POLITECNICO DI TORINO

Master's Degree in Computer Engineering
Artificial Intelligence and Data Analytics

Master's Degree Thesis

Inverse Reinforcement Learning for Mastering Long-Horizon Procedural Tasks from Visual Demonstrations

Relatori
prof. Giuseppe Bruno Averta
prof. Francesca Pistilli
M.Sc. Andrea Protopapa
firma dei relatori

. . . . . .
. . . . . .

Candidato
Luca Ianniello

firma del candidato

. . . . . .

Anno Accademico 2024-2025

Voglio dedicare questa tesi alla mia famiglia che mi ha sempre sostenuto sia da vicino che da lontano in questi due anni, ai miei amici e ai miei coinquilini che hanno alleggerito le lunghe giornate di Torino, alla testa che fa sempre le scelte più difficili e al cuore che deve sopportarne le conseguenze.

Summary

Robotic manipulation represents one of the most challenging domains in robotics, requiring precise coordination and adaptability to complex environments. While Reinforcement Learning approaches show promise, they face significant limitations in practical applications: reward engineering can be prohibitively complex, exploration in high-dimensional spaces can be inefficient, and physical robot training requires extensive resources. Imitation Learning (IL), particularly Inverse Reinforcement Learning (IRL), offers an alternative by learning directly from demonstrations rather than explicit reward signals.

However, current IRL approaches face several fundamental challenges when applied to robotic manipulation tasks. Long-horizon manipulation tasks with multiple sequential stages are difficult to learn end-to-end due to sparse rewards and temporal complexity. Additionally, the effectiveness of different visual representation learning architectures for IRL in manipulation contexts remains under-explored, especially when combined with procedural decomposition strategies.

With procedural decomposition, complex tasks are broken down into manageable subtasks, allowing the agent to focus on mastering each subtask before integrating them into a complete solution. However, choosing the appropriate number of subtasks is an important challenge, as too few may not capture the task's complexity, whereas too many can lead to overfitting and increased computational costs. Furthermore, balancing exploration and exploitation becomes even more critical in this context because the agent must learn from demonstrations while exploring new strategies.

In this thesis, we present INEST-IRL, a novel framework that integrates inverse Reinforcement Learning with procedural decomposition and active exploration strategies to effectively learn complex manipulation tasks from visual demonstrations. The framework leverages state-of-the-art visual representation learning models to extract meaningful features from demonstrations, which are then used to learn the reward functions for each subtask through embedding diversity enhancement. The active exploration strategy incorporates a reward component based on the difference between the current and previous embedding frames within the same subtask, promoting more effective exploration.

Our results demonstrate that INEST-IRL outperforms existing IRL methods on complex manipulation tasks in the X-Magical environment, particularly in scenarios that require long-horizon planning and sequential decision-making. We also provide an extensive evaluation of the impact of different numbers of procedural steps and the contribution of each component of the proposed framework to the learning performance.

Acknowledgements

Questa è molto probabilmente la parte più difficile da scrivere dell'intera tesi, perchè le conoscenze, le nozioni, le idee si possono imparare o emulare, ma i sentimenti no. Ciò che una persona prova non si può imparare e nemmeno percepire, anche se si pensa di aver provato qualcosa di simile. Durante la mia esperienza a Torino, ho imparato a gestire questi sentimenti, passando da momenti di gioia a quelli di solitudine e infine a momenti in cui l'unica cosa da dire è "Grazie". Il valore di questa parola è mistico. Ogni giorno la pronunciamo per qualcuno soltanto per aver fatto qualcosa per noi o per averci aiutato. Dietro di essa, però, nasconde amore, fiducia. Il potere che ha un Grazie è sconfinato. E' capace di scacciare le nubi che riempiono la nostra mente. Ha anche il potere di allegerire ogni peso che ci portiamo appresso ogni giorno ininterrottamente. Perchè un Grazie, accompagnato da un sorriso, fa capire che c'è qualcuno di cui ci si può fidare e che è pronto a dare una mano.

Quando ho scritto i ringraziamenti per la tesi triennnale, la vita era molto diversa. Vivevo con i miei genitori e mio fratello in una città che al tempo ritenevo grigia, sognando di girare per altri posti in cerca di qualcosa che mi mancava. Adesso che sto scrivendo i ringraziamenti per la tesi magistrale, soltanto 2 anni dopo, vivo con 3 persone di cui conoscevo poco o nulla in quel momento di 2 anni fa e in una città diversa, ma che è un pò meno grigia. Non intendo dire che Torino sia una noiosa e monotona città, ma che le manca qualcosa di importante. Qualcosa che ho imparato proprio stando a Torino e andando via da Caserta.

Ciò che manca a Torino è molto semplice: la mia famiglia. Questo molto probabilmente è l'insegnamento più importante che mi ha dato questa esperienza. In passato ho sempre sottovalutato la quotidianità di vivere in famiglia. Ogni momento insieme, che fosse un pranzo della domenica o un uscita in giro, era sempre uguale. Adesso invece, pagherei per rivivere ogni singolo momento del passato. Pagherei per godermi tutti quei momenti che avevo considerato uguali. E se ora ho imparato ciò, lo devo proprio alla mia famiglia. Per loro non basterebbero milioni di Grazie per far capire quanto sono grato per avermi dato l'opportunità di essere qui. Per tutti i sacrifici fatti, per tutte le chiamate ricevute ogni giorno, per tutte le volte in cui ho ritrovato il calore di casa quando tornavo dopo mesi e mesi. Per tutti gli abbracci e i saluti dati alla stazione, per tutte le volte in cui ho disfatto la valigia e per tutte le volte in cui mi hanno aiutato a rifarla prima di partire. Grazie di tutto ciò, Mamma e Papà. Grazie per avermi permesso di vivere tutto ciò e di aver potuto imparare dai miei errori e dai vostri consigli.

Oltre a Mamma e Papà, devo ringraziare anche mio fratello, Marco. Ma a lui non devo

dire solo Grazie, ma anche Scusa. Scusa per non esserci stato sempre in questi 2 anni, come avrei voluto. Scusa per non averti potuto veder crescere negli anni più importanti della tua vita. Ogni volta che sono tornato a casa, ho sempre visto come eri cambiato e come eri cresciuto e ciò ha fatto crescere anche me. Sono molto contento della persona che sei e stai diventando, anche grazie ai tuoi amici. Ci sono stati tanti momenti in passato in cui non ci sopportavamo a vicenda. Poi siamo cresciuti, e siamo finalmente diventati fratelli. Probabilmente anche questo è stato frutto della lontananza. Entrambi abbiamo capito quanto l'uno era importante per l'altro nella quotidianità e tu sei importantissimo per me. Grazie per avermi distratto da tutti i problemi che avevo quando tornavo a casa. Grazie per avermi coinvolto sempre di più nella tua vita, anche quando ero a 800km di distanza. Grazie per essere diventato ciò che ho sempre voluto e forse ciò che avrei sempre voluto diventare io.

A Torino, però, non sono stato solo. Anche se non c'è la mia famiglia, qui si è formata un'altra vita, diversa, ma piacevole sotto alcuni aspetti. A renderla piacevole, ci sono i miei amici e i miei coinquilini, con cui ho vissuto moltissime esperienze e non basterebbe una tesi per raccontarle tutte.

Prima di parlare della mia nuova "famiglia", devo ringraziare due persone, senza le quali non sarei ciò che sono diventato ora e forse senza di loro non avrei superato molte avversità facilmente: Matteo e Domenico.

Ormai sono 5 anni che io e Matteo ci conosciamo. In questo tempo, il nostro rapporto è cambiato moltissimo e sono sicuro che continuerà a cambiare anche in futuro. Matteo, per me, è come un fratello adottivo. E' quel tipo di amico che se deve dire qualcosa, lo fa, anche se so che non mi piacerà. E' quell'amico con cui ti puoi confrontare su tutto, anche su stupidaggini. E' quell'amico a cui racconto tutto ciò che faccio, dalla mattina alla sera. Matteo è quell'amico che risponde sempre ai miei racconti, alle mie richieste, ai miei dubbi, ed è quell'amico che c'è sempre, indipendentemente da tutto e da tutti. E la cosa che mi piace di più di Matteo è che siamo simili, ma diversi. Paradossalmente, ogni situazione la possiamo affrontare sia nello stesso modo sia in modo diverso e ciò permette di imparare qualcosa di nuovo l'uno dall'altro. Forse questo è il segreto della nostra amicizia. Prima di conoscere Matteo, venivo da svariate delusioni. Amici che ritenevo come fratelli che se ne sono andati senza salutare e altri con cui non è mai scattato qualcosa. Con Matteo invece, quel qualcosa è scattato fin da subito. L'anno in cui è stato a Stoccolma è stato particolare. Nonostante lo sentissi ogni giorno, avevo paura che quando sarebbe tornato, lo avrei trovato diverso oppure che anche lui se ne sarebbe andato. Invece, non solo è rimasto, ma siamo cresciuti entrambi e con noi il nostro rapporto. Grazie Matteo per esserci sempre stato e spero di poterti ringraziare altre mille volte in futuro.

La mia amicizia con Domenico è una vera eccezione alla regola. Ho sempre dubitato delle amicizie a distanza, perchè avevo sempre il dubbio che si potesse perdere l'intesa tra le parti. Invece, mi sbagliavo. Perchè, purtroppo, la mia amicizia con Domenico è stata solo un'amicizia a distanza. Il purtroppo è doveroso. Avrei voluto tantissimo vivere tante avventure e tante esperienze con Domenico. Avrei voluto averlo nella mia quotidianità. E un pò è ciò che ho fatto. Anche Domenico è quel tipo di amico come Matteo, che ritengo come un fratello acquisito. Ma il rapporto che ho con Domenico è diverso. E' un legame che accomuna persone con un animo affine. Abbiamo interessi molto simili, idee di pensiero equivalenti e sempre qualcosa di cui parlare. Proprio questa è la forza del

nostro legame. In Domenico ho trovato una persona con cui potrei parlare per giornate intere senza stancarmi e sapendo sempre che dai lui otterrò sempre una risposta. Che sia un messaggio scritto o un vocale da 30 minuti, lui risponderà sempre. E nel parlare ogni giorno, siamo cresciuti molto. Sono veramente contento di averti visto raggiungere molti traguardi e di averti visto superare molte difficoltà. Grazie per avermi fatto crescere con la tua fiducia e presenza e grazie per avermi reso il confidente della tua storia.

Oltre a Matteo e Domenico, devo ringraziare la mia "famiglia". Questa nuova famiglia è composta da tre persone, una più strana dell'altra e una più diversa dell'altra. Proprio questa diversità (e stranezza) mi hanno permesso di crescere molto durante questi 2 anni. Il rapporto tra me e Michele è stato molto particolare. Siamo passati da momenti di confidenze reciproche a momenti di astio e avversità. La maggior parte delle volte in cui si creavano conflitti è stata per colpa mia, forse perchè la diversità tra me e Michele è molta, sia dal punto di vista caratteriale sia per gli interessi. Proprio questi momenti mi hanno dato una grande lezione: Tutti siamo diversi, ma tutti vogliamo qualcuno che ci stia vicino. Prima di questo viaggio, conoscevo molto poco Michele, ma devo dire di essere contento di avere avuto una persona come lui al mio fianco. E' la persona che ti riesce ad alleggerire momenti stressanti con una battuta o semplicemente parlando. E quando vedi che ciò non succede, vuol dire che devi essere tu ad alleggerire i suoi momenti stressanti. Grazie Michele per avermi sopportato e supportato in questi momenti e per esserci stato quando ne avevo più bisogno. A differenza di Michele, conoscevo già Luca. Nella tesi triennale, avevo espresso il desiderio di volerlo conoscere di più perchè sapevo che mi avrebbe potuto dare tanto. Così è stato. Luca è quella persona che vuoi sempre avere vicino, perchè sai che dirà sempre qualcosa o di divertente o di utile. E' quella persona che c'è sempre quando gli si propone di fare qualcosa o spesso è lui a proporre qualcosa da fare. E' una persona semplice, e nella sua semplicità riesce a trasmetterti calore e felicità. Poi diciamo anche che Luca è una persona con cui si può inciuciare tranquillamente, ma quello è per altre storie. Sono molto contento di averti avuto vicino durante questo percorso. Grazie dottore per avermi insegnato un pò di semplicità in una vita complicata. Il terzo membro della famiglia è Francesco e purtroppo è anche la persona che conosco da meno tempo e con la quale ho vissuto meno esperienze. Tuttavia, in questi pochi momenti ho capito perfettamente che tipo di persona è. Francesco ha una caratteristica che purtroppo mi manca: l'adeguarsi ad ogni situazione. Può succedere qualunque cose, lui si adeguerà e la renderà molto semplice. Magari non è la persona più affidabile che ho conosciuto, però è l'amico con cui puoi parlare di tutto e ti darà sempre dei buoni consigli. Spero che la nostra amicizia continui e si fortifichi di più in futuro e spero che possa imparare altro da te.

Oltre a loro, voglio ringraziare tutti gli amici che ci sono stati sempre durante questa esperienza. Voglio ringraziare Tonino, Manuel, Raffaele, Pietro e Andrea, per aver contribuito alla mia crescita, qui a Torino e per tutti gli insegnamenti e le bellissime esperienze che abbiamo vissuto. Voglio ringraziare anche Pietro e Manuel, che, nonostante la distanza, ci sono sempre stati dalla triennale e ancora mi aiutano ad imparare cose nuove ogni giorno.

Voglio ringraziare anche i miei relatori, il Professor Averta, la Professoressa Pistilli, e Andrea. Grazie mille per tutto l'aiuto che mi avete dato in questi 8 mesi affinchè tutto ciò fosse possibile. Specialmente voglio ringraziare Andrea. Sei la persona che più mi

ha aiutato a crescere dal punto di vista lavorativo durante questa esperienza e mi voglio anche scusare con te per tutti i momenti tesi che ci sono stati durante il percorso.

Infine, voglio ringraziare il me del passato. Questa volta però, il ruolo che ha avuto il me del passato è stato molto, ma molto più marginale rispetto al ruolo che hanno avuto tutte le persone che ho nominato in questa pagina. Diciamo che il suo ruolo è stato quello di accettare una decisione difficile come quella di lasciare casa e cercare nuove sfide. Se potessi incontrarti, avrei tante cose da dirti e da insegnarti, ma adesso ciò che conta è dove siamo arrivati e ciò che faremo da ora in avanti, perchè con questa tesi finisce un percorso di studio di 18 anni e adesso inizia la vita vera, quella che tutti temono e anche tu, come tutti, dovrai affrontare, dando sempre tutto, come hai sempre fatto.

Contents

List of Tables

List of Figures

Flectere si nequeo superos, Acheronta movebo
[Publio Virgilio Marone, Eneide]

Ti conviene lasciare un posto dove sei qualcuno per andare in un posto dove sei solo un numero?
[Un professore, 2 anni fa]

Despite everything, it's still you
[Undertale]

Chapter 1

Introduction

Robotic manipulation is one of the most challenging domains in robotics. At its core, robotic manipulation involves the interaction between robots and objects in their environment through physical contact to achieve the desired operations. As highlighted by Kroemer et al. [1], manipulation encompasses a diverse set of tasks, including grasping, pushing, lifting, assembling, and rearranging objects. All these tasks require precise control and sophisticated perception capabilities.

Despite significant progress, robotic manipulation faces several fundamental challenges that limit its practical application.

- Perception challenges, as highlighted by Kroemer et al. [1], require robots to accurately capture object properties including shape and pose. Many physical properties must be inferred through interactions, creating a complex relationship between perception and action.
- The physics of contact introduces additional complexity, with Billard et al. [2] noting that contact dynamics are highly sensitive to initial conditions and involve transitions between different states that require specialized control strategies.
- Long-horizon planning presents computational challenges due to the combinatorial nature of action sequences and interdependencies between decisions, as identified by Han et al. [3]. These planning challenges are compounded by the need for robust adaptation when confronted with perceptual or execution-related uncertainties.
- Kroemer et al. [1] emphasize that generalization across novel objects, tasks, and environments remains a persistent limitation, as robots trained in controlled settings often struggle to transfer their skills to new situations. This generalisation gap significantly constrains the practical utility of robotic manipulation systems in environments where adaptability is essential, necessitating solutions that can extract underlying manipulation principles rather than memorising specific instances.

Reinforcement Learning has emerged as a promising approach to address many of the challenges in robotic manipulation. In contrast to traditional methods that rely on explicit models and handcrafted control strategies, RL enables robots to learn manipulation

policies through trial-and-error interactions with their environment. This learning-based approach offers several theoretical advantages, particularly for complex tasks in which analytical solutions are difficult to derive.

1.1 Problem Statement

Reinforcement Learning allows to handle long-horizon tasks by maximising cumulative rewards. This characteristic is particularly relevant for manipulation tasks that require sequences of coordinated actions to achieve the desired object configurations. The ability to learn from delayed rewards theoretically enables agents to discover action sequences with beneficial long-term consequences, even when the immediate outcomes appear suboptimal.

However, the use of Reinforcement Learning in robotic manipulation tasks presents several challenges. One of these is the reward specification problem. Designing reward functions that effectively guide learning toward desired manipulation behaviours is challenging. As Billard et al. [2] noted, manipulation tasks often involve multiple objectives and constraints that are challenging to capture in a single scalar reward signal. Sparse rewards lead to inefficient learning, whereas highly shaped rewards may introduce unintended biases that result in suboptimal or unstable policies.

Another significant challenge is learning to solve long horizon tasks. As highlighted by Wang et al. [4], complex robotic manipulation tasks often require sequences of coordinated actions that extend across long temporal horizons, creating several fundamental difficulties for standard RL approaches to address.

The primary challenge in long-horizon tasks stems from the temporal credit assignment problem, in which it becomes increasingly difficult to determine which actions in a long sequence contribute to the final outcome. In Reinforcement Learning, this is shown as sparse reward signals that provide limited guidance during the learning process, as the agent may only receive meaningful feedback upon completing the entire task.

1.2 Objective and Contributions

The purpose of this thesis is to explore and overcome the challenges discussed above. The proposed approach comprises three main components:

- Inverse Reinforcement Learning from visual demonstrations (Section 1.2.1), which enables the learning of reward functions directly from video and image data, to address the reward specification problem.

- Procedural Learning (Section 1.2.2), which decomposes long-horizon tasks into manageable subtasks, facilitating more efficient learning and planning.

- An Intrinsic Reward contribution (Section 1.2.3) that encourages the agent to explore diverse states and actions within each subtask, thereby enhancing its ability to discover effective strategies for the overall task.

The goal of this study is to develop an approach that enables robots to learn complex manipulation skills from visual demonstrations, addressing the limitations of traditional Reinforcement Learning methods. By leveraging the strengths of imitation learning, particularly inverse Reinforcement Learning (IRL), and incorporating Procedural Learning techniques and Intrinsic Reward mechanisms, this study aims to create a robust and efficient system for teaching robots long-horizon manipulation tasks.

1.2.1 Inverse Reinforcement Learning from Visual Demonstrations

Imitation learning has emerged as a powerful paradigm for training agents to perform complex tasks by observing expert performance. This enables agents to acquire skills without explicit programming or reward engineering, making it particularly suitable for domains in which defining appropriate reward functions is challenging.

The three main categories of imitation learning approaches are:

- Behaviour Cloning: learning a direct mapping from states to actions by mimicking expert demonstrations.
- Adversarial Imitation Learning: employing a generative adversarial method where a policy is trained to produce behaviors indistinguishable from expert demonstrations.
- Inverse Reinforcement Learning: recovering the underlying reward function that explains expert behavior from demonstrations.

Among the different imitation learning approaches, Inverse Reinforcement Learning (IRL) represents the most relevant paradigm for learning complex manipulation tasks from visual demonstrations. Unlike behaviour cloning, which directly mimics actions, or adversarial imitation learning, which matches state-action distributions, IRL aims to recover the underlying reward function that explains expert behaviour directly from video and image data.

The application of IRL to visual data, particularly demonstration videos, involves a structured pipeline consisting of two primary phases: representation learning and policy learning. In the representation learning phase, visual demonstrations are processed through encoder architectures to extract meaningful embeddings that capture the task-relevant features. These embeddings act as compact representations of the visual state space, facilitating subsequent reward inference.

The output of the representation learning phase is a trained encoder that maps visual observations to embeddings that capture the essential task features and progress. These embeddings serve as the foundation for inferring the reward function, which is typically formulated as a distance metric in embedding space. States similar to those demonstrated by experts receive higher rewards, whereas dissimilar states receive lower rewards. This approach enables the system to learn reward functions directly from visual observations without requiring explicit state or action labels to be defined.

In the policy learning phase, the learned reward function guides the optimisation of a policy using standard Reinforcement Learning algorithms. As the agent interacts with

the environment, its current visual observations are processed through an encoder to determine rewards based on their similarity to the demonstrated states.

1.2.2 Procedural Learning

Procedural Learning has emerged as a fundamental paradigm for addressing the challenges mentioned above by decomposing complex, long-horizon tasks into manageable subtasks that can be learned and executed sequentially. As demonstrated by Liu et al. [5], this approach leverages the insight that many complex behaviours can be understood as compositions of simpler and more atomic actions. By breaking down a complex manipulation task into a sequence of intermediate objectives, Procedural Learning transforms a single difficult learning problem into multiple, more tractable subproblems.

This decomposition offers several advantages. First, it simplifies the learning process by allowing the agent to focus on mastering individual subtasks before integrating them into a coherent, overall strategy. This stepwise approach can lead to more efficient learning because each subtask can be optimised independently, reducing the complexity of the policy search space. Second, Procedural Learning facilitates better exploration of state spaces. By defining clear intermediate goals, the agent can explore relevant regions of the environment more effectively, thereby leading to improved sample efficiency and faster convergence. Finally, it enhances the interpretability and modularity of the learned behaviours. Each subtask can be understood and analysed independently, allowing for easier debugging and refinement of specific components of the overall task.

1.2.3 Intrinsic Reward

One issue that characterizes Procedural Learning in Reinforcement Learning is the exploration-exploitation trade-off. In Reinforcement Learning, agents must balance the exploration of new actions to discover potentially better strategies and the exploitation of known actions that yield high rewards. This trade-off is particularly pronounced in long-horizon tasks, where the consequences of actions may not be immediately evident, and the state space can be vast and complex.

Dividing a long-horizon task into smaller subtasks can create issues in which the agent focuses too much on exploiting known strategies for individual subtasks, potentially neglecting the exploration of alternative approaches that could lead to better overall performance. This can result in suboptimal policies that fail to generalise across the entire task.

To address this challenge, an Intrinsic Reward contribution is proposed to encourage the agent to explore diverse sets of actions and states within each subtask. This strategy involves an additional contribution to the reward function based on the diversity of the visited embeddings. By rewarding the agent for visiting a wide range of states, the Intrinsic Reward contribution incentivises the agent to explore beyond the immediate goals of each subtask, thereby enhancing its ability to discover novel strategies that may lead to improved performance in the overall task.

1.2.4 INEST-IRL

The proposed approach, named INEST-IRL, integrates the strengths of Inverse Reinforcement Learning from visual demonstrations, Procedural Learning for long-horizon tasks, and an Intrinsic Reward contribution to create a robust framework for teaching robots complex manipulation skills. By leveraging visual data to infer reward functions, decomposing tasks into manageable subtasks, and encouraging exploration through diversity rewards, INEST-IRL aims to overcome the limitations of traditional Reinforcement Learning methods and enable robots to learn effective manipulation policies in complex environments.

This strategy is particularly relevant for robotic manipulation tasks, where the ability to learn from visual demonstrations and adapt to new situations is crucial for practical applications. The integration of these components creates a comprehensive learning framework that addresses the challenges of reward specification, long-horizon planning, and exploration in a unified way.

1.3 Structure of the Contents

The thesis is structured as follows:

- Chapter 2 provides an overview of the key concepts and techniques relevant to this thesis, including Reinforcement Learning, imitation learning, and representation learning.
- Chapter 3 reviews the existing literature on robotic manipulation, imitation learning, and inverse Reinforcement Learning, highlighting the strengths and limitations of current approaches.
- Chapter 4 details the proposed approach, including the dataset used, the structure of the inverse Reinforcement Learning pipeline, and the contributions of Procedural Learning and Intrinsic Reward.
- Chapter 5 presents the experimental setup, evaluation metrics, and results of applying the proposed approach to various robotic manipulation tasks. It includes comparisons with baseline methods and ablation studies to assess the contribution of each component.
- Chapter 6 summarises the findings of the thesis, discusses their implications for robotic manipulation, and outlines potential directions for future research.

Chapter 2

Background

This chapter provides an overview of the key concepts and techniques useful for understanding this thesis. In detail, this chapter covers the fundamental concepts of Reinforcement Learning, starting from the Markov Decision Process (MDP) and the different types of algorithms, such as value-based, policy-based, and actor-critic methods. Then, it introduces the concept of Imitation Learning, with a focus on Inverse Reinforcement Learning (IRL) from images. Finally, the techniques used for representation learning, including Convolutional Neural Networks (CNNs), Residual Networks, Vision Transformers, and Temporal Cycle-Consistency (TCC), are discussed. These concepts are essential for a better understanding of the methodology and experiments presented in the following sections of this paper.

2.1 Reinforcement Learning

2.1.1 What is Reinforcement Learning?

Reinforcement Learning (RL) is the fundamental paradigm of Machine Learning and Artificial Intelligence that focuses on training agents to make sequences of decisions by interacting with an environment. As defined by Sutton and Barto [6], an agent learns to achieve a goal by receiving feedback in the form of a reward or penalty.

To properly contextualise Reinforcement Learning, it is important to understand Artificial Intelligence (AI) and Machine Learning (ML) and how Reinforcement Learning differs from other learning paradigms. Artificial Intelligence refers to the development of computational systems capable of performing tasks that would typically require human intelligence. These tasks include reasoning, problem solving, understanding natural language, and visual perception and recognition. Machine Learning, as a subset of AI, represents a paradigm shift from traditional programming approaches. Rather than explicitly programming a computer with specific rules for every situation, machine learning algorithms enable systems to automatically learn patterns from data and improve their performance based on experience. This capability allows machine learning systems to adapt to new scenarios and handle complex problems that would be impractical to solve

using conventional programming techniques. Machine Learning can be categorised into different paradigms, including supervised, unsupervised, self-supervised, and Reinforcement Learning.

In supervised learning, algorithms learn a function from labelled training data by mapping inputs to known outputs under direct supervision. In this case, the given dataset is composed of input-output pairs, and the model learns to predict the output from the input on unseen new data. On the other hand, unsupervised learning works with unlabelled data to discover hidden patterns or intrinsic structures without receiving explicit feedback. Self-supervised learning, a relatively new paradigm, autonomously generates supervisory signals from the data structure, creating pseudo-labels for training.

Reinforcement Learning differs from these approaches in several key aspects, as highlighted by Arulkumaran et al. [7]. Unlike supervised learning, RL does not require pre-labelled examples but learns through trial-and-error interactions. Kaelbling et al. [8] describe how RL agents must sequentially make decisions and observe outcomes to maximise cumulative reward over time. The agent receives feedback only after completing actions, often with delayed rewards, making it challenging to determine which actions are advantageous. This temporal credit assignment problem is unique to Reinforcement Learning. In addition, RL must balance exploration (trying new actions to discover better strategies) and exploitation (using known effective actions), a trade-off that is not present in other learning paradigms, such as supervised learning. The agent must balance exploiting what it already knows to earn rewards and exploring less tested actions that might lead to greater long-term success.

In the context of robotic manipulation, Kober et al. [9] demonstrated how Reinforcement Learning offers distinct advantages, as it allows robots to learn complex manipulation skills through interaction with the physical world, gradually refining their behaviour based on task success rather than requiring explicit programming of every movement.

2.1.2 Markov Decision Process

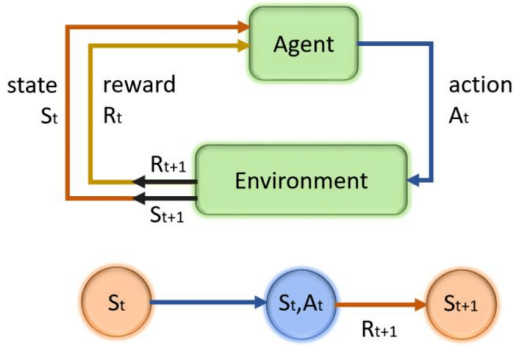


Figure 2.1: Markov Decision Process (MDP) Representation. The figure illustrates the key components of an MDP, including states, actions, transition dynamics, and rewards.

To better understand the fundamental concepts and elements of Reinforcement Learning, it is essential to introduce the Markov Decision Process (MDP), which provides a mathematical foundation for modelling decision-making problems in RL.

As described by Kaelbling et al. [8], an MDP is defined as a tuple (S, A, P, R, γ) where

- S is a finite set of states representing all possible configurations of the environment. Each state encodes the complete observable information necessary to determine the optimal action at any time.
- A is a finite set of actions available to the agent. It includes all possible moves that an agent can make in the environment.
- P is the state transition probability function, where $P(s'|s, a)$ represents the probability of transitioning to state s' when taking action a in state s.
- R is the reward function, where $R(s, a, s')$ defines the immediate reward received after transitioning from state s to state s' due to action a. It can be positive or negative, depending on whether the action leads to desirable or undesirable results. The return is the total discounted reward received over time, starting from a given state and following a specific policy.
- $\gamma \in [0,1]$ is the discount factor that determines the importance of future rewards relative to immediate rewards.

The Markov Decision Process is based on the Markov property, which states that the future state depends only on the current state and action and not on the entire sequence of states and actions. Formally, this means that

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, ..., s_0, a_0) = P(s_{t+1}|s_t, a_t) \tag{2.1}$$

This property significantly simplifies the Reinforcement Learning problem by eliminating the need to consider the entire history of states and actions. Reinforcement Learning algorithms rely on this property to learn optimal policies efficiently.

The goal in an MDP is to find a policy π that maps states to actions ($\pi : S \to A$) in a way that maximises the expected cumulative discounted reward, defined as

$$E_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t), s_{t+1}) \right] \tag{2.2}$$

This policy guides the agent's behaviour in the environment, determining which actions to take in each state to achieve the highest cumulative discounted reward.

Kaelbling et al. [8] highlight two fundamental functions used to evaluate policies: the state-value function $V^\pi(s)$, which represents the expected return when starting in a state s and following policy π, and the action-value function $Q^\pi(s, a)$, which represents the expected return when starting in a state s, taking action a, and following policy π thereafter. These functions form the basis of many Reinforcement Learning algorithms.

2.1.3 Model-based vs Model-free Methods

In Reinforcement Learning, algorithms can be categorised into two main approaches based on how they learn and make decisions: model-based and model-free approaches. This distinction is based on whether the agent explicitly learns a model of the dynamics of the environment or directly learns optimal behaviours without modelling the environment.

Model-based methods, as described by Sutton and Barto [6], involve learning an explicit model of the environment that predicts state transitions and rewards. This model represents the agent's understanding of how the environment responds to its actions. Once a model is learned, the agent can use it to simulate experiences and plan without directly interacting with the robot's actual environment. Using the learned model, the agent can improve its ability by considering actions that allow it to achieve the best rewards. The key advantage of model-based approaches is their high efficiency. They typically require fewer interactions with the environment to learn effective policies because they can generate synthetic experiences via simulations.

Notable model-based algorithms include the Monte Carlo Tree Search (MCTS), which builds a search tree to find optimal decisions by simulating possible future states, and Dyna, which integrates both real and simulated experiences from the learned model to update the policy. As highlighted by Arulkumaran et al. [7], model-based methods are particularly valuable in domains where interactions with the environment are costly, such as robotics.

In contrast, model-free methods learn directly from experience without explicitly modelling the environmental dynamics. These approaches focus on estimating value functions or directly optimising policies based on observed rewards and state transitions. Sutton and Barto [6] emphasised that model-free methods are often simpler to implement and can be more effective in complex environments where accurate models are difficult to learn.

Model-free methods can be divided into two main categories based on how the policy is updated: on-policy and off-policy methods. On-policy methods, such as SARSA, update the policy based on the actions taken by the same current policy. In contrast, off-policy methods, such as Q-learning, consider two policies: the target policy, which is to be improved, and the behaviour policy, which the agent follows and uses to generate the data. This allows off-policy methods to learn from experiences generated by different policies, thereby enhancing the exploration and learning efficiency.

The most famous model-free algorithm is Q-learning and its variants. Q-learning is a fundamental model-free algorithm that learns the value function by mapping the state-action pairs to the expected returns. Its simplicity and effectiveness make it a popular choice for various Reinforcement Learning tasks.

The choice between model-based and model-free approaches is based on the specific problem and available resources. Model-based methods typically offer better sample efficiency but may suffer from model bias if the learned model is inaccurate. Model-free methods are generally straightforward to implement and can handle complex environments; however, they often require more interaction with the environment to achieve comparable performance. Recent research, as noted by both Sutton and Barto [6] and Arulkumaran et al. [7], has focused on hybrid approaches that combine the strengths of both paradigms to develop more efficient and effective Reinforcement Learning algorithms.

Both approaches play important roles in the context of robotic manipulation. Model-based methods can reduce the need for extensive real-world training, which is particularly valuable when working with physical robots in real-world environments. However, the complexity of modelling physical interactions in manipulation tasks often makes pure model-free or hybrid approaches more practical for several applications.

2.1.4 Value-based Methods

Another important distinction between Reinforcement Learning algorithms is their representation and optimisation of the agent's behaviour. Three main categories can be identified: value-based, policy-based, and actor-critic methods.

Value-based methods focus on estimating the value of states or state-action pairs to derive optimal policies for the agents. These methods are based on the approximation of the state-value function $V(s)$ and action-value function $Q(s, a)$.

As described by Pashenkova et al. [10], the core idea of value-based methods is to iteratively improve the estimates of these value functions until they converge to their optimal values. The two classical approaches are value and policy iterations. The value iteration works by repeatedly applying the Bellman optimality equation as follows:

$$V_{i+1}(s) = \max_a \left[R(s, a) + \gamma \sum_{s'} P(s'|s, a) V_i(s') \right] \tag{2.3}$$

This update rule propagates the value information backward from the future states to the current states. In contrast, policy iteration alternates between policy evaluation (computing the value function for the current policy) and policy improvement (making the policy greedy with respect to the current value function). Pashenkova et al. [10] analyse these algorithms in terms of their computational complexity and convergence properties, noting that while policy iteration often requires fewer iterations, each iteration can be more computationally intensive.

A particularly influential value-based algorithm is the previously mentioned Q-learning, which learns the optimal action-value function directly as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{2.4}$$

where α is the learning rate. Q-learning is an off-policy algorithm, which means that it can learn the optimal policy while following a different behaviour policy, making it particularly efficient.

The integration of deep neural networks with value-based methods has significantly expanded their applicability to complex problems that involve high-dimensional state spaces. Wang et al. [11] introduce the Duelling Network Architecture, a novel neural network architecture designed specifically for value-based Reinforcement Learning. This architecture explicitly separates the representation of the state values from the action advantages.

$$Q(s, a) = V(s) + A(s, a) - \frac{1}{|A|} \sum_{a'} A(s, a') \tag{2.5}$$

where $V(s)$ estimates the value of being in state s and $A(s, a)$ estimates the relative advantage of taking action a in state s. The advantage function, defined as $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$, measures how much better a specific action is compared to the average performance of the policy in the state. Wang et al. [11] demonstrate that this decomposition helps the network learn which states are valuable without having to learn the effect of each action in each state, leading to more efficient learning in many environments. The authors showed that this architecture consistently outperformed standard Q networks in multiple Atari games, particularly in environments with many actions.

Another significant advancement in value-based methods is the Deep Q-Network (DQN), which combines Q-learning with deep convolutional neural networks to handle high-dimensional visual inputs. DQN employs two key mechanisms to stabilise learning: experience replay, which breaks the correlation between consecutive samples by randomly sampling from a buffer of past experiences, and target networks, which reduce the moving-target problem by periodically updating a separate network used to compute target values.

2.1.5 Policy-based Methods

While value-based methods focus on learning value functions and deriving policies from them, policy-based methods take a different approach by directly parameterising and optimising the policy itself. As Peters [12] explains, these methods work by explicitly representing the policy as a parameterised function $\pi_\theta(a|s)$, which gives the probability of taking action a in state s under the policy parameters θ. The goal is to determine the parameters that maximise expected returns.

The fundamental theoretical foundation of policy-based methods is the Policy Gradient Theorem, which provides an analytical expression for the gradient of the expected return with respect to the policy parameters. According to Peters [12], this gradient can be expressed as

$$\nabla_\theta J(\theta) = \mathbb{E}_{(s,a)\sim\rho_{\pi_\theta}} \left[\nabla_\theta \log \pi_\theta(a|s) \cdot Q^{\pi_\theta}(s, a)\right] \tag{2.6}$$

where $J(\theta)$ is the expected return and $Q^{\pi_\theta}(s, a)$ is the action-value function for the current policy. This formulation allows for the direct optimisation of the policy through gradient ascent, without the need to compute the value functions.

One of the most fundamental policy gradient algorithms is REINFORCE, which estimates the gradient using Monte Carlo (MC) sampling. The update rule for REINFORCE is the following:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot G_t \tag{2.7}$$

where G_t is the observed return starting from time step t and α is the learning rate of the model. Peters [12] noted that while REINFORCE is unbiased, it often suffers from high variance in gradient estimates, leading to slow and unstable learning.

To address these limitations, various improvements have been made to the policy gradient methods. Schulman et al. [13] introduced Proximal Policy Optimisation (PPO), a significant advancement that offers better sample efficiency and stability. PPO addresses a key challenge in policy optimisation: determining the appropriate step size for policy

updates. Taking a step that is too large can lead to destructive updates that collapse the performance, whereas steps that are too small result in slow learning.

As described by Schulman et al. [13], PPO achieves this balance through a clipped objective function as follows:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right] \tag{2.8}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the ratio of the probability of the action under the new policy to that under the old policy, A_t is the advantage estimate at time t, and ϵ is a hyperparameter that controls the clipping range. This clipping mechanism prevents excessive policy updates by limiting the incentive for the new policy to deviate significantly from the previous one.

Schulman et al. [13] demonstrate that PPO achieves state-of-the-art performance across a variety of benchmark tasks while being much simpler to implement and tune than previous policy optimisation methods. This combination of simplicity and effectiveness has made PPO one of the most widely used Reinforcement Learning algorithms in practice, particularly for continuous control tasks and robotics applications.

2.1.6 Actor-Critic Methods

However, both Value-based and Policy-based methods have their limitations. Value-based methods struggle with high-dimensional action spaces and may not effectively handle continuous actions. On the other hand, policy-based methods can suffer from high variances in gradient estimates, leading to slow and unstable learning.

Actor-critic methods represent a hybrid approach that combines the strengths of value-based and policy-based methods. As their name suggests, these algorithms consist of two primary components: an actor that determines the policy (how to act) and a critic that evaluates the policy by estimating value functions.

The seminal work of Konda and Tsitsiklis [14] established the theoretical foundations of actor–critic algorithms. In their formulation, the critic estimates the action-value function $Q^\pi(s,a)$ or the advantage function $A^\pi(s,a)$, which is then used to guide the actor's policy updates.

The update mechanism for the basic actor-critic method can be formulated as follows.

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{2.9}$$
$$\theta_v \leftarrow \theta_v + \alpha_v \delta_t \nabla_{\theta_v} V(s_t) \tag{2.10}$$
$$\theta_\pi \leftarrow \theta_\pi + \alpha_\pi \delta_t \nabla_{\theta_\pi} \log \pi(a_t|s_t) \tag{2.11}$$

where δ_t is the temporal difference (TD) error, θ_v is the parameter of the value function approximator, and θ_π is the policy parameter. The temporal difference (TD) error represents the difference between the observed reward and expected value, providing a signal for both the actor and critic to update their parameters. The critic updates the value function to minimise the TD error, whereas the actor updates the policy in the direction suggested by it.

Actor-critic methods offer several advantages over pure policy or value based approaches. By using a critic to provide a baseline for policy updates, they significantly reduced the variance of the gradient estimates compared to methods such as REINFORCE, leading to more stable and efficient learning. Additionally, they can operate in an online, step-by-step manner rather than requiring complete episodes for updates, making them more suitable for continuous tasks and environments in which the episodes may be very long.

These methods have been particularly successful in robotics applications because of their ability to handle continuous action spaces and sampling efficiency. In the context of robotic manipulation, actor-critic methods can learn fine-grained control policies while making efficient use of interaction data, which is often expensive to collect in physical systems.

Several advanced actor-critic algorithms have emerged in recent years, with notable examples including the Deep Deterministic Policy Gradient (DDPG) and Soft Actor-Critic (SAC) algorithms. DDPG extends the actor-critic method to deep Reinforcement Learning with continuous action spaces using deterministic policy gradients and techniques, such as target networks and experience replay, to stabilise learning.

Although Actor-Critic methods have shown impressive performance, they also face challenges. Balancing the learning rates of the actor and critic is crucial for stable convergence because the two components must learn at appropriate speeds. Additionally, these methods can suffer from high bias in their value estimates if the critic approximation is inaccurate.

More recent algorithms have introduced various improvements to address these challenges, including entropy regularisation to encourage exploration, multiple critics to reduce the overestimation bias, and sophisticated architectures that better handle complex state spaces.

Soft Actor-Critic (SAC)

Soft Actor-Critic (SAC) is an advanced off-policy actor-critic algorithm introduced by Haarnoja et al. [15] which represents a significant innovation in Reinforcement Learning for continuous action spaces. Unlike traditional Reinforcement Learning algorithms, which seek to maximise the expected cumulative reward, SAC incorporates an entropy maximisation objective that encourages exploration by rewarding the agent for behaving randomly, in addition to seeking high rewards.

The foundation of SAC lies in the maximum entropy Reinforcement Learning framework, where the objective is to maximise both the expected return and entropy of the policy:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \right] \tag{2.12}$$

where $\mathcal{H}(\pi(\cdot|s_t))$ is the entropy of the policy at state s_t, and α is a temperature parameter that determines the relative importance of the entropy versus the reward. This entropy term encourages exploration by rewarding the agent for maintaining stochastic

behaviour, which helps to prevent premature convergence to suboptimal deterministic policies.

As Haarnoja et al. [15] explained, SAC employs a stochastic policy that outputs a probability distribution over actions, which is typically represented as a Gaussian distribution in continuous action spaces. The algorithm concurrently learns three functions as follows:

- The policy (actor) $\pi_\phi(a|s)$, which is responsible for selecting actions based on the current state.
- Two Q-value functions (critics) $Q_{\theta_1}(s,a)$ and $Q_{\theta_2}(s,a)$, which estimate the expected return of taking action a in state s.

The use of two Q-networks is a critical design choice that addresses the overestimation bias common in value-based methods by taking the minimum of the two Q-values:

$$\hat{Q}(s_t, a_t) = \min_{i=1,2} Q_{\theta_i}(s_t, a_t) \tag{2.13}$$

The Q-functions are updated to minimise the soft Bellman residual as follows:

$$J_Q(\theta_i) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_{\theta_i}(s_t,a_t) - \left(r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}}\left[V(s_{t+1})\right]\right)\right)^2\right] \tag{2.14}$$

where $\mathcal{D}$ is a replay buffer of past experiences, and the soft value function $V(s_{t+1})$ is implicitly defined as

$$V(s_{t+1}) = \mathbb{E}_{a_{t+1}\sim\pi_\phi}\left[\hat{Q}(s_{t+1},a_{t+1}) - \alpha\log\pi_\phi(a_{t+1}|s_{t+1})\right] \tag{2.15}$$

The policy (actor) is updated to minimise the Kullback-Leibler divergence between the policy and a Boltzmann distribution induced by the Q-function:

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{D}}\left[\mathbb{E}_{a_t\sim\pi_\phi}\left[\alpha\log\pi_\phi(a_t|s_t) - \hat{Q}(s_t,a_t)\right]\right] \tag{2.16}$$

A key innovation in SAC is the automatic tuning of the temperature parameter α, which controls the trade-off between exploration (entropy maximisation) and exploitation (reward maximisation) of the policy. This is achieved by formulating α as a learnable parameter that is adjusted to maintain the target-entropy level.

Haarnoja et al. [15] demonstrate that SAC achieves state-of-the-art performance on a range of continuous control benchmark tasks, outperforming prior methods in terms of sample efficiency, stability, and final performance. The algorithm's robustness to hyperparameter settings makes it particularly valuable for practical applications, including robotics, where sample efficiency is crucial because of the high cost of collecting real-world data required for training.

2.2 Imitation Learning

Imitation Learning (IL) is a subtype of Reinforcement Learning in which an agent learns to perform tasks by observing and mimicking expert demonstrations rather than through explicit programming or trial-and-error interactions with the environment. As defined by Zare et al. [16], imitation learning aims to enable agents to acquire skills or behaviours by observing and replicating the actions of an expert demonstrator and inferring the underlying policy or reward function that describes the expert's behaviour.

The fundamental motivation for imitation learning stems from the challenges inherent in traditional Reinforcement Learning approaches. In complex environments with high-dimensional state and action spaces, sparse rewards, or long-horizon tasks, designing appropriate reward functions is exceedingly difficult. Additionally, many real-world tasks, particularly in robotics, require extensive interactions with the environment to learn effective policies through trial and error, which can be impractical, time-consuming or potentially dangerous. Imitation learning overcomes these challenges by using expert demonstrations to accelerate the learning process and reduce the need for environmental interactions.

According to Zare et al. [16], imitation learning algorithms can be categorised into three primary approaches, each with distinct methodologies and theoretical foundations.

- *Behaviour Cloning (BC)* treats imitation as a supervised learning problem, directly mapping states to actions by mimicking the expert's behaviour through demonstrated state-action pairs.

- *Adversarial Imitation Learning (AIL)* formulates the imitation problem as a distribution matching task, where the goal is to train a policy that generates state-action distributions indistinguishable from those of the expert demonstrations.

- *Inverse Reinforcement Learning (IRL)* aims to infer the underlying reward function that explains the expert's behaviour, under the assumption that the expert is optimising some unknown reward function. Once this reward function is learned, it can be used with standard Reinforcement Learning algorithms to derive a policy.

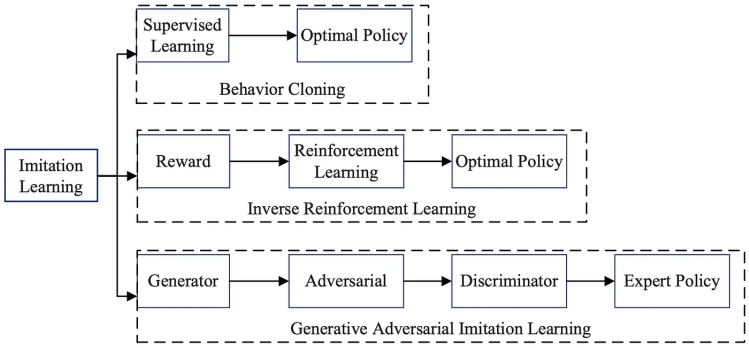


Figure 2.2: Categories of Imitation Learning. The figure illustrates the three main categories of imitation learning: behaviour cloning, adversarial imitation learning, and inverse Reinforcement Learning. Each category has distinct methodologies for learning from expert demonstrations.

The strengths of imitation learning are particularly evident in complex high-dimensional tasks, in which reward engineering is challenging. By observing expert demonstrations, agents can efficiently learn to perform tasks without requiring carefully designed rewards. This approach is especially valuable in robotics and human-computer interaction scenarios, where natural demonstrations can guide the learning process. Furthermore, imitation learning enables agents to acquire behaviours that may be difficult to specify but are easy to demonstrate, such as natural movement patterns or task-specific heuristics.

Despite these advantages, imitation learning has some limitations. As highlighted by Zare et al. [16], one significant challenge is the distribution shift problem, in which the agent encounters states during execution that were not present in the demonstrations, resulting in compounding errors. Additionally, imitation learning approaches often assume that the demonstrations are optimal or near-optimal, which may not always be true. The quality and quantity of demonstrations significantly impact learning outcomes, potentially limiting generalisation to novel scenarios. Moreover, pure imitation approaches may struggle with tasks that require exploration beyond the demonstrated behaviours.

Recent studies on imitation learning have focused on addressing these limitations using hybrid approaches that combine elements from different paradigms. These include methods that integrate Reinforcement Learning with imitation rewards, approaches that handle suboptimal demonstrations, and techniques that enable generalisation to novel environments and tasks. These developments have expanded the applicability of imitation learning to increasingly complex real-world problems, particularly in robotic manipulation tasks, where traditional Reinforcement Learning approaches encounter significant challenges.

2.2.1 Behavior Cloning

Behaviour Cloning (BC) is the first category of Imitation Learning algorithm. This is based on the formulation of the problem as a supervised learning task. As described by Zare et al. [16], behaviour cloning directly learns a mapping from states to actions by

treating expert demonstrations as labelled training data, where states are inputs and the corresponding expert actions are target outputs.

The fundamental principle of behaviour cloning is to minimise the discrepancy between the actions chosen by the learned policy and those demonstrated by the expert. Formally, given a dataset $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^{N}$ of state-action pairs collected from expert demonstrations, Behaviour Cloning aims to find a policy π_θ parameterised by θ that minimises the following objective:

$$\mathcal{L}_{BC}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\pi_\theta(s_i), a_i) \tag{2.17}$$

where $\mathcal{L}$ is a suitable loss function, such as the mean squared error for continuous action spaces or the cross-entropy loss for discrete action spaces. The policy π_θ is typically represented as a neural network that can approximate complex mappings from high-dimensional state spaces to action spaces.

The primary strength of behaviour cloning lies in its simplicity and efficiency. It does not require interaction with the environment during training, making it particularly valuable in scenarios in which such interactions are costly, dangerous, or impractical. For instance, in robotic manipulation tasks, collecting expert demonstrations can be significantly more efficient than allowing robots to explore the environment via trial and error.

However, behaviour cloning suffers from several limitations. The most significant challenge, as noted by Zare et al. [16], is the distribution shift problem, also known as the covariate shift or compounding error. During execution, small deviations from the states observed in the training data can lead the agent to unfamiliar states, where the policy may make errors, potentially resulting in catastrophic failure. This issue arises because the policy is trained only on states from expert trajectories and does not learn how to recover from errors or handle novel situations.

In addition, behaviour cloning typically requires a large amount of demonstration data to achieve good performance, particularly for complex tasks with high-dimensional state spaces. It also assumes that expert demonstrations are optimal or near-optimal, which may not always be the case. Suboptimal demonstrations can lead to the replication of inefficiencies or errors present in expert behaviour.

To address these limitations, several extensions of the basic behaviour cloning framework have been proposed. These include data augmentation techniques to improve generalisation, interactive approaches that allow the agent to query the expert in uncertain states, and hybrid methods that combine behaviour cloning with Reinforcement Learning to enable learning from both demonstrations and environmental interactions.

2.2.2 Adversarial Imitation Learning

Adversarial Imitation Learning (AIL) is the second category of Imitation Learning methodologies. This approach addresses some of the fundamental limitations of the behaviour cloning techniques. As described by Zare et al. [16], AIL formulates the imitation problem as a distribution-matching task, where the objective is to train a policy that generates

state-action distributions that are indistinguishable from those demonstrated by an expert.

The foundational concept behind AIL is inspired by Generative Adversarial Networks (GANs) [17], where two neural networks, a generator and a discriminator, are trained simultaneously in a min-max game. In the context of imitation learning, the policy functions as a generator, producing actions given states, whereas the discriminator attempts to differentiate between state-action pairs from expert demonstrations and those from the learned policy. Formally, this adversarial framework can be expressed as

$$\min_{\pi} \max_{D} \mathbb{E}_{\tau \sim \pi_E}[\log D(\tau)] + \mathbb{E}_{\tau \sim \pi}[\log(1 - D(\tau))] \tag{2.18}$$

where π is the policy being learned, π_E is the expert policy, D is the discriminator, and τ represents the trajectory or state-action pair. The policy aims to minimise this objective, whereas the discriminator aims to maximise it, resulting in a policy that generates trajectories that are indistinguishable from the expert demonstrations.

A key strength of AIL lies in its ability to address the distribution shift problem that affects behavioural cloning. By considering imitation as distribution matching rather than supervised learning, AIL implicitly accounts for the sequential nature of decision-making and the temporal dependencies between states and actions. This allows AIL methods to generalise better to novel situations and recover from errors, as the learned policy is trained to match the expert's state visitation distribution rather than simply mimicking individual actions.

Furthermore, AIL approaches do not require explicit reward function engineering because the discriminator effectively learns an implicit reward function that guides the policy optimisation. This characteristic makes AIL particularly valuable in complex environments in which reward designs are challenging. Additionally, AIL methods can often learn from fewer demonstrations than behaviour cloning, as they leverage the power of Reinforcement Learning to explore and improve on the demonstrated behaviours.

However, AIL has some limitations. As noted by Zare et al. [16], adversarial methods typically involve a more complex optimisation process than behaviour cloning, requiring careful balancing of the generator and discriminator training. This complexity can lead to training instability and sensitivity to hyperparameter settings. Moreover, the adversarial training process is generally more computationally intensive and requires environmental interactions during training, which may be impractical in certain real-world applications, particularly in robotics, where physical interactions are costly.

Recent advancements in AIL have focused on addressing these limitations using various techniques, including more stable adversarial training procedures, sample-efficient policy optimisation, and methods for extracting more information from limited demonstration datasets. These developments continue to enhance the applicability of AIL to increasingly complex imitation learning scenarios, particularly in domains requiring sophisticated behavioural patterns beyond what can be achieved through simple mimicry.

2.2.3 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) is the third category of Imitation Learning. As initially proposed by Russell [18], IRL aims to recover the underlying reward function from expert demonstrations, operating under the assumption that an expert optimises an unknown reward function. This approach is particularly valuable in scenarios where the expert's objectives are complex or implicit, making direct reward specification impractical.

The core idea of IRL is that the reward function can be directly understood from the representations of the task. By inferring this function from demonstrations, IRL can potentially capture an expert's intentions more robustly than methods that directly mimic actions. Formally, given expert demonstrations $\mathcal{D}_E = \{\tau_1, \tau_2, ..., \tau_n\}$ consisting of state-action trajectories, IRL seeks to find a reward function $R(s, a)$ such that the expert's policy π_E is optimal with respect to the reward function. This can be expressed as finding R such that

$$\pi_E = \arg\max_{\pi} \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)\right] \tag{2.19}$$

where γ is the discount factor. Once the reward function is recovered, standard Reinforcement Learning algorithms can be used to derive a policy that optimises the reward.

Trajectories can be represented in different ways, such as sequences of state-action pairs or more structured forms, such as graphs or images, depending on the nature of the task and the available data. The choice of representation can significantly affect the effectiveness of the IRL process.

Graph-based IRL methods leverage structured representations of the environment and tasks in the form of graphs, where nodes typically represent entities or states, and edges represent relationships or transitions. Kumar et al. [19] demonstrated that these structured representations can capture causal relationships and compositional knowledge more explicitly than raw sensory inputs, enabling more sample-efficient and interpretable reward learning. Similarly, Rodin et al. [20] utilised action scene graphs to represent the temporal dynamics of egocentric videos, allowing for more effective modelling of long-term dependencies and sequencing actions. These graph-based approaches excel in domains where the relevant entities and relationships can be readily identified and modelled, providing a compact and human-interpretable representation of the task's structure.

In contrast, image-based IRL methods work directly with visual observations and learn reward functions from raw pixel inputs without requiring explicit feature engineering or structural assumptions to be made. These approaches, exemplified by the work of Ma et al. [21], leverage deep neural networks to extract meaningful representations from visual data, thereby enabling end-to-end learning of reward functions from demonstrations.

The pipeline of Inverse Reinforcement Learning from videos and frame images is composed of two steps:

- Representation Learning: The first step involves learning a compact and informative representation of the visual input, typically using convolutional neural networks (CNNs) or other deep learning architectures. This representation should capture the essential features of the scene that are relevant to the task. In this phase, a reward

function was learned from visual data, allowing the model to associate specific visual features with the corresponding rewards.

- Policy Learning: In the second step, the learned reward function is used to guide the agent's behaviour. The second step aims to learn a policy that maximises the expected reward and can guide the agent to correctly complete the task learned in the Representation Learning phase. The second phase typically involves the use of Reinforcement Learning algorithms to optimise the agent's policy based on the inferred reward function.

The primary advantage of image-based methods is their flexibility and applicability to diverse scenarios without requiring domain-specific knowledge or feature designs. As demonstrated by Zakka et al. [22], image-based IRL can effectively learn cross-embodiment reward functions, allowing robots to learn from demonstrations performed by different embodiments, including humans, which is particularly valuable for robotic manipulation tasks.

For the research presented in this thesis, image-based IRL was selected as the primary approach for the following reasons. First, robotic manipulation tasks typically involve complex visual scenes with numerous objects and spatial relationships that are difficult to model explicitly as graphs without significant domain engineering. Image-based methods offer a more direct path from perception to action, which aligns with the end-to-end learning paradigm that has proven successful in many modern robotic applications. Second, as Jiang et al. [23] observed, although graph-based methods offer interpretability advantages, they often struggle with scalability in diverse environments and require substantial prior knowledge of the relevant entities and relationships. Conversely, image-based approaches can adapt more readily to new scenarios without requiring task-specific graph construction or feature-extraction pipelines.

Furthermore, image-based IRL methods have demonstrated superior performance in cross-embodiment settings, where the expert and learner may have different morphologies or action spaces. This capability is crucial for the long-term goal of enabling robots to learn from human demonstrations because humans and robots inherently possess different embodiments. By learning reward functions directly from visual observations, image-based IRL can abstract the specific details of the demonstrator's embodiment and focus on the task-relevant features of the environment, facilitating a more effective transfer of skills from humans to robots.

A significant strength of IRL lies in its ability to generalise beyond the demonstrated behaviours. By learning the underlying objectives rather than specific actions, IRL methods can adapt to novel situations in which optimal behaviour differs from demonstrations but still serves the same goals. This is particularly valuable in robotics applications, where environments and tasks may vary considerably between the training and deployment conditions. Furthermore, as Kumar et al. [19] noted, IRL provides interpretable reward functions that can offer insights into the decision-making process of experts, potentially facilitating better human-AI collaboration.

However, IRL faces several challenges. This problem is fundamentally ill-posed because multiple reward functions can explain the same observed behaviour, necessitating

additional constraints or assumptions to achieve unique solutions. This challenge can be addressed by optimising the Representation Learning phase such that the learned reward is more robust to variations in the input data and better aligned with the true objectives of the task.

2.3 Representation Learning

Video and image processing form the foundation of modern visual perception systems, particularly in the context of imitation learning from visual demonstrations. As highlighted by Li et al. [24], the ability to effectively extract, encode, and decode visual information is crucial for enabling agents to understand and replicate complex behaviours demonstrated through visual media.

Encoