[d_{comb}(\tau_i, \tau_e) | \tau_e \in R_E] \tag{4.5}$$

The episode cumulative returns of τ_i and $\tau_{e(min)}$ are then compared. If the episode cumulative return of τ_i's episode cumulative return exceeds that of $\tau_{e(min)}$, it is deemed a higher-quality trajectory for the current policy and subsequently stored in the expert buffer R_E.

Sometimes the goal pair distribution within the expert demonstrations may not be uniformly distributed, often resulting in most demonstrations being of a similar, relatively easy level of difficulty. This scenario is particularly common with sub-optimal experts who can only offer success trajectories of lower difficulty levels. Consequently, the expert trajectory $\tau_{e(min)}$ identified as most similar to τ^i, might still have a high combined goal pair distance, rendering it substantially different from τ^i. Under these circumstances, comparing the episode returns of these two trajectories becomes less meaningful. To address this issue, we introduce a threshold, C_{comb}during the search for the minimum combined goal pair distance. If the smallest combined goal pair distance found in the expert buffer exceeds this threshold, $d_{comb}(\tau_i, \tau_{e(min)}) > C_{comb}$, it indicates that the trajectory τ^i significantly diverges from any existing demonstrations and can therefore be directly added to the expert buffer.

Algorithm

Goal-SGAIL adopts the Goal-GAIL learning framework, utilising the off-policy RL method DDPG and Hindsight Experience Replay (HER) for the RL learning component. To manage trajectories, we maintain two replay buffers, R_E for expert demonstrations and R_B for self-generated trajectories. We have expanded the expert buffer R_E to include high-quality self-generated trajectories, enhancing its compatibility with self-adaptive learning. To prevent the expert buffer R_E from becoming overwhelmed with outdated samples, we cap its maximum size and employ a first-in-first-out (FIFO) strategy for incorporating new trajectories. This approach not only keeps the buffer current but also allows for the removal of older demonstrations that may no longer be optimal for the agent's current learning phase.

During the experience replay stage, samples are drawn from both the expert buffer R_E and the self-generated buffer R_B. with HER applied to experiences from both sources. We use a mixed distribution for sampling experiences, balancing between expert (d_E) and self-generated (d_B) trajectories as follows:

$$d_{mix} = \alpha d_E + (1 - \alpha) d_B \tag{4.6}$$

where α is the expert demonstration's sample rate.

The discriminator training in Goal-SGAIL follows the same approach as in Goal-GAIL. The loss function is presented in Equation4.3 in Section 4.1.4. For training the discriminator, experiences are sampled using Hindsight Experience Replay (HER) for both expert and self-generated experiences. The reward function for RL incorporates the output from the discriminator. Various methods can be applied to utilise the discriminator's output as the GAIL reward, for example, use $D(s, g, a)$, $log(sigmoid(D(s, g, a)))$, or $sigmoid(D(s, g, a))$. In our experiments, we opt for $sigmoid(D(s, g, a))$ as the GAIL reward. The

Algorithm 5 Goal-based Self-adaptive Generative Adversarial Imitation Learning (Goal-SGAIL)

```
Require:
  - An off-policy algorithm $\mathbb{A}$                                    ▷ e.g. DDPG
  - An discriminator $\mathbb{D}$                                           ▷ usually MLP
  - A goal-based reward function $\mathcal{S} \times \mathcal{G} \times \mathcal{A} \to \mathbb{R}$   ▷ e.g. $r(s, a, g_d) = -1$ if fails, 0 if success
  - A self-collected replay buffer $\mathbb{R}_{\mathbb{B}}$ and an expert replay buffer $\mathbb{R}_{\mathbb{E}}$
  Initialise actor and critic networks for $\mathbb{A}$ and discriminator network for $\mathbb{D}$
  Initialise $\mathbb{R}_{\mathbb{B}}$, initialise $\mathbb{R}_{\mathbb{E}}$ with expert demonstration trajectories
  for $episode = (1, M)$ do                        ▷ Step 1: data collection and storage
      Collect trajectory $\tau$ using behaviour policy $\pi$ from $\mathbb{A}$
      Find expert trajectory $\tau_{e(min)}$ using Equations (4.4) and (4.5)   ▷ Check if high-quality
      if $d_{combine}(\tau, \tau_{e(min)}) > C_{d_{combine}}$ then
          Store $\tau$ in $\mathbb{R}_{\mathbb{E}}$ if its episode return is higher than $\tau_{e(min)}$
      else
          Directly store $\tau$ in $\mathbb{R}_{\mathbb{E}}$
      end if
  end for
  for $batch = (1, N_d)$ do                        ▷ Step 2: Train discriminator
      Sample demonstration transitions in $\mathbb{R}_{\mathbb{E}}$, sample self-collected transitions in $\mathbb{R}_{\mathbb{B}}$
      Train $\mathbb{D}$ using Equation (4.3)
  end for
  for $batch = (1, N)$ do                          ▷ Step 3: policy update with HER
      Sample transitions in $\mathbb{R}_{\mathbb{E}}$ and $\mathbb{R}_{\mathbb{B}}$, apply the ratio stated in Equation 4.6
      Combine these transitions and store them in $\mathbb{B}$
      Apply HER on transitions in $\mathbb{B}$, recalculate the reward
      Calculate combined reward for all samples using Equation (4.7)   ▷ Add GAIL reward
      Perform one step optimisation for $\mathbb{A}$ using $\mathbb{B}$
  end for
```

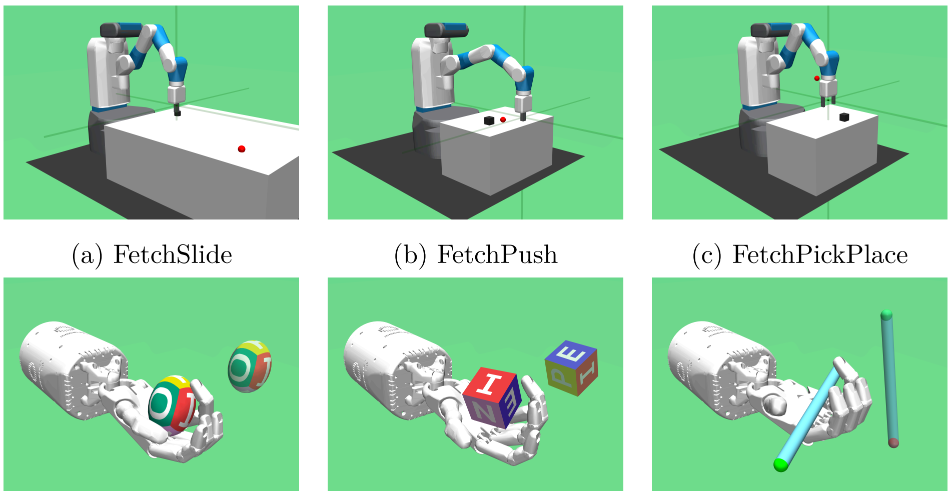
combined reward used for learning is as follows:

$$r_{combined} = (1 - \delta_{GAIL}) * r_{env} + \delta_{GAIL} * sigmoid(D(s, g, a)) \qquad (4.7)$$

where the GAIL weight δ_{GAIL} is used to control how much the GAIL reward affects the learning. The complete algorithm of Goal-SGAIL is shown in Algorithm 5.

4.2.3 Experimental Results

Environments



(a) FetchSlide (b) FetchPush (c) FetchPickPlace

(d) HandEgg (e) HandBlock (f) HandPen

Figure 4.2: OpenAI Gym robotics multi-goal RL environments [83]

The proposed Goal-SGAIL method is tested on various OpenAI Gym simulated multi-goal robotic manipulation tasks, see Figure 4.2. These environments are all designed for robot manipulation and are goal-conditioned:

- **FetchPush-v1**: Push a block on a table to reach a target position with the Fetch robot;
- **FetchSlide-v1**: Hit a block on a slippery table to slide to a target position with the Fetch robot;
- **FetchPickAndPlace-v1**: Pick a block on the table with the Fetch robot and reach a target position in the air;

- **HandManipulateEggRotate-v0**: Rotate an egg-shaped object within hand to reach a target orientation using the Shadow hand robot;
- **HandManipulateBlockRotateXYZ-v0**: Rotate a block-shaped object within hand to reach a target orientation using the Shadow hand robot;
- **HandManipulatePenRotate-v0**: Rotate a pen-shaped object within hand to reach a target orientation using the Shadow hand robot.

Implementation details and hyperparameters

In our experiments, each epoch is composed of 50 training cycles. Within each cycle, we first gather data from 40 episodes, followed by training the policy networks across 40 batches, each with a mini-batch size of 5120. After the completion of each epoch, we conduct an evaluation over 100 episodes to assess performance. Both the policy and discriminator networks are structured as MLPs, featuring 4 layers with 256 nodes per layer.

The proportion of expert to self-collected samples utilised for training adheres to Equation 4.6 with α set at 0.3. In both Goal-GAIL and Goal-SGAIL implementations, we apply a GAIL weight δ_{GAIL} of 0.1. The discriminator undergoes training for 40 batches per cycle, with a mini-batch size of 512

Specifically for Goal-SGAIL, we cap the maximum size of the expert buffer R_E at ten times the count of initial expert trajectories. The calculation of the combined goal pair distance in Goal-SGAIL takes into account the task-specific goal distance $d(g_a, g_b)$. For robot manipulation tasks, object position control tasks (Fetch tasks) utilise Euclidean distance, whereas object rotation tasks (Shadow hand tasks) employ the quaternion rotation difference as the distance metric. Furthermore, the threshold C_{comb} for combined goal pair distance checking varies between task types. In our experiments, we set $C_{comb} = 0.05$ for object position control tasks (Fetch tasks), and $C_{comb} = 1.0$for object rotation tasks (Shadow hand tasks).

4.2.4 Performance comparison

In our experiments, Goal-SGAIL is evaluated alongside the vanilla HER (as the RL baseline), and the imitation learning methods DDPGfD+HER and Goal-GAIL. For all LfD algorithms, expert demonstrations are generated with

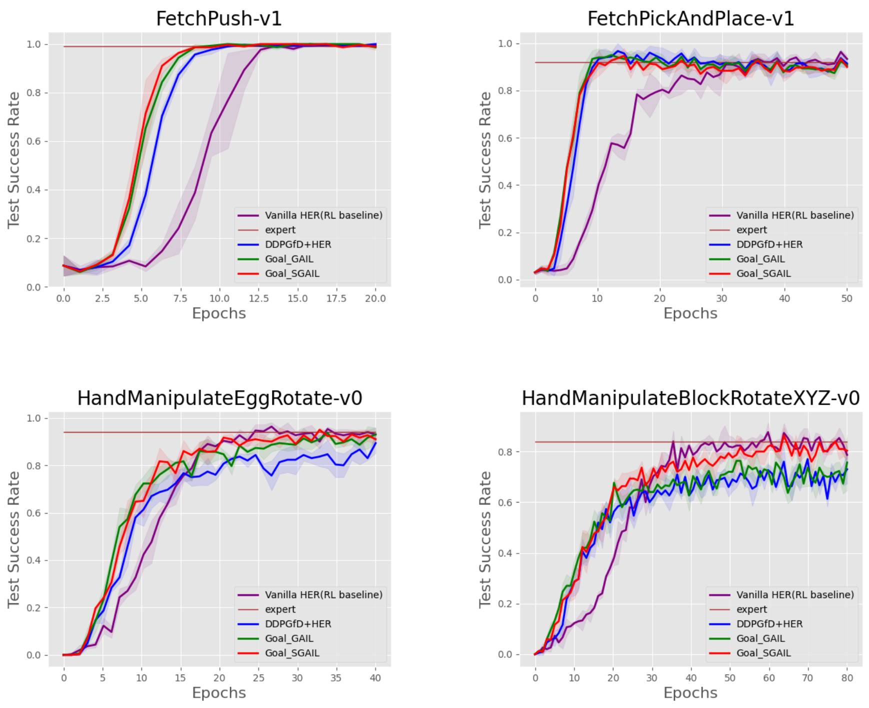


Figure 4.3: The learning curve for different LfD methods with optimal demonstration provided by a fully-trained RL agent. The purple, blue, green and red lines represent the learning curve for vanilla HER, DDPGfD+HER, Goal-GAIL, and Goal-SGAIL, respectively. The brown line represents the average success rate for the expert agent that is used to produce the demonstrations

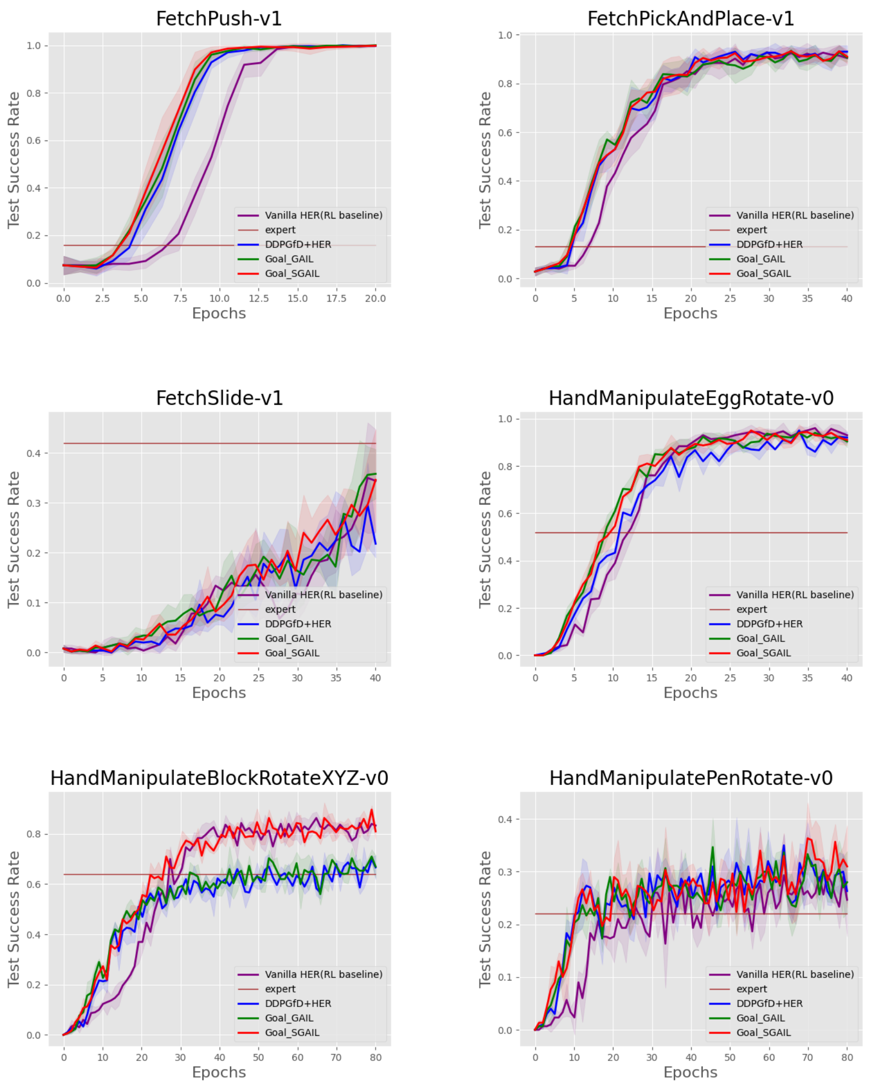


Figure 4.4: The learning curve for different LfD methods with sub-optimal demonstration provided by a semi-trained RL agent. The purple, blue, green and red lines represent the learning curve for vanilla HER, DDPGfD+HER, Goal-GAIL, and Goal-SGAIL, respectively. The brown line represents the average success rate for the expert agent that is used to produce the demonstrations

