Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Control Automático

Robot Behavior Learning through Biased Exploration in Reinforcement Learning from Expert Demonstrations

Tesis que presenta

Jorge Armando Ramírez Díaz

Para obtener el grado de

Doctor en Ciencias

En la especialidad de

Control Automático

Director de Tesis

Dr. Wen Yu Liu

Ciudad de México Agosto, 2024

En memoria de mi abuelo,
que dejó una huella
imborrable en mi vida.

Agradecimientos

Agradezco a mis padres, que siempre me han apoyado e impulsado para que siga mis sueños y que, con su fortaleza, valor y cariño, me siguen enseñando a no rendirme.

Agradezco a Marcela, que ha compartido conmigo mis triunfos, mis alegrías, mis fracasos y mis tristezas. Su amor ha sido un pilar que me ha mantenido soñando.

Agradezco a mi hermano, mi mejor amigo, por su constante apoyo y consejo.

Agradezco a mi asesor, el Dr. Wen, que a lo largo de estos años me ha guiado con paciencia y esmero.

Agradezco al DCA, al CINVESTAV y al CONAHCYT por el apoyo brindado para poder continuar con mis estudios de posgrado.

Contents

List of Figures

Abstract

This thesis delves into the synergistic integration of imitation and reinforcement learning principles through Reinforcement Learning from Expert Demonstrations (RLED), proposing a cohesive framework that significantly enhances behavior learning in robotics. By establishing a foundational understanding of RLED, this work identifies three indispensable assumptions for its application: agent-environment interaction, environmental reward feedback, and incorporation of knowledge from demonstrations. These tenets lay the foundation for the innovative methodologies introduced here.

The first method, Biased Exploration RLED, represents a paradigm shift in reinforcement learning exploration strategies. By preferentially guiding the agent towards states and actions evidenced in expert demonstrations, this method notably accelerates the learning process. It efficiently circumvents the inefficiency of random exploration, particularly in high-dimensional spaces, thereby optimizing the acquisition of behavior policies that not only mimic but surpass demonstrated performances. This approach is empirically validated on a redundant 7-degree-of-freedom manipulator robot, demonstrating its effectiveness in reducing learning times while achieving superior operational behaviors.

Building on the Biased Exploration foundation, the thesis further introduces the Monte Carlo (MC) RLED method. An evolution of its predecessor, MC RLED enhances the biased exploration concept with Monte Carlo Dropout for advanced uncertainty estimation, enabling safer and more reliable exploration strategies. This second method, validated on a 2-degree-of-freedom planar robot, showcases a robust capability to safely navigate the learning environment, achieving high performance with an emphasis on safety and reliability.

The juxtaposition of these methods with conventional approaches highlights their novel contribution to reducing learning times, improving safety, and improving performance efficiency. The practical application of both methodologies underscores a transformative advancement in the field of robotics, offering a new lens through which efficient and safe behavior learning can be achieved.

This thesis marks a significant step forward in the realm of reinforcement learning and

robotics by rigorously developing and validating two pioneering RLED methods. By addressing the challenges of sample efficiency, safe exploration, and accelerated learning, this research paves the way for future innovations in autonomous systems. Through its foundational review, methodological innovations, and empirical validations, this work contributes a novel perspective to the discourse on behavior learning in robotics, setting the stage for further explorations in complex, dynamic environments.

Chapter 1

Introduction

In the interdisciplinary nexus of robotics and Automatic Control, the quest for superior behavior learning stands paramount. Beyond the traditional paradigms of robot task proficiency is a nuanced challenge: How can a control policy not only emulate specific behaviors such as those exhibited by humans, but also outperform established policies, achieving superior performance and precision?

Although imitation learning (IL) [1] and Reinforcement Learning (RL) [2] have predominantly charted the course for behavior learning, each comes with its set of challenges. IL, anchored in expert demonstrations, and RL, driven by reward shaping, have encountered obstacles ranging from local generalization, sample efficiency, managing high dimensionality, ensuring safety in learning, to the intricacies of formulating rewards.

This thesis unveils an innovative approach, named "Reinforcement Learning from Expert Demonstrations" (RLED) [3]. Prior explorations have hinted at the tenets of RLED, but here lies a thorough dissection of its essence, pinpointing where IL and RL converge. This analytical deep dive reveals critical commonalities and paves the way for a consolidated and methodical RLED framework. Rather than a simplistic fusion, this approach carefully assimilates the strengths of IL and RL, aiming to present a holistic resolution to their individual drawbacks.

As this exploration advances, the thesis delineates two novel methodologies encapsulated within the RLED paradigm. Crafted to influence reinforcement learning's exploration trajectory, these methods are steeped in insights derived from expert demonstrations. Alongside their exposition, there is a robust convergence analysis that underscores their theoretical foundation. Transitioning from the conceptual to the empirical, the subsequent sections showcase experiments that bring to light the prowess and practicality of the presented methodologies.

With this outlined roadmap, the stage is set. The journey ahead will unravel the motivations that underpin this research, subsequently transitioning to the innovative contributions that form its core.

1.1 Motivation

The realm of robotics and automatic control has witnessed relentless advances, driving the continuous evolution of behavior learning mechanisms. Central to this evolution is a profound question: Can robotic systems merely replicate complex behaviors, or can they surpass the prevailing standards, showcasing unparalleled agility and precision?

To underscore the urgency and significance of this research, we delve into key challenges:

- **Sample Efficiency: The Data Dilemma.** The heart of behavior learning lies in the data: the more we have, the better we can train robots. Traditional RL techniques, while powerful, are notoriously data-hungry [2]. They require vast amounts of exploratory data to grasp even basic tasks. This inefficiency in data usage poses a significant challenge, especially in real-world applications where collecting extensive data can be impractical or costly. In contrast, IL achieves proficiency with just a snippet of expert demonstrations. This work seeks to strike a balance by amalgamating RL and IL, aiming to achieve the finesse of expert demonstrations with a fraction of the data traditionally required by RL, thereby enhancing sample efficiency.

- **The Curse of Dimensionality: Navigating Complex Spaces.** High-dimensional spaces, often termed "the curse of dimensionality" [4, 5], pose a monumental challenge. As the dimensions of state, action, and outcome spaces grow, so do the data requirements, intertwining the problem with sample efficiency. Traditional RL methods struggle with this complexity, requiring exponentially more data to achieve reliable performance. This work leverages Deep Reinforcement Learning (DRL), which harnesses the power of neural networks, offering a promising avenue to navigate these intricate spaces more efficiently and effectively.

- **Safety: A Non-Negotiable Imperative.** In a world where robots increasingly interact with humans and delicate environments, safety is paramount. Traditional RL methods, characterized by extensive exploration, often venture into hazardous terrains, posing risks to both robots and their surroundings. IL, guided by expert demonstrations, offers a more restrained approach, inherently prioritizing safety. This work aims

to develop exploration strategies that advance learning effectively and securely, ensuring adherence to safety constraints and minimizing risks [6].

- **Reward Specification: Framing the Goalpost.** Reward functions are foundational to RL, serving as the guiding compass for agents to align their actions with the desired outcomes. As Sutton and Barto [2] aptly put it, the success of RL hinges on how effectively the reward signal reflects the designer's intent. Designing dense reward functions, which provide constant feedback, can be complex and cumbersome. Sparse rewards, which focus on significant milestones, offer simplicity but can lead to sparse feedback, complicating learning. This work adopts sparse reward functions infused with insights from expert demonstrations, aiming to guide agents towards desired behaviors without the distractions of complex reward structures.

- **Generalization: Broadening Horizons.** One of IL's limitations is its comparatively narrow generalization [7], stemming from its reliance on supervised learning. Since the training data might not exhaustively cover all possible scenarios, the learned behavior can sometimes falter in uncharted territories. RL, on the other hand, typically exhibits more robust generalization due to its exploratory nature. The aspiration is to cultivate algorithms that not only learn efficiently from limited samples but also maintain a broad and effective generalization scope, ensuring robust performance across diverse and novel scenarios.

In response to these challenges, this thesis crystallizes its motivation: to forge a pathway that amalgamates the virtues of RL and IL, effectively countering their intrinsic limitations. This endeavor extends beyond academic rigor, emphasizing pragmatic real-world solutions, thus setting the tone for the ensuing discourse of this thesis.

1.2 Objectives

The focus of this thesis is on developing algorithms with the following specific objectives, each addressing a limitation of traditional RL:

1. **Efficient Learning from Experts:** Derive control policies capable of proficiently capturing and imitating behaviors from expert demonstrations. This addresses the challenge of sample inefficiency in traditional RL by leveraging the efficiency of IL.

2. **Prioritizing Generalization with Efficient Sampling:** Design algorithms that emphasize robust generalization similar to RL. While improved data efficiency is pursued, ensuring strong generalization takes precedence, even if it necessitates trade-offs in sample efficiency. This objective tackles the limitation of narrow generalization in IL and aims to achieve the broader generalization capabilities of RL.

3. **High-Dimensional Space Mastery:** Guarantee that the algorithms function effectively in high-dimensional state and action spaces, ensuring consistent performance regardless of the complexity of the environment. This aims to overcome the challenge posed by the curse of dimensionality, which traditional RL methods struggle with.

4. **Safety-Centric Exploration:** Prioritize safety throughout the learning process. Exploration strategies should be designed to minimize risks and avoid potential pitfalls, especially in environments where safety is paramount. This addresses the significant safety concerns associated with the extensive exploration characteristic of traditional RL methods.

5. **Simple Reward Design:** Focus on sparse reward functions as a means to streamline the process of guiding agents towards desired behaviors. Sparse rewards can bypass the challenges and ambiguities often associated with denser reward functions. This objective simplifies the reward specification process, making it more practical and less error-prone.

By addressing these objectives, this thesis aims to integrate the strengths of RL and IL, effectively countering their intrinsic limitations, and to foster advances in the field of behavior learning. This will streamline the process of obtaining competent control policies for robotic systems.

1.3 Contributions

The pursuit of advancing behavior learning mechanisms, especially in the contexts of robotics and Automatic Control, demands novel strategies, rigorous analysis, and empirical validations. This thesis makes several key contributions to the body of knowledge:

1. **Conceptualization and Formation of RLED:**

   - Consolidated diverse findings from numerous behavior learning studies to establish the foundational principles of RLED.

- Defined the core requirements and types of knowledge from demonstrations intrinsic to the RLED approach, providing a clear framework for future research.
- Highlighted both addressed and outstanding challenges in the domain, offering a structured perspective on the progression of the field.

2. **Development and Validation of RLED Algorithms:**

   - Introduced a foundational RLED algorithm that integrates the strengths of both Reinforcement Learning and Imitation Learning, serving as the cornerstone for more advanced methods presented in this thesis.
   - Advanced this foundation with a refined RLED algorithm, which is more adept at addressing the specific challenges of sample efficiency, safety, high-dimensional spaces, reward specification, and generalization.
   - Conducted comprehensive empirical experiments to assess the efficiency and effectiveness of both algorithms, demonstrating their potential to outperform traditional methods in various tasks.

Through these contributions, this thesis provides both theoretical insights and empirical tools, paving a comprehensive path forward for the field of behavior learning. It aims to elevate the state-of-the-art by integrating Reinforcement Learning and Imitation Learning to overcome their intrinsic limitations, ultimately advancing the capabilities of robotic systems in complex environments.

1.4 Thesis Structure

The thesis chapters unfold as follows, illustrating the evolution from foundational theories to practical applications:

- **Chapter 2: Behavior Learning Approaches** delves into the evolution of behavior learning, beginning with an examination of Imitation Learning and its progression from robotics to a cornerstone of AI behavior instruction. Transitioning to Reinforcement Learning, this chapter contrasts its exploratory learning approach with IL's demonstration-based learning.
- **Chapter 3: Reinforcement Learning from Expert Demonstration** introduces the convergence of IL and RL into Reinforcement Learning from Expert Demonstrations, laying the foundation for combining the strengths of both approaches. This

chapter sets the stage by categorizing RLED methods, addressing existing challenges, and paving the way for the innovative methodologies presented in subsequent chapters.

- **Chapter 4: Biased Exploration RLED** presents a novel RL method that leverages biased exploration to utilize expert demonstrations efficiently. By enhancing the Twin Delayed Deep Deterministic Policy Gradient algorithm, this chapter introduces a methodology to guide exploration towards valuable insights from expert demonstrations. It offers a theoretical convergence analysis and demonstrates practical application through redundant robot manipulator control, illustrating improved learning efficiency and generalization capabilities.

- **Chapter 5: Safe Exploration RLED** details a strategic approach to safe exploration within the RLED framework, focusing on safety constraints and uncertainty estimation via Monte Carlo Dropout. By presenting a dual Markov Decision Process approach, this chapter underlines the critical role of safety in learning from demonstrations. It provides a thorough convergence analysis and showcases application to a 2-degree-of-freedom planar robot, balancing safety with learning efficiency. This chapter underscores the trade-offs in setting the uncertainty threshold, contributing valuable insights to the field of safe RL.

- **Chapter 6: Conclusions and Future Work** synthesizes the thesis's contributions towards RLED for enhanced behavior learning and safe exploration. It revisits the innovative methods introduced, emphasizing their potential and versatility. This chapter discusses the broader impacts of these contributions on advancing RL applications and lays out future research directions, including extending RLED to more complex scenarios and exploring adaptive strategies for uncertainty management. It highlights the thesis's role in paving the way for further innovations in autonomous systems and artificial intelligence.

This thesis structure provides a logical flow from basic concepts to practical applications, underscoring the contributions to behavior learning in robotics. It establishes a foundation for future research in reinforcement learning, particularly in developing safer and more efficient exploration methods.

1.5 Publications

The journey of this doctoral research is captured not only within the confines of this thesis. It extends to meticulously crafted manuscripts that have undergone rigorous peer review and have culminated in five publications, of which two are Q1 journal publications. These works, spread over international journals and conferences, mark significant milestones in the research trajectory, encapsulating detailed methodologies, insights, and results. They are a testament to the depth of the research and its pivotal role in advancing the ongoing discourse in behavior learning and robotics. Presented below is a compilation of these scholarly contributions:

International journal

1. **Ramírez, Jorge**, Wen Yu, and Adolfo Perrusquía. "Model-free reinforcement learning from expert demonstrations: a survey." Artificial Intelligence Review 55.4 (2022): 3213-3241.

2. **Ramirez, Jorge**, and Wen Yu. "Reinforcement learning from expert demonstrations with application to redundant robot control." Engineering Applications of Artificial Intelligence 119 (2023): 105753.

3. **Ramirez, Jorge**, and Wen Yu. "Safe Reinforcement Learning for Learning from Human Demonstrations." Under Review in Soft Computing (2024).

International Conference

4. **Ramirez, Jorge**, and Wen Yu. "Redundant robot control with learning from expert demonstrations." 2022 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2022.

5. **Ramirez, Jorge**, and Wen Yu. "Safe Exploration in Reinforcement Learning for Learning from Human Experts." 2023 IEEE International Conference on Artificial Intelligence, Blockchain, and Internet of Things (AIBThings). IEEE, 2023.

Chapter 2

Behavior Learning Approaches

In the rapidly advancing landscape of artificial intelligence, behavior learning from human demonstrations has emerged as a critical foundation. Known AI applications such as AlphaGo [8] and AlphaStar [9] vividly illustrate this. Both these sophisticated agents employ a hybrid approach: They initiate their training with supervised learning policies built on human expert demonstrations, which are subsequently fine-tuned via reinforcement learning. Their success underscores the profound impact of harnessing human demonstrations in shaping agent behavior.

A dominant approach to this kind of behavior learning is IL. Historically, terms such as Learning from Demonstration [1] [10] and Programming by Demonstration [11] have been prevalent in robotics, while IL remains the preferred term in machine learning circles. These terminologies essentially refer to the same concept: the process of learning based on demonstrations.

The roots of IL can be traced back to robotics manufacturing where it was used to guide robot movements [12][13]. However, as machine learning techniques evolved, the popularity of IL increased, especially with its integration with supervised learning for local generalization [7]. In this paradigm, an apprentice strives to emulate the behavior of a teacher. Although idealized as "optimal", in reality, human behaviors often encompass mistakes or suboptimal strategies. The true strength of IL lies in its efficiency, especially when dealing with limited demonstration data [1].

RL offers a contrasting approach. Instead of relying on demonstrations, RL advocates a trial-and-error method, navigating through states and actions based on reward feedback [14]. Despite its data-intensive nature, deep learning advancements have propelled RL to the forefront, especially with the rise of DRL.

In this context, RLED emerges as a groundbreaking intersection between IL and RL. It synthesizes the strengths of both paradigms, aiming for global generalization while ensuring sample efficiency. Using demonstrations, RLED enhances sample efficiency in high-dimensional spaces, assists in sparse reward scenarios, and offers potential for safe exploration. Pioneers have highlighted how the blend of RL and IL can be mutually beneficial [3].

However, RLED should not be viewed in isolation. It shares conceptual space with approaches such as Inverse RL (IRL) [15] and Batch RL [16] [17]. Although IRL also leverages knowledge from expert demonstrations, it operates without a given reward function, instead inferring it from demonstration trajectories. Batch RL, sometimes referred to as Offline RL, presents another nuanced perspective: It operates solely on prior knowledge, without further interactions with the environment, requiring the agent to extract the best policy from this confined dataset. RLED distinguishes itself by uniquely blending demonstrations with reinforcement learning, offering a practical middle ground in behavior learning.

2.1 Imitation Learning

In the realm of artificial intelligence, behavior learning encompasses the development of systems capable of acquiring and replicating complex behaviors through data-based learning. This approach is fundamental in creating intelligent agents that can perform intricate tasks by learning from examples or interactions with their environment. Among the methods of behavior learning, IL stands out due to its reliance on demonstrations provided by experts.

IL involves an apprentice system learning to replicate the behavior of a teacher, typically an expert, to perform specific tasks. Unlike other learning methods, IL relies on direct observation and emulation of demonstrations provided by experts, such as humans or other established control policies. This approach allows the apprentice to acquire complex skills without the need to manually design detailed rules or models of the desired behavior.

The underlying principle of IL is to use the experience and expertise of skilled individuals to train intelligent systems. By observing and mimicking the actions of an expert, the apprentice can learn to perform tasks at a high level of proficiency. This method is particularly advantageous in scenarios where defining explicit rules for behavior is challenging or infeasible.

In theory, the teacher's demonstrations are considered optimal since they come from experts in the specific task. However, it is pragmatic to recognize that human demonstrations can be suboptimal due to potential errors and non-ideal strategies. Acknowledging this suboptimality introduces a realistic perspective to IL, highlighting the complexity of transferring human skills to machines. The variations and errors in human demonstrations not only reflect

the inherently noisy nature of human behavior, but also present a significant challenge for the apprentice's learning process. The ability of a system to generalize from imperfect data and still successfully execute the task is a testament to the robustness of the IL approach.

One of the primary advantages of IL lies in its sample efficiency and practicality. Compared to other learning methods, such as RL, IL generally requires fewer data to capture complex behaviors. This is because IL directly uses demonstrations to guide the apprentice's learning. In practical applications, this means that IL can be implemented with a relatively modest set of demonstration data, making the training process faster and less resource-intensive in terms of computational power and time.

The ability of IL to leverage human demonstrations also facilitates the transfer of skills between humans and machines. Instead of designing specific algorithms for each new task, developers can simply provide new demonstrations, allowing the apprentice to adapt its capabilities flexibly and efficiently. This is particularly beneficial in dynamic environments where tasks and contexts can change rapidly and adaptability is crucial for sustained performance.

However, IL is not without challenges. One of the main issues is local generalization, which is the ability of the apprentice to apply what it has learned from the demonstrations to unseen situations. Since demonstrations typically cover only a fraction of the possible state space, the apprentice must effectively extrapolate from the available data. This problem is exacerbated in tasks with high variability or highly dynamic environments, where conditions can change unpredictably. Additionally, reliance on suboptimal demonstrations can lead to the propagation of errors and the adoption of non-ideal strategies.

To better understand IL, it is essential to explore its primary paradigms: Behavioral Cloning and Apprenticeship Learning. These paradigms offer different approaches to the use of demonstrations for learning, each with its advantages and limitations.

2.1.1 Demonstration Strategies

Choosing the right demonstration strategy in IL depends on various factors, including the nature of the task, the environment, and the capabilities of both the teacher and the apprentice. Each method has its own advantages and limitations that must be carefully considered to ensure effective learning. The primary methods of demonstration in IL are Teleoperation, Kinetic Teaching, Sensors on Teacher, and External Observation.

Teleoperation

Teleoperation involves the teacher remotely controlling the apprentice using devices such as joysticks, haptic interfaces, or other remote control systems. This method facilitates the direct transfer of control strategies from the teacher to the apprentice. It is particularly useful in scenarios where precise control is necessary, such as in surgical robots or remote-operated vehicles.

In surgical robotics, for example, teleoperation allows a surgeon to perform delicate procedures remotely, with the robotic system mimicking the surgeon's hand movements. The haptic feedback provided by the interface ensures that the surgeon can feel the resistance and texture of the tissues, allowing precise manipulation [18]. Similarly, in remote-operated vehicles, teleoperation allows operators to control vehicles in hazardous environments where direct human presence is not feasible.

However, teleoperation also has limitations. It requires high-fidelity communication links to ensure that control signals are transmitted accurately and without latency. Any delay or distortion in the signal can result in suboptimal performance or even accidents. Additionally, the teacher's expertise and ability to control the system accurately are crucial, as any errors made by the teacher will be directly replicated by the apprentice. In addition, teleoperation can induce fatigue in the teacher during prolonged operations, which can affect the quality of the demonstrations.

Kinetic Teaching

Kinetic teaching involves the teacher physically manipulating the apprentice's mechanisms. This method is ideal for learners designed to interact physically with human operators, such as robotic arms or exoskeletons. By physically guiding the apprentice through the desired movements, the teacher can impart complex motor skills that might be difficult to convey by remote control or verbal instructions.

This hands-on approach ensures that the apprentice receives precise feedback on the desired movements, making it suitable for tasks that require fine motor control and dexterity. For example, in robotics, kinetic teaching is often used to program robotic arms for tasks such as welding, assembly, and material handling. The teacher can guide the robotic arm through the desired movements, and the robot records these movements for future replication. This method ensures that the robot learns the exact movements needed to complete the task, reducing the need for complex programming and trial-and-error adjustments [19].

Despite its advantages, kinetic teaching has limitations. It requires the teacher to be

physically present and capable of performing the task, which may not always be feasible. Additionally, the method is less effective for tasks that require high-speed or high-precision movements that are difficult to replicate through physical guidance alone. The physical effort required from the teacher can also be a limiting factor, especially for tasks that involve repetitive or strenuous actions.

Sensors on Teacher

This strategy involves affixing sensors to the teacher's body or equipment to capture the nuances of human motion for later replication. Sensors such as accelerometers, gyroscopes, and motion capture suits record detailed motion data, which can then be used to train the apprentice. This method is beneficial for tasks that require accurate replication of human movements, such as walking, running, or performing complex gestures.

Motion capture technology, for instance, uses a combination of cameras and reflective markers to track the teacher's movements in three-dimensional space. This data can then be used to create a digital representation of the teacher's actions, which the apprentice can use to learn the desired behavior. This method is particularly useful for tasks that require the replication of complex, dynamic movements, such as dancing, sports, or rehabilitation exercises. By capturing the teacher's movements in high detail, the apprentice can learn to replicate these actions with a high degree of fidelity. Motion capture technology is also widely used in the entertainment industry to create realistic animations for movies and video games.

However, the use of sensors on the teacher has its challenges. The equipment can be cumbersome and may restrict the teacher's natural movements. In addition, the data captured by the sensors must be processed and interpreted accurately to ensure that the apprentice can learn the correct movements. This requires sophisticated software and algorithms to analyze the data and generate appropriate training inputs for the apprentice. Furthermore, sensor calibration and data stream synchronization can be technically demanding.

External Observation

External observation utilizes sensors placed in the environment to observe and record the teacher's actions. Commonly used sensors include cameras, laser range finders, and depth sensors. These sensors capture the teacher's movements from an external perspective, which are then translated into formats understandable by the apprentice.

This method is often used in laboratory settings, where precise control over the environment is possible. External observation allows for the capture of holistic movement patterns

and interactions with the environment, making it useful for tasks that involve navigation and interaction with objects. For example, in autonomous driving, external cameras and LIDAR sensors can capture the driving behavior of a human driver, which can then be used to train an autonomous vehicle.

One of the main advantages of external observation is that it does not require the teacher to wear any special equipment, allowing for more natural and unrestricted movements. However, this method also has limitations. It requires a controlled environment with carefully placed sensors to ensure accurate data capture. In addition, the data captured by the sensors must be accurately processed and interpreted, which can be challenging in complex or cluttered environments. External sensors may also have limited resolution and field of view, which can affect the quality of captured data.

Comparison of Strategies

Each of these demonstration strategies offers unique advantages and is suitable for different types of tasks and environments. Teleoperation is ideal for tasks that require real-time control and precision, while kinetic teaching provides a hands-on approach to transfer physical skills. The teacher sensor offers detailed and accurate movement data, which makes it suitable for replicating complex dynamic movements. External observation provides a flexible and nonintrusive way to capture demonstrations, making it ideal for tasks that involve interaction with the environment.

Choosing the right strategy involves considering factors such as the complexity of the task, the environment in which the apprentice will operate, and the capabilities of both the teacher and the apprentice. In some cases, a combination of strategies can be used to leverage the strengths of each method and ensure effective learning.

2.1.2 Teacher and Apprentice

The effectiveness of IL is heavily dependent on the roles and characteristics of both the teacher and the apprentice. The teacher, typically an expert, provides demonstrations that the apprentice learns to replicate. The quality of these demonstrations is crucial, as it directly impacts the learning efficiency and performance of the apprentice.

Roles and Characteristics of the Teacher and Apprentice

The teacher's role is to exhibit complex, natural motions that are representative of the task at hand. The teacher should have a high level of experience and skill in performing the task to

ensure that the demonstrations are accurate and reliable. Characteristics such as precision, consistency, and the ability to perform the task without introducing significant errors are critical. In scenarios in which human teachers are used, their ability to provide intuitive and adaptable guidance can significantly enhance the learning process.

For example, in surgical training, an expert human surgeon can demonstrate the precise movements and techniques required for complex procedures. The expert's ability to perform these tasks accurately and consistently ensures that the apprentice receives high-quality data to learn from. Additionally, the expert's experience allows them to adapt their demonstrations to highlight critical aspects of the task that might not be evident through observation alone.

The apprentice, on the other hand, must possess the capacity to learn from these demonstrations efficiently. The key characteristics of an effective apprentice include high adaptability, the ability to generalize from limited data, and robust learning algorithms that can process and replicate the demonstrated behaviors. The apprentice's design should allow for flexibility in learning, enabling it to improve its performance over time as it receives more demonstrations.

For example, in robotic systems, the apprentice might be a robotic arm programmed to mimic the movements demonstrated by the teacher. The robotic arm must be equipped with advanced sensors and learning algorithms that allow it to accurately capture and replicate movements. Furthermore, the system should be able to refine its actions based on feedback, continually improving its performance.

Choosing the Demonstrator

Selecting the right demonstrator is crucial to the effectiveness of the IL process. Human demonstrators are often preferred for their ability to exhibit complex natural motions. The decision-making process should consider both the control and execution aspects of demonstrations. For example, an expert on a particular task will be able to provide high-quality demonstrations that accurately capture the desired behavior. However, the availability and consistency of human demonstrators can be a limitation.

Human demonstrators have the advantage of being able to perform tasks in a nuanced and flexible manner, which is particularly beneficial for complex and dynamic tasks. Their ability to intuitively adjust their actions based on the context of the task provides rich data that can greatly enhance the learning process of the apprentice. However, human demonstrators can introduce variability in the demonstrations due to fatigue, differences in technique, or

other factors. In addition, the availability of human experts can be limited and their time can be costly.

In some cases, robot demontrators may be used to provide consistent and repeatable demonstrations. These alternatives can be particularly useful for tasks that require high precision or for training apprentices in environments that are hazardous or difficult for humans to operate in. Robotic demonstrators can perform tasks with high repeatability and precision, ensuring that the apprentice receives consistent data for learning. This consistency is crucial for tasks that require fine motor skills or precise movements, where even slight variations can significantly impact the learning outcome.

For example, in manufacturing, robotic arms can be used to demonstrate assembly processes repeatedly and accurately. These robotic demonstrators can operate continuously without the risk of fatigue, providing a steady stream of high-quality demonstration data. Similarly, in hazardous environments, such as nuclear facilities or deep-sea operations, robotic demonstrators can safely perform tasks that would be dangerous to human workers.

Virtual simulations offer another alternative, providing a controlled environment in which demonstrations can be performed without physical constraints. Simulations can model complex scenarios and allow for the adjustment of variables to test different conditions. This flexibility makes virtual simulations an excellent tool for tasks that require extensive testing and iteration.

The use of robotic demonstrators also allows for the generation of large amounts of training data, which can be used to improve the apprentice's learning process. By programming the robotic demonstrators to perform a wide range of variations on a task, a comprehensive dataset can be created, covering different scenarios and conditions. This data can be invaluable for training robust learning algorithms capable of generalizing in different contexts.

When choosing a demonstrator, factors such as the complexity of the task, the required precision, the environment, and the availability of the demonstrator should be considered. The impact of the demonstrator on learning outcomes should also be evaluated to ensure that the apprentice can achieve optimal performance. In some cases, a combination of human and robotic demonstrators may be the best approach, leveraging the strengths of both to provide high-quality demonstrations that enhance the learning process.

The roles and characteristics of the teacher and the apprentice, as well as the choice of the demonstrator, are critical components of the IL process. By carefully selecting and using appropriate demonstrators, human, robotic, or virtual, the effectiveness of IL can be significantly enhanced, leading to better learning outcomes and more efficient training processes.

2.1.3 Data Collection and Analysis

Data collection and analysis are fundamental to the success of IL. Effective data collection methods ensure that the apprentice has access to high-quality demonstrations, which are critical for accurate learning. Data analysis techniques help preprocess and interpret this data, making it usable for training the apprentice.

Data Collection Methods

Various methods are used to collect demonstration data, depending on the nature of the task and the environment. These methods include manual recording, automated sensors, and motion capture systems.

Manual Recording: Manual recording involves capturing demonstrations using simple tools such as video cameras. This method is straightforward and cost-effective but may lack the precision required for certain tasks. For example, recording a teacher performing a task with a video camera provides visual data that can be analyzed later. However, the quality of the data is highly dependent on the recording conditions, such as lighting and camera angles. Manual recording is best suited for tasks where visual cues are sufficient for learning and where high precision is not critical.

Automated Sensors: Automated sensors provide a more precise and reliable method of data collection. Sensors such as accelerometers, gyroscopes, and force sensors can capture detailed motion and force data. These sensors are often attached to the teacher or the objects being manipulated. For instance, in a robotic assembly task, sensors can be attached to the teacher's hands and the objects to capture precise motion data and forces applied during the task. This method ensures high-quality data but may require more sophisticated equipment and setup. The use of automated sensors is particularly beneficial in environments where precision and reliability are paramount, such as in industrial automation and robotics.

Motion Capture Systems: Motion capture systems offer the highest level of detail and accuracy in capturing human movements. These systems use multiple cameras and reflective markers placed on the teacher's body to track movements in three-dimensional space. The data collected from motion capture systems can be extremely detailed, capturing even the smallest nuances of the teacher's actions. This method is widely used in applications such as animation, sports science, and rehabilitation. However, motion capture systems can be expensive and require a controlled environment to function effectively. They are ideal for tasks that require high fidelity and detailed motion data, such as biomechanics studies and advanced robotic control.

Preprocessing and Analysis Techniques

Once the data is collected, it must be preprocessed to remove noise and irrelevant information. Preprocessing techniques such as data cleaning, normalization, and feature extraction are commonly used to prepare data for analysis.

Data Cleaning: Data cleaning involves removing errors, inconsistencies, and outliers from the dataset. For example, if a sensor malfunctions during a demonstration, the erroneous data points must be identified and removed to prevent them from affecting the learning process. This step is crucial to ensure the integrity of the data and to enhance the accuracy of the learning algorithms.

Normalization: Normalization scales the data to a standard range, ensuring that all features contribute equally to the learning process. This is particularly important when dealing with data from multiple sensors that may have different units and ranges. Normalization helps in reducing biases in the data and makes it easier to apply machine learning algorithms.

Feature Extraction: Feature extraction is the process of identifying and isolating the most relevant aspects of the data. This can involve techniques such as principal component analysis to reduce the dimensionality of the data while retaining the most significant features. For instance, in motion capture data, features such as joint angles, velocities, and accelerations can be extracted and used as inputs for the learning algorithm. Feature extraction helps in focusing on the most important data points and reduces the computational complexity of the learning process.

Advanced Analysis Techniques: After preprocessing, advanced analysis techniques, including machine learning algorithms, can be applied to extract meaningful patterns and insights from the data. These techniques help to identify correlations and dependencies between different variables, enabling the apprentice to learn the underlying structure of the task. For example, clustering algorithms can be used to identify different phases of a task, such as preparation, execution, and completion, allowing the apprentice to learn context-specific behaviors.

Data Handling Approaches

Demonstration strategies can also be distinguished by their data handling approaches: batch learning and iterative learning. The choice between these approaches depends on the nature of the task and the availability of the demonstration data.

Batch Learning: In batch learning, the apprentice processes the entire demonstration dataset at once. This approach allows for comprehensive analysis and training based on a

complete set of data. Batch learning is beneficial for tasks where the demonstration data is static and can be fully utilized for training. This method ensures that the apprentice has access to all available information before making decisions, which can improve the accuracy of the learning process. However, batch learning requires significant computational resources and may not be suitable for tasks that need real-time updates or have large datasets.

Iterative Learning: Iterative learning allows the apprentice to incrementally learn as new data becomes available. This approach is useful in dynamic environments where new demonstrations can be continuously integrated into the training process. Iterative learning enables the apprentice to adapt to changes and improve its performance over time. This method is particularly advantageous for tasks that evolve or where new data is generated continuously, such as in autonomous driving or adaptive robotics. Iterative learning allows for ongoing refinement and adjustment of the learning model, making it more flexible and responsive to new information.

The choice between batch learning and iterative learning depends on the specific requirements of the task and the characteristics of the available data. For tasks that require ongoing adaptation and improvement, iterative learning provides the flexibility needed to incorporate new information and refine the apprentice's skills. In contrast, for tasks that benefit from a thorough analysis of a static dataset, batch learning may be the preferred approach.

The selection of demonstration strategies in IL is a multifaceted decision that impacts the overall effectiveness of the learning process. By carefully considering the characteristics and applications of each data collection method and choosing the appropriate data handling approach, practitioners can optimize the IL process to achieve the best possible outcomes. Effective data collection, preprocessing, and analysis are essential components of a successful IL implementation, ensuring that the apprentice can learn accurately and efficiently from the demonstrations provided.

2.1.4 Correspondence Mappings

Acquiring demonstrations requires taking into account the morphological differences between the master and the apprentice. There are two types of correspondence mappings: recording mapping and embodiment mapping.

Recording Mapping

Recognizing the morphological distinctions between the expert and the apprentice requires the incorporation of a recording mapping $g_{rec} : \mathcal{S}_m \times \mathcal{A}_m \rightarrow \mathcal{S}_d \times \mathcal{A}_d$. This mapping effectively

translates expert behaviors into a format that contains the desired information for training, where $\mathcal{S}_m$ represents the set of states experienced by the master, $\mathcal{A}_m$ represents the set of actions made by the master, $\mathcal{S}_d$ represents the set of states to be stored, and $\mathcal{A}_d$ encompasses the subset of actions to be stored.

Formally, the recording mapping can be defined as follows:

$$(s_d, a_d) = g_{rec}(s_m, a_m), \tag{2.1}$$

where $s_d \in \mathcal{S}_d$ is the state of the stored demonstration, $a_d \in \mathcal{A}_d$ is the action of the stored demonstration, $s_m \in \mathcal{S}_a$ is the state of the master, and $a_a \in \mathcal{A}_a$ is the action of the master.This mapping ensures that the demonstrations can be stored and later used for training despite the morphological differences between the master and the desired data.

Embodiment Mapping

Establishing the embodiment mapping involves aligning the states and actions between the stored demonstrations and the apprentice's executions. This requires defining a correspondence function $g_{emb} : \mathcal{S}_d \times \mathcal{A}_d \to \mathcal{S}_a \times \mathcal{A}_a$, where $\mathcal{S}_d$ and $\mathcal{A}_d$ are the state and action spaces of the stored expert demonstrations, and $\mathcal{S}_a$ and $\mathcal{A}_a$ are the state and action spaces of the apprentice.

The embodiment mapping can be expressed as follows:

$$(s_a, a_a) = g_{emb}(s_d, a_d), \tag{2.2}$$

where $s_a \in \mathcal{S}_a$ is the state of the apprentice, and $a_a \in \mathcal{A}_a$ is the action of the apprentice. This function ensures that the states and actions taken by the apprentice correspond to those stored in the demonstrations.

Required Correspondence Mappings

Depending on the demonstration strategy used, different types of correspondence mappings are required:

- **Kinetic Teaching and Teleoperation:** These methods use the apprentice's body to record demonstrations. The master experiences the states and actions through the apprentice's body, so there is no need for a recording mapping. Additionally, the stored states and actions will be the same as those the apprentice needs to execute, eliminating the need for an embodiment mapping. These are the most straightforward strategies.

- **Sensors on Teacher:** This method uses sensors on the master's body, so the states and actions experienced by the master are direct and do not require a recording mapping. However, the states and actions recorded by the sensors are not directly executable by the apprentice, necessitating the use of an embodiment mapping to translate these into the apprentice's action space.

- **External Observation:** This is the least straightforward strategy, as the observations are not directly experienced by the master. Therefore, a recording mapping is needed to translate the external observations into a format suitable for storage. Additionally, since the stored states and actions are not directly executable by the apprentice, an embodiment mapping is also required to align these with the apprentice's capabilities.

2.1.5 Control Policies

Control policies are a crucial component of IL, where the goal is to map states to amisible control accions. This involves defining a policy $\pi : \mathcal{S}_a \rightarrow \mathcal{A}_a$ that translates the apprentice states $\mathcal{S}_a$ from demonstrations into actions $\mathcal{A}_a$ that the apprentice can perform. Formally, we can express this as follows:

$$a_a = \pi(s_a) \tag{2.3}$$

where $s_a \in \mathcal{S}_a$ and $a_a \in \mathcal{A}_a$. The policy π must be learned from the demonstrated data, ensuring that the apprentice can accurately replicate the behavior of the expert in its own state and action space.

Training the apprentice to follow the expert's behavior often involves supervised learning techniques. In this context, the demonstration data serves as the labeled dataset, where the states observed by the expert are the inputs, and the corresponding actions taken by the expert are the labels.

The supervised learning process can be formalized as follows:

1. **Data Collection**: Gather a batch of state-action pairs (s_i, a_i) from the expert's demonstrations dataset.

2. **Model Selection**: Choose a model architecture (e.g., neural networks, decision trees) to represent the policy π.

3. **Loss Function**: Define a loss function $\mathcal{L}(\pi(s_d), a_d)$ that measures the discrepancy between the predicted action $\pi(s_d)$ and the expert action a_d.

4. **Optimization**: Use an optimization algorithm (e.g., gradient descent) to minimize the loss function over the dataset, updating the model parameters to improve policy accuracy.

The optimization problem can be expressed as follows:

$$\min_{\pi} \mathbb{E}_{(s,a)\sim D}[\mathcal{L}(\pi(s), a)] \tag{2.4}$$

where $\mathcal{L}(\cdot)$ is the loss function and D is the distribution of the demonstration data.

Choosing the right model for π is critical. Common choices include:

- Linear Models: Simple and interpretable, but may struggle with complex behaviors.
- Decision Trees: Handle non-linearities well, but can be prone to overfitting.
- Neural Networks: Highly flexible and powerful, capable of learning complex mappings, but require large amounts of data and computational resources.

Once trained, the policy π must be evaluated to ensure that it accurately replicates the expert's behavior. This involves:

- Cross-Validation: Splitting the data into training and validation sets to test the policy's generalization ability.
- Performance metrics: Using metrics such as accuracy, precision, recall, and F1 score to assess performance.

In many real-world applications, the environment may be non-stationary, meaning its dynamics can change over time. To address this, the apprentice must be capable of dynamically adapting its policy π. Techniques for handling non-stationarity include:

- Incremental Learning: Continuously updating the policy as new demonstration data becomes available.
- Transfer Learning: Using knowledge gained from previous tasks to adapt to new but related tasks.
- Robustness Training: Training the policy to handle a range of scenarios and uncertainties, improving its ability to generalize to new conditions.

In conclusion, the design and training of control policies are fundamental to the success of IL. By effectively modeling the environment, using supervised learning to train policies, selecting appropriate models, and addressing non-stationarity, the apprentice can accurately replicate the expert's behavior and adapt to new challenges.

2.1.6 Imitation Learning Paradigms

Behavioral Cloning

Behavioral Cloning (BC) is a straightforward and widely used paradigm within Imitation Learning. It involves training a model to replicate the behavior of an expert by mimicking their actions in given states. This process typically uses supervised learning techniques, where the model learns from pairs of states and actions recorded during the expert's demonstrations.

In Behavioral Cloning, the apprentice system is provided with a set of state-action pairs (s, a), where s represents the state of the environment and a represents the action taken by the expert in that state. The goal of BC is to learn a policy π that maps states to actions, effectively mimicking the expert's behavior.

$$a = \pi(s) \tag{2.5}$$

The BC training process involves several steps, including data collection, model selection, loss function definition, optimization, and evaluation. These steps are crucial in developing an effective policy. For a detailed explanation of these steps, refer to the *Control Policies* section.

One of the primary advantages of BC is its sample efficiency. Since the learning process is guided by expert demonstrations, it requires fewer data to capture complex behaviors compared to methods that rely solely on trial and error, such as Reinforcement Learning. This makes BC particularly appealing for applications where collecting extensive data is impractical or costly.

Despite its simplicity and efficiency, Behavioral Cloning has significant limitations. One major issue is overfitting to the training data. Because BC relies on supervised learning, the model can become overly specialized to the specific examples seen during training. This leads to poor generalization when the apprentice encounters states that were not present in the training data. This limitation is often referred to as the covariate shift problem, where the distribution of states encountered during training differs from the distribution encountered during deployment.

Another challenge with BC is its inability to handle unseen states effectively. Since BC models are trained to imitate the expert's actions in specific states, they struggle when faced with new or unexpected situations. Without additional mechanisms to explore and adapt, the apprentice may not perform adequately in these scenarios.

Despite these challenges, BC remains a powerful tool for behavior learning, particularly when used in combination with other techniques that address its limitations. For example, integrating BC with Reinforcement Learning can help overcome issues of generalization and adaptability, as discussed in the next subsection on Apprenticeship Learning.

Apprenticeship Learning

Apprenticeship Learning extends the concept of Behavioral Cloning by incorporating a reward signal to guide the learning process. This paradigm combines elements of both Imitation Learning and Reinforcement Learning, allowing the apprentice to learn from demonstrations while also optimizing a reward function. The introduction of the reward signal helps the apprentice generalize better to unseen states and improve its performance over time.

In Apprenticeship Learning, the apprentice begins by imitating the expert's demonstrations, similar to Behavioral Cloning. However, instead of solely relying on the state-action pairs provided during training, the apprentice also uses a reward signal to refine its policy. This approach helps the system not only replicate the expert's behavior but also optimize it according to a specified goal.

The learning process in Apprenticeship Learning can be described as follows:

- Imitation Phase: The apprentice uses supervised learning to mimic the expert's behavior from the provided demonstrations, learning an initial policy π_0.
- Reinforcement Phase: The apprentice then uses this initial policy as a starting point and employs reinforcement learning techniques to further optimize the policy by maximizing a reward function $R(s, a)$.

Formally, the optimization problem in the reinforcement phase can be expressed as:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R(s_t, \pi(s_t))\right] \tag{2.6}$$

where γ is a discount factor, s_t represents the state at time t, and T is the time horizon.

The integration of the reward signal offers several advantages over pure Behavioral Cloning. First, it provides a mechanism for the apprentice to improve beyond simply copying the ex-

pert, enabling it to handle variations and unforeseen states more effectively. This ability to adapt and optimize based on feedback is crucial for achieving robust performance in dynamic and unpredictable environments.

Moreover, Apprenticeship Learning mitigates the covariate shift problem inherent in Behavioral Cloning. By iteratively refining the policy through reinforcement learning, the apprentice can adjust to discrepancies between the training and deployment distributions of states, thus enhancing generalization.

Despite its strengths, Apprenticeship Learning also poses challenges. The primary difficulty lies in defining an appropriate reward function that accurately reflects the desired behavior. The design of such a reward function can be complex and may require domain-specific knowledge. Additionally, the reinforcement learning phase can be computationally intensive and time-consuming, especially for high-dimensional state spaces or tasks requiring long-term planning.

In summary, Apprenticeship Learning represents a powerful approach to behavior learning that leverages the strengths of both Imitation Learning and Reinforcement Learning. By incorporating a reward signal, it addresses some of the key limitations of Behavioral Cloning, offering a more flexible and robust framework for training intelligent agents.

2.2 Reinforcement Learning

RL is deeply rooted in dynamic programming principles that serve as its foundation [20, 21]. RL operates on the premise that an agent learns to make decisions by interacting with its environment, with the aim of maximizing cumulative rewards over time. The connection to dynamic programming lies in solving complex problems by breaking them down into simpler, recursive decision steps, which is central to both fields.

At the heart of RL is the concept of Markov Decision Process (MDP), a mathematical framework used to describe the environment in which the RL agent operates [14]. MDPs provide a structured way to model decision-making situations where outcomes are partly random and partly under the control of a decision-maker. This concept aligns with dynamic programming's approach to solving problems where the future state is dependent on the current state and the action taken, embodying the Markov property.

In RL, an agent learns an optimal policy that dictates the best action to take in each state to maximize future rewards. This process is inherently iterative and adaptive, reflecting dynamic programming's iterative approach to optimizing a given policy. RL extends dynamic programming by dealing with the exploration-exploitation trade-off, where the agent must

balance between exploring new actions to discover potentially better future rewards and exploiting known actions to maximize immediate rewards.

The learning process in RL involves estimating the value functions which are a measure of the expected cumulative reward of following a particular policy from a given state that are central to both RL and dynamic programming. In dynamic programming, value functions are computed through known transition dynamics and rewards, while in RL, these functions are learned from the agent's experiences, often under conditions of uncertainty and incomplete knowledge of the environment.

2.2.1 Markov Decision Process

MDPs offer a robust framework for modeling environments in decision-making processes where outcomes are partially random and partially under the control of a decision maker. This framework is fundamental in many fields, including robotics and RL.

The Markov Property

At the heart of MDPs lies the Markov property, which posits that the future state of a process depends only on the current state, not on the path taken to reach it. Formally, a process with a set of states $\mathcal{S}$ satisfies the Markov property if for any sequence of states $s_1, s_2, \ldots, s_t$, the following holds:

$$\Pr(S_{t+1} = s | S_t = s_t, S_{t-1} = s_{t-1}, \ldots, S_1 = s_1) = \Pr(S_{t+1} = s | S_t = s_t).$$

This property ensures that the state transitions are memoryless, simplifying the modeling of such processes.

Markov Process (Markov Chain)

A Markov Process, or Markov Chain, is a stochastic model that describes a sequence of possible events in which the probability of each event depends only on the state attained in the previous event. A Markov Chain is formally defined as a tuple $\langle \mathcal{S}, \mathcal{T} \rangle$ where:

- $\mathcal{S}$ is a finite set of states.
- $\mathcal{T}$ is a transition probability matrix which describes the probability of moving from state s to state s'.

Markov Reward Process

A Markov Reward Process extends a Markov Chain by associating rewards with transitions between states. It is formally defined as a tuple $\langle \mathcal{S}, \mathcal{T}, \mathcal{R} \rangle$ where:

- $\mathcal{S}$ and $\mathcal{T}$ are as defined in the Markov Process.
- $\mathcal{R}$ is a reward function associated with each state representing the expected reward received after transitioning from state s.

Markov Decision Process

An MDP incorporates decisions into the Markov Reward Process, allowing an agent to influence the state transitions through its actions. An MDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \rangle$ where:

- $\mathcal{S}$ is a set of states.
- $\mathcal{A}$ is a set of admissible control actions.
- $\mathcal{T}$ is a transition probability matrix dependent on both the current state and the control action taken.
- $\mathcal{R}$ is a reward function that depends on the state and the control action.

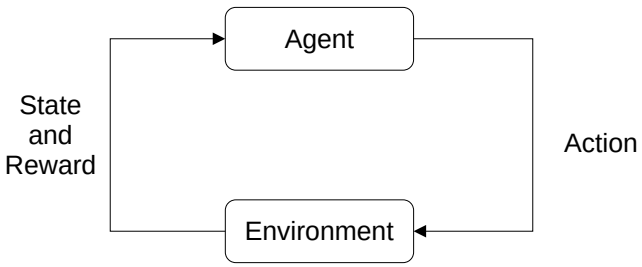

Figure 2.1: Common MDP diagram showing the interaction between agent and environment.

Each element of this structure, states, actions, rewards, and transitions, is crucial to understanding and implementing decision-making algorithms that optimize long-term rewards

in stochastic environments. This formal structure not only clarifies the components and interactions within an MDP but also underscores the utility of this framework in modeling complex decision-making scenarios where outcomes are uncertain.

2.2.2 Goal and Return

In RL, the concepts of goals and returns are fundamental and guide the behavior of an agent within an environment.

Reward Function

The reward function quantifies the immediate gain an instant reward r that an agent receives from being in a state, after taking a certain action and transitioning to a new state. Formally, the reward function in a Markov Decision Process is expressed as $\mathcal{R}$, where $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ represents the expected reward:

$$\mathcal{R}(s, a) = \mathbb{E}[r_{t+1}|S_t = s, A_t = a]. \tag{2.7}$$

This function is essential, as it provides the evaluative feedback necessary for the agent to learn and make decisions.

Return and Goal

The return, denoted as G_t, represents the total accumulated reward an agent collects over time, starting from the time step t. It captures the long-term consequences of the control actions taken by the agent. Let $h_t \in \mathcal{H}$ be the trajectory from instant 0 to t in the combined state, action, reward spaces:

$$h_t = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, a_{t-1}, r_{t-1}, s_t). \tag{2.8}$$

For a policy π, the goal $G^\pi(s)$ is defined as the expected return (or expected cumulative reward signal) when the system starts from $s_0 = s$:

$$G = r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots = \lim_{T\to\infty} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t | s_0 = s, \pi\right], \tag{2.9}$$

where γ $(0 \leq \gamma < 1)$ is the discount factor. The discount factor prioritizes immediate rewards over distant ones, influencing the agent's decision-making strategy higher values of

γ encourage the agent to consider long-term rewards more significantly.

Episodes and Time Steps

In RL, an episode encapsulates a sequence of actions and observations, beginning from an initial state and ending at a terminal state. Each episode begins at the time step $t = 0$ and terminates either when a natural terminal state of the environment is reached or when predefined conditions such as a maximum number of time steps or a specific failure state are met. Within an episode, each time step t involves the agent perceiving its current state s_t, performing an action a_t, receiving an immediate reward r_t, and transitioning to a subsequent state s_{t+1}.

This episodic approach segments the RL process into discrete, manageable units, facilitating the computation of returns and the evaluation of learning progress. Although natural for environments that are reset, such as simulations or games, episodic learning is also applied in continuous environments by imposing reset conditions. These forced resets help manage complex continuous spaces by providing structured learning intervals and resetting the environment to a standard initial state at the beginning of each new episode.

Moreover, the episodic structure allows for the adjustment of policies based on the accumulated experience from multiple episodes, rather than a continuous interaction flow. This distinction is crucial for algorithms that depend on clear delimitations of the trial and the outcome. Thus, understanding episodes and time steps is fundamental for developing strategies that optimize long-term rewards across various RL environments.

2.2.3 Control Policies

Control policies in RL define the method by which an agent selects actions based on its current state within the environment. A policy, denoted as π, is essential in guiding the agent's decisions to achieve its objectives by mapping states to actions. The effectiveness and optimality of a policy are crucial as they are measured by the goal as stated in Eq. (2.9).

Value Functions and Bellman Equations

Value functions are essential to evaluate and improve decision-making in reinforcement learning, estimating the expected cumulative discounted reward or goal G from specific states or actions under a given policy.

State-Value Function $V^\pi(s)$: Represents the expected goal G when starting from the state s and following the control policy π. It is defined as:

$$V^\pi(s) = \mathbb{E}[G_t|S_t = s, \pi], \tag{2.10}$$

where G_t is the cumulative discounted reward from state s at time t under control policy π.

Action-Value Function $Q^\pi(s, a)$: Estimates the expected goal G when starting from the state s, taking an initial action a, and following the control policy π. It is expressed as:

$$Q^\pi(s, a) = \mathbb{E}[G_t|S_t = s, A_t = a, \pi]. \tag{2.11}$$

The Bellman equations provide a recursive decomposition to calculate these values: - **Bellman Expectation Equation for** $V^\pi(s)$:

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V^\pi(s')], \tag{2.12}$$

- **Bellman Optimality Equation for** $V^*(s)$:

$$V^*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V^*(s')], \tag{2.13}$$

- **Bellman Expectation Equation for** $Q^\pi(s, a)$:

$$Q^\pi(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \sum_{a'} \pi(a'|s')Q^\pi(s', a')], \tag{2.14}$$

- **Bellman Optimality Equation for** $Q^*(s, a)$:

$$Q^*(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} Q^*(s', a')], \tag{2.15}$$

where γ is the discount factor, emphasizing the trade-off between immediate and future rewards.

Types of Control Policies

Control policies in RL are broadly classified into two types: deterministic and stochastic, each suitable for different kinds of environments and objectives.

Deterministic Policies directly map states to a specific action, formulated as:

$$\pi(s) = a. \tag{2.16}$$

This direct mapping is beneficial for environments where there is a clear best action for each state, making the policy simpler and often faster to compute. However, deterministic policies might not handle the variability and uncertainty present in more complex or dynamic environments effectively.

Stochastic Policies, in contrast, provide a probabilistic approach to action selection, formulated as:

$$\pi(a|s) = P(A = a|S = s), \tag{2.17}$$

which describes the probability of selecting action a when in state s. This type allows for exploration of the action space, which can be crucial for discovering more effective strategies in environments with high uncertainty or variability. It supports exploration inherently and can prevent the policy from getting stuck in local optima during the learning process.

The distinction between deterministic and stochastic policies raises fundamental questions about the nature of policy optimality in various settings. How can we determine which policy is optimal in a given environment? Under what conditions can we assert that one policy is better than another? These questions are not only theoretical but also have practical implications in the design of RL agents. To address these, we turn to formal theoretical results that provide foundational insight into policy performance and optimality.

Theorem 1 (Existence of an Optimal Policy)**.** *Let $\pi^* \in \Pi$ be a policy such that for all $s \in \mathcal{S}$,*

$$G^{\pi^*}(s) = \max_{\pi \in \Pi} G^{\pi}(s). \tag{2.18}$$

Under some mild assumptions about the dynamics of the environment and the reward function, such a policy π^* indeed exists.

Theorem 2 (Ordering of Policies by Value Functions)**.** *Let $\pi_1, \pi_2 \in \Pi$ be two policies in an MDP. We say that policy π_1 is less than or equal to policy π_2 (denoted $\pi_1 \leq \pi_2$), if and only if the value function associated with π_2 is greater than or equal to the value function associated with π_1 for all states in the state space $\mathcal{S}$, i.e.,*

$$\pi_1 \leq \pi_2 \iff V^{\pi_1}(s) \leq V^{\pi_2}(s) \quad \forall s \in \mathcal{S}. \tag{2.19}$$

This ordering reflects that a policy π_2 is at least as good as a policy π_1 in terms of the expected

returns from all states.

Policy Formulation

Formulating a policy involves selecting a representation that can efficiently and effectively capture the complexities of the environment and the learning task. This choice significantly impacts the learning performance and generalizability of the policy.

Tabular Methods are straightforward and involve mapping each state to an action in a table or matrix. This method is practical in environments with a limited number of discrete states and actions but becomes infeasible as the state or action space grows due to exponential growth in the size of the table.

Function Approximators are used in more complex or continuous environments. These methods involve parameterizing the policy with a set of parameters (e.g., the weights of a neural network). Neural networks, for example, input the state of the environment and output either the probabilities of each action (in the case of stochastic policies) or the best action directly (in deterministic policies). This approach allows for generalization across states that may not be explicitly presented during training, which is crucial for handling large state spaces.

Deep learning methods, in particular, have revolutionized policy formulation by enabling the approximation of policies in high-dimensional spaces that would otherwise be intractable with traditional methods. Techniques such as convolutional neural networks (CNNs) [22, 23] to process visual input from the environment, recurrent neural networks (RNNs) [24] to handle temporal dynamics, and feedforward neural networks (FNNs) [25] are commonly employed. Feedforward neural networks are especially useful for their ability to approximate complex functions from a large set of inputs to outputs, making them ideal for environments where decisions are based on a static snapshot of the state without the need for temporal context. These networks typically consist of multiple layers of neurons, each layer fully connected to the next, which allows them to learn a wide variety of patterns in data and play a crucial role in tasks that require a straightforward, yet powerful, interpretation of the environmental state to make decisions.

Exploration and Exploitation Trade-off

Balancing the need for exploration and the tendency to exploit known rewarding actions is a fundamental challenge in reinforcement learning. Exploration enables the agent to attempt new actions to discover potentially more rewarding strategies, while exploitation leverages

existing knowledge to maximize immediate rewards.

Epsilon-Greedy Strategy: One of the most straightforward and commonly used methods for balancing exploration and exploitation. With this approach, the agent selects the best-known action with a probability of $1 - \epsilon$ and a random action with a probability of ϵ, where ϵ is a parameter that can be adjusted over time. The strategy is formalized as:

$$\pi(a|s) = \begin{cases} \max_a Q(s,a) & \text{with probability } 1-\epsilon, \\ \text{random action} & \text{with probability } \epsilon. \end{cases} \tag{2.20}$$

Softmax Selection: A generalization of epsilon-greedy that selects actions according to a probability distribution that favors better actions but still allows all actions to be chosen. Actions are chosen based on their action values relative to each other, weighted by a temperature parameter τ that controls the degree of exploration:

$$\pi(a|s) = \frac{e^{Q(s,a)/\tau}}{\sum_b e^{Q(s,b)/\tau}}, \tag{2.21}$$

where $\tau > 0$ determines how much the probabilities are spread out over different actions.

Gaussian Noise for Exploration in Deterministic Policies: Particularly relevant when employing deep-deterministic policy gradient methods, where policies are typically deterministic. Gaussian noise is added to the chosen action to ensure sufficient exploration. This method involves adding a normally distributed random variable $\mathcal{N}(0, \sigma^2)$ to the selected action:

$$a = \pi(s) + \mathcal{N}(0, \sigma^2), \tag{2.22}$$

where σ^2 is the variance of the Gaussian distribution, which can be adapted over time to balance exploration and exploitation more effectively. This strategy allows the agent to explore the action space around the optimal actions suggested by the deterministic policy, thereby preventing premature convergence to suboptimal policies.

These methods ensure that the agent does not prematurely converge to suboptimal policies and continues to learn about the environment throughout its operation. Each strategy has its specific use cases and effectiveness, depending on the nature of the task and the environment.

Types of Model-Free Algorithms

Model-free reinforcement learning algorithms are categorized based on how they approach the problem of learning optimal policies without a model of the environment. These algorithms

can be broadly classified into three types: value-based methods, policy-based methods, and actor-critic methods.

Value-Based Methods Value-based methods focus on learning value functions, such as the state-value function $V(s)$ and the action-value function $Q(s, a)$. The central idea is to use these value functions to derive the optimal policy. Some of the most common value-based methods include:

- **Q-Learning:** Q-learning [26] is an off-policy algorithm that seeks to learn the optimal action-value function $Q^*(s, a)$, which represents the maximum expected future rewards for an action taken in a given state. It updates the Q values iteratively based on the rewards received and the maximum future Q values.
- **SARSA (State-Action-Reward-State-Action):** SARSA [27] is an on-policy algorithm that updates the action-value function based on the action taken by the current policy. It learns the Q values based on the action actually taken in the next state, ensuring consistency with the current policy.
- **Deep Q-Network (DQN):** DQN [28] extends Q-learning by using deep neural networks to approximate the action-value function. This allows DQN to handle high-dimensional state spaces, such as those in video games. DQN incorporates techniques like experience replay and target networks to stabilize training.
- **Double DQN (DDQN):** DDQN [29] addresses the overestimation bias in Q-learning by decoupling the selection and evaluation of the action. It uses the main network to select actions and the target network to evaluate them, leading to more accurate value estimates.

Policy-Based Methods Policy-based methods directly parameterize the policy and optimize the policy parameters θ to maximize the expected return. These methods are particularly useful in environments with high-dimensional or continuous action spaces. Common policy-based methods include:

- **REINFORCE:** REINFORCE [30] is a Monte Carlo policy gradient method that updates the policy parameters using the gradient of the expected return. It uses the returns from complete episodes to adjust the policy, encouraging actions that lead to higher returns.

- **Proximal Policy Optimization (PPO):** PPO [31] improves the stability of training by limiting the magnitude of policy updates. It uses a clipped objective function to prevent large updates that can destabilize the training process, making it more robust and reliable.

- **Trust Region Policy Optimization (TRPO):** TRPO [32] is another policy gradient method that ensures large policy updates do not lead to performance collapse. It optimizes a surrogate objective subject to a trust region constraint, which limits the deviation between the new and old policies.

Actor-Critic Methods Actor-critic methods combine the advantages of both value-based and policy-based approaches by maintaining two separate structures: the actor, which updates the policy parameters, and the critic, which evaluates the policy by estimating the value function. Common actor-critic methods include:

- **Advantage Actor-Critic (A2C/A3C):** A2C and A3C [33] are synchronous and asynchronous variants of the advantage actor-critic algorithm, respectively. These methods use a centralized or distributed approach to update the policy (actor) and value function (critic) simultaneously, leveraging multiple workers to stabilize training.

- **Deep Deterministic Policy Gradient (DDPG):** DDPG [34] is an off-policy actor-critic algorithm that extends DQN to continuous action spaces. The actor network outputs deterministic actions, while the critic network evaluates these actions. DDPG uses experience replay and target networks to stabilize training.

- **Twin Delayed DDPG (TD3):** TD3 [35] builds on DDPG by addressing overestimation bias and introducing techniques like clipped double Q-learning and delayed policy updates. These improvements lead to more stable and reliable learning in continuous action spaces.

- **Soft Actor-Critic (SAC):** SAC [36] is an off-policy actor-critic algorithm that optimizes a stochastic policy to maximize a trade-off between expected return and entropy. The entropy term encourages exploration by ensuring the policy remains stochastic, improving performance in complex environments.

2.2.4 Actor Critic Learning

The synthesis of value-based and policy-based methods in RL has culminated in the development of actor-critic methods, a class of algorithms that elegantly balances the strengths of its constituent approaches. At its core, the actor-critic framework is a two-pronged strategy: the actor proposes actions to be taken given the current state, driven by a policy that seeks to maximize cumulative rewards; concurrently, the critic evaluates the potential of these actions using a value function, which estimates the expected returns. This cooperative dynamic allows the system not only to gauge the immediate efficacy of its actions but also to forecast their long-term consequences, making actor-critic methods particularly powerful in domains where decisions have far-reaching impacts.

Central to the success of actor-critic methods in environments with continuous action spaces is their ability to operate without the need to discretize the action space, a limitation that hampers many traditional algorithms. In such spaces, the actions are not a set of discrete choices but rather a continuum, allowing for an infinite number of possible actions. Actor-critic algorithms navigate these spaces with finesse, enabling precise control that is often required in complex tasks such as robotic manipulation or autonomous driving. Using the continuous nature of the action space, the actor can adjust its policy in infinitely fine gradations, guided by the critic's assessment to improve policy performance iteratively and efficiently.

The marriage of policy and value function within the actor-critic architecture addresses the exploration-exploitation dilemma with a unique balance. The actor, seeking to optimize the policy, must explore the environment to discover rewarding actions, while the critic's value function tempers this exploration with exploitation, guiding the actor towards actions that are known to yield high returns. This interplay results in a learning process that is both exploratory, pushing the boundaries of the unknown, and grounded, capitalizing on accumulated knowledge. The dual nature of the actor-critic method also lends itself to a more stable learning process compared to methods that solely rely on value functions or policies, as it mitigates the risk of large policy updates that could derail learning. This balance is especially critical in environments where the agent must learn to make decisions over continuous actions, and any erratic behavior can have significant consequences.

Foundations of Actor-Critic Architecture

Actor-critic algorithms represent a sophisticated blend of policy-based and value-based reinforcement learning, capturing the strengths of both in a singular framework. In this archi-

tecture, the actor, delineated by the parameter θ, specifies a policy $\pi_\theta(a|s)$ that governs the choice of actions in any given state. In parallel, the critic, encapsulated by the parameter w, assesses these actions by estimating a value function $V_w(s)$ that predicts the expected returns from a state s under the current policy. This synergistic operation enables continuous refinement of the policy in tandem with the value predictions, a method particularly adept for complex, high-dimensional environments where actions form a continuous space.

The advantage function, $A^\pi(s,a)$, quantifies the excess in expected returns of choosing a particular action a in state s over what is predicted by the current policy. It is defined as the difference between the action-value function, which represents the expected return of taking action a in state s, and the state-value function, which gives the expected return of just being in state s under policy π:

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s). \tag{2.23}$$

Building upon this, the policy gradient theorem articulates the mechanism of policy optimization. It delineates how policy parameters θ can be fine-tuned to increase the expected return, as captured by the gradient of the performance objective $J(\theta)$ with respect to θ:

Theorem 3 (Policy Gradient)**.** *The gradient of the expected return with respect to the policy parameters is directly proportional to the expectation of the product of the logarithmic gradient of the policy and the advantage function:*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) A^\pi(s,a)\right]. \tag{2.24}$$

This establishes a concrete direction for updating the policy parameters to prioritize actions that are projected to yield better-than-average returns.

Critical to the learning process is the temporal-difference (TD) error, δ_t, which serves as a pivotal estimator for the advantage function:

$$\delta_t = R_{t+1} + \gamma V_w(s_{t+1}) - V_w(s_t), \tag{2.25}$$

where R_{t+1} is the reward after taking action a_t in state s_t. The parameterized state-value function $V_w(s)$, appearing in the TD error, should reflect the critic's current estimate, thus the inclusion of the subscript w. This error informs the updates for both the actor's and the critic's parameters.

The policy parameters θ are adjusted to maximize expected returns, guided by the ad-

vantage function as estimated by the TD error:

$$\theta_{k+1} = \theta_k + \alpha_{actor} \nabla_\theta \log \pi_\theta(a_t|s_t)\delta_t. \tag{2.26}$$

Simultaneously, the critic refines the parameters of the value function w to minimize the TD error, improving the precision of the value estimates:

$$w_{k+1} = wk + \alpha_{critic}\delta_t \nabla_w V_w(s_t). \tag{2.27}$$

The iterative application of these update rules propels the actor-critic method toward increasingly effective policies. The assumption underlying this process is that both the actor and the critic are learning at rates that promote the convergence of policy to optimality and maintain the stability of the learning trajectory.

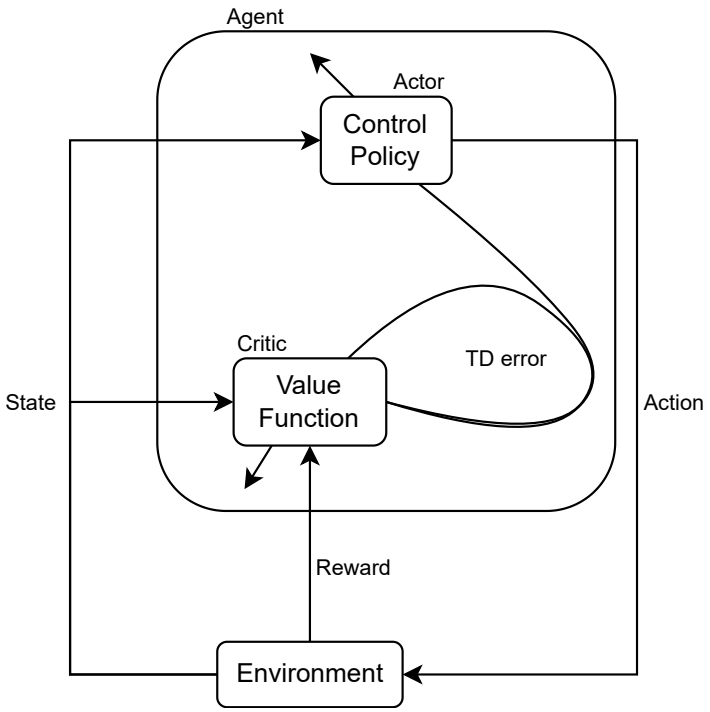

Figure 2.2: Common MDP diagram showing the actor and the critic interacting with the environment.

Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient (DDPG) is an actor-critic algorithm that uses deep function approximators to handle continuous action spaces, capable of learning policies in

complex, high-dimensional environments. The DDPG algorithm can be viewed as an extension of the Deep Q-Network (DQN) algorithm for continuous action spaces. Although DQN focuses on discrete action problems using a deep neural network to approximate the action-value function $Q(s, a)$, DDPG adopts a similar approach but incorporates a deterministic policy and an actor-critic structure. This makes DDPG particularly suitable for problems where actions are continuous and the state-action space is large.

To evaluate the accuracy of the critic, DDPG minimizes the Mean Squared Bellman Error (MSBE), defined as:

$$J_c = \frac{1}{|B|} \sum_{(s,a,r,s') \in B} \left[\left(r + \gamma Q_{\phi'}(s', \pi_{\theta'}(s')) - Q_{\phi}(s, a) \right)^2 \right], \tag{2.28}$$

where (s, a, r, s') are a minibatch B of state transitions sampled from a replay buffer, and $Q_{\phi'}$ and $\pi_{\theta'}$ are the target networks for the critic and actor, respectively.

It is worth noting that while the MSBE is used for evaluating the Q-function, it is closely related to the TD error used in traditional actor-critic methods for evaluating the V-function. In contrast, the MSBE focuses on the Q-function and provides a measure of how well the critic approximates the action-value function.

Minimizing the MSBE helps ensure that the critic provides accurate value estimates, which are crucial for guiding the updates of the actor.

Replay Buffer The replay buffer is a crucial component in DDPG, used to store transitions (s, a, r, s') observed during training. The replay buffer addresses the issue of correlation between consecutive transitions by allowing the algorithm to sample random mini-batches of transitions for training. This random sampling breaks the correlation between consecutive samples, which is essential for stabilizing training and ensuring that updates are more robust.

By using a replay buffer, DDPG can improve the efficiency of learning in several ways:

1. Breaking Correlation: The random sampling of transitions from the buffer mitigates the risk of correlated data, which can lead to suboptimal learning and instability in neural network updates.

2. Data Efficiency: The buffer allows the algorithm to reuse past experiences multiple times, making the learning process more data-efficient. For example, a transition (s, a, r, s') that was experienced at the beginning of the training can be reused in multiple updates, providing more learning opportunities.

3. Smoothing Updates: The buffer helps smooth the learning updates by providing a more varied set of training samples, which contributes to more stable and consistent gradient estimates.

The replay buffer enables the algorithm to learn from a diverse set of experiences, enhancing the overall stability and performance of the training process.

Target Networks To stabilize training and avoid divergence, DDPG uses target networks for the critic and actor. The target networks are copies of the original networks that are slowly updated to track the learned networks. This mechanism addresses the issue of instability that arises when the target value, used in the MSBE, is dependent on the same parameters being updated. Frequent and large updates to the parameters can cause the target values to change rapidly, leading to unstable learning. By having a separate target network that updates more slowly, we ensure that the target values change more smoothly, contributing to the stability of the learning process.

Specifically, the target networks for the critic and actor are updated using a soft update mechanism:

$$\phi'_{k+1} = \tau\phi_k + (1-\tau)\phi'_k, \tag{2.29}$$

$$\theta'_{k+1} = \tau\theta_k + (1-\tau)\theta'_k, \tag{2.30}$$

where $\tau \ll 1$ is a parameter controlling the update speed. This means that the target networks ϕ' and θ' are updated to slowly track the parameters of the critic and actor networks ϕ and θ, respectively. The use of target networks helps smooth out the updates, preventing oscillations and divergence, and ultimately leading to a more stable training process.

Critic Update The value function (critic) is updated using the following update rule:

$$\phi_{k+1} = \phi_k - \alpha_c \nabla_\phi J_c, \tag{2.31}$$

where α_c is the learning rate for the critic. This gradient minimizes the MSBE and adjusts the critic's parameters ϕ.

Actor Update The policy gradient (actor) is calculated to maximize the expected return:

$$J_a = \frac{1}{|B|} \sum_{s \in B} Q_\phi(s, \pi_\theta(s)), \tag{2.32}$$

The actor's policy π_θ is then updated using the deterministic policy gradient:

$$\theta_{k+1} = \theta_k + \alpha_a \nabla_\theta J_a, \tag{2.33}$$

where this gradient is used to adjust the actor's parameters θ.

Exploration in DDPG For exploration in action selection, DDPG injects noise into the deterministic policy:

$$a_t = \pi_\theta(s_t) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2), \tag{2.34}$$

where $\mathcal{N}(0, \sigma^2)$ represents the Gaussian noise process. This exploration is crucial to balance the exploitation of known policies with the exploration of new policies.

Algorithm 1 Deep Deterministic Policy Gradient (DDPG)

Initialize critic network Q_ϕ and actor network π_θ with random parameters ϕ and θ
Initialize target networks $\phi' \leftarrow \phi$, $\theta' \leftarrow \theta$
Initialize replay buffer
for each episode **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for $t = 1$ to T **do**
    Select action $a_t = \pi_\theta(s_t) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and next state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in replay buffer
    if t mod $train\_every = 0$ **and** $t \geq train\_after$ **then**
      for $k = 0$ to K **do**
        Sample a random minibatch B of transitions (s_i, a_i, r_i, s_{i+1}) from replay buffer
        Set $y_i = r_i + \gamma Q_{\phi'}(s_{i+1}, \pi_{\theta'}(s_{i+1}))$
        Update critic by minimizing the loss:

$$J_c = \frac{1}{|B|} \sum_i \left(y_i - Q_\phi(s_i, a_i)\right)^2 \tag{2.35}$$

        Update the critic parameters using gradient descent:

$$\phi_{k+1} = \phi_k - \alpha_c \nabla_\phi J_c \tag{2.36}$$

        if k mod $delay = 0$ **then**
          Update the actor policy using the sampled policy gradient:

$$J_a = \frac{1}{N} \sum_i Q_\phi(s_i, \pi_\theta(s_i)) \tag{2.37}$$

          Update the actor parameters using gradient ascent:

$$\theta_{k+1} = \theta_k + \alpha_a \nabla_\theta J_a \tag{2.38}$$

          Update the target networks:

$$\phi' \leftarrow \tau\phi + (1 - \tau)\phi' \tag{2.39}$$

$$\theta' \leftarrow \tau\theta + (1 - \tau)\theta' \tag{2.40}$$

        end if
      end for
    end if
  end for
end for

Chapter 3

Reinforcement Learning from Expert Demonstrations

The convergence of RL and IL represents a compelling evolution in the field of machine learning, aiming to harness the strengths of both paradigms while mitigating their individual limitations. RL excels in environments where the agent can explore and learn from interactions, optimizing its policy to maximize long-term rewards. However, this process can be data-intensive and risky, especially in complex or safety-critical domains. IL, on the other hand, leverages expert demonstrations to bootstrap the learning process, providing a more efficient and safer starting point. However, IL often struggles with generalization beyond the demonstrated scenarios and may require high-quality expert data.

RLED bridges these methodologies, offering a hybrid approach that leverages the structured guidance of expert demonstrations with the adaptive learning capabilities of RL. By incorporating demonstration data, RLED accelerates the learning process, improves policy performance, and improves safety during exploration. This integration allows agents to benefit from the expert's knowledge, reducing the sample complexity and addressing the exploration-exploitation dilemma inherent in traditional RL.

The rationale for combining RL and IL in RLED lies in their complementary strengths. Although RL provides the robustness and adaptability needed for learning in dynamic and uncertain environments, IL offers a shortcut to acquiring foundational knowledge that would be costly to learn from scratch. This synergy not only expedites the learning process, but also imbues the agent with a higher baseline performance from the outset.

In practice, RLED operates by integrating the demonstration data at various stages of the learning process. Prior knowledge, acquired before the RL training begins, provides initial

guidance that shapes the agent's early interactions. Online knowledge, sourced during the RL training, offers intermittent corrections and refinements based on the expert's insights. This knowledge integration ensures that the agent continuously aligns its learning trajectory with expert-provided benchmarks while retaining the flexibility to adapt and optimize its policy through interaction.

RLED has shown remarkable applicability across various domains, from robotics and video games to finance and urban planning. In each application, the ability to combine RL's learning dynamics with IL's expert guidance has led to significant improvements in performance, safety, and efficiency.

However, the path forward is not without challenges. The quality and quantity of demonstrations, handling system delays, managing partial observability, and mitigating the impact of noisy or detrimental demonstrations remain critical areas for further research. Addressing these challenges will be crucial to fully realize the potential of RLED and extend its applicability to even more complex and nuanced environments.

3.1 Foundations of RLED

RLED harnesses two primary knowledge sources to augment the RL learning process: 1) prior knowledge, gathered from demonstrations before the RL process begins, and 2) online knowledge, occasionally sourced from demonstrations during the RL training phase [3]. This knowledge supplements the insights an agent gradually accumulates through its interactions with the environment. RLED operates under three pivotal assumptions:

1. The agent can interact with the environment.

2. Feedback on environment rewards is accessible.

3. At least one type of knowledge is available from demonstrations.

Drawing parallels with traditional RL, RLED is formalized in the context of two MDP. The primary MDP corresponds to the common RL approach [14], which is defined as the 4-tuple:

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T} \rangle,$$

where:

- $\mathcal{S} \in \mathbb{R}^{\Lambda}$ represents the state space.

- $\mathcal{A} \in \mathbb{R}^{\Omega}$ denotes the admissible control action space.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ encapsulates the deterministic transition function that dictates the behavior of the environment.

Within this framework, the agent executes a control action $a \in \mathcal{A}$ in a given state $s \in \mathcal{S}$, subsequently earning an instant reward $r = \mathcal{R}(s, a)$ and transitioning to the next state $s' = \mathcal{T}(s, a)$.

The secondary MDP models expert behavior and is articulated as the 3-tuple:

$$\mathcal{D} = \langle \mathcal{C}, \mathcal{O}, \mathcal{L} \rangle,$$

where:

- $\mathcal{C} \in \mathbb{R}^{\Upsilon}$ represents the experienced state space.
- $\mathcal{O} \in \mathbb{R}^{\Gamma}$ characterizes the control action space executed by the expert.
- $\mathcal{L} : \mathcal{C} \times \mathcal{O} \to \mathcal{C}$ defines the deterministic transition function in the context of the expert.

Here, the expert takes a control action $o \in \mathcal{O}$ for each state $c \in \mathcal{C}$, and the subsequent state c' is determined by $\mathcal{L}(c, o)$.

To address disparities in morphologies between the expert and the apprentice, a recording mapping is introduced as

$$g_{rec} : \mathcal{C} \times \mathcal{O} \to \mathcal{S}^d \times \mathcal{A}^d.$$

where:

- $\mathcal{S}^d \subset \mathcal{S}$ signifies the demonstrated state subset.
- $\mathcal{A}^d \subset \mathcal{A}$ designates the subset of demonstrated control actions.

Then, we can formulate the expert's control policy as:

$$\pi^d(s^d) = a^d, \tag{3.1}$$

where $s^d \in \mathcal{S}^d$ is the demonstrated state, and $a^d \in \mathcal{A}^d$ corresponds to the demonstrated action.

The overarching goal is to obtain a policy $\pi^- : \mathcal{S} \to \mathcal{A}$ similar to the expert policy π^d but with superior efficacy, while simultaneously optimizing the return. Herein, the return is expressed as:

$$G_t = \sum_{k=t}^{T-1} \gamma^{k-t} \mathcal{R}(s_k, a_k), \tag{3.2}$$

with $\gamma \in [0, 1)$ symbolizing the discount rate.

The state-action value function is articulated as:

$$Q(s_t, a_t) = \mathbb{E}\pi[Gt|s_t, a_t],$$

its recursive formulation governed by the Bellman equation [20] [14] posits:

$$Q(s_t, a_t) = \mathbb{E}\pi[\mathcal{R}(st, a_t) + \gamma Q(s_{t+1}, a_{t+1})].$$

The optimal policy emerges as:

$$\pi^*(s_t) = \arg\max_{a'} Q(s_t, a'),$$

and the optimal equation governing the state-action value function is:

$$Q^*(s_t, a_t) = \mathbb{E}[\mathcal{R}(s_t, a_t) + \gamma \max_{a'} Q^*(s_{t+1}, a')].$$

The assumption rests on the expert being an authority in the domain, possessing a deep understanding or substantial experience regarding the task. However, it is crucial to underscore that labeling the teacher as an "expert" does not automatically equate their behavior to the optimal policy. In numerous instances, especially when humans are the demonstrators, it is plausible to assume that the demonstrations lean more towards suboptimal control policies. Human factors such as sporadic errors, micro-tremors, and coordination challenges are likely culprits.

Theorem 4. *The expert policy is postulated as suboptimal. Furthermore, there exists a locally optimal policy* π^- *in* Π *within the larger sets* $\mathcal{S}$ *and* $\mathcal{A}$*, such that:*

$$Q(s_t, \pi^d(s_t)) \leq Q(s_t, \pi^-(s_t)) \tag{3.3}$$

RLED methods are classified on the basis of their utilization of demonstration trajectories in the formulation of the control policy.

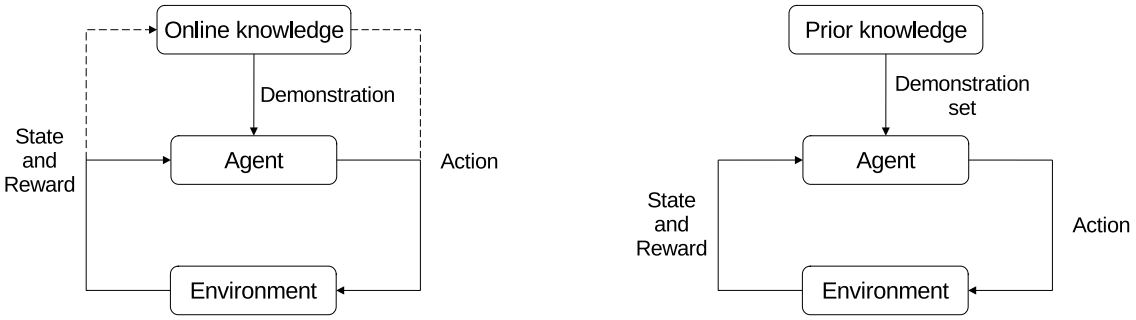

Figure 3.1: These diagrams depict the interaction between knowledge sources, the agent, and the environment, which guide the RL process. Dashed line: Intermittent interaction. In RLED with prior knowledge, the teacher offers a series of demonstration trajectories (prior to the RL process) that serve as a knowledge source for the agent in the RL process. In RLED with online knowledge, the teacher sporadically provides a demonstration trajectory (by assuming control of the agent) that the agent uses as a knowledge source in the RL process.

3.2 RLED from Prior Knowledge

In this approach, the teacher offers a collection of demonstration trajectories before beginning the reinforcement learning process. These demonstrations guide the agent during its learning phase, enabling it to emulate the teacher's behavior.

3.2.1 Biased Exploration

Biased exploration in RLED aims to obtain a control policy that aligns with the optimal aspects of the teacher's behavior. Thus, the RL agent is incentivized to actively explore and assess the states and actions evident in the demonstration trajectories. The goal is to discern when to adopt demonstration strategies or explore potentially superior alternatives.

Human-Agent Transfer (HAT) Taylor, Suay, and Chernova [37] proposed the Human-Agent Transfer (HAT) method, a three-step approach designed to transfer knowledge from human demonstrations to an RL agent. The steps are as follows:

1. The agent logs demonstrations of all state and action transitions.

2. From these recordings, a decision tree learning method is used to derive a summary control policy in the form of a list of rules.

3. These rules are then transferred to an RL agent, which further improves and refines the summary control policy.

Three methods are proposed for transferring the summarized control policy to an RL agent: Value Bonus, Extra Action, and Probabilistic Policy Reuse. Value Bonus assigns a Q-value to the actions specified by the summarized policy, compelling the RL agent to execute these actions for a number of episodes. Extra Action provides the RL agent the option to decide between taking a pseudo-action (the summarized control policy actions) or a random action, thereby learning through exploration when to follow the summarized control policy and when to opt for a different control action. Probabilistic Policy Reuse is similar to the ϵ-greedy method [2], assigning probabilities ψ to take actions from the summarized control policy, ϵ to random actions and $1 - \psi - \epsilon$ to take the greedy action.

Confidence-HAT (CHAT) Confidence-HAT (CHAT) [38] extends the HAT approach by introducing confidence-aware classifiers in the second and third steps. In the second step, three classifiers are trained from the recorded demonstrations: Gaussian Process HAT (GPHAT), Neural Network HAT (NNHAT), and Decision Tree HAT (DTHAT). GPHAT uses a Gaussian model to estimate confidence, NNHAT applies a neural network with a softmax layer for confidence estimation, and DTHAT uses the accuracy of leaf nodes as an estimate of confidence. In the third step, Probabilistic Policy Reuse is used to transfer the summarized control policy to an RL agent, with an added restriction to execute actions suggested by the demonstrations only when the confidence level exceeds a certain threshold.

Dynamic Reuse of Prior (DRoP) Dynamic Reuse of Prior (DRoP) [39] further enhances HAT by incorporating an online confidence measure. This measure uses a temporal difference model to evaluate the performance of source actions for a given state. The confidence measure is updated using a rule that considers the type of confidence, either prior knowledge (CP) or Q knowledge (CQ). The source actions are selected based on a probability distribution that favors actions with higher confidence, either through a hard decision rule that maximizes the confidence or a soft decision rule that allows for exploration of lower-confidence actions.

Reward Reshaping Brys et al. [40] proposed a method for reward reshaping in RLED. This approach introduces a potential function to complement the base reward function, guiding the RL agent towards states and actions demonstrated by the expert. The potential function models the distribution of states and actions in the demonstrations using a non-normalized multi-variate Gaussian. High values of the potential function indicate that an

action has been demonstrated for a state near the current state, encouraging the agent to follow the demonstrated behavior. This idea was also applied to inverse reinforcement learning (IRL) by Suay et al. [41].

Introspective Reinforcement Learning Li, Brys, and Kudenko [42] extended reward reshaping through an introspective RL agent. This agent records its state-action decisions and experiences during training in a priority queue, estimating a Monte Carlo Q-value to calculate the potential function. The introspective agent does not require optimal demonstrations, but leverages prior knowledge to gradually improve its policy, replacing lower action-values in the queue with higher ones as better decisions are found.

Bayesian Reward Shaping Gimelfarb, Sanner, and Lee [43] applied a Bayesian framework to reward shaping. This approach integrates expert knowledge in a nuanced manner, enhancing the ability of the RL agent to adapt to new and varied scenarios by using probabilistic models to infer the most beneficial actions based on previous demonstrations.

3.2.2 Extended Optimization Criterion

This approach augments the optimization problem by incorporating additional terms, which guide the agent to emulate the teacher's actions but with enhanced efficiency. A natural extension of the optimization incorporates a pure IL loss, compelling the agent to conform to the teacher's decisions. However, reliance exclusively on IL can result in overfitting and reduced generalization capabilities, a concern highlighted by Lakshminarayanan et al. [44]. The optimization integrates a variety of methods, including:

With Pre-training Phase

The pre-training phase in RLED methodologies involves initially training the agent using only the demonstration data, without any interaction with the environment. This phase aims to provide a solid foundation by leveraging supervised learning techniques to closely follow expert demonstrations.

Deep Q-learning from Demonstrations (DQfD) One of the most notable works in this area is Deep Q-learning from Demonstrations (DQfD) proposed by Hester et al. [45]. DQfD combines Prioritized Double DQN (PDD DQN) with Imitation Learning (IL). During the pre-training phase, the RL agent acts as a supervised learning algorithm, utilizing only

the demonstration data stored in a replay buffer. The optimization process involves multiple loss functions: a one-step and an n-step double Q-learning loss, L2 regularization, and a supervised large-margin classification loss. This ensures that the agent learns a policy closely aligned with the demonstrated actions before interacting with the environment. After the pre-training phase, the agent continues to learn from its interactions with the environment, using a mixture of its experiences and the demonstrations to refine its policy. The large-margin classification loss particularly helps in maintaining a margin between the action-values of demonstrated and non-demonstrated actions, thus guiding the agent towards the demonstrated behaviors more robustly.

DDPG from Demonstrations (DDPGfD) DDPG from Demonstrations (DDPGfD) by Vecerik et al. [46] extends the DDPG algorithm to incorporate demonstrations. Like DQfD, DDPGfD begins with a pre-training phase where the agent learns in a supervised manner using demonstration data. The agent stores these demonstrations in a prioritized replay buffer and samples mini-batches to update the critic and actor networks. The fine-tuning phase involves interacting with the environment while continually refining the policy using both the demonstration data and the newly gathered experiences. This method is particularly notable for its applicability in continuous action spaces, which is a significant advancement over the limitation of DQfD to discrete action spaces. The combination of one-step and n-step return losses helps spread the sparse rewards more effectively, thereby enhancing learning efficiency.

Cycle-of-Learning (CoL) Cycle-of-Learning (CoL), proposed by Goecks et al. [47], builds on DDPG by integrating a behavior cloning loss into the learning process. The agent first undergoes a pre-training phase using the behavior cloning loss to learn from expert demonstrations stored in an expert buffer. This pre-training helps initialize the agent's policy, making it more robust when it starts interacting with the environment. During the fine-tuning phase, the agent learns from both its interactions and a fixed ratio of expert demonstrations, effectively balancing between exploration and exploitation. The behavior cloning loss function ensures that the agent mimics the expert's actions closely, providing a strong starting point for further reinforcement learning.

Demonstration Augmented Policy Gradient (DAPG) Demonstration Augmented Policy Gradient (DAPG) by Rajeswaran et al. [48] introduces a method that incorporates a pre-training phase based on maximum-likelihood estimation of the expert's policy, followed by a reinforcement learning phase using natural policy gradients. The pre-training phase

provides a good initialization, significantly reducing the sample complexity required for effective learning. This method ensures that the agent starts with a strong policy derived from demonstrations and then fine-tunes it through interactions with the environment. The use of maximum-likelihood estimation allows the agent to align its initial policy closely with expert demonstrations, facilitating a smoother transition to the reinforcement learning phase.

Other Notable Methods Other methods, such as those proposed by Cruz et al. [49] and Gao et al. [50], also follow a similar approach. Cruz et al. utilized a deep neural network trained on human demonstrations, which then initializes a DQN or A3C agent. Gao et al.'s Normalized Actor-Critic (NAC) method avoids the overfitting problem by not using a supervised loss in the pre-training phase, making it more robust to imperfect or noisy demonstrations. The NAC method parameterizes the action-value function and derives the state-value and policy from it, ensuring a stable learning process. The emphasis on robustness to noisy demonstrations makes NAC particularly suitable for real-world applications where perfect demonstrations are not always available.

Without Pre-training Phase

In scenarios where a pre-training phase is not employed, demonstration data is integrated throughout the learning process. This approach ensures that the agent continuously refines its behavior, aligning it more closely with the teacher's actions.

Lakshminarayanan et al.'s Method Lakshminarayanan et al. [44] proposed a method that incorporates expert demonstrations into DQN by adding a weighted IL loss function to the original DQN loss. This method ensures that the action-values for demonstrated actions are high, guiding the agent to prefer these actions during learning. The integration of imitation loss helps the agent to explore effectively while still adhering to the expert's demonstrated behavior. This approach mitigates the risk of overfitting by continuously balancing the reinforcement learning objectives with the imitation learning losses.

Nair et al.'s Method Nair et al. [51] extended the DDPG framework by integrating a behavior cloning loss and a filter that applies the IL loss only when the critic determines that the demonstrated actions are better than the agent's actions. This method not only helps the agent find better control actions, but also incorporates a random reset to demonstration states, starting learning episodes from demonstrated states to enhance learning efficiency.

This ensures that the agent benefits from the demonstration data throughout the learning process, improving its ability to generalize and perform well in varied scenarios.

Direct Policy Iteration with Demonstrations (DPID) Direct Policy Iteration with Demonstrations (DPID) by Chemali et al. [52] builds on Direct Policy Iteration (DPI) by integrating demonstration data into the set of training samples. This method constructs a cost-sensitive training set that combines states sampled from distribution and demonstration trajectories. The loss functions in DPID ensure that the agent's policy closely follows the demonstrated actions while allowing for exploration to improve upon these actions. This approach effectively combines demonstration data with reinforcement learning to create a robust learning framework.

Policy Optimization from Demonstration (POfD) Policy Optimization from Demonstration (POfD) by Kang et al. [53] introduces a practical algorithm based on Jensen-Shannon divergence, similar to a Generative Adversarial Network (GAN). POfD guides the RL agent's exploration near the demonstrated policy by minimizing the difference between the agent's policy and the expert's policy. This method uses occupancy measures to better exploit demonstrations and facilitates the optimization process. The integration of occupancy measures allows for a more refined approach to leveraging demonstration data, ensuring that the agent's policy remains close to the expert's policy while still exploring new strategies.

Soft Expert Guidance from Demonstrations Jing et al. [54] proposed Soft Expert Guidance from Demonstrations, which constrains the control policy optimization within a bounded tolerance factor, ensuring that the agent's policy stays close to the demonstration policy. This method limits exploration to regions near the demonstrated trajectories, providing a balance between following the expert's behavior and exploring new strategies. The use of bounded tolerance factors allows for a controlled exploration process, ensuring that the agent does not deviate significantly from the demonstrated behavior while still learning to improve upon it.

Distributed Setup

In distributed reinforcement learning setups, methods such as Ape-X DQfD and Expert Augmented ACKTR (EA-ACKTR) exemplify the integration of demonstrations to enhance scalability and efficiency.

Ape-X DQfD Ape-X DQfD [55] combines the principles of DQfD with the distributed architecture of Ape-X DQN. This approach removes the pre-training phase and introduces a transformed Bellman operator for the temporal difference loss function. The method uses a fixed ratio of agent-generated transitions and expert transitions to stabilize learning and improve performance across distributed nodes. Using the distributed setup, Ape-X DQfD ensures that the learning process is scalable and efficient, allowing the simultaneous training of multiple agents.

Expert Augmented ACKTR (EA-ACKTR) Expert Augmented ACKTR (EA-ACKTR) [56] extends the ACKTR algorithm by incorporating expert demonstrations stored in a separate replay buffer. This method adds a new term to the ACKTR loss function to consider the demonstration trajectories, ensuring that the agent benefits from expert knowledge while interacting with the environment. The use of a separate replay buffer for demonstrations helps in maintaining a clear distinction between the expert data and the agent's own experiences, facilitating a more organized and effective learning process.

Recurrent Replay Distributed DQN with Demonstrations (R2D3) R2D3 (Recurrent Replay Distributed DQN with Demonstrations) by Paine et al. [57] combines the R2D2 algorithm with expert demonstrations. In this method, multiple actors run independent copies of the behavior policy, storing interactions in a shared buffer prioritized based on TD-error. The demonstrations are stored in a separate buffer and the learning process involves sampling from both buffers to optimize the policy. The integration of recurrent replay mechanisms ensures that the agent can effectively leverage both its own experiences and expert demonstrations to improve its performance.

Acceleration Techniques Acceleration techniques such as Dynamic Frame Skipping-Experience Replay (DFS-ER) and Frame Skipping-Experience Replay (FS-ER) by Yeo et al. [58][59] speed up training by managing replay buffers and incorporating frame skipping. These methods use dual replay buffers for demonstration trajectories and agent interactions, applying online and dynamic frame skipping schemes to enhance learning efficiency. The use of frame skipping allows the agent to focus on the most relevant parts of the demonstration trajectories, thereby improving the efficiency of the learning process.

3.2.3 Episode Initialization

Demonstrated trajectories are employed to establish the starting point of learning episodes, enabling the agent to begin from a demonstrated state proximate to a high-reward state. As learning progresses, episodes begin from increasingly distant demonstrated states, facilitating a gradual learning curve. This method, proposed by Salisman and Chen [60] and Resnick et al. [61], uses the states of a single demonstration as starting points for each episode in the RL process. Furthermore, the technique introduced by Nair et al. [51] also aligns with the concept of episode initialization, incorporating a mechanism that randomly resets to the demonstration states, thus allowing the learning episode to begin from a random state within the demonstration trajectory.

3.2.4 Reasoning

This unique approach draws inspiration from human cognitive processes, specifically causal reasoning. Torrey [62] introduces a concept known as reasoning, which mimics how humans reason and acquire knowledge from task demonstrations. In RfD, the task is segmented to discern the relationships between cause and effect, allowing a deeper understanding of the dynamics of the task.

Reasoning from Demonstration (RfD) RfD is a novel approach introduced by Torrey [62], which aims to emulate human cognitive processes in understanding and learning from task demonstrations. This approach goes beyond simple imitation by breaking down tasks into fundamental components and establishing causal relationships between actions and outcomes.

In RfD, the agent begins by identifying the objects present in each state and observing their attributes such as position, velocity, and the regions they occupy within the environment. Events are defined as interactions between these objects, which are crucial for understanding the dynamics of state transitions. By focusing on these object interactions, the agent can discern the essential elements that drive the execution of the task.

The environment in RfD is characterized by its state space, action space, and an unknown probability distribution governing state transitions. This probabilistic approach helps the agent manage the uncertainties and variabilities inherent in real-world environments. Understanding how different actions lead to state transitions allows the agent to build a comprehensive model of its operational context.

Tasks in RfD are structured around specific subsets of state spaces: the beginning, the end,

success feedback states, and failure feedback states. This structure provides clear indicators of progress, helping the agent recognize successful and unsuccessful outcomes. Objectives are defined as desirable interactions that lead to task success, while anti-objectives are undesirable interactions that should be avoided to prevent task failure.

During the training process, RfD develops three main components: a theory, a map, and a set of policies. The theory consists of cause-effect hypotheses that link object interactions (causes) to environmental feedback (effects). This causal framework enables the agent to understand the impact of its actions on the results of the task. The map is a graphical representation of the environment, illustrating regions and their connections, which aids in spatial planning and navigation. Policies are derived from the theory and map, guiding the agent's actions to achieve objectives while avoiding anti-objectives.

A key advantage of RfD is its robustness to low-quality and imperfect demonstrations. Traditional imitation learning methods often struggle with suboptimal demonstrations as they focus primarily on mimicking observed actions. In contrast, RfD's emphasis on causal reasoning allows it to extract valuable insights even from imperfect data, making it more versatile and effective in varied scenarios.

In summary, RfD uses human-like reasoning to enhance the learning process in RLED. By breaking down tasks into core components and establishing causal relationships, RfD enables agents to better understand and adapt to complex environments. This approach not only improves learning efficiency, but also enhances the agent's ability to generalize from demonstrations, leading to more robust and adaptable performance.

3.3 RLED with Online Knowledge

In the context of RLED with online knowledge, the agent dynamically interacts with the teacher, querying which behavior to follow upon observing certain states. This interaction involves the teacher taking control over the agent for several consecutive actions, providing demonstrations that are then stored and used by the agent to improve its proficiency. After the query period, the control is reverted to the agent, and the standard RL process resumes until the next interaction. The primary challenge in this approach is determining the optimal timing to request a demonstration, as the teacher's time is often more valuable than the agent's. This section details several methodologies that address these challenges and optimize the request process for demonstrations.

Exploration from Demonstration (EfD) Subramanian, Isbell Jr., and Thomaz introduced Exploration from Demonstration (EfD) [63], which guides the agent's exploration using statistical measures to identify influential regions in the learning algorithm. EfD is based on Q-Learning with function approximation, where the Q-values are estimated using state-action features. The agent computes the influence as a combination of leverage and discrepancy measures. Leverage assesses how far a specific observation is from the known observations' convex hull, while discrepancy measures the observation's contribution to the model error. When the influence of an observation exceeds a certain threshold, it is flagged as influential, prompting the agent to request a demonstration from the teacher. This targeted querying helps the agent focus on crucial areas of the state space that require further exploration and learning.

Active Reinforcement Learning with Demonstrations (ARLD) Active Reinforcement Learning with Demonstrations (ARLD) by Chen et al. [64] is a framework based on DQN that estimates uncertainty in the agent's current state to decide when to ask for a demonstration. ARLD employs two methods to estimate uncertainty: the divergence of bootstrapped DQN and the predictive variance of noisy DQN. The divergence method uses bootstrapped estimates of the Q-value and measures the Jensen-Shannon divergence between these estimates to determine uncertainty. The variance method uses noisy networks, where the weights and biases are perturbed by noise, and the predictive variance is calculated to estimate the uncertainty. When the uncertainty in a state exceeds a threshold, the agent requests a demonstration, which is then used along with a supervised loss and the usual DQN loss to enhance learning. This approach ensures that the agent seeks guidance in states where its predictions are most uncertain, thereby improving its performance and learning efficiency.

Minimal Human Effort Framework Rigter, Lacerda, and Hawes proposed a framework to minimize human effort by intelligently switching control between the human teacher and the agent [65]. In this framework, the agent decides whether to request a demonstration or use its learned policy based on a cost function that considers the human teacher's time. The selection of the control policy is formulated as a contextual Multi-Armed Bandit (MAB) problem, where probabilities are estimated using a continuous correlated beta process. The agent chooses a control policy that minimizes the estimated cost of failure and the cost of human demonstration time. By dynamically deciding when to involve the human teacher, this method reduces unnecessary interruptions and optimizes the use of human demonstrations,

ensuring that the agent only seeks help when it is likely to be most beneficial.

3.4 Applications of RLED

RLED has found numerous applications across various fields due to its ability to leverage demonstration data to improve sample efficiency and performance. This section explores the practical applications of RLED in areas such as robotics, video games, natural language processing, finance, and urbanism.

Robotics

RLED has shown significant promise in the field of robotics, where learning from demonstrations can accelerate the training process and improve performance in real-world tasks. One notable application is in robotic insertion tasks. Vecerik et al. [46] applied DDPGfD to four simulated robot insertion tasks: classic peg-in-hole, hard-drive insertion into a computer chassis, two-pronged deformable plastic clip insertion into a housing, and a cable insertion task. The robotic system used was a Sawyer robot with 7 degrees of freedom. For the physical hard-drive insertion task, demonstration trajectories were provided through kinesthetic teaching, while for simulated tasks, a hard-coded joint space P-controller was used to generate demonstration trajectories.

Similarly, Wu et al. [66] adapted DQfD for a robotic precision insertion problem, dividing the task into pose alignment and peg-in-hole insertion phases. The robot system consisted of a 3 degree-of-freedom manipulator, a 4 degree of freedom adjustable platform, three microscopic cameras, and a high-precision force sensor. Human kinesthetic teaching provided the demonstration trajectories.

Another significant application in robotics was demonstrated by Zhu et al. [67] using DAPG. This method was applied to dexterous multi-fingered robotic hands, including a ROBEL 3-fingered manipulator with 9 degrees of freedom and a 4-fingered Allegro hand with 16 degrees of freedom. The tasks involved rigid and deformable objects, such as valve rotation, box flipping, and door opening. A small number of kinesthetic demonstrations provided by humans significantly accelerated the reinforcement learning process.

In the medical field, Keller et al. [68] utilized a DDPGfD-based approach for autonomous robotic deep anterior lamellar keratoplasty (DALK) needle insertion, an ophthalmic microsurgery procedure. Experienced corneal surgeons provided the demonstration trajectories, highlighting the potential of RLED to perform highly precise and delicate surgical tasks.

Video Games

Video games have been a popular domain for evaluating reinforcement learning algorithms due to their complexity and well-defined environments. RLED methods have been successfully applied to challenging games such as Montezuma's Revenge and Minecraft. The Atari Grand Challenge dataset [69] includes five challenging games with almost 45 hours of gameplay, providing a valuable resource to evaluate RLED algorithms.

Ecoffet et al. [70] successfully solved Montezuma's Revenge and Pitfall with superhuman performance using the Backplay method as part of their Go-Explore algorithm. This method employs episode initialization from states in the demonstration trajectories, facilitating easier exploration and reward acquisition.

The Minecraft demonstrations dataset (MineRL) [71] is another significant resource, consisting of 60 million automatically annotated state-action pairs and more than 500 hours of recorded human demonstrations. The dataset encourages the development of RLED algorithms by providing diverse and complex tasks within the Minecraft environment.

Natural Language Processing

In natural language processing (NLP), RLED has been applied to task-oriented conversational AI. Gordon-Hall et al. [72, 73] used DQfD to develop a dialog system to help users book flights or hotels. The system operates in an environment with large state and action spaces and sparse rewards, making it an ideal candidate for RLED. The use of expert demonstrations improved the dialog policy, which was initially based on a rule-based handwritten system. This approach was later enhanced to work with weak and cheap expert demonstrations, further demonstrating the flexibility and robustness of RLED in NLP applications.

Finance

RLED has also been applied in the financial sector for dynamic pricing and trading strategies. Liu et al. [74] proposed a dynamic pricing framework for e-commerce platforms that uses DQfD and DDPGfD to optimize long-term revenue while avoiding capital loss. The demonstration trajectories were derived from historical sales data, providing a solid starting point for the reinforcement learning process.

In quantitative trading, Liu et al. [75] developed an adaptive trading framework combining DDPGfD pretraining strategies and behavior cloning loss from exploration-based methods. This approach used recurrent deterministic policy gradient algorithms, significantly

outperforming traditional quantitative trading methods.

Urbanism

In urbanism, RLED has been applied to traffic signal control. An effective controller [76] that adjusts timing plans according to traffic dynamics provided demonstration trajectories. By adapting DQfD losses into the A2C algorithm, the learned policy outperformed the initial control policy, demonstrating the potential of RLED in optimizing urban traffic systems.

3.5 RLED Challenges

Implementing RLED in real-world environments involves navigating a labyrinth of challenges, some of which have been overcome, while others still lurk in the path ahead. We highlight both addressed and pending challenges. Additionally, we underscore the potential of advanced guidance methods, which may not always be human-derived, as tools to negotiate these hurdles. Works like [77, 78, 79, 80, 81], among others, offer insights into these challenges.

3.5.1 Addressed challenges in RLED

RLED, with its innovative approach, has found solutions to a gamut of challenges in RL's real-world applications:

Sample Efficiency Sample efficiency refers to the imperative quantum of data to craft a competent control policy. Classic DRL methods can be ravenous for data and often require large amounts of data for accurate learning. In stark contrast, IL techniques, leaning on the crutch of demonstrations, are the epitome of sample thriftiness. RLED ingeniously bridges this gap by drawing on the demonstration data. However, the efficacy of this method is contingent on these demonstrations providing a comprehensive overview of the pertinent states and their potential outcomes. A small set containing relevant demonstration trajectories can significantly speed up the learning process, making RLED a more practical approach in data-scarce environments.

Dimensionality The intimidating shadow of "the curse of dimensionality" looms large when dealing with expansive spaces [4]. Conventional RL agents are designed to guzzle copious amounts of data when learning within such high-dimensional arenas. Powell [5] identifies

three different spaces affected by the curse of dimensionality: the state space, the action space, and the result space. RLED has gracefully sped up around this challenge, ensuring that learning remains optimal without losing the connection to sample efficiency. Using demonstration data, RLED methods help reduce the exploration needed in high-dimensional spaces, thus addressing one of the most significant challenges in RL.

Safety In the realm of RL, safety is not just a buzzword but a doctrine. It is the art and science of balancing the pursuit of maximum rewards with the sanctity of system stability, all while ensuring that critical safety boundaries are not breached [6]. RLED illuminates the path to safety, using demonstrations as guiding beacons, thereby bypassing the pitfalls of excessive exploration, a common peril in traditional RL. Although RLED methods require some exploration, strategies such as those discussed in Section 2 improve safety by guiding the agent's behavior through demonstrations, thus preventing it from exploring dangerous states and actions.

Reward Specification Sutton and Barto's 2018 observations bring to the fore the paramount importance of a well-articulated reward signal [2]. While dense reward mechanisms such as the Cartesian distance can help in directing smooth agent trajectories, there is always the lurking danger of them deviating into uncharted territories of undesirable behaviors. Through RLED, we can employ demonstrations to elegantly dance around these pitfalls, ensuring that the agent remains on track towards its destined target states. Demonstrations help simplify the reward specification by highlighting relevant states and actions, thereby providing a clearer path to the goal.

3.5.2 Remaining RLED Challenges

Despite the laurels, RLED is not without its Achilles heel. Several challenges await, particularly those related to its robustness in a myriad of real-world scenarios.

System Delays The RL process is a delicate dance, easily tripped up by delays at various junctures. Katsikopoulos [82] identifies three types of delays in MDPs: observation delay, control delay, and reward delay. Observation delay occurs when status information is not available instantly, control delay when control action is not reflected immediately, and reward delay when the reward for an action is not obtained immediately. Although considerable strides have been made in understanding the impacts of consistent and random delays, the exact ramifications of such interruptions on RLED remain tantalizingly out of reach.

Observability The enigma of partial observability, where vital state data remains obscured, is a formidable adversary. This refers to situations where potentially important state information is not directly observable. The existing literature is replete with proposed strategies for taming this beast, yet the full arsenal of RLED's capabilities in this arena remains shrouded in mystery.

Noise A common problem with physical systems is signal noise from system sensors and signal noise going to system actuators. This noise can distort the data used for learning, making it difficult for the agent to distinguish between meaningful information and irrelevant noise.

Detrimental Demonstrations Akin to the proverbial double-edged sword, demonstrations in real-world settings can sometimes mislead, either through inadvertence or malice. This is likely to happen in real-world scenarios, where any user could deploy and train an RLED agent. Users might provide harmful demonstrations that introduce dangerous behavior, confusing behaviors that show divergent solutions for the same task, or useless behavior that lacks relevant data. The broader implications and potential strategies to harness or deflect these demonstrations in the context of RLED methods warrant a deeper exploration.

Quality and Number of Demonstrations By adding demonstration paths to the RL process, an additional hyperparameter is introduced: the amount of relevant data needed to speed up the learning process and achieve the desired behavior. It is unclear how much relevant data a single demonstration should contain and how many demonstrations should be provided. Understanding the optimal quantity and quality of demonstrations required to accelerate learning remains an intriguing puzzle.

RL Acceleration via Remonstrations When we introduce demonstrations into the RL tableau, it adds a layer of complexity. Deciphering the optimal blend and volume of demonstrations that would catalyze the learning process remains an intriguing puzzle. Ensuring that demonstrations are effective without overwhelming the agent with redundant or irrelevant data is crucial for efficient learning.

3.5.3 Other Forms of Human Guidance

Human-guided RL is limited not only to approaches such as IRL, Batch RL, or RLED. Other forms of human guidance can also provide valuable information to guide the RL process.

These include human evaluative feedback, human preferences, hierarchical imitation, human attention, and no-action state sequences [83, 84].

Combining multiple sources of knowledge is a promising approach that could help train sufficiently robust RL algorithms. Ibarz et al. [85] propose a method for training a reward model where the agent learns from demonstrations during the pretraining phase and then uses human preferences to refine its reward function. Unlike the common RLED approach, this method assumes that the reward is unavailable and relies on human intent communicated through demonstrations and preferences. This approach guides the agent's learning process even when explicit rewards are sparse or absent, demonstrating the potential of integrating diverse forms of human guidance in RL.

In summary, while RLED has addressed several key challenges in RL, significant hurdles remain. Addressing these challenges through advanced and potentially nonhuman-derived guidance methods will be crucial for the continued evolution and practical application of RLED in complex, real-world environments.

Chapter 4

Exploration Algorithms for Reinforcement Learning from Expert Demonstrations

4.1 Biased Exploration Reinforcement Learning from Expert Demonstrations

4.1.1 Biased Exploration

To effectively tackle the solutions for (3.1) and (3.1), particularly when dealing with the complexities of high-dimensional spaces and continuous action spaces, the Twin Delayed Deep Deterministic Policy Gradient (TD3) methodology emerges as a pivotal tool. This strategy, as fully described in [35], is selected for its notable performance superiority compared to the classical Deep Deterministic Policy Gradient (DDPG) approach [34]. TD3 distinguishes itself as a model-free Deep Reinforcement Learning (DRL) method, employing an innovative actor-critic architecture. The actor network is central to policy estimation, captured in the equation:

$$NN_a = \pi_\theta(s_t) \approx \pi^*(s_t). \tag{4.1}$$

Meanwhile, the critic network focuses on estimating the state-action value function. To counteract the overestimation of state-action values, a strategy of employing dual critic networks

is used, according to the approach recommended in [35]:

$$\begin{aligned} NN_{c_1} &= Q_{\phi_1}(s_t, a_t) \approx Q^*(s_t, a_t) \\ NN_{c_2} &= Q_{\phi_2}(s_t, a_t) \approx Q^*(s_t, a_t). \end{aligned} \tag{4.2}$$

The lesser of the two values from the dual critic networks is used to calculate the mean squared Bellman error (MSBE) target. The stability of the learning process is further reinforced by introducing auxiliary neural network function approximators for both the state-action value function and the policy [34, 35]. These secondary networks are denoted as:

$$\begin{aligned} NN_{a'} &= \pi_{\theta'}(s_t) \approx \pi_\theta(s_t) \\ NN_{c'_1} &= Q_{\phi'_1}(s_t, a_t) \approx Q_{\phi_1}(s_t, a_t) \\ NN_{c'_2} &= Q_{\phi'_2}(s_t, a_t) \approx Q_{\phi_2}(s_t, a_t). \end{aligned} \tag{4.3}$$

The MSBE target is formulated as:

$$y = \mathcal{R}(s_k, a_k) + \gamma \min_{i=1,2} Q_{\phi'_i}(s_{k+1}, \tilde{a}), \tag{4.4}$$

where $\tilde{a} = \pi_{\theta'}(s_{k+1}) + \epsilon$, and $\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c)$, with σ and c being hyper-parameters.The optimization of the primary critics NN_{c_1} and NN_{c_2} involves minimizing the MSBE loss functions, which are defined as:

$$\begin{aligned} J_{c_1} &= \tfrac{1}{|B|} \textstyle\sum_{(s_k, a_k, s_{k+1}) \in B} (y - Q_{\phi_1}(s_k, a_k))^2 \\ J_{c_2} &= \tfrac{1}{|B|} \textstyle\sum_{(s_k, a_k, s_{k+1}) \in B} (y - Q_{\phi_2}(s_k, a_k))^2, \end{aligned} \tag{4.5}$$

where B represents a mini-batch of samples drawn from the replay buffer. In parallel, the primary actor NN_a is updated by maximizing the following loss function:

$$J_a = \frac{1}{|B|} \sum_{s_k \in B} Q_{\phi_1}(s_k, \pi_\theta(s_k)).$$

Secondary neural networks $NN_{a'}$, $NN_{c'_1}$, and $NN_{c'_2}$ are updated following the soft update laws [34]. Thus, the updating law for the primary critics is:

$$\phi_{i,k+1} = \phi_{i,k} - \alpha_c \nabla_{\phi_i} J_{c_i}, \tag{4.6}$$

and for the secondary critics is:

$$\phi'_{i,k+1} = \tau\phi_{i,k+1} + (1-\tau)\phi'_{i,k}, \tag{4.7}$$

where $i = 1, 2$. The updating law for the primary actor is:

$$\theta_{k+1} = \theta_k + \alpha_a \nabla_\theta J_a, \tag{4.8}$$

and for the secondary actor is:

$$\theta'_{k+1} = \tau\theta_{k+1} + (1-\tau)\theta'_k. \tag{4.9}$$

To mitigate the risk of overestimating the actor, [35] suggests a strategy of updating the actor less frequently than the critic. This approach also involves updating the weights of all neural networks only after accumulating a significant amount of agent experience in the replay buffer, ensuring a more data-informed adjustment process.

The final neural network introduced in this context is dedicated to estimating the expert policy:

$$NN_d = \hat{\pi}^d_\omega(s_t) \approx \pi^d(s_t).$$

This neural network, aimed at approximating the expert policy, is trained using data from ζ, gathered through the secondary MDP $\mathcal{D}$. The training employs the mean square error loss function:

$$J_d = \frac{1}{|B_d|} \sum_{(s^d_k, a^d_k) \in B_d} ||\hat{\pi}^d_\omega(s^d_k) - a^d_k||^2_2,$$

where $B_d \subset \zeta$ represents a mini-batch. The neural network NN_d is updated by the law:

$$\omega_{k+1} = \omega_k - \alpha_d \nabla_\omega J_d. \tag{4.10}$$

In model-free RL, the necessity to explore a substantial number of states to gain adequate experience for the convergence of the value function is a well-recognized requirement. This exploration imperative is particularly crucial in continuous spaces, where the prevalent method involves adding Gaussian noise to the control action derived from the policy:

$$a^+_t = \pi_\theta(s_t) + \varepsilon,$$

with a^+ representing the action taken in the environment and $\varepsilon \sim \mathcal{N}(0, \delta)$ the noise induced

to explore. In an effort to streamline this exploration and avoid visiting irrelevant states, the concept of biased exploration is proposed. Rather than visiting random states in each episode, the agent, with a specific probability in each episode, will focus on relevant states as directed by the estimated expert policy $\hat{\pi}\omega^d$. The exploration bias ξ is defined as follows:

$$\xi = \begin{cases} 1 & with \quad probability \quad \Phi \\ 0 & with \quad probability \quad 1-\Phi \end{cases}, \tag{4.11}$$

where Φ represents the Probability Policy Reuse (PPR) hyper-parameter. The control action to take is taken by the bias ξ as:

$$a_t^+(\xi) = \begin{cases} \hat{\pi}_\omega^d(s_t) & \longrightarrow \quad \xi = 1 \\ \pi_\theta(s_t) + \epsilon & \longrightarrow \quad \xi = 0 \end{cases}. \tag{4.12}$$

The central challenge in biased exploration RLED involves developing a policy π_θ that exceeds the expert policy π^d, utilizing the insights from the estimated expert policy $\hat{\pi}\omega^d$. The fundamental objective can be articulated as:

$$Q(s_t, \pi^d(s_t)) \leq Q(s_t, \pi_\theta(s_t)). \tag{4.13}$$

In this process, the RL process starts from a state demonstrated by the expert, denoted as $s_0^d \sim \zeta$. This initial state selection is strategic, with the aim of facilitating the discovery of positive rewards more readily. The effectiveness of this approach is underpinned by the bounded local generalization capability of the estimated expert policy. Specifically, for any state $\bar{s}t$ that lies within the vicinity of a demonstrated state, characterized by $\|\bar{s}t - s_t^d\|_2 \leq \epsilon_s$, the estimated expert policy adheres to a consistency condition. This condition is expressed as:

$$\|T(\bar{s}_t^d, \hat{a}_t^d) - T(s_t^d, a_t^d), \|_2 \leq \epsilon_s.$$

In refining the equation (4.13) within the framework of the applied algorithm and considering the consistency condition, it can be reformulated as:

$$\min_{i=1,2} Q_{\phi_i}(\bar{s}_t, \pi_\omega^d(\bar{s}_t)) \leq \min_{i=1,2} Q_{\phi_i}(\bar{s}_t, \pi_\theta(\bar{s}_t)). \tag{4.14}$$

Expanding on the left side of (4.14) yields:

$$\min_{i=1,2} Q_{\phi_i}(\bar{s}_t, \hat{a}_t^d) \approx \mathcal{R}(\bar{s}_t, \hat{a}_t^d) + \sum_{l=t+1}^{T-1} \gamma^{l-t} \mathcal{R}(\bar{s}_l, \hat{a}_l^d), \tag{4.15}$$

where $\hat{a}_t^d = \pi_\omega^d(s_t)$ represents the estimated action demonstrated by the expert.

In the context of a sparse reward function, where a positive reward is allocated only upon reaching a specific target state s^* within the environment, the reward function is characterized as follows:

$$\mathcal{R}(s_t, a_t) = \begin{cases} r \longrightarrow T(s_t, a_t) = s^* \\ 0 \longrightarrow T(s_t, a_t) \neq s^* \end{cases} .$$

Then (4.15) can be simplified as:

$$\min_{i=1,2} Q_{\phi_i}(\bar{s}_t, \hat{a}_t^d) \approx \gamma^{T-t-1} r, \tag{4.16}$$

and (4.14) follows:

$$\gamma^{T-t-1} r \leq \min_{i=1,2} Q_{\phi_i}(\bar{s}_t, \pi_\theta(\bar{s}_t)). \tag{4.17}$$

Finally, applying the biased exploration (4.12) to find the solution for (4.17) yields:

$$\gamma^{T-t-1} r \leq \min_{i=1,2} Q_{\phi_i}(\bar{s}_t, a_t^+(\xi)). \tag{4.18}$$

Through the implementation of biased exploration, a distinction is made in each episode regarding the agent's approach: whether to engage in normal exploration to discover superior actions or to bias its exploration by emulating similar demonstrations. This dichotomy allows for a more targeted and efficient learning process. The proposed algorithm, which integrates this concept of biased exploration, is succinctly summarized in Algorithm 2. This algorithmic approach strategically balances between exploring new actions and leveraging existing expert knowledge, thereby optimizing the learning trajectory.

4.1.2 Convergence Analysis

The convergence of the algorithm is analyzed using a methodology similar to that employed in the convergence studies presented in [86] and [87]. It is essential to note that this analysis assumes a fixed policy associated with ordinary differential equations (ODE).

The analysis begins by defining the Markov chain with the fixed estimated expert policy

Algorithm 2 Biased exploration RLED

Collect demonstration data (s^d, a^d) in ζ
Train estimated expert policy $\hat{\pi}^d_\omega$ (Eq. (4.10))
Initialize primary critic networks Q_{ϕ_1}, Q_{ϕ_2}, and actor network π_θ with random parameters ϕ_1, ϕ_2, θ
Initialize secondary networks $\phi'_1 \leftarrow \phi_1$, $\phi'_2 \leftarrow \phi_2$, $\theta' \leftarrow \theta$
Initialize replay buffer
Set PPR Φ
for each episode **do**
  Select the initial state $s^d_0 \sim \zeta$
  Calculate ξ (Eq. (4.11))
  for $t = 0$ to T **do**
    Take control action $a^+_t(\xi)$ (Eq. (4.12))
    Observe reward $\mathcal{R}(s_t, a^+_t)$ and next state s_{t+1}
    Store $(s_t, a^+_t, \mathcal{R}(s_t, a^+_t), s_{t+1})$ in the replay buffer
    if t mod $train\_every = 0$ **and** $t \geq train\_after$ **then**
      for $k = 0$ to K **do**
        Sample a batch of transitions $B = \{(s, a^+, \mathcal{R}(s, a^+), s'\}$ from replay buffer
        Calculate target y (Eq. (4.4))
        Update primary critics (Eq. (4.6))
        if k mod $delay = 0$ **then**
          Update primary actor (Eq. (4.8))
          Update secondary networks (Eq. (4.7) and Eq. (4.9))
        end if
      end for
    end if
  end for
end for

π^d_ω as:

$$\mathcal{N} = \langle \mathcal{S}, \mathcal{T}_{\pi^d_\omega} \rangle,$$

where $\mathcal{T}\pi\omega^d$ represents the transition function under the fixed policy. The chain $\mathcal{N}$ is postulated to be uniformly ergodic with an invariant probability measure μ_S. Furthermore, the policy is constructed such that $P_{\pi^d_\omega}\left[\hat{a}^d|\bar{s}\right] > 0$ for all $\hat{a}^d \in \mathcal{A}$ and μ_S-almost all $\bar{s} \in \mathcal{S}$.

The convergence analysis also considers a function approximation of the state-value function for both the primary and secondary critics, formulated as follows:

$$\begin{array}{l} Q_{\phi_i}(s,a) = \sum_{j=1}^{N} f_j(s,a)\phi_{i,j} = f^T(s,a)\phi_i \\ Q_{\phi'_i}(s,a) = \sum_{j=1}^{N} f_j(s,a)\phi'_{i,j} = f^T(s,a)\phi'_i \end{array}. \tag{4.19}$$

Their associated ODE under the estimated expert policy π_ω^d is:

$$\dot{\phi}_i = \frac{\alpha_c}{|B|} \sum_{(\bar{s}_k, \hat{a}_k^d, \bar{s}_{k+1}) \in B} f(\bar{s}_k, \hat{a}_k) \Delta_{i,k}. \tag{4.20}$$

The ODE (4.20) is expanded in matrix form as:

$$\begin{array}{c} \dot{\phi}_i = \frac{\alpha_c}{|B|} \sum_{(\bar{s}_k, \hat{a}_k^d, \bar{s}_{k+1}) \in B} f_{\bar{s}_k}^T (\mathcal{R}(\bar{s}_k, \hat{a}_k^d) \\ + \gamma \min_{l=1,2} g_{\bar{s}_{k+1}} \phi_l' - f_{\bar{s}_k} \phi_i) \end{array}, \tag{4.21}$$

where

$$f_{\bar{s}_k} = f^T(\bar{s}_k, \hat{a}_k^d), \quad g_{\bar{s}_k} = f^T(\bar{s}_k, \tilde{a}_k).$$

The convergence of the algorithm is analyzed in terms of the expanded ODE (4.21) following Theorem 17 of [88]. This approach provides a theoretical foundation for assessing the effectiveness of the algorithm and its potential to reach convergence. The critical aspect of this analysis is summarized in the following theorem:

Theorem 5. *In the context of biased exploration RLED, the algorithm achieves convergence with probability 1, provided that the learning rate $\alpha_{c,k}$ meets specific criteria. The conditions for the learning rate are as follows:*

$$\sum_k \alpha_{c,k} = \infty \qquad \sum_k \alpha_{c,k}^2 < \infty. \tag{4.22}$$

Under these stipulations, the learning process described in (4.6) *is guaranteed to converge.*

Proof. Let $\phi_i^1(k)$ and $\phi_i^2(k)$ be two trajectories of the ODE starting at different initial conditions, and $\tilde{\phi}_i(k) = \phi_i^1(k) - \phi_i^2(k)$. Then, from (4.21) yield:

$$\begin{aligned} \frac{d}{dk}\|\tilde{\phi}_i\|_2^2 =& \frac{2\alpha_c}{|B|} \tilde{\phi}_i^T \sum_{(\bar{s}_k, \hat{a}_k^d, \bar{s}_{k+1}) \in B} f_{\bar{s}_k}^T (\mathcal{R}(\bar{s}_k, \hat{a}_k^d) \\ &+ \gamma \min_{l=1,2} g_{\bar{s}_{k+1}} \phi_l' - f_{\bar{s}_k} \phi_i^1 - \mathcal{R}(\bar{s}_k, \hat{a}_k^d) \\ &- \gamma \min_{l=1,2} g_{\bar{s}_{k+1}} \phi_l' + f_{\bar{s}_k} \phi_i^2), \end{aligned}$$

simplifying as follows:

$$\frac{d}{dk}\|\tilde{\phi}_i\|_2^2 = \frac{2\alpha_c}{|B|} \tilde{\phi}_i^T \sum_{(\bar{s}_k, \hat{a}_k^d, \bar{s}_{k+1}) \in B} -f_{\bar{s}_k}^T f_{\bar{s}_k} (\phi_i^1 - \phi_i^2),$$

and now defining a matrix of the form as:

$$\Omega = \sum_{(\bar{s}_k, \hat{a}_k^d, \bar{s}_{k+1}) \in B} f(\bar{s}_k, \hat{a}_k^d) f^T(\bar{s}_k, \hat{a}_k^d).$$

Finally yields:

$$\frac{d}{dk}\|\tilde{\phi}_i\|_2^2 = -\frac{2\alpha_c}{|B|}\tilde{\phi}_i^T \Omega \tilde{\phi}_i. \tag{4.23}$$

Here it is concluded that $\frac{d}{dk}\|\tilde{\phi}_i\|_2^2 < 0$. For common RL methods, it is usual to select a constant learning rate

$$\alpha_k = \alpha, \quad 0 < \alpha \leq 1.$$

If it is taken as:

$$\alpha_k = \frac{\eta}{1 + \frac{1}{\beta}k}, \quad 0 < \eta \leq 1, \quad \beta >> 1,$$

where η is a constant. Since β is very big, for finite time k, the learning rate $\alpha_k \approx \eta$. Considering

$$\begin{array}{c} \sum_{k=1}^{\infty} \frac{\eta}{1+\frac{1}{\beta}k} = \infty \\ \sum_{k=1}^{\infty} \left(\frac{\eta}{1+\frac{1}{\beta}k}\right)^2 = \eta^2\beta^2\psi\left(\beta, 1\right) - \eta^2 < \infty \end{array},$$

where $\psi\left(\beta, 1\right)$ is the Digamma function. Condition (4.22) is satisfied. $\Delta_{i,k}$ converges to zero, and Q_k converges to Q^* w.p.1. Therefore, the ODE is globally asymptotically stable and (4.6) converges w.p.1. □

Condition $\sum_k \alpha_{c,k}^2 < \infty$ in (4.22) requires that all state-action pairs be visited for the algorithm to converge effectively. This requirement is critical in the context of the convergence analysis for the biased exploration RLED. It is important to note that this analysis focuses primarily on the presence and implications of the estimated expert policy. In this context, the role and influence of the actor policy are not considered within the scope of this convergence study.

This emphasis on the estimated expert policy highlights its importance in the learning process, especially in guiding the exploration and decision-making aspects of the algorithm. However, it also implies that the dynamics and potential impact of the actor's policy on the overall convergence of the algorithm remain unaddressed in this specific analysis. Understanding the limitations of this approach is essential to accurately interpret the results and implications of the convergence study within the wider framework of the RLED.

4.1.3 Robot Control

Numerous studies have implemented traditional behavior learning techniques, employing IL [19] for redundant robot manipulators and RL [89, 90, 91] for their non-redundant counterparts, aiming to derive straightforward control policies. However, IL often encounters generalization challenges, whereas RL is hampered by inefficiencies in sample utilization and the complexity of designing effective reward functions. Directly learning control policies that mimic human behavior, as explored in [92, 93, 94], presents a viable solution to these issues.

The implementation of biased exploration RLED emerges as a groundbreaking approach in the control of redundant robot manipulators. These robots, distinguished by their additional degrees of freedom, pose distinct challenges that demand innovative control strategies. The integration of biased exploration RLED harnesses the power of expert knowledge along with adaptive learning techniques to effectively manage the intricate control dynamics inherent to these advanced robotic systems.

Addressing the control of redundant robot manipulators illuminates the critical need for precision and adaptability within robotic control mechanisms. Operating in multifaceted environments, these systems often encounter scenarios in which conventional control methods are inadequate. Through biased exploration RLED, a path is forged to significantly boost robots' autonomy and operational efficiency, establishing a comprehensive framework for optimal control amidst variable and unpredictable conditions. This approach not only utilizes valuable insights from expert demonstrations, but also exhibits remarkable adaptability to novel situations, thereby ensuring that robots are equipped to undertake an extensive array of tasks with unparalleled flexibility and accuracy.

A robotic system's dynamics [95] can typically be represented by the equation:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau, \tag{4.24}$$

where $M(q)$ denotes the positive definite inertia matrix, $C(q,\dot{q})$ the Coriolis matrix, and $G(q)$ the gravitational torques vector, with τ representing the applied torque. The variables $q, \dot{q}$, and $\ddot{q}$, respectively, correspond to the joint angles' position, velocity, and acceleration, all within $\mathbb{R}^n$.

Expert knowledge typically operates within the task-space (the world frame $x-y-z$), necessitating a translation to the joint-space, denoted by q, for robot control. The crucial relationship between the joint angles $q \in \mathbb{R}^n$ and the task-space $X \in \mathbb{R}^m$ is described as:

$$X = f(q), \quad q = f^{-1}(X), \tag{4.25}$$

where $f(\cdot) : \mathbb{R}^n \to \mathbb{R}^m$ represents the forward kinematics function, and $f^{-1}(\cdot) : \mathbb{R}^m \to \mathbb{R}^n$ signifies the inverse kinematics.

A redundant robot manipulator is characterized by having a task-space dimension that is smaller than its joint-space dimension ($n > m$). In such cases, the inverse kinematic function $f^{-1}(\cdot)$ is not uniquely defined, leading to multiple joint-space configurations q corresponding to a single task-space position X. Consequently, a direct inverse kinematics solution for redundant robots is not straightforwardly applicable.

The dynamics of velocity kinematics are captured by the equation:

$$\dot{x} = \frac{\partial f(q)}{\partial q}\dot{q} = J(q)\dot{q}, \tag{4.26}$$

where $J(q) \in \mathbb{R}^{m \times n}$ represents the Jacobian matrix, facilitating the translation from joint velocities to task-space velocities.

The relationship defining the change in task-space position, ΔX, in response to a change in joint-space position, Δq, is described as:

$$\Delta X = J(q)\Delta q. \tag{4.27}$$

For redundant robot manipulators, equation (4.27) presents an under-determined system, implying the existence of an infinite number of solutions.

Addressing the challenge of learning the inverse kinematics of a redundant robot manipulator involves establishing a mapping from $(\Delta X, q)$ to Δq, where ΔX denotes the change in task-space coordinates and Δq represents the corresponding change in joint-space coordinates [96]. This learning process facilitates understanding of how adjustments in the robot's joints affect its position in the task space.

The MDP for controlling redundant robot manipulators through biased exploration RLED is formalized as a 4-tuple:

$$\mathcal{M} = \langle \mathcal{X}, \Delta\mathcal{Q}, \mathcal{R}, f \rangle,$$

where $\mathcal{X} \in \mathbb{R}^{m+n}$ comprises the fully observable state set, denoting displacements in task-space alongside their corresponding joint positions. The set of admissible control actions, $\Delta\mathcal{Q} \in \mathbb{R}^n$, corresponds to changes in joint positions. The function $f : \mathcal{X} \times \Delta\mathcal{Q} \to \mathcal{X}$ is the state transition function, determined by forward kinematics, and $\mathcal{R} : \mathcal{X} \times \Delta\mathcal{Q} \to \mathbb{R}$ defines the scalar reward function, guiding the learning towards optimal control actions.

Each state $x \in \mathcal{X}$ is represented as:

$$x = \begin{bmatrix} \Delta X \\ q \end{bmatrix},$$

where ΔX is the difference between the target position X_T in the task-space and the current position X, with $X_T \in \mathbb{R}^m$. In response, the agent executes control actions that correspond to minor joint displacements $\Delta q \in \Delta\mathcal{Q}$. The outcome of applying the forward kinematics function $f(x_t, \Delta q_t)$ is the subsequent state $x_{t+1} \in \mathcal{X}$, delineated as:

$$x_{t+1} = f(q_t + \Delta q_t). \tag{4.28}$$

Following this state transition, the agent is awarded a scalar reward $\mathcal{R}(x_t)$ that assesses the immediate impact of the displacement Δq_t, effectively evaluating the movement from x_t to x_{t+1}.

In the context of teleoperation demonstrations, where robot maneuvers are exhibited through the robot itself, as discussed in the recent literature [10], expert-demonstrated behavior is encapsulated in the 3-tuple as:

$$\mathcal{D} = \langle \mathcal{X}^d, \Delta Q^d, f \rangle,$$

comprising $\mathcal{X}^d$ as the subset of states experienced by the expert, $\Delta Q^d \in \Delta Q$ as the subset of permissible demonstrated control actions, and f the state transition function. At each timestep t, the demonstrated state-action pairs $(x^d, \Delta q^d)$ are archived in the demonstration set ζ. The aim is to devise a deterministic policy $\pi_\theta(x_t) = \Delta q_t$ that addresses the inverse kinematic challenge, essentially resolving 4.27 for Δq. The policy π_θ aims to align closely with the expert policy π^d.

The schematic for controlling redundant robot manipulators using biased RLED exploration is depicted in Figure 4.1. This framework incorporates "prior demonstrations" to facilitate biased exploration. This integrative approach underpins the effective use of RLED in achieving precise control over redundant robot manipulators, using both prior demonstrations and real-time feedback to dynamically refine control strategies.

Task Design

The task involves navigating a robot manipulator's end-effector to a target position within the task-space, illustrated as a red sphere in Figure 4.2. Initially, the manipulator starts from

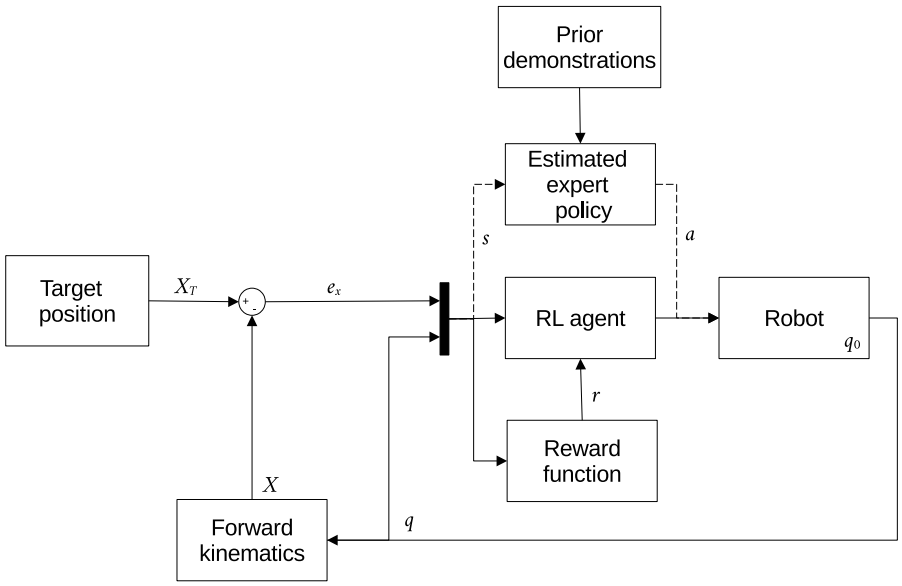

Figure 4.1: Redundant robot control diagram with biased exploration RLED.

a starting point denoted by a green sphere. The process unfolds in several key stages:

1. The manipulator begins its journey from the initial position, ensuring a methodical approach to reaching the designated target.

2. Movement is constrained to small joint displacements, adhering to the dynamics outlined in (4.28).

3. A new target position is generated upon the end-effector reaching the current target with a positional error that falls below a predetermined threshold. Consequently, the recent target transitions into the new starting point for the subsequent trajectory.

4. An episode concludes once the manipulator successfully reaches a predefined count of target positions, marking the completion of the task cycle.

For the purpose of collecting demonstration trajectories through teleoperation of the robot manipulator, the demonstration set ζ is bifurcated into two distinct collections: the demonstration points dataset and the demonstration trajectories dataset. The demonstration points dataset $\zeta^a = (X_i^d, q_i^d)$ encompasses a series of discrete time-independent points that reflect similar operational patterns, where X^d represents the task-space position and q^d the

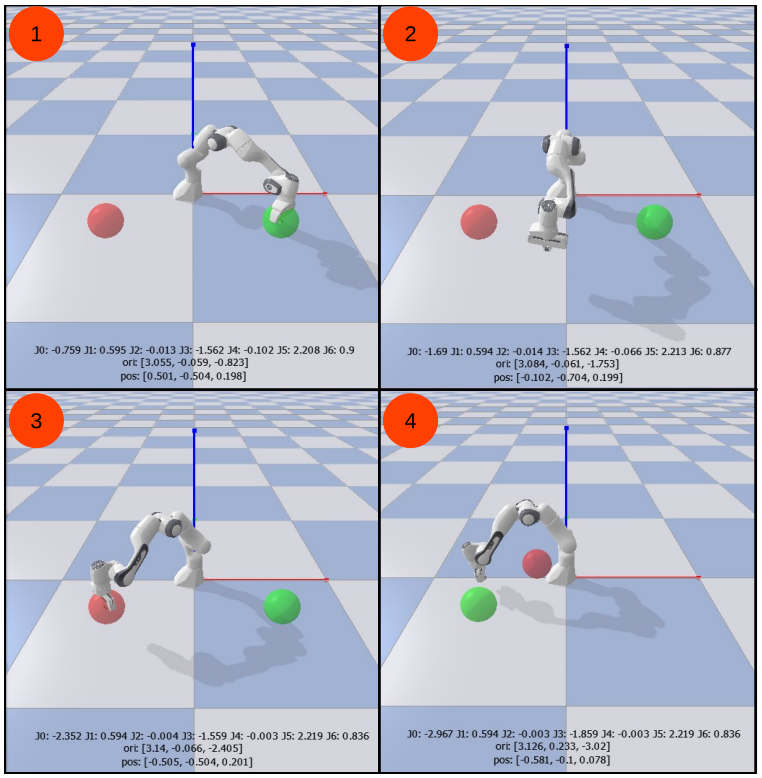

Figure 4.2: Task sequence in the environment, divided into four steps: (1) Starting from the initial position (green), (2) Moving through small joint adjustments, (3) Generation of a new target upon successful reach, and (4) Conclusion of an episode after achieving a set number of targets

corresponding joint-space position for each demonstration. In contrast, the demonstration trajectories dataset $\zeta_j^b = (x_{j,t}^d, \Delta q_{j,t}^d)$ captures temporal sequences of the demonstrated state-action pairs, offering dynamic insight into manipulative maneuvers.

To gather demonstrations, the following steps are undertaken:

1. Teleoperate the robot manipulator across various locations to exhibit desired poses, recording both the task-space position X^d and its corresponding joint-space position q_i^d within ζ^a.

2. To generate demonstration trajectories, begin by selecting an initial point $(X_0^d, q_0^d) \sim \zeta^a$ and a target point $(X_T^d, q_T^d) \sim \zeta^a$. Subsequently, execute a demonstration action embodying the desired behavior, quantified as follows:

$$e_{qmax} = \max_k |q_{k,T}^d - q_{k,t}^d|$$

$$\Delta q_{k,t}^{d} = \frac{clip(e_{qmax}, 0, \Delta q_{max})(q_{k,T}^{d} - q_{k,t}^{d})}{e_{qmax}},$$

where k indexes each robot joint vector element and Δq_{max} stipulates the maximum permissible joint alteration. This systematic approach to demonstration collection and trajectory generation facilitates a structured framework for replicating and learning from expert maneuvers, thereby informing the development of robust control strategies within the biased exploration RLED paradigm.

Training the estimated expert policy $\hat{\pi}_{\omega}^{d}$ using the dataset of demonstration trajectories ζ^{b} adheres to the procedure described in (4.10).

To assess the efficacy of biased exploration RLED, two distinct reward functions are proposed. The dense reward function is defined as:

$$R(s) = \begin{cases} u + ||e_x||_2^2 & \longrightarrow ||e_x||_2^2 < \epsilon_x \\ ||e_x||_2^2 & \longrightarrow ||e_x||_2^2 \geq \epsilon_x \end{cases},$$

and the sparse reward function is given by:

$$R(s) = \begin{cases} u & \longrightarrow ||e_x||_2^2 < \epsilon_x \\ 0 & \longrightarrow ||e_x||_2^2 \geq \epsilon_x \end{cases},$$

where $||e_x||2^2$ represents the squared error position, ϵx is a predefined threshold, and u is a positive reward. When the error position falls below ϵ_x, the agent receives an immediate positive reward, prompting the generation of the next target position. Learning episodes conclude once the agent achieves a predetermined count of target positions within the task-space or exceeds a specified number of time steps.

Episode initialization plays a pivotal role in the RL process, ensuring proximity to demonstrated states at each time step of an episode through the reuse of the estimated expert policy. Episodes commence from a state at the beginning of a demonstration trajectory. The initial joint-space position q_0^d and the target task-space position X_T^d are selected from ζ^a.

This initialization strategy simplifies task resolution under the sparse reward framework, especially when using the estimated expert policy. Consequently, the agent is presented with a less challenging exploration problem, significantly increasing the likelihood of encountering positive rewards.

Throughout the learning episodes, the estimated expert policy is consistently employed to maintain proximity to states akin to those demonstrated, thereby guiding the exploration

process. Before each episode, the exploration bias factor Φ is calculated in accordance with (4.11) and (4.12), optimizing the exploration strategy by balancing between adhering to demonstrated behaviors and exploring new state-action pairs.

Settings

The evaluation of the proposed method was carried out in a simulated environment using the Pybullet simulator [97], integrated with the OpenAI Gym framework [98]. The Panda 7-DoF robot manipulator served as the robot model for these experiments. Each episode was limited to a maximum of 999 time steps. The demonstration points dataset, ζ^a, contained 1572 paired elements, while the demonstration trajectories dataset, ζ^b, comprised 142199809 paired elements. The maximum allowable joint change, Δq_{max}, was fixed at 0.04 radians. The target positions for the trials were generated using the parametric functions of Bernoulli's Lemniscate, with the precision threshold to reach a point established at $\epsilon_x = 3 \times 10^{-3}$. The orientation of the end-effector was not considered in these experiments. Given that $n = 7$ and $m = 3$, this configuration resulted in a continuous state-space 10-dimensional coupled with a continuous action-space 7-dimensional.

The agent was developed using Tensorflow [99] in the Python programming environment [100]. Tensorflow's capabilities for model weight management were utilized for saving and loading network parameters. Input states and actions were normalized prior to processing by the neural network. The network architecture for training the estimated expert policy included an input layer with 10 units, followed by two hidden layers containing 400 and 300 units, respectively, each employing a leaky ReLU activation function. The output layer consisted of 7 units activated by a hyperbolic tangent function. The training process used a learning rate of 1×10^{-3}, with mini-batch processing involving 256 samples.

The agent and critic networks were structured with two hidden layers, each comprising 400 and 300 units. The discount factor was set at $\gamma = 0.99$. The learning rates for the actor and critic networks were matched at 1×10^{-3}. The target action noise was adjusted with $\sigma = 0.2$ and the clipping at $c = 0.5$. Updates to the actor network were scheduled every 2 steps. Learning started after accumulating 1×10^4 samples in the replay buffer, with network updates following every 1×10^3 samples collected. During updates, 1×10^3 gradient steps were performed. The replay buffer was allocated a capacity for 1×10^6 samples, with mini-batch updates drawing on 100 random samples. The update of target networks used a soft update rate of 5×10^{-3}. The environmental reward factor was established at $u = 1000$, with the objective of guiding the end effector to 15 target positions within the task space,

marking the criterion for successful completion of the task.

Results

The initial phase of applying the proposed method involves collecting data from expert demonstrations showcasing the desired behavior. Following data collection, the next step involves training an estimated expert policy using supervised learning techniques. The performance of this policy is crucial for achieving the desired outcomes within the task-space. The learning curve obtained from supervised learning, prior to participating in the RL process, is shown in Figure 4.3. The aim is to evaluate the behavior of the estimated expert policy, especially in scenarios involving points that are not encountered during training. This evaluation helps identify the limitations of local generalization commonly observed in IL policies. Although the estimated expert policy demonstrates the ability to reach certain new points smoothly, its effectiveness decreases as the target positions diverge from those seen during training, as illustrated in Figure 4.4.

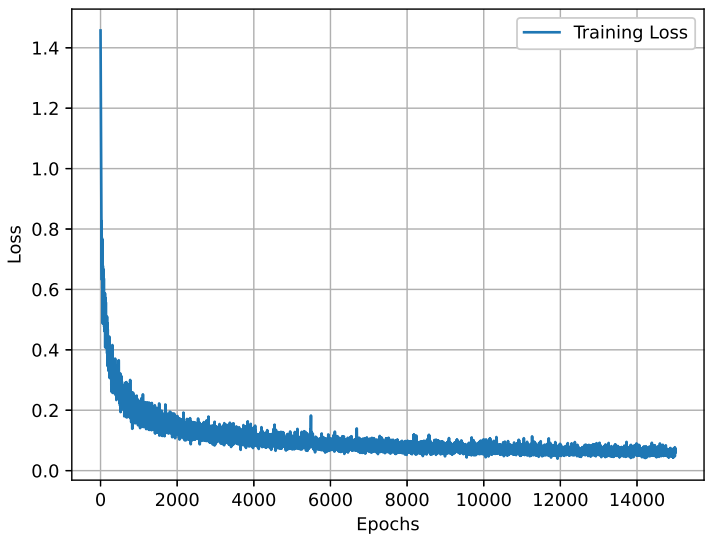

Figure 4.3: Training curve for the estimated expert policy using supervised learning, preceding the RL process.

The role of the estimated expert policy within the proposed framework is to facilitate the RL process. The subsequent analysis investigates the influence of the estimated expert policy on accelerating the RL process compared to scenarios where it is not used. The dynamics of training with a dense reward function is shown in Figure 4.5, with sparse reward

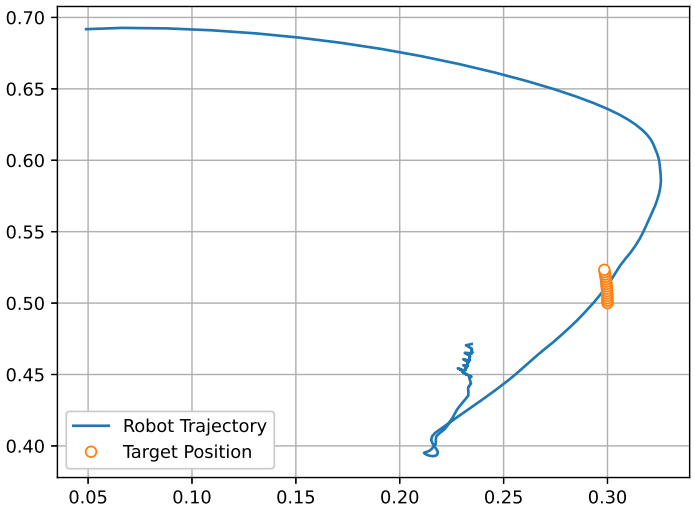

Figure 4.4: Testing the estimated expert policy in the task-space with unseen target positions.

function training depicted in Figure 4.6. Despite the limited local generalization of the estimated expert policy, its application proves beneficial in accelerating the RL process through exploration biasing, especially in the presence of dense reward functions. Implementing the estimated expert policy with a PPR of 0.1 significantly enhances the learning pace compared to the TD3 baseline method, which corresponds to a PPR of 0.0.

The effectiveness of the estimated expert policy becomes more pronounced when faced with sparse rewards. In scenarios where pure TD3 (PPR = 0.0) is applied, the agent struggles to learn an effective policy due to the low probability of encountering positive rewards. Incorporating the estimated expert policy with a PPR of 0.1 emerges as the most efficient strategy for policy learning, outperforming configurations with PPR values of 0.2 and 0.3.

Post-RLED process, the refined policy's performance is scrutinized, particularly in its ability to reach previously unseen points within the task-space, thereby validating its enhanced generalization capabilities. Observations reveal that the refined policy maintains smooth operation at a distance from target positions but exhibits less stability as it approaches them, with a noticeable increase in abrupt behavior for unseen targets (Figures 4.7 and 4.8). The general behavior of the refined policy on multiple unseen targets is documented in Figure 4.9.

The estimated expert policy requires a minimal amount of data for training, in contrast to the more extensive data demands of pure TD3. The biased exploration RLED achieves

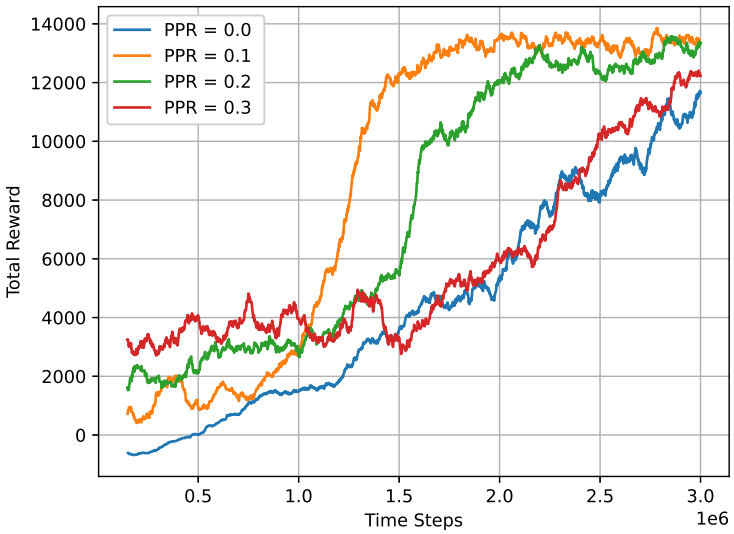

Figure 4.5: RLED training curves under various PPR factors with a dense reward function.

similar efficiency with approximately half the data required by TD3, highlighting its sample efficiency and faster training potential, provided the demonstration trajectories encompass a broad spectrum of relevant state-action pairs.

Despite the limited generalization of the estimated expert policy, akin to typical IL policies, it is proving instrumental in guiding the RL process by influencing exploration. The biased exploration RLED policy, in contrast, exhibits robust generalization capabilities, similar to standard RL policies, indicating a notable improvement over the estimated expert policy. This method demonstrates versatility in handling both sparse and dense rewards, with sparse rewards being simpler to design, thus positioning biased exploration RLED as a viable strategy for circumventing complex reward function designs and achieving satisfactory performance levels. However, while the estimated expert policy predominantly promotes safe exploration, the standard exploration component within RLED may expose the agent to potentially hazardous states and actions, underscoring a limitation in the method's safety assurances.

Human demonstrations provide a foundational solution approach, regardless of the reward function's density. However, these solutions may not always represent the optimal strategy due to the potential imperfections in human execution. Biased exploration RLED enables the agent to scrutinize these demonstrations alongside its exploratory findings, fostering a refinement of sub optimal human behaviors into more optimal agent strategies.

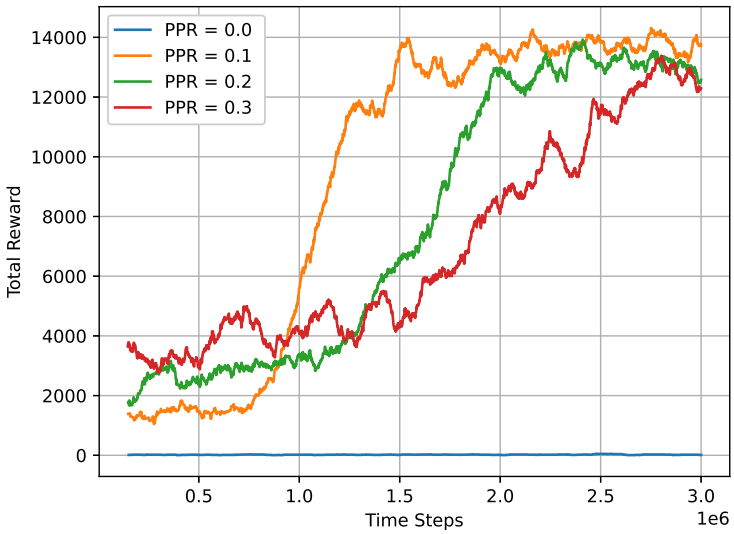

Figure 4.6: RLED training curves under various PPR factors with a sparse reward function.

Although comprehensive, this study is conducted exclusively in simulation, potentially overlooking real-world factors that significantly impact system dynamics. The investigation operates under assumptions of full-state observability, absence of delays or disturbances in the learning process, and stationarity in the environment's dynamic transition and reward functions. The generalizability of the proposed method extends beyond robotics, adaptable to any domain or application involving demonstrable state-action pairs, and the outlined pseudocode remains applicable across varied contexts.

4.2 Safe Exploration Reinforcement Learning from Expert Demonstrations

Safe exploration in RLED is conceptualized through the framework of two distinct MDPs. Initially, a constrained MDP is introduced as per [101], structured as a 5-tuple:

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \Psi \rangle,$$

where $\mathcal{S} \in \mathbb{R}^{\Lambda}$ represents the state space, $\mathcal{A} \in \mathbb{R}^{\Omega}$ denotes the set of admissible control actions, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ defines the deterministic transition function, and Ψ is the set of safety constraints. Within this framework, the agent

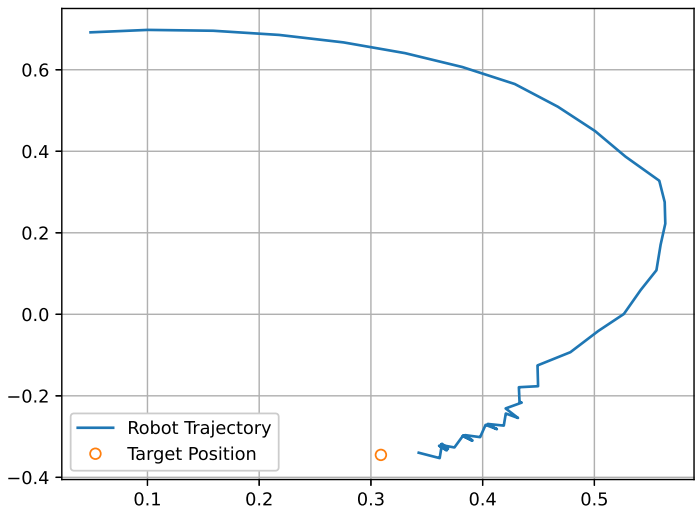

Figure 4.7: Testing the RLED refined policy with a seen target position in the task-space.

selects a control action $a \in \mathcal{A}$ in each state $s \in \mathcal{S}$, subsequently receives a reward $\mathcal{R}(s, a)$, and transitions to the next state s' as dictated by $\mathcal{T}(s, a)$. According to Garcia and Fernandez [6], every state-action pair must adhere to each safety constraint $\psi_h \in \Psi$, expressed as:

$$\psi_h = \{u_h(s, a) \leq \xi_h\}, \tag{4.29}$$

where u represents the safety function and ξ the safety threshold.

The second MDP delineates the expert behavior and is constituted as a 3-tuple:

$$\mathcal{D} = \langle \mathcal{C}, \mathcal{O}, \mathcal{L} \rangle,$$

with $\mathcal{C} \in \mathbb{R}^{\Upsilon}$ being the set of states experienced by the expert, $\mathcal{O} \in \mathbb{R}^{\Gamma}$ the set of control actions executed by the expert, and $\mathcal{L} : \mathcal{C} \times \mathcal{O} \to \mathcal{C}$ the deterministic transition function guiding the progression to subsequent states. In this context, the expert performs a control action $o \in \mathcal{O}$ within each state $c \in \mathcal{C}$, facilitating the transition to the next state c' according to $\mathcal{L}(c, o)$.

Recognizing the morphological distinctions between the expert and the apprentice requires the incorporation of a recording mapping $g_{rec} : \mathcal{C} \times \mathcal{O} \to \mathcal{S}^d \times \mathcal{A}^d$. This mapping effectively translates expert behaviors into a format applicable to the apprentice's context, where $\mathcal{S}^d \subset \mathcal{S}$ represents the subset of states demonstrated by the expert, and $\mathcal{A}^d \subset \mathcal{A}$ encompasses the

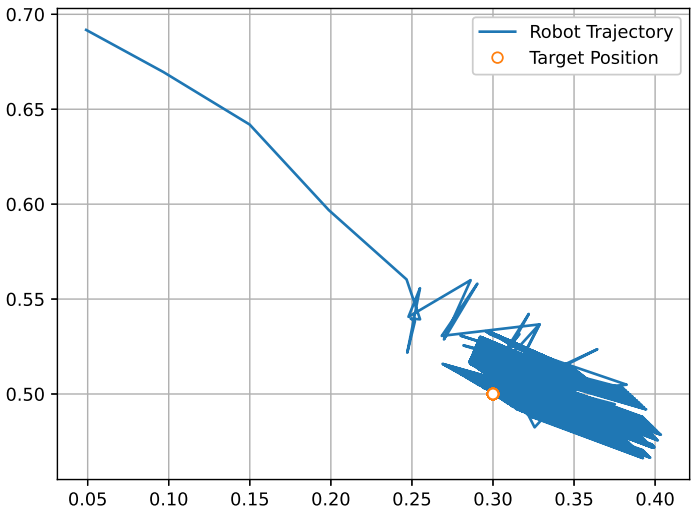

Figure 4.8: Testing the RLED refined policy with an unseen target position in the task-space.

subset of control actions shown during demonstrations. The expert control policy is thus articulated as:

$$\pi^d(s_t^d) = a_t^d, \tag{4.30}$$

with $s^d \in \mathcal{S}^d$ identifying the demonstrated state, and $a^d \in \mathcal{A}^d$ specifying the demonstrated action at any given time step.

To ensure safety within the learning environment, the concept of safe states $\mathcal{S}^u \subset \mathcal{S}$ and safe actions $\mathcal{A}^u \subset \mathcal{A}$ is introduced. Under the following assumption, the framework presupposes the expert's behavior as inherently safe. This premise is grounded in the notion that every demonstrated state agrees with the safety criteria.

Assumption 1. *The expert behavior is safe, such that for every demonstrated state:*

$$\mathcal{S}^d = \{s^d \in \mathcal{S}^u | \psi_h, a\},$$

and similarly, each demonstrated action adheres to these safety constraints:

$$\mathcal{A}^d = \{a^d \in \mathcal{A}^u | \psi_h, s\}.$$

This assumption is crucial for the integrity of the learning process, ensuring that the apprentice's exploration and learning phases are conducted within the bounds of predefined

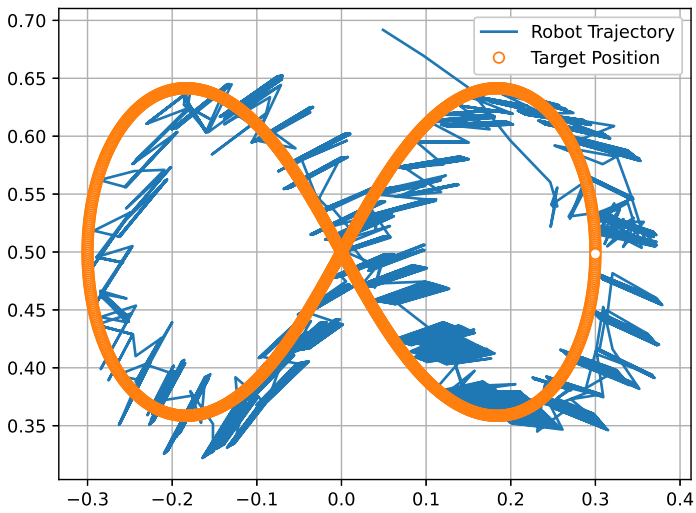

Figure 4.9: Testing the RLED refined policy with several unseen target positions.

safety measures, thereby mitigating risks and promoting a secure learning trajectory.

4.2.1 Safe Exploration with Uncertainty Estimation

The goal is to devise a policy $\pi : \mathcal{S}^u \to \mathcal{A}^u$ that not only aligns with the expert policy π^d, but also exceeds its performance by maximizing the return of Eq (3.2) implemented in Section 2. This approach seeks to enhance the efficacy of the policy while adhering to the principles of safety and optimality within the designated subsets of states $\mathcal{S}^u$ and actions $\mathcal{A}^u$.

Theorem 6. *It is posited that the expert policy, while proficient, is not the pinnacle of optimality. A superior and locally optimal policy π^- exists within the larger sets $\mathcal{S}$ and $\mathcal{A}$, which, within the confines of $\mathcal{S}^u$ and $\mathcal{A}^u$, is considered the globally optimal policy. This relationship is characterized by the inequality:*

$$Q(s_t, \pi^d(s_t)) \leq Q(s_t, \pi^-(s_t)) \tag{4.31}$$

To navigate the challenges of identifying solutions for the local optimal policy and state-action value function, particularly in high-dimensional spaces with continuous actions, and continuing the method implemented in Section 3 TD3 [35] is employed. The actor is tasked with the estimation of the policy:

$$\pi_\theta(s_t) \approx \pi^-(s_t),$$

, while the critic is responsible for estimating the state-action value function. To mitigate the overestimation of state-action values, dual critics are used:

$$Q_{\phi_1}(s_t, a_t) \approx Q^{-}(s_t, a_t)$$

and

$$Q_{\phi_2}(s_t, a_t) \approx Q^{-}(s_t, a_t),$$

, with the lesser of the two estimations being incorporated into the calculation of the mean squared Bellman error (MSBE) target. The stability of the learning process is further enhanced by the integration of auxiliary neural network function approximators for both the state-action value function and the policy, as recommended in [34, 35]. These auxiliary networks are represented as:

$$\pi_{\theta'}(s_t) \approx \pi_{\theta}(s_t),$$

$$Q_{\phi' 1}(s_t, a_t) \approx Q_{\phi_1}(s_t, a_t),$$

and

$$Q_{\phi' 2}(s_t, a_t) \approx Q_{\phi_2}(s_t, a_t).$$

The MSBE plays a pivotal role in the reinforcement learning process, guiding the update mechanism with its target defined as:

$$y = \mathcal{R}(s_k, a_k) + \gamma \min_{i=1,2} Q_{\phi'_i}(s_{k+1}, \tilde{a}), \tag{4.32}$$

where $\tilde{a} = \pi_{\theta'}(s_{k+1}) + \bar{\epsilon}$ introduces a noise element to the action, with $\bar{\epsilon}$ following a clipped normal distribution $\text{clip}(\mathcal{N}(0, \sigma), -c, c)$, and σ and c serving as hyperparameters. This formulation is designed to temper the estimation with a degree of exploratory noise, fostering robustness in the learning process.

The optimization of primary critics hinges on minimizing the loss functions:

$$J_{c_1} = \frac{1}{|B|} \sum_{(s_k, a_k, s_{k+1}) \in B} (y - Q_{\phi_1}(s_k, a_k))^2$$
$$J_{c_2} = \frac{1}{|B|} \sum_{(s_k, a_k, s_{k+1}) \in B} (y - Q_{\phi_2}(s_k, a_k))^2,$$

with B denoting a mini-batch of samples drawn from the replay buffer. Concurrently, the

primary actor's policy is refined by maximizing the loss function:

$$J_a = \frac{1}{|B|} \sum_{s_k \in B} Q_{\phi_1}(s_k, \pi_\theta(s_k)). \tag{4.33}$$

The update equations for the primary critics are articulated as:

$$\phi_{i,k+1} = \phi_{i,k} - \alpha_c \nabla_{\phi_i} J_{c_i}, \tag{4.34}$$

with a similar principle applied to the secondary critics, ensuring their parameters are gradually adjusted towards the primary critics' newly updated values:

$$\phi'_{i,k+1} = \tau \phi_{i,k+1} + (1 - \tau)\phi'_{i,k}, \tag{4.35}$$

where τ represents the rate of mixing between the primary and secondary critics, enhancing stability in the learning dynamics.

The policy updates for the primary actor follow the law:

$$\theta_{k+1} = \theta_k + \alpha_a \nabla_\theta J_a, \tag{4.36}$$

with the secondary actor's parameters being updated in a manner that mirrors the approach taken for the secondary critics:

$$\theta'_{k+1} = \tau \theta_{k+1} + (1 - \tau)\theta'_k. \tag{4.37}$$

Action Uncertainty Estimation

Monte Carlo (MC) dropout is used to establish a safety constraint that meets (4.29). Introduced by Gal and Ghahramani [102], MC Dropout serves as a technique for estimating uncertainty within standard neural network models.

The output of the actor network, $\hat{a}$, with its various layers L and parameters $\theta = \theta_1, ..., \theta_L$ as weight matrices, aims to find the local optimal action a^* for the state s^*. For a given set of states $\bar{S}t = st, 1, ..., s_{t,\alpha}$ and actions $\bar{A}t = at, 1, ..., a_{t,\alpha}$, the predictive distribution is expressed as follows:

$$p(a_t^*|s_t^*, \bar{S}_t, \bar{A}_t) = \int p(a_t^*|s_t^*, \theta) p(\theta|\bar{S}_t, \bar{A}_t) d\theta,$$

where $p(a^*t|s^*t, \theta)$ denotes the model's likelihood, and $p(\theta|\bar{S}t, \bar{A}t)$ symbolizes the posterior

over the weights. This distribution suggests a mean and variance that predict uncertainty. However, the posterior distribution over weights is impractical for analytical evaluation. An approximating variational distribution $q(\theta)$, inspired by Gaussian processes, is proposed to approximate $p(\theta|\bar{S}t, \bar{A}t)$ as closely as possible. The optimization thus pivots on minimizing the Kullback-Leibler divergence:

$$KL(q(\theta) \mid p(\theta|\bar{S}_t, \bar{A}_t))$$

yielding the approximate predictive distribution:

$$q(a_t^*|s_t^*) = \int p(a_t^*|s_t^*, \theta)q(\theta)d\theta.$$

Per [102], $q(\theta)$ is selected as a distribution over matrices with columns nullified randomly in line with a Bernoulli distribution:

$$\theta_i = M_i diag([z_{ij}]_{j=1}^{K_i}),$$

where $z_{ij} \sim Bernoulli(p_i)$ for layers $i = 1, ..., L$ and element $j = 1, ..., K_{i-1}$, with $\theta \in \mathbf{R}^{K_i \times K_{i-1}}$, p_i indicating the dropout probability, and M_i the matrix of variational parameters. Drawing N samples from this distribution results in $\theta_1^n, ..., \theta_L^n n = 1^N$. Consequently, the predictive mean action is estimated as follows:

$$E_{q(a^*|s^*)(a^*)} \approx \frac{1}{N} \sum_{n=1}^{N} \hat{a}^*(s^*, \theta_1^n, ..., \theta_L^n) = a_{MC}, \tag{4.38}$$

where $\hat{a}^*$ is the mapping of s^* by the actor network, equating to conducting N stochastic forward passes through the actor network and averaging the outcomes. The actor network, equipped with dropout layers, yields diverse outputs for the same state input, enabling uncertainty analysis through variance, entropy, or mutual information metrics. The Quartile Coefficient of Dispersion is chosen for uncertainty quantification:

$$u_t = \frac{1}{\Omega} \sum_{\omega=1}^{\Omega} \left| \frac{q_3(\tilde{A}_t^*) - q_1(\tilde{A}_t^*)}{q_3(\tilde{A}_t^*) + q_1(\tilde{A}_t^*)} \right|, \tag{4.39}$$

where Ω denotes the action space dimension, $q3$ and q_1 the third and first quartiles respectively, for the set of predictive actions $\tilde{A_t}^* = \{\hat{a}_{t,1}^*, ..., \hat{a}_{t,N}^*\}$ with N stochastic forward passes over the actor network. This formulation allows for a safety constraint for exploration to be

established based on action uncertainty:

$$a_t^+ = \begin{cases} \pi^d(s_t) \longrightarrow u_t \geq \xi \\ a_{MC} + \epsilon \longrightarrow u_t < \xi, \end{cases} \tag{4.40}$$

where a^+ signifies the action executed in the environment, and $\epsilon \sim \mathcal{N}(0, \delta)$ introduces the exploration noise.

Aligning with the safety levels described by Brunke et al. [103], the estimation of uncertainty by (4.39) fulfills the first and third safety criteria, offering a methodological approach to integrate safety within the reinforcement learning paradigm, thereby ensuring that the actions taken are within a defined threshold of certainty.

4.2.2 Convergence Analysis

The convergence of the MC RLED algorithm is rigorously evaluated using an approach that mirrors the analysis performed in Section 3 for biased exploration RLED. This parallel analysis leverages the theoretical framework established in [104, 105, 86, 88], focusing on the expanded ODE to scrutinize the algorithm's capacity for achieving convergence. The foundational principles that underpin this convergence analysis are encapsulated in the following theorem, explicitly aligning the MC RLED algorithm with the methodologies outlined in Section 3:

Theorem 7. *For the MC RLED, the convergence to the optimal policy is guaranteed with probability 1, contingent on the adherence to the specific learning rate $\alpha_{c,k}$ conditions. The requisite conditions for the learning rate are as follows:*

$$\sum_k \alpha_{c,k} = \infty \quad \textit{and} \quad \sum_k \alpha_{c,k}^2 < \infty. \tag{4.41}$$

Compliance with these criteria ensures that the MC RLED algorithm's learning process, delineated in a manner akin to the approach described in Section 3, is poised for successful convergence.

This theorem reinforces the notion that the convergence analysis for MC RLED adheres to the same foundational approach as that established for biased exploration RLED. By ensuring that the cumulative sum of the learning rates over all iterations is infinite, the algorithm is encouraged to continually adapt and refine its parameters. In contrast, maintaining the sum of the squared learning rates within a finite range acts as a stabilizing constraint, essential

for the sustainability of the learning process. Through this alignment, the MC RLED algorithm not only subscribes to the proven theoretical underpinnings of its predecessor but also promises a coherent and structured pathway to achieving convergence, thereby optimizing policy performance within the domain of reinforcement learning.

4.2.3 Robot Control

Task Design

This study evaluates the MC RLED algorithm's performance in guiding a 2-degree-of-freedom planar robot from one point to another within its task-space. The goal is to demonstrate the algorithm's ability to converge to an optimal policy under safety constraints. Furthermore, this research explores the algorithm's sensitivity to variations in the uncertainty threshold hyperparameter, examining its impact on convergence rate and overall performance.

The analysis includes the average time steps required for convergence, alongside the final policy's accuracy and adherence to safety parameters. These findings contribute to a comprehensive understanding of the strengths, limitations, and response of the MC RLED algorithm to adjustments in the uncertainty threshold.

The performance of the MC RLED algorithm was compared with the biased exploration RLED, as documented in [106, 107]. Both algorithms were trained within the same environment and their performance was evaluated on the basis of the rewards accumulated during the training sessions. Additionally, this study varied the uncertainty threshold hyperparameter of the MC RLED algorithm, recording its influence on the algorithm's convergence velocity.

Settings

To evaluate the performance of the MC RLED algorithm, we conducted experiments on a simple task of navigating a 2 degree-of-freedom (DOF) planar robot from one point to another in the robot task-space. The environment consisted of one robot that had to move from the start position to the goal position. We used a simulation environment to simulate the robot and the environment. The simulation environment was implemented in Python using the PyBullet physics engine [97]. The robot was equipped with a two-joint manipulator, and the position of the end-effector was used to define the state of the robot. The state space consisted of the Cartesian coordinates of the end-effector and the action space consisted of the joint angles of the robot.

This section elaborates on the selection and impact of various hyperparameters critical to

fine-tuning the MC RLED algorithm's performance. A comprehensive list of these parameters, along with the explored ranges, is provided in the following.

- **Discount Factor** (γ): The discount factor γ influences the importance of future rewards, where $\gamma = 0.95$ suggests a preference for long-term rewards.
- **Network architecture** (ϕ, θ): The architecture, defined by the number of neurons and layers in the critic and actor networks, dictates the model's complexity. Optimal configurations are problem-specific and determined through experimentation. In this study, both critics and actors comprise 2 hidden layers with 128 neurons each.
- **Learning rates** (α_c, α_a): The learning rates $\alpha_c = 1 \times 10^{-5}$ and $\alpha_a = 1 \times 10^{-5}$ regulate the update magnitude of the critic and actor networks, respectively, balancing stability and convergence speed.
- **Target policy smoothing** ($\bar{\epsilon}$, c): To mitigate Q-value overestimation, Gaussian noise $\bar{\epsilon} = 0.2$ is added to target actions, with clipping at $\pm c$, where $c = 0.5$.
- **Delayed policy update** ($delay$): Updating the actor network every $delay = 2$ critic iterations decouples actor from critic updates, enhancing stability.
- **Training intervals**: Training commences after $train\_after = 1000$ time steps and occurs every $train\_every = 100$ steps to ensure diverse experiences and stabilize updates.
- **Gradient steps** ($gradient\_steps$): The number of updates per training iteration is fixed at $gradient\_steps = 100$, a balance between convergence speed and risk of overfitting.
- **Soft update** (τ): The target networks are softly updated towards the current networks at a rate of $\tau = 5 \times 10^{-3}$, ensuring gradual learning.
- **Exploration noise** (ϵ): Gaussian noise $\epsilon = 0.1$ promotes policy exploration during training.
- **Batch size** ($|B|$): Each training iteration processes $|B| = 100$ samples, optimizing learning efficiency.
- **Replay buffer** ($buffer$): The buffer capacity is set to 5×10^4 transitions, supporting learning from a rich set of experiences.

- **Uncertainty management**: The exploration strategy is fine-tuned with an uncertainty threshold u within $[1 \times 10^{-7}, 1 \times 10^{-1}]$ and a dropout rate $drop = 0.1$ to manage exploration safety and network regularization.

- **Monte Carlo samples** (N): The uncertainty of predictions is estimated using $N = 30$ dropout mask samples per forward pass, enhancing the reliability of uncertainty estimates.

Results Analysis

This study investigates the effect of the uncertainty threshold ξ, on the performance and safety of the algorithm through extensive experimentation.

Impact on performance and safety: Analysis of the reward-time step relationship revealed that a lower ξ enhances safety by reducing occurrences of low rewards, indicative of risky states or actions. In particular, at $\xi = 1 \times 10^{-3}$, the algorithm achieved complete safety with no low rewards throughout training. Despite variations in ξ, all configurations converged to similar final rewards, suggesting that adjusting ξ does not compromise ultimate performance. The learning curves showed consistent shapes across the ξ values, with variations in the speed at which to reach performance plateaus and safety levels. However, the biased exploration strategy in the RLED algorithm exhibited the slowest progression, attributed to its lack of an uncertainty-based exploration mechanism.

Demonstration dependence: The algorithm's reliance on demonstrations inversely correlated with ξ, with lower values prolonging the use of demonstrations, as shown in the second graph. This indicates a trend towards increased confidence in the algorithm's policy with decreasing ξ.

Uncertainty reduction: The third graph illustrates that lower ξ values slow the reduction of prediction uncertainty, suggesting a more cautious approach to building confidence in the algorithm predictions. The Biased exploration RLED variant demonstrated the least confidence, as indicated by its gradual uncertainty reduction.

Comparative analysis: The MC RLED algorithm outperformed its Biased exploration RLED counterpart in terms of learning efficiency and safety, highlighting the benefits of integrating uncertainty estimates for more effective and safer exploration.

Trade-offs and system design considerations: The findings underscore a trade-off between safety and learning efficiency associated with ξ adjustments. Lower ξ values favor safety at the expense of slower learning rates, necessitating a balanced approach in system design to meet specific safety and performance goals.

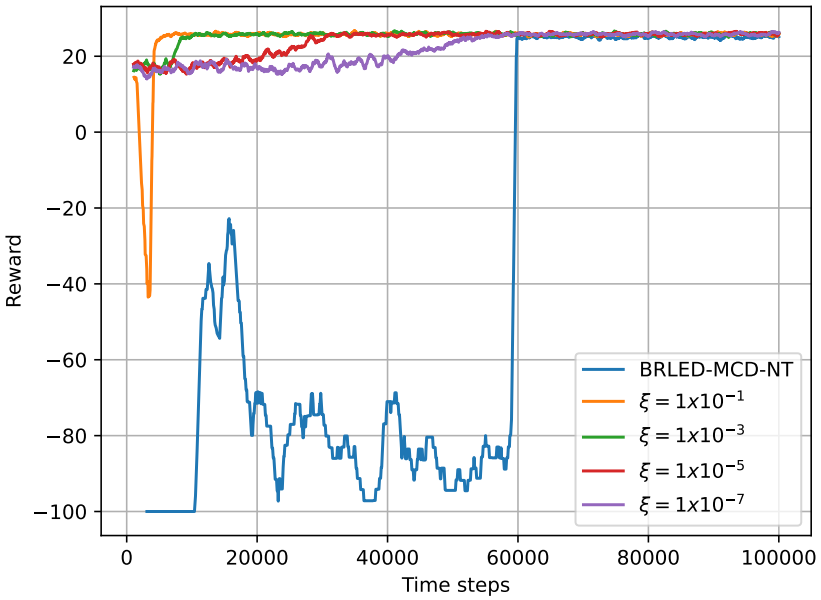

Figure 4.10: Comparison of reward learning curves across different uncertainty threshold parameters.

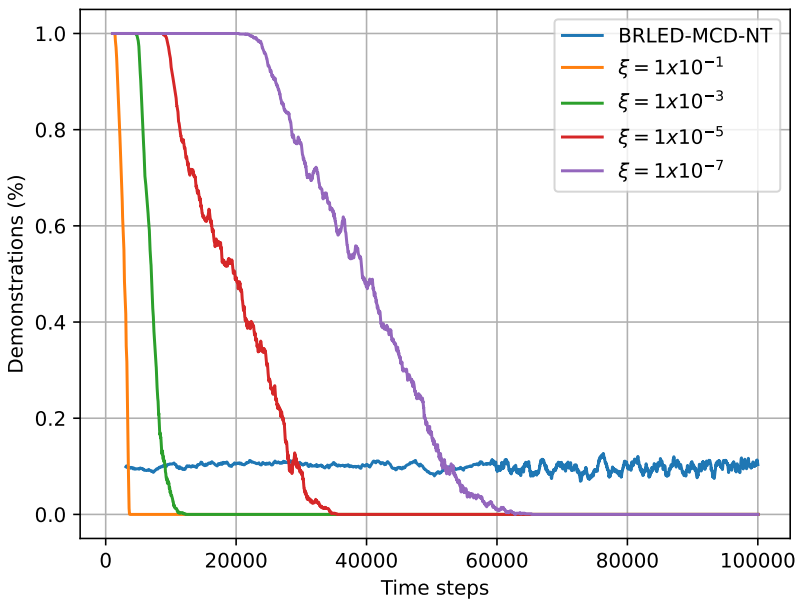

Figure 4.11: Comparison of demonstration percentage curves with varying uncertainty threshold parameters.

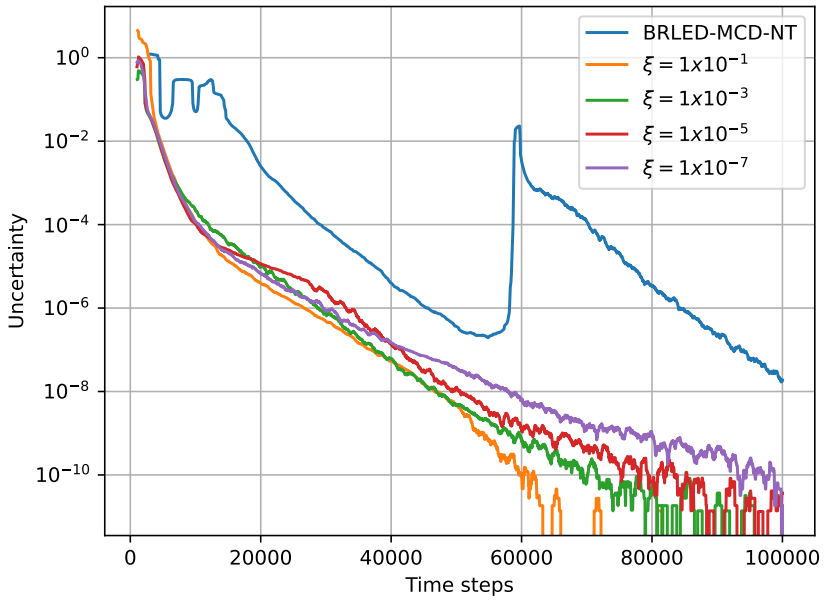

Figure 4.12: Comparison of uncertainty reduction curves across different uncertainty threshold parameters.

Limitations and future work: The experiments were confined to a single, relatively simple task environment, raising questions about the generalizability of the findings. Future research should explore the applicability of the method across diverse environments to validate its effectiveness and adaptability.

Chapter 5

Conclusions and Future Work

5.1 Conclusion

This thesis has made significant contributions to the field of RL, with a particular focus on augmenting RL with expert demonstrations and integrating considerations of safety and exploration efficiency in high-dimensional or continuous spaces. Through comprehensive reviews, development, and empirical validation in various studies, this work has advanced the understanding and application of RL in complex and uncertain environments.

5.1.1 Advancements in RLED

The initial investigation into RLED underscored its transformative potential to improve behavioral learning through expert demonstrations. Using demonstration trajectories, RLED enables the agent to integrate prior knowledge and refine its policy through reinforcement learning. This learning framework provides a robust foundation for the management of complex environments and tasks.

Incorporating expert demonstrations significantly reduces the complexity of the sample, allowing the agent to achieve high performance with fewer interactions with the environment. This is particularly valuable in domains where data collection is expensive or time-consuming. Starting with a policy informed by expert demonstrations, RLED agents exhibit superior initial performance compared to traditional RL agents. This head start can be crucial in applications where early-stage performance is critical.

Furthermore, RLED's use of expert demonstrations helps mitigate the risks associated with exploration, guiding the agent toward safer and more effective behaviors. This is essential in safety-critical applications, such as autonomous driving or medical robotics.

Despite its promise, several challenges must be addressed to fully realize the potential of RLED. The effectiveness of RLED is highly dependent on the quality and quantity of the demonstration data. Future research should focus on methods to enhance the robustness of RLED to suboptimal or sparse demonstrations. Real-world environments often introduce delays and noise, which complicate the learning process. Developing RLED algorithms that can effectively manage these issues is a critical area for future exploration. In many practical scenarios, agents operate with incomplete information about the environment. Extending RLED to handle partial observability will significantly broaden its applicability.

The exploration of innovative classifications and methodologies in RLED has opened new avenues for research. Investigating techniques to utilize demonstrations initially deemed detrimental can offer insight into improving agent resilience and adaptability. Integrating knowledge from diverse sources to effectively refine control policies can lead to more robust and versatile RLED agents.

By addressing these challenges and exploring new methodologies, RLED can continue to advance, providing a powerful framework to combine the strengths of RL and IL. This will enable the development of agents capable of performing complex tasks in dynamic and uncertain environments with greater efficiency and safety.

5.1.2 Biased Exploration RLED

The exploration of Biased Exploration RLED has demonstrated its effectiveness in addressing exploration challenges, particularly in environments characterized by high-dimensional or continuous action spaces. Using the TD3 methodology, Biased Exploration RLED significantly enhances policy learning by integrating demonstration trajectories into the reinforcement learning process. The TD3 approach, known for its superior performance compared to the classical DDPG, employs a model-free DRL method with an actor-critic architecture that ensures stability and effectiveness in policy learning.

The core innovation in Biased Exploration RLED is the integration of expert policy estimation to guide the exploration process. By approximating the expert policy through neural networks, the method biases the agent's exploration towards states and actions demonstrated by the expert, thereby reducing the likelihood of visiting irrelevant or less promising states. This biased exploration is controlled by a PPR parameter, which balances between leveraging expert knowledge and allowing for autonomous exploration.

The efficacy of this approach is evident in its ability to accelerate the learning process, particularly in sparse reward environments where traditional RL methods struggle to find

positive rewards. By starting the learning process from demonstrated states and employing biased exploration, the agent is more likely to encounter rewarding states, thereby improving overall learning efficiency and effectiveness.

Empirical studies have shown that Biased Exploration RLED outperforms baseline TD3 methods in various tasks, including those that involve redundant robot manipulators. The method's ability to utilize both dense and sparse reward functions, combined with its robustness to sub optimal demonstrations, makes it a versatile tool for a wide range of applications. The strategic use of demonstration data not only enhances sample efficiency but also ensures safer exploration by guiding the agent's behavior based on expert knowledge.

The introduction of Biased Exploration RLED marks a significant advancement in the field of reinforcement learning from expert demonstrations. Its ability to effectively combine demonstration trajectories with reinforcement learning principles addresses critical challenges such as exploration efficiency, safety, and sample complexity. This approach paves the way for more efficient and robust learning algorithms capable of tackling complex and high-dimensional environments with greater efficacy.

5.1.3 Safe RLED

The integration of safety within RLED has been advanced through the development of the MC RLED algorithm. This approach introduces uncertainty estimation using Monte Carlo dropout, providing a robust mechanism to ensure safer exploration strategies. By estimating the uncertainty of actions, the algorithm can effectively navigate the trade-offs between exploration and exploitation, thereby enhancing the agent's ability to avoid unsafe states and actions. This innovation is particularly crucial in environments where safety is a paramount concern, such as in robotic control and autonomous systems.

Empirical evaluations of the MC RLED, particularly in the context of a 2-degree-of-freedom planar robot, have demonstrated its superiority over the biased exploration RLED in terms of both reward acquisition and adherence to safety constraints. The experiments highlighted the importance of the uncertainty threshold hyperparameter, which significantly influences the balance between convergence rate and safety. Lower uncertainty thresholds improve safety by reducing risky actions, but may slow down the learning process. In contrast, higher thresholds can accelerate learning, but at the potential cost of increased risk. This trade-off necessitates a careful tuning of the hyperparameter to achieve optimal performance and safety.

The success of MC RLED in maintaining safety without compromising learning efficiency

underscores its potential for a wider application in real-world scenarios. By incorporating expert demonstrations and using uncertainty estimation, the algorithm not only achieves higher rewards, but also ensures that the learning process adheres to predefined safety constraints. This dual focus on performance and safety makes MC RLED a promising approach for complex, high-stakes environments where traditional reinforcement learning methods may falter.

The introduction of MC RLED marks a significant advance in the field of safe RL. The ability to integrate expert knowledge with robust uncertainty estimation techniques paves the way for more effective and reliable learning algorithms. Future research should focus on refining these methods, exploring their applicability in diverse and more complex tasks, and further investigating the optimal tuning of uncertainty thresholds. The advancements presented in this thesis contribute to the ongoing efforts to develop reinforcement learning systems that are not only efficient and high performing, but also inherently safe and reliable.

5.2 Future Directions

The promising results achieved in this thesis open up several avenues for future research in RLED. One key area for future exploration is the enhancement of RLED's robustness to suboptimal and sparse demonstrations. Real-world environments often present imperfect and limited demonstration data, which can hinder the learning process. Developing methods that can effectively handle and learn from such data will significantly improve the applicability and resilience of RLED in practical scenarios.

Another critical direction is the integration of RLED with techniques for managing delays and noise in real-world environments. Many real-world applications, such as autonomous driving and robotic manipulation, operate under conditions in which sensor readings and actions are subject to delays and noise. Research focused on adapting RLED algorithms to account for these factors will help to create more reliable and accurate models that can perform well under uncertain conditions.

Expanding RLED to handle partially observable environments is also a crucial area for future research. In many practical situations, agents do not have access to complete information about their environment. Incorporating techniques such as POMDP (Partially Observable Markov Decision Processes) into the RLED framework can enhance the agent's ability to make informed decisions despite limited visibility, thereby broadening the scope of applications where RLED can be effectively utilized.

Lastly, future research should explore the development of adaptive exploration strate-

gies within the RLED framework. Balancing exploration and exploitation is a fundamental challenge in reinforcement learning, and adaptive methods that can dynamically adjust the exploration strategy based on the agent's learning progress and environmental conditions can lead to more efficient learning processes. In addition, integrating these adaptive strategies with safety considerations will ensure that the agent can learn effectively while maintaining a high level of operational safety.

By addressing these challenges and pursuing these research directions, the potential of RLED can be further realized, leading to more robust, efficient, and safe learning algorithms capable of tackling an even broader range of real-world applications.

5.3 Synthesis and Final Thoughts

In summary, this thesis has helped advance the state of the art in RL by addressing critical challenges in learning from expert demonstrations, improving exploration strategies, and integrating safety measures. The foundation laid by this work for future research aims to refine RL methods further, making them more robust, efficient, and applicable to a wider range of real-world challenges. The continuous push to expand the boundaries of what is possible with RL, bolstered by the insights from this thesis, is expected to significantly contribute to advancing artificial intelligence, potentially revolutionizing how autonomous systems learn and interact with their environments.

Bibliography

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

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

[3] J. Ramirez, W. Yu, and A. Perrusquia, "Model-free reinforcement learning from expert demonstrations: a survey," *Artificial Intelligence Review*, vol. 55, no. 4, pp. 3213–3241, 2022.

[4] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[5] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007, vol. 703.

[6] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[7] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[8] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[9] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[10] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, 2020.

[11] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Handbook of robotics chapter 59: Robot programming by demonstration," *Handbook of Robotics. Springer*, 2008.

[12] A. Segre and G. DeJong, "Explanation-based manipulator learning: Acquisition of planning ability through observation," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 1985, pp. 555–560.

[13] T. Lozano-Perez, "Robot programming," *Proceedings of the IEEE*, vol. 71, no. 7, pp. 821–841, 1983.

[14] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[15] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 663–670.

[16] S. Lange, T. Gabel, and M. Riedmiller, "Batch reinforcement learning," in *Reinforcement learning*. Springer, 2012, pp. 45–73.

[17] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[18] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5628–5635.

[19] J. Ramírez and W. Yu, "Human behavior learning in joint space using dynamic time warping and neural networks," in *2019 16th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*. IEEE, 2019, pp. 1–6.

[20] R. Bellman, "On the theory of dynamic programming," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 38, no. 8, p. 716, 1952.

[21] R. S. Sutton, A. G. Barto, and R. J. Williams, "Reinforcement learning is direct adaptive optimal control," *IEEE Control Systems Magazine*, vol. 12, no. 2, pp. 19–22, 1992.

[22] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 12, pp. 6999–7019, 2021.

[23] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, "Recent advances in convolutional neural networks," *Pattern recognition*, vol. 77, pp. 354–377, 2018.

[24] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[25] F. Scarselli and A. C. Tsoi, "Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results," *Neural networks*, vol. 11, no. 1, pp. 15–37, 1998.

[26] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[27] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*. University of Cambridge, Department of Engineering Cambridge, UK, 1994, vol. 37.

[28] 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*, vol. 518, no. 7540, pp. 529–533, 2015.

[29] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[30] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.

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

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

[33] 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*, 2016, pp. 1928–1937.

[34] 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," in *International Conference on Learning Representations*, 2016.

[35] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1587–1596.

[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] M. E. Taylor, H. B. Suay, and S. Chernova, "Integrating reinforcement learning with human demonstrations of varying ability," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 617–624.

[38] Z. Wang and M. E. Taylor, "Improving reinforcement learning with confidence-based demonstrations," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 3027–3033.

[39] ——, "Interactive reinforcement learning with dynamic reuse of prior knowledge from human and agent demonstrations," in *Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019.

[40] T. Brys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé, "Reinforcement learning from demonstration through shaping," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[41] H. B. Suay, T. Brys, M. E. Taylor, and S. Chernova, "Learning from demonstration for shaping through inverse reinforcement learning," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, 2016, pp. 429–437.

[42] M. Li, T. Brys, and D. Kudenko, "Introspective q-learning and learning from demonstration," *The Knowledge Engineering Review*, vol. 34, 2019.

[43] M. Gimelfarb, S. Sanner, and C.-G. Lee, "Reinforcement learning with multiple experts: A bayesian model combination approach," in *Advances in Neural Information Processing Systems*, 2018, pp. 9528–9538.

[44] A. S. Lakshminarayanan, S. Ozair, and Y. Bengio, "Reinforcement learning with few expert demonstrations," in *NIPS Workshop on Deep Learning for Action and Interaction*, vol. 2016, 2016.

[45] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband *et al.*, "Deep q-learning from demonstrations," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[46] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[47] V. G. Goecks, G. M. Gremillion, V. J. Lawhern, J. Valasek, and N. R. Waytowich, "Integrating behavior cloning and reinforcement learning for improved performance in dense and sparse reward environments," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 465–473.

[48] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," in *Robotics: Science and System XIV*, 2018.

[49] G. V. Cruz Jr, Y. Du, and M. E. Taylor, "Pre-training neural networks with human demonstrations for deep reinforcement learning," *Workshop on Adaptive and Learning Agents (ALA) at the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2018.

[50] Y. Gao, H. Xu, J. Lin, F. Yu, S. Levine, and T. Darrell, "Reinforcement learning from imperfect demonstrations," *arXiv preprint arXiv:1802.05313*, 2018.

[51] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6292–6299.

[52] J. Chemali and A. Lazaric, "Direct policy iteration with demonstrations," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[53] B. Kang, Z. Jie, and J. Feng, "Policy optimization with demonstrations," in *International Conference on Machine Learning*, 2018, pp. 2469–2478.

[54] M. Jing, X. Ma, W. Huang, F. Sun, C. Yang, B. Fang, and H. Liu, "Reinforcement learning from imperfect demonstrations under soft expert guidance," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 5109–5116.

[55] T. Pohlen, B. Piot, T. Hester, M. G. Azar, D. Horgan, D. Budden, G. Barth-Maron, H. Van Hasselt, J. Quan, M. Večerík *et al.*, "Observe and look further: Achieving consistent performance on atari," *arXiv preprint arXiv:1805.11593*, 2018.

[56] M. Garmulewicz, H. Michalewski, and P. Miłoś, "Expert-augmented actor-critic for vizdoom and montezumas revenge," *arXiv preprint arXiv:1809.03447*, 2018.

[57] C. Gulcehre, T. Le Paine, B. Shahriari, M. Denil, M. Hoffman, H. Soyer, R. Tanburn, S. Kapturowski, N. Rabinowitz, D. Williams *et al.*, "Making efficient use of demonstrations to solve hard exploration problems," in *International Conference on Learning Representations*, 2019.

[58] S. Yeo, S. Oh, and M. Lee, "Accelerated deep reinforcement learning with efficient demonstration utilization techniques," *World Wide Web*, pp. 1–23, 2020.

[59] ——, "Accelerating deep reinforcement learning using human demonstration data based on dual replay buffer management and online frame skipping," in *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 2019, pp. 1–8.

[60] T. Salimans and R. Chen, "Learning montezuma's revenge from a single demonstration," *arXiv preprint arXiv:1812.03381*, 2018.

[61] C. Resnick, R. Raileanu, S. Kapoor, A. Peysakhovich, K. Cho, and J. Bruna, "Backplay:" man muss immer umkehren"," in *Workshop on Reinforcement Learning in Games at AAAI-19*, 2019.

[62] L. Torrey, "Reinforcement learning via reasoning from demonstration," *Workshop on Adaptive and Learning Agents (ALA) at the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2020.

[63] K. Subramanian, C. L. Isbell Jr, and A. L. Thomaz, "Exploration from demonstration for interactive reinforcement learning," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, 2016, pp. 447–456.

[64] S.-A. Chen, V. Tangkaratt, H.-T. Lin, and M. Sugiyama, "Active deep q-learning with demonstration," *Machine Learning*, pp. 1–27, 2019.

[65] M. Rigter, B. Lacerda, and N. Hawes, "A framework for learning from demonstration with minimal human effort," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2023–2030, 2020.

[66] X. Wu, D. Zhang, F. Qin, and D. Xu, "Deep reinforcement learning of robotic precision insertion skill accelerated by demonstrations," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2019, pp. 1651–1656.

[67] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar, "Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3651–3657.

[68] B. Keller, M. Draelos, K. Zhou, R. Qian, A. N. Kuo, G. Konidaris, K. Hauser, and J. A. Izatt, "Optical coherence tomography-guided robotic ophthalmic microsurgery via reinforcement learning from demonstration," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1207–1218, 2020.

[69] V. Kurin, S. Nowozin, K. Hofmann, L. Beyer, and B. Leibe, "The atari grand challenge dataset," *arXiv preprint arXiv:1705.10998*, 2017.

[70] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune, "First return, then explore," *Nature*, vol. 590, no. 7847, pp. 580–586, 2021.

[71] W. H. Guss, B. Houghton, N. Topin, P. Wang, C. Codel, M. Veloso, and R. Salakhutdinov, "Minerl: A large-scale dataset of minecraft demonstrations," *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019.

[72] G. Gordon-Hall, P. J. Gorinski, G. Lampouras, and I. Iacobacci, "Show us the way: Learning to manage dialog from demonstrations," in *The Eight Dialog System Technology Challenge (DSTC-8) at AAAI 2020*, 2020.

[73] G. Gordon-Hall, P. J. Gorinski, and S. B. Cohen, "Learning dialog policies from weak demonstrations," in *Proceedings of the 58th Annual Meeting of the Association for*

Computational Linguistics. Association for Computational Linguistics, 2020, pp. 1394–1405.

[74] J. Liu, Y. Zhang, X. Wang, Y. Deng, and X. Wu, "Dynamic pricing on e-commerce platform with deep reinforcement learning," *arXiv preprint arXiv:1912.02572*, 2019.

[75] Y. Liu, Q. Liu, H. Zhao, Z. Pan, and C. Liu, "Adaptive quantitative trading: an imitative deep reinforcement learning approach," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 2128–2135.

[76] S.-B. Cools, C. Gershenson, and B. D'Hooghe, "Self-organizing traffic lights: A realistic simulation," in *Advances in applied self-organizing systems*. Springer, 2013, pp. 45–55.

[77] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[78] P. Kormushev, S. Calinon, and D. G. Caldwell, "Reinforcement learning in robotics: Applications and real-world challenges," *Robotics*, vol. 2, no. 3, pp. 122–148, 2013.

[79] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[80] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "An empirical investigation of the challenges of real-world reinforcement learning," *arXiv preprint arXiv:2003.11881*, 2020.

[81] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine, "The ingredients of real-world robotic reinforcement learning," in *International Conference on Learning Representations*, 2020.

[82] K. V. Katsikopoulos and S. E. Engelbrecht, "Markov decision processes with delays and asynchronous cost collection," *IEEE transactions on automatic control*, vol. 48, no. 4, pp. 568–574, 2003.

[83] M. E. Taylor, "Improving reinforcement learning with human input," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 5724–5728.

[84] R. Zhang, F. Torabi, L. Guan, D. H. Ballard, and P. Stone, "Leveraging human guidance for deep reinforcement learning tasks," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019.

[85] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei, "Reward learning from human preferences and demonstrations in atari," in *Advances in Neural Information Processing Systems*, 2018, pp. 8011–8023.

[86] F. S. Melo, S. P. Meyn, and M. I. Ribeiro, "An analysis of reinforcement learning with function approximation," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 664–671.

[87] A. A. Shahri, C. Shan, and S. Larsson, "A novel approach to uncertainty quantification in groundwater table modeling by automated predictive deep learning," *Natural Resources Research*, vol. 31, no. 3, pp. 1351–1373, 2022.

[88] A. Benveniste, P. Métivier, and P. Priouret, *Adaptive algorithms and stochastic approximations*. Springer Science & Business Media, 1990.

[89] X. Xing and D. E. Chang, "Deep reinforcement learning based robot arm manipulation with efficient training data through simulation," in *2019 19th International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2019, pp. 112–116.

[90] C. Liu, L. Zhu, X. Ji, and X. Zheng, "Application of deep reinforcement learning in control of drawing robots," in *Journal of Physics: Conference Series*, vol. 1732, no. 1. IOP Publishing, 2021, p. 012050.

[91] A. R. Mahmood, D. Korenkevych, B. J. Komer, and J. Bergstra, "Setting up a reinforcement learning task with a real-world robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4635–4640.

[92] A. Perrusquía, "Human-behavior learning: A new complementary learning perspective for optimal decision making controllers," *Neurocomputing*, vol. 489, pp. 157–166, 2022.

[93] A. Perrusquía, W. Yu, and X. Li, "Nonlinear control using human behavior learning," *Information Sciences*, vol. 569, pp. 358–375, 2021.

[94] W. Yu and A. Perrusquía, *Human-Robot Interaction Control Using Reinforcement Learning*. Wiley-IEEE Press, 2021.

[95] M. W. Spong, S. Hutchinson, M. Vidyasagar *et al.*, *Robot modeling and control*. Wiley New York, 2006, vol. 3.

[96] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, vol. 1. IEEE, 2001, pp. 298–303.

[97] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[98] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016, (Version 0.21). [Online]. Available: https://www.gymlibrary.ml

[99] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, (Version 2.3). [Online]. Available: https://www.tensorflow.org

[100] G. Van Rossum, "Python," 1991, (Version 3.7.13). [Online]. Available: https://www.python.org

[101] E. Altman, *Constrained Markov decision processes*. CRC press, 1999, vol. 7.

[102] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[103] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.

[104] J. Ramirez and W. Yu, "Reinforcement learning from expert demonstrations with application to redundant robot control," *Engineering Applications of Artificial Intelligence*, vol. 119, p. 105753, 2023.

[105] ——, "Redundant robot control with learning from expert demonstrations," in *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2022, pp. 715–720.

[106] ——, "Safe exploration in reinforcement learning for learning from human experts," in *2023 IEEE International Conference on Artificial Intelligence, Blockchain, and Internet of Things (AIBThings)*. IEEE, 2023, pp. 1–5.

[107] ——, "Safe reinforcement learning for learning from human demonstrations," 2024.