FetchPush					**FetchSlide**				
stage	1	2	3	4	stage	1	2	3	4
seed1	0.03	0.44	0.11	0	seed1	0	0	0.04	0.05
seed2	0.03	0.52	0.04	-0.02	seed2	0	0.01	0.01	0.03
seed3	0	0.38	0.11	0.01	seed3	0	0.03	0.07	0
seed4	0.05	0.46	0.09	0.01	seed4	0	-0.01	0.10	0.06
seed5	0.04	0.41	0.04	-0.01	seed5	0	0.01	-0.03	0.03
mean	**0.03**	**0.44**	**0.08**	0	mean	0	0.01	**0.04**	**0.04**
std	**0.02**	**0.05**	**0.03**	0.01	std	0	0.01	**0.04**	**0.02**
p-value	**0.022**	**0**	**0.001**	0.335	p-value	0.378	0.172	**0.005**	**0.009**
FetchPickAndPlace					**HandManipulateEggRotate**				
stage	1	2	3	4	stage	1	2	3	4
seed1	0.10	0.03	-0.01	0.01	seed1	0.11	0.13	-0.01	-0.01
seed2	0.06	0.06	-0.04	-0.03	seed2	0.09	0.08	-0.03	-0.02
seed3	0.10	0.04	0	-0.02	seed3	0.11	0.14	-0.01	-0.03
seed4	0.11	0.06	0	0.01	seed4	0.10	0.08	-0.01	-0.01
seed5	0.12	0.15	0.09	0.03	seed5	0.15	0.09	-0.02	0.02
mean	**0.10**	**0.07**	0.01	0	mean	**0.11**	**0.10**	-0.02	-0.01
std	**0.02**	**0.04**	0.04	0.02	std	**0.02**	**0.03**	0.01	0.02
p-value	**0**	**0**	0.269	1.000	p-value	**0**	**0**	0.012	0.211
HandManipulateBlockRotateXYZ					**HandManipulatePenRotate**				
stage	1	2	3	4	stage	1	2	3	4
seed1	0.06	0.25	0.05	-0.12	seed1	0.04	0.11	0.06	0.06
seed2	0.05	0.20	0.14	-0.03	seed2	0.08	0.07	0.03	-0.02
seed3	0.06	0.23	0.09	-0.06	seed3	0.05	0.13	0.03	0.08
seed4	0.04	0.17	0.10	0	seed4	0.04	0.11	0.08	0.03
seed5	0.07	0.24	0.06	-0.13	seed5	0.05	0.09	0.03	0.03
mean	**0.06**	**0.22**	**0.09**	-0.07	mean	**0.05**	**0.10**	**0.05**	**0.04**
std	**0.01**	**0.03**	**0.03**	0.05	std	**0.01**	**0.02**	**0.02**	**0.04**
p-value	**0**	**0**	**0**	0	p-value	**0**	**0**	**0**	**0.001**

Table 4.1: The statistical analysis of stage improvements: **Comparing Goal-SGAIL to HER using sub-optimal RL demonstrations**. The stage improvement during each learning stage is calculated for each run with a different seed. The mean and standard deviation across five seeds are also given. The p-value of a one-sample t-test is given to indicate whether the improvement is considered statistically significant and not random ($p - value < 0.05$). We specifically highlighted stages showing significant positive improvements, where the mean is greater than 0 and the p-value is less than 0.05

FetchPush					**FetchSlide**				
stage	1	2	3	4	stage	1	2	3	4
seed1	0.01	0.11	0.01	0.01	seed1	-0.01	0.01	0.01	-0.02
seed2	-0.01	0.08	0	-0.02	seed2	0	0	-0.02	-0.04
seed3	-0.01	-0.03	0	0	seed3	0	-0.02	-0.01	0
seed4	-0.01	0.06	0	0	seed4	0	-0.01	0.05	0.06
seed5	-0.01	0	0	0	seed5	0	-0.01	-0.01	0.10
mean	0	**0.04**	0	0	mean	0	-0.01	0	0.02
std	0.01	**0.05**	0.01	0.01	std	0.01	0.01	0.02	0.05
p-value	0.177	**0.006**	0.302	0.273	p-value	0.646	0.301	0.817	0.186
FetchPickAndPlace					**HandManipulateEggRotate**				
stage	1	2	3	4	stage	1	2	3	4
seed1	0.02	-0.02	0	0	seed1	0.02	0.01	0	0.02
seed2	0	-0.05	-0.01	0	seed2	-0.01	-0.03	-0.03	-0.01
seed3	-0.03	-0.08	-0.03	-0.01	seed3	-0.01	0.01	-0.01	0.01
seed4	-0.02	0.03	0.03	0.02	seed4	0	-0.03	0.02	0.01
seed5	-0.01	0.10	0.11	0.04	seed5	0	0.05	0.02	0.03
mean	-0.01	0	**0.02**	**0.01**	mean	0	0	0	**0.01**
std	0.02	0.07	**0.05**	**0.02**	std	0.01	0.03	0.02	**0.01**
p-value	0.127	0.686	**0.025**	**0.028**	p-value	0.974	0.768	0.901	**0.046**
HandManipulateBlockRotateXYZ					**HandManipulatePenRotate**				
stage	1	2	3	4	stage	1	2	3	4
seed1	0.01	0	0.02	0.04	seed1	0.01	0.01	0.01	0.03
seed2	-0.02	-0.02	0.09	0.15	seed2	0.03	0	0	-0.02
seed3	-0.02	0.04	0.07	0.11	seed3	0.01	0	-0.01	0.01
seed4	0	0	0.07	0.15	seed4	0	-0.01	0.02	0
seed5	0	-0.02	0.03	0	seed5	0	0.01	-0.02	0.01
mean	-0.01	0	**0.06**	**0.09**	mean	0.01	0	0	0.01
std	0.01	0.02	**0.03**	**0.06**	std	0.01	0.01	0.02	0.01
p-value	0.291	0.938	**0**	**0**	p-value	0.096	0.695	0.953	0.282

Table 4.2: The statistical analysis of stage improvements: **Comparing Goal-SGAIL to Goal-GAIL using sub-optimal RL demonstrations**. The stage improvement during each learning stage is calculated for each run with a different seed. The mean and standard deviation across five seeds are also given. The p-value of a one-sample t-test is given to indicate whether the improvement is considered statistically significant and not random ($p < 0.05$). We specifically highlighted stages showing significant positive improvements, where the mean is greater than 0 and the p-value is less than 0.05

random goals, collecting only successful trajectories. We utilise 10 demonstrations for the FetchPush and FetchPickAndPlace tasks, 100 demonstrations for the FetchSlide task, and 200 demonstrations for all three hand rotation tasks. Each experiment is conducted with five random seeds per environment/LfD method, and results are averaged across these seeds.

Initially, we assess Goal-SGAIL's performance relative to other LfD methods using optimal demonstrations provided by a fully trained RL agent. Subsequently, we explore LfD with demonstration datasets deemed sub-optimal due to inadequate goal space coverage. These sub-optimal demonstrations, collected from a semi-trained RL agent, are all successful trajectories but exhibit imbalanced goal coverage, particularly lacking in challenging goals. Our aim is to determine how Goal-SGAIL enhances learning efficiency in comparison to DDPGfD+HER and Goal-GAIL, especially in the context of demonstration sub-optimality.

Performance with optimal demonstration: The learning curves utilising different methods with optimal demonstrations are depicted in Figure 4.3. The results indicate that all three LfD algorithms (DDPGfD+HER, Goal-GAIL, and Goal-SGAIL) markedly outperform the vanilla HER baseline in terms of learning speed, particularly during the initial and intermediate stages of the learning process across all tested environments In specific tasks, such as FetchPush and HandEggRotation, both Goal-GAIL and Goal-SGAIL exhibit a marginally faster learning rate than DDPGfD+HER. However, the difference in performance between Goal-GAIL and Goal-SGAIL is not statistically significant.

While DDPGfD+HER and Goal-GAIL do not reach the same performance level as the RL baseline in certain tasks (notably HandEggRotation and HandBlockRotation), Goal-SGAIL consistently achieves an optimal performance level comparable to the RL baseline within the allocated learning time frame.

Upon examining the demonstration dataset, it becomes apparent that the demonstrations provide comprehensive and evenly distributed coverage of the goal space. The majority of the demonstration trajectories are near-perfect, achieving their goals within just a few steps. Leveraging HER for experience relabelling, the basic DDPGfD+HER method extracts sufficient information to enhance the learning process. Goal-GAIL sees a slight improvement in learning performance, thanks to the additional GAIL reward for all expert-like self-collected experiences. However, since only a limited number of self-collected trajectories surpass the demonstrations during learning, Goal-SGAIL does not notably increase the learning rate. Nonetheless, in scenarios where demonstrations are insufficient for LfD to achieve peak performance, Goal-SGAIL offers a valuable means to bypass this limitation.

Performance with sub-optimal expert demonstration: The learning curves comparing various methods with sub-optimal demonstrations are illustrated in Figure 4.4. A statistical analysis was performed on the learning results from these demonstrations to quantify the performance improvement attributed to Goal-SGAIL. The *improvement* is defined as the success rate difference between Goal-SGAIL and the methods it's compared against, such as the vanilla HER (the RL baseline) and Goal-GAIL (the IL baseline), to evaluate how significantly Goal-SGAIL accelerates the learning process. We regard each set of 10 epochs (or 5 for FetchPush) as a learning stage and calculate the *stage improvement* by averaging the *improvement* during this stage. The *stage improvements* from five runs of each experiment, including their mean and standard deviation, are detailed in Table 4.1 for comparisons between Goal-SGAIL and HER, and in Table 4.2 for comparison between Goal-SGAIL and Goal-GAIL. To determine the consistency of the improvement and ensure it's not due to chance, we conduct a one-sample t-test on the *improvements* collected from five runs at each learning stage. The resulting p-values for each learning stage are also listed in the mentioned tables. An improvement is deemed statistically significant and not random if $p < 0.05$.

From the learning curves and statistical analysis, we observe that in relatively simple robot manipulation tasks (FetchPush, FetchPickAndPlace, and HandEggRotate), where learning can converge within the specified simulation window, all LfD methods are capable of learning an optimal policy with sub-optimal demonstrations. In these instances, Goal-SGAIL demonstrates significant improvement during early and intermediate learning stages when compared to HER, though it shows only slight improvement at some stages compared to Goal-GAIL. In more complex tasks, such as HandBlockRotation, DDPGfD+HER and Goal-GAIL exhibit faster convergence rates in the initial stages of learning but fail to learn the optimal policy. While displaying a convergence rate similar to that of Goal-GAIL, Goal-SGAIL overcomes the bottleneck and ultimately achieves a similar final performance compared with optimal policy. For tasks such as FetchSlide and HandPenRotate, mastering an optimal policy through RL presents a significant challenge. Despite this, all

Learning from Demonstration (LfD) methods, supported by successful demonstrations, demonstrate slight advantages in achieving final performance over the HER baseline. In terms of learning speed, all LfD approaches exhibit marginally faster progress than HER in FetchSlide and are notably quicker in HandPenRotate. However, in these scenarios, Goal-SGAIL does not exhibit improvement over Goal-GAIL.

Upon reviewing the demonstration dataset, we find that despite the sub-optimal demonstrations lacking comprehensive coverage of the goal space, these trajectories excel at achieving their local goals. Consequently, they still offer sufficient guidance to enable all LfD methods to surpass the performance of the expert in most environments. However, in practical application scenarios where an RL agent is unavailable and demonstrations are typically gathered through human teleoperation, the quality of these sub-optimal demonstrations may be further compromised. This is attributed to the challenges human operators face in remotely controlling the robot to execute such complex tasks. In the following section, we will explore a case study that delves into using human teleoperation for LfD.

4.3 Learning from demonstration with data collected from teleoperation (Leap motion controller)

In this section, we present an LfD case study utilising a human expert and data gathered through teleoperation. We focus on a block rotation task with a Shadow Hand in the PyBullet [16] physics engine, as detailed by Zahlner et al. [122]. Additionally, they have developed a teleoperation system employing a Leap Motion Controller (LMC) for collecting demonstration data.

We will begin by outlining the setup of the environment and the procedure for acquiring human demonstration data. Following this, we will explore various LfD strategies, including our own Goal-SGAIL method, using human demonstrations and evaluating their respective performances.

4.3.1 Environment setup

The learning environment is based on a shadow hand block rotation task within the OpenAI Gym environment, utilising PyBullet [121]. The structure of the teleoperation system implemented by Zahlner et al. [122] and a visualization of the task are depicted in Figure 4.5. In this setup, a block is positioned

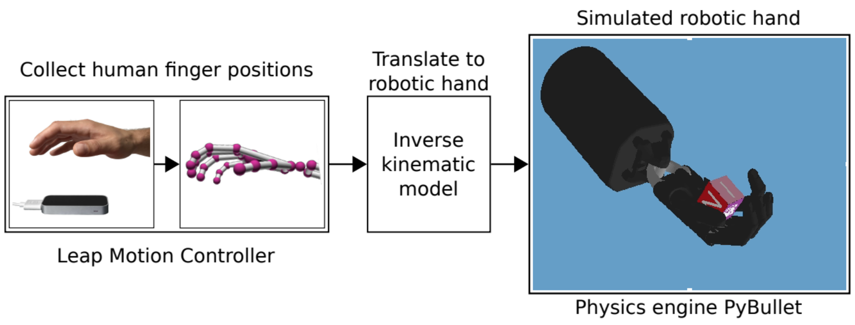


Figure 4.5: PyBullet shadow hand block rotation task and the teleoperation system structure in [122], copied from the original paper.

in the hand with a randomly assigned initial orientation. The objective is to manipulate the object in-hand to achieve and maintain a specific target orientation, within a predefined threshold of $\epsilon = 0.1\ rad$. Each episode spans 99 timesteps.

The action space is comprised of 20 dimensions, corresponding to the absolute positions of all controllable hand joints. The observation space includes the positions and velocities of the 20 non-coupled hand joints, the object's current Cartesian pose, its linear and angular velocities, and the desired target pose. Since the environment is designed for goal-conditioned learning, this environment facilitates the extraction of the object's desired and achieved goals for separate use. A sparse binary reward system is employed, where the agent receives a score of -1 when the target goal is not achieved and 0 upon successful achievement.

4.3.2 Data collection with human teleoperation

The teleoperation system employs the LMC, which uses an infrared stereo camera to track human hand movements. The LMC comes equipped with an SDK that fits a hand model to the image stream, providing the positions of the hand's joints. This system leverages an inverse kinematic model of the shadow robot hand [19] to translate human finger movements into robotic joint positions within the PyBullet simulation platform. Human demonstrators control the simulated robot hand, manipulating the object while observing the action through PyBullet's real-time rendering on the screen.

During the data collection phase for the shadow hand block rotation task, hand joint data from the LMC is captured in real-time across each 99-timestep episode. Successful episodes, where the object is manipulated as required, are stored in an expert buffer along with state, action, and episode-specific information, forming the demonstration dataset.

4.3.3 Experimental Results

We compare the LfD performance over two demonstration datasets: 50 successful trajectories produced by a fully-trained RL agent (RL demos), and 50 successful trajectories collected through human teleoperation (Human teleop demos).

Analysis for demonstration datasets

Before comparing the learning performance, we discuss some differences between the two types of demonstration datasets. The teleoperation system's setup is relatively simple (especially when compared to CyberGloves and VR simulation) and lacks feedback, making it challenging for human demonstrators to control the simulated robot hand smoothly and accurately. Additionally, the RL-trained agent controls the robot hand in a unique manner that diverges from typical human habits.

The average episode cumulative return for the human teleoperation demonstrations is -76, whereas for RL demonstrations, it is -15, indicating superior performance by the RL demos compared to the human demos. Additionally, we examined the distribution of goal distances between the initial position and the target goal $d(g_{init}, g_o)$ across all trajectories in each dataset, as illustrated in Figure 4.6. For a multi-goal learning problem, LfD stands to gain from demonstrations that adequately cover the goal space. Ideally, the goal distances $d(g_{init}, g_o)$ across all trajectories should span all ranges and be more uniformly distributed, a pattern observed in the RL demos. In contrast, the majority of human teleoperation demos feature shorter goal distances (representing easier goals), with scarcely any coverage of longer goal distance areas (indicating more challenging goals). This suggests that while human teleoperation demos provide ample examples of easier sub-tasks, they fall short in offering demonstrations for more challenging sub-tasks. In summary, the human teleoperation demonstrations seem less optimal compared to the RL demonstrations.

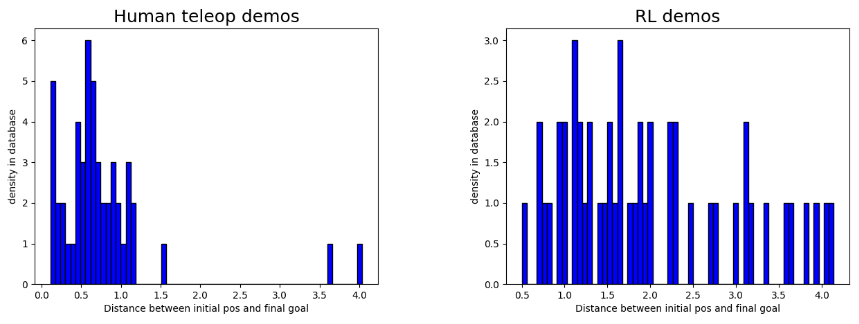


Figure 4.6: The distribution of the initial and target goal distances of all trajectories for the two types of data sets in LfD experiments: we calculate the goal distance $d(g_{init}, g_o)$ between the initial pos g_{init} and the target goal g_o for each demonstration trajectory, and classify them into different distance intervals. The density of the distance intervals of all trajectories is plotted for the two datasets.

Learning performance comparison

In our LfD experiments, we compare three LfD algorithms: DDPGfD with HER, Goal-GAIL, and our proposed method, Goal-SGAIL, using vanilla HER RL as the baseline. The implementation parameters for all algorithms remain consistent with those detailed in Section 4.2.3, with the exception of setting the expert ratio to $\alpha = 0.5$ and the GAIL weight to $\delta_{GAIL} = 0.3$.

The learning curves for both human teleoperation demonstrations and RL demonstrations are illustrated in Figure 4.7, with the results averaged across three random seeds for each method. We perform a statistical analysis similar to the one in Section 4.2.4, focusing on the data obtained from human teleoperation demonstrations. Our analysis specifically evaluates the improvement of Goal-SGAIL over HER and Goal-SGAIL over Goal-GAIL. The findings are presented in Table 4.3.

The analysis indicates that with optimal RL demonstrations, all Learning from Demonstration (LfD) methods outperform the vanilla DDPG+HER. Specifically, DDPGfD+HER and Goal-GAIL enhance the learning process more effectively than Goal-SGAIL. When employing human teleoperation demonstrations, both DDPGfD+HER and Goal-GAIL experience a reduction in learning rates during the intermediate and final stages, failing to achieve the performance level of the RL baseline. Conversely, Goal-SGAIL markedly

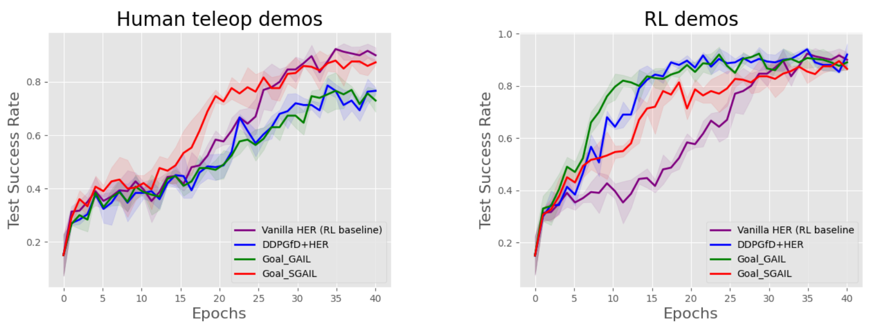


Figure 4.7: The learning curve for different LfD methods with human teleop demos and RL demos on PyBullet Shadowhandblockrotate environment. The purple, blue, green and yellow lines represent the learning curve for vanilla DDPG+HER, DDPGfD+HER with demonstration, Goal-GAIL, and Goal-SGAIL, respectively.

Improvement of Goal-SGAIL over HER					**Improvement of Goal-SGAIL over Goal-GAIL**				
stage	1	2	3	4	stage	1	2	3	4
seed1	0	0.06	0.07	-0.05	seed1	0.02	0.08	0.14	0.06
seed2	0.02	0.07	0.05	-0.01	seed2	0.03	0.09	0.22	0.15
seed3	0.02	0.13	0.07	-0.03	seed3	0.05	0.17	0.22	0.17
mean	0.01	**0.09**	**0.06**	-0.03	mean	**0.04**	**0.11**	**0.19**	**0.13**
std	0.01	**0.03**	**0.01**	0.01	std	**0.01**	**0.04**	**0.04**	**0.05**
p-value	0.083	**0**	**0.001**	0	p-value	**0**	**0**	**0**	**0**

Table 4.3: The statistical analysis of stage improvements: **Comparing Goal-SGAIL to HER/Goal-GAIL using human demonstrations**. The stage improvement during each learning stage is calculated for each run with a different seed. The mean and standard deviation across three seeds are also given. The p-value of a one-sample t-test is given to indicate whether the improvement is considered statistically significant and not random ($p < 0.05$). We specifically highlighted stages showing significant positive improvements, where the mean is greater than 0 and the p-value is less than 0.05

boosts the learning rate during the intermediate stages and attains a final performance that is on par with the RL baseline, showcasing notable advancements over HER during these phases. Furthermore, it consistently surpasses Goal-GAIL at every stage. However, it is noted that Goal-SGAIL's final performance is marginally lower than HER's, a pattern also observed in HandEggRotation and HandBlockRotation with sub-optimal RL demonstrations in Table 4.1. This may suggest that relying more on self-generated demonstrations in the final learning stages does not benefit the learning process.

In our experiments, the human teleoperation demonstrations were markedly sub-optimal, predominantly focusing on simpler sub-tasks beneficial only in the initial learning phases. As LfD algorithms' RL components begin to master these easier sub-tasks, they move on to tackle more challenging ones, rendering the human teleoperation demonstrations less instructive. Goal-SGAIL, by augmenting the demonstration dataset with successful self-generated trajectories that surpass the original demonstrations, manages to outperform the other methods.

4.4 Summary

This chapter explores Learning from Demonstration (LfD) methods tailored for multi-goal learning challenges. We have examined the Generative Adversarial Imitation Learning (GAIL) method, along with enhancements to GAIL aimed at overcoming the challenges posed by sub-optimal demonstrations (SAIL) and adapting GAIL for multi-goal contexts (Goal-GAIL).

We introduced the **Goal-based Self-adaptive Generative Adversarial Imitation Learning (Goal-SGAIL)** method, which merges Goal-GAIL with the self-adaptive concept inspired by SAIL. This incorporation of the self-adaptive approach transitions the Learning from Demonstration (LfD) process from relying on existing demonstrations to leveraging high-quality, self-generated experiences as the agent's performance surpasses that of the demonstrations. This transition aims to address the challenges associated with the sub-optimal quality of the demonstrations.

The initial experimental outcomes using demonstrations from RL-trained agents show that with optimal RL demonstrations, even simple LfD methods like DDPGfD with HER, as well as more advanced methods like Goal-GAIL, can enhance learning speed. However, when faced with sub-optimal

RL demonstrations, both DDPGfD+HER and Goal-GAIL encounter limitations. Goal-SGAIL, leveraging its self-adaptive strategy, assists the learning process in surmounting the sub-optimality of the demonstrations. Moreover, in real-world LfD settings, such as multi-goal in-hand object rotation tasks using demonstrations collected via human teleoperation, the quality of the demonstrations is significantly sub-optimal in terms of both goal space coverage and the speed of achieving individual local goals. When implementing various LfD approaches with human teleoperation demonstrations, methods like DDPGfD+HER and Goal-GAIL do not achieve the same optimal level as the RL baseline. In contrast, Goal-SGAIL demonstrates resilience against the profound sub-optimality of the demonstrations, managing to learn an optimal policy and markedly outperforming both DDPGfD and Goal-GAIL.

The **contribution of Goal-SGAIL** is its development of a strategy for selecting high-quality, self-generated trajectories within a multi-goal learning framework and incorporating these into the GAIL demonstration dataset to facilitate a self-adaptive transition during the learning process. However, the **limitation of Goal-SGAIL** lies in: Firstly, in scenarios where demonstrations are less sub-optimal, such as those provided by RL, Goal-SGAIL does not demonstrate significant improvement over Goal-GAIL. Secondly, there are instances where the final performance of the learning process falls short of the RL baseline. This suggests that Goal-SGAIL may need to be annealed during the final stages of learning to encourage more exploration through RL itself.

Chapter 5

Conclusions and Future Work

In this dissertation, we explored reinforcement learning (RL) and imitation learning (IL) methods for in-hand dexterous manipulation tasks (InDex) involving anthropomorphic robot hands, with a particular emphasis on enhancing learning efficiency for multi-goal-oriented tasks. Through this endeavour, we developed several methods within both RL and IL frameworks:

(1) An experience prioritisation method designed to boost RL learning efficiency by optimising the use of experiences;

(2) An automatic goal selection method intended to enhance RL learning efficiency through improved exploration of the task space;

(3) A GAIL-based imitation learning approach aimed at increasing the efficiency of GAIL imitation learning in multi-goal contexts.

This chapter wraps up the dissertation by summarising their key contributions and limitations, and by proposing directions for future research.

5.1 Conclusions

5.1.1 Goal Density-based Hindsight Experience Prioritisation (GDP)

In Chapter 3 (Section 3.2), we addressed the research question RQ.1 (outlined in Section 1.1)by exploring how to effectively leverage the experience accumulated during the RL learning process. We introduced the GDP method, which priorities experiences based on the density distribution of achieved points, specifically targeting those rarely seen in the replay buffer to enhance the learning process's sample efficiency. Furthermore, we developed the Prioritisation Switching with Ensemble Strategy (PSES), a strategy that amalgamates various experience prioritisation methods to optimise overall performance by selectively applying the most effective component from each method throughout the learning process.

The **contribution of GDP** is providing a novel approach to analyzing experience density, specifically through the utilization of goal-pairs. This approach results in decreased computational effort compared to previous density-based methods such as MEP.

The **limitation of GDP**, on the other hand, stems from its limited applicability across a wide range of manipulation tasks. Merely focusing on prioritizing newly encountered goals proves insufficient.

5.1.2 Policy-level-based Curriculum Goal Selection (PL-CGS)

In Chapter 3 (Section 3.3), we continued addressing the research question RQ.1 by investigating efficient strategies for exploring the goal space through curriculum-based goal generation during the learning process. We introduced the PL-CGS method, which assesses the goal-pair distribution of the most recently collected experiences and adopts a curriculum approach to select goals of intermediate difficulty appropriate for the current agent's learning stage. To analyse the goal-pair distribution, we implemented two types of distribution analysis strategies: GMM and MLP discriminator. Experimental findings indicated that PL-CGS, when paired with GMM, enhances both the learning speed and the final outcomes in challenging robot manipulation tasks. However, the PL-CGS approach utilising a discriminator encounters difficulties in effectively training a discriminator to accurately identify suitable goals, leading to subpar performance.

The **contribution of PL-CGS**lies in its novel method for analyzing Goals of Intermediate Difficulty (GOID) to guide goal selection and generation, specifically within the framework of multi-goal learning.

The **limitation of PL-CGS** is highlighted by a noticeable drop in success rates as learning approaches convergence in various tasks. This approach's emphasis on GOID tends to favour achievable goals, thereby neglecting the learning opportunities presented by unexplored goals during advanced stages.

5.1.3 Goal-based Self-Adaptive Generative Adversarial Imitation Learning (Goal-SGAIL)

In Chapter 4, we tackled research question RQ.2 (outlined in Section 1.1), focusing on Generative Adversarial Imitation Learning (GAIL) [39] and its enhancements. We introduced the Goal-SGAIL method, aimed at enhancing learning efficiency by selecting high-quality, self-generated trajectories for inclusion in the GAIL demonstration dataset. Goal-SGAIL proved effective in addressing the sub-optimality of demonstrations, demonstrating improved learning outcomes in certain tasks. Additionally, we explored a practical Learning from Demonstration (LfD) scenario involving a multi-goal in-hand object rotation task, utilising demonstrations gathered through human teleoperation. We found that the quality of demonstrations significantly impacts performance, especially when compared with RL outcomes for traditional LfD methods like DDPGfD+HER and Goal-GAIL. Nevertheless, Goal-SGAIL managed to surpass RL performance, even with highly sub-optimal demonstrations from human teleoperation.

The **contribution of Goal-SGAIL** lies in its innovative approach to selecting high-quality, self-generated trajectories within a multi-goal learning framework and integrating these into the GAIL demonstration dataset to facilitate self-adaptive learning.

The **limitation of Goal-SGAIL** is twofold: Firstly, in situations where the demonstrations are less sub-optimal, such as those generated by RL, Goal-SGAIL does not exhibit a marked improvement over Goal-GAIL. Secondly, there are occasions when the final performance of the learning process does not meet the RL baseline. This indicates that Goal-SGAIL might require tapering off during the latter stages of learning to promote increased exploration driven by the RL component.

5.2 Future work

Machine learning has demonstrated significant capabilities in addressing a wide range of robotic tasks. However, when it comes to object manipulation tasks involving humanoid robot hands, the high degree of freedom in the robot's structure and the complexity of operating environments limit its application in real-world scenarios, necessitating further enhancements.

In Reinforcement Learning (RL), tackling multi-goal-oriented in-hand dexterous manipulation (InDex) tasks presents a notable challenge. While Hindsight Experience Replay (HER) has enhanced sample efficiency for such scenarios, there's ample room to boost learning efficiency through various approaches.

Our investigation into existing methods led us to propose a novel approach for experience prioritization. However, few studies address the need for adaptive experience sampling strategies across different learning stagesmost focus on only one aspect. In the initial stages of learning, sampling should predominantly come from achieved areas, enabling the agent to swiftly grasp a basic model of the state-goal and action space mapping. In contrast, during the later stages, as the agent becomes proficient with easily achieved goals, prioritising experiences related to seldom-achieved goals becomes crucial for advancing the agent's capabilities. Developing a curriculum strategy that transitions experience sampling from 'most seen' to 'rarely seen' experiences warrants further exploration.

Curriculum learning strategies are applicable not only in experience utilisation but also in data collection for goal generation in goal-conditioned tasks. We've introduced an automated goal-generation method that suggests goals based on the current capabilities of the agent. However, the approach to adapting the policy level for different goals remains simplistic. The reliance on prior experiences, even if recent, introduces a delay in goal generation. Enhancing the analysis strategy to determine appropriate goals, which are challenging enough to guide the agent without being overly difficult, requires additional research effort.

In the field of Imitation Learning (IL), leveraging learning from demonstration has proven highly effective in accelerating learning speeds across many straightforward robot manipulation tasks. However, algorithms tailored for multi-goal-oriented tasks still require significant research attention. While we've introduced a method to enhance learning efficiency for the goal-based Generative Adversarial Imitation Learning (GAIL) approach, there's room for improvement in the strategy for selecting high-quality self-collected experiences as demonstrations. Expanding beyond merely selecting from successful trajectories, we could consider incorporating unsuccessful trajectories and adjusting experiences through hindsight experience relabelling. Additionally, the quality of self-collected experiences should be evaluated in relation to different learning stages, ensuring that appropriate experiences are presented as demonstrations for each stage of learning.

In practical scenarios, gathering demonstrations with human experts is often deemed the most efficient approach, with teleoperation being a viable method. However, as outlined in Section 4.1.5, the quality of collected demonstrations may not always meet the standards required for Learning from Demonstration (LfD). And it is time-consuming to collect enough demonstrations for LfD to utilise. Therefore, it becomes imperative to develop methods aimed at enhancing the quality and quantity of demonstrations provided by human experts. Recent advancements have enabled the collection of vision-based demonstrations from humans without the need for teleoperation [87]. Further research can explore the integration of this technique with our proposed Goal-SGAIL method to achieve LfD from human demonstrations.

References

[1] N. Aghasadeghi and T. Bretl. Maximum entropy inverse reinforcement learning in continuous state spaces with path integrals. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1561–1566. IEEE, 2011.

[2] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[3] 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. *arXiv preprint arXiv:1808.00177*, 2018.

[4] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[5] S. P. Arunachalam, S. Silwal, B. Evans, and L. Pinto. Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation. In *2023 ieee international conference on robotics and automation (icra)*, pages 5954–5961. IEEE, 2023.

[6] C. Bai, P. Liu, W. Zhao, and X. Tang. Guided goal generation for hindsight multi-goal reinforcement learning. *Neurocomputing*, 2019.

[7] Y. Bai and C. K. Liu. Dexterous manipulation using both palm and fingers. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1560–1565. IEEE, 2014.

[8] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[9] A. Bicchi and R. Sorrentino. Dexterous manipulation through rolling. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, volume 1, pages 452–457. IEEE, 1995.

[10] I. M. Bullock, R. R. Ma, and A. M. Dollar. A hand-centric classification of human and robot dexterous manipulation. *IEEE transactions on Haptics*, 6(2):129–144, 2012.

[11] W. Cai, X. Liu, X. Liu, and B. Tang. Goal-conditioned imitation learning (goalgail) replication, 2020. Submitted to NeurIPS 2019 Reproducibility Challenge.

[12] A. Carfì, T. Patten, Y. Kuang, A. Hammoud, M. Alameh, E. Maiettini, A. I. Weinberg, D. Faria, F. Mastrogiovanni, G. Alenyà, et al. Hand-object interaction: From human demonstrations to robot manipulation. *Frontiers in Robotics and AI*, page 316, 2021.

[13] N. Chavan-Dafle and A. Rodriguez. Prehensile pushing: In-hand manipulation with push-primitives. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6215–6222. IEEE, 2015.

[14] C. Colas, P.-Y. Oudeyer, O. Sigaud, P. Fournier, and M. Chetouani. Curious: Intrinsically motivated modular multi-goal reinforcement learning. In *International Conference on Machine Learning*, pages 1331–1340, 2019.

[15] C. Coppola, G. Solak, and L. Jamone. An affordable system for the teleoperation of dexterous robotic hands using leap motion hand tracking and vibrotactile feedback. In *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 920–926. IEEE, 2022.

[16] E. Coumans and Y. Ba. Pybullet, a python module for physics simulation for games, robotics and machine learning. `https://pybullet.org`, 2016.

[17] M. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[18] M. P. Deisenroth, C. E. Rasmussen, and D. Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. *Robotics: Science and Systems VII*, pages 57–64, 2011.

[19] J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. 1955.

[20] A. D. Deshpande, Z. Xu, M. J. V. Weghe, B. H. Brown, J. Ko, L. Y. Chang, D. D. Wilkinson, S. M. Bidic, and Y. Matsuoka. Mechanisms of the anatomically correct testbed hand. *IEEE/ASME Transactions on Mechatronics*, 18(1):238–250, 2011.

[21] Y. Ding, C. Florensa, P. Abbeel, and M. Phielipp. Goal-conditioned imitation learning. *Advances in neural information processing systems*, 32, 2019.

[22] M. Eppe, S. Magg, and S. Wermter. Curriculum goal masking for continuous deep reinforcement learning. In *2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pages 183–188. IEEE, 2019.

[23] P. Falco, A. Attawia, M. Saveriano, and D. Lee. On policy learning robust to irreversible events: An application to robotic in-hand manipulation. *IEEE Robotics and Automation Letters*, 3(3):1482–1489, 2018.

[24] B. Fang, S. Jia, D. Guo, M. Xu, S. Wen, and F. Sun. Survey of imitation learning for robotic manipulation. *International Journal of Intelligent Robotics and Applications*, 3:362–369, 2019.

[25] T. Feix, J. Romero, H.-B. Schmiedmayer, A. M. Dollar, and D. Kragic. The grasp taxonomy of human grasp types. *IEEE Transactions on Human-Machine Systems*, 46(1):66–77, 2015.

[26] C. Florensa, D. Held, X. Geng, and P. Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR, 2018.

[27] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pages 482–495. PMLR, 2017.

[28] J. Fu, K. Luo, and S. Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[29] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[30] I. Gaiser, S. Schulz, A. Kargov, H. Klosek, A. Bierbaum, C. Pylatiuk, R. Oberle, T. Werner, T. Asfour, G. Bretthauer, et al. A new anthropomorphic robotic hand. In *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*, pages 418–422. IEEE, 2008.

[31] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[32] M. Grebenstein, A. Albu-Schäffer, T. Bahls, M. Chalon, O. Eiberger, W. Friedl, R. Gruber, S. Haddadin, U. Hagn, R. Haslinger, et al. The dlr hand arm system. In *2011 IEEE International Conference on Robotics and Automation*, pages 3175–3182. IEEE, 2011.

[33] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[34] A. Gupta, C. Eppner, S. Levine, and P. Abbeel. Learning dexterous manipulation for a soft robotic hand from human demonstrations. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3786–3793. IEEE, 2016.

[35] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[36] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[37] A. Handa, K. Van Wyk, W. Yang, J. Liang, Y.-W. Chao, Q. Wan, S. Birchfield, N. Ratliff, and D. Fox. Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9164–9170. IEEE, 2020.

[38] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[39] J. Ho and S. Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[40] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[41] R. D. Howe. Tactile sensing and control of robotic manipulation. *Advanced Robotics*, 8(3):245–261, 1993.

[42] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[43] J. M. Hyde and M. R. Cutkosky. A phase management framework for event-driven dextrous manipulation. *IEEE Transactions on Robotics and Automation*, 14(6):978–985, 1998.

[44] S. Jacobsen, E. Iversen, D. Knutti, R. Johnson, and K. Biggers. Design of the utah/mit dextrous hand. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 1520–1532. IEEE, 1986.

[45] D. Jain, A. Li, S. Singhal, A. Rajeswaran, V. Kumar, and E. Todorov. Learning deep visuomotor policies for dexterous hand manipulation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3636–3643. IEEE, 2019.

[46] S. M. Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

[47] A. Kargov, T. Asfour, C. Pylatiuk, R. Oberle, H. Klosek, S. Schulz, K. Regenstein, G. Bretthauer, and R. Dillmann. Development of an

anthropomorphic hand for a mobile assistive robot. In *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.*, pages 182–186. IEEE, 2005.

[48] K. D. Katyal, E. W. Staley, M. S. Johannes, I.-J. Wang, A. Reiter, and P. Burlina. In-hand robotic manipulation via deep reinforcement learning. In *Proceedings of the Workshop on Deep Learning for Action and Interaction, in Conjunction with Annual Conference on Neural Information Processing Systems, Barcelona, Spain*, volume 9, 2016.

[49] D. Katz, Y. Pyuro, and O. Brock. Learning to manipulate articulated objects in unstructured environments using a grounded relational representation. In *In Robotics: Science and Systems*. Citeseer, 2008.

[50] H. Kawasaki, T. Komatsu, and K. Uchiyama. Dexterous anthropomorphic robot hand with distributed tactile sensor: Gifu hand ii. *IEEE/ASME transactions on mechatronics*, 7(3):296–303, 2002.

[51] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[52] O. Kroemer, S. Niekum, and G. Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *The Journal of Machine Learning Research*, 22(1):1395–1476, 2021.

[53] Y. Kuang, G. Vogiatzis, and D. R. Faria. Improving exploration efficiency of single-goal in-hand manipulation reinforcement learning by progressive goal generation. 2020.

[54] Y. Kuang, A. I. Weinberg, G. Vogiatzis, and D. R. Faria. Goal density-based hindsight experience prioritization for multi-goal robot manipulation reinforcement learning. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 432–437. IEEE, 2020.

[55] V. Kumar. *Manipulators and Manipulation in high dimensional spaces*. PhD thesis, 2016.

[56] V. Kumar, A. Gupta, E. Todorov, and S. Levine. Learning dexterous manipulation policies from experience and imitation. *arXiv preprint arXiv:1611.05095*, 2016.

[57] V. Kumar and E. Todorov. Mujoco haptix: A virtual reality system for hand manipulation. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 657–663. IEEE, 2015.

[58] V. Kumar, E. Todorov, and S. Levine. Optimal control with learned local models: Application to dexterous manipulation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 378–383. IEEE, 2016.

[59] D. Leidner, C. Borst, A. Dietrich, M. Beetz, and A. Albu-Schäffer. Classifying compliant manipulation tasks for automated planning in robotics. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1769–1776. IEEE, 2015.

[60] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[61] S. Levine and V. Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9, 2013.

[62] R. Li, H. Wang, and Z. Liu. Survey on mapping human hand motion to robotic hands for teleoperation. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(5):2647–2665, 2021.

[63] S. Li, X. Ma, H. Liang, M. Görner, P. Ruppel, B. Fang, F. Sun, and J. Zhang. Vision-based teleoperation of shadow dexterous hand using end-to-end deep neural network. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 416–422. IEEE, 2019.

[64] T. Li, W. Xi, M. Fang, J. Xu, and M. Q.-H. Meng. Learning to solve a rubik's cube with a dexterous hand. *arXiv preprint arXiv:1907.11388*, 2019.

[65] Y. Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[66] 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. *2016 International Conference on Learning Representations (ICLR)*, 2016.

[67] Y. Lin and Y. Sun. Robot grasp planning based on demonstrated grasp strategies. *The International Journal of Robotics Research*, 34(1):26–42, 2015.

[68] J. Liu, F. Feng, Y. C. Nakamura, and N. S. Pollard. A taxonomy of everyday grasps in action. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 573–580. IEEE, 2014.

[69] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848*, 2018.

[70] R. R. Ma and A. M. Dollar. On dexterity and dexterous manipulation. In *2011 15th International Conference on Advanced Robotics (ICAR)*, pages 1–7. IEEE, 2011.

[71] B. Manela and A. Biess. Bias-reduced hindsight experience replay with virtual goal prioritization. *arXiv preprint arXiv:1905.05498*, 2019.

[72] 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 *International conference on machine learning*, pages 1928–1937, 2016.

[73] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[74] 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, 2015.

[75] M. Mudigonda, P. Agrawal, M. Deweese, and J. Malik. Investigating deep reinforcement learning for grasping objects with an anthropomorphic hand. 2018.

[76] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

[77] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

[78] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *The Journal of Machine Learning Research*, 21(1):7382–7431, 2020.

[79] H. Nguyen and H. La. Review of deep reinforcement learning for robot manipulation. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pages 590–595. IEEE, 2019.

[80] H. Nguyen, H. M. La, and M. Deans. Hindsight experience replay with experience ranking. In *2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pages 1–6. IEEE, 2019.

[81] T. Okada. Object-handling system for manual industry. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(2):79–89, 1979.

[82] A. M. Okamura, N. Smaby, and M. R. Cutkosky. An overview of dexterous manipulation. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 255–262. IEEE, 2000.

[83] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[84] R. Portelas, C. Colas, L. Weng, K. Hofmann, and P.-Y. Oudeyer. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*, 2020.

[85] U. Prieur, V. Perdereau, and A. Bernardino. Modeling and planning high-level in-hand manipulation actions from human knowledge and active learning from demonstration. In *2012 IEEE/RSJ International Conference on intelligent Robots and Systems*, pages 1330–1336. IEEE, 2012.

[86] Y. Qin, H. Su, and X. Wang. From one hand to multiple hands: Imitation learning for dexterous manipulation from single-camera teleoperation. *IEEE Robotics and Automation Letters*, 7(4):10873–10881, 2022.

[87] 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. In *European Conference on Computer Vision*, pages 570–587. Springer, 2022.

[88] I. Radosavovic, X. Wang, L. Pinto, and J. Malik. State-only imitation learning for dexterous manipulation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7865–7871. IEEE, 2021.

[89] M. Ragaglia et al. Robot learning from demonstrations: Emulation learning in environments with moving obstacles. *Robotics and autonomous systems*, 101:45–56, 2018.

[90] 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.

[91] A. Rajeswaran, K. Lowrey, E. V. Todorov, and S. M. Kakade. Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems*, pages 6550–6561, 2017.

[92] Z. Ren, K. Dong, Y. Zhou, Q. Liu, and J. Peng. Exploration via hindsight goal generation. *arXiv preprint arXiv:1906.04279*, 2019.

[93] S. robot company. Shadowrobot dexterous hand. `https://www.shadowrobot.com/products/dexterous-hand/`.

[94] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings*

of the fourteenth international conference on artificial intelligence and statistics, pages 627–635, 2011.

[95] P. Ruppel and J. Zhang. Learning object manipulation with dexterous hand-arm systems from human demonstration. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5417–5424. IEEE, 2020.

[96] M. Saeed, M. Nagdi, B. Rosman, and H. H. Ali. Deep reinforcement learning for robotic hand manipulation. In *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, pages 1–5. IEEE, 2021.

[97] T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *International Conference on Machine Learning*, pages 1312–1320, 2015.

[98] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

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

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

[101] L. Sciavicco and B. Siciliano. *Modelling and control of robot manipulators*. Springer Science & Business Media, 2012.

[102] 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.

[103] J. Shi, J. Z. Woodruff, P. B. Umbanhowar, and K. M. Lynch. Dynamic in-hand sliding manipulation. *IEEE Transactions on Robotics*, 33(4):778–795, 2017.

[104] W. Si, N. Wang, and C. Yang. A review on manipulation skill acquisition through teleoperation-based learning from demonstration. *Cognitive Computation and Systems*, 3(1):1–16, 2021.

[105] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, pages 387–395, 2014.

[106] R. Suárez, J. Cornella, and M. R. Garzón. *Grasp quality measures.* Citeseer, 2006.

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

[108] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[109] T. Takaki and T. Omata. High-performance anthropomorphic robot hand with grasping-force-magnification mechanism. *IEEE/ASME Transactions on mechatronics*, 16(3):583–591, 2010.

[110] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[111] J. C. Trinkle and R. P. Paul. Planning for dexterous manipulation with sliding contacts. *The International Journal of Robotics Research*, 9(3):24–48, 1990.

[112] M. L. Turner. *Programming dexterous manipulation by demonstration.* PhD thesis, Citeseer, 2001.

[113] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[114] H. Van Hoof, T. Hermans, G. Neumann, and J. Peters. Learning robot in-hand manipulation with tactile features. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 121–127. IEEE, 2015.

[115] M. Večerík, 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. *arXiv preprint arXiv:1707.08817*, 2017.

[116] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[117] P. Weber, E. Rueckert, R. Calandra, J. Peters, and P. Beckerle. A low-cost sensor glove with vibrotactile feedback and multiple finger joint and hand motion sensing for human-robot interaction. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 99–104. IEEE, 2016.

[118] Z. Xu, V. Kumar, and E. Todorov. A low-cost and modular, 20-dof anthropomorphic robotic hand: design, actuation and modelling. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 368–375. IEEE, 2013.

[119] Z. Xu and E. Todorov. Design of a highly biomimetic anthropomorphic robotic hand towards artificial limb regeneration. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3485–3492. IEEE, 2016.

[120] C. Yu and P. Wang. Dexterous manipulation for multi-fingered robotic hands with reinforcement learning: a review. *Frontiers in Neurorobotics*, 16:861825, 2022.

[121] Zahlner and Hirschmanner. Openaai gym shadow dexterous hand robot environment based on pybullet. `https://github.com/szahlner/shadowhand-gym`, 2021-2023.

[122] S. Zahlner, M. Hirschmanner, T. Patten, and M. Vincze. Teleoperation system for teaching dexterous manipulation. *Google Scholar*, 2020.

[123] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.

[124] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke. Towards vision-based deep reinforcement learning for robotic motion control. *arXiv preprint arXiv:1511.03791*, 2015.

[125] R. Zhao, X. Sun, and V. Tresp. Maximum entropy-regularized multi-goal reinforcement learning. In *International Conference on Machine Learning*, pages 7553–7562, 2019.

[126] R. Zhao and V. Tresp. Energy-based hindsight experience prioritization. In *Conference on Robot Learning*, pages 113–122, 2018.

[127] R. Zhao and V. Tresp. Curiosity-driven experience prioritization via density estimation. *arXiv preprint arXiv:1902.08039*, 2019.

[128] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[129] Z. Zhu, K. Lin, B. Dai, and J. Zhou. Self-adaptive imitation learning: Learning tasks with delayed rewards from sub-optimal demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9269–9277, 2022.

[130] G. Zuo, Q. Zhao, K. Chen, J. Li, and D. Gong. Off-policy adversarial imitation learning for robotic tasks with low-quality demonstrations. *Applied Soft Computing*, 97:106795, 2020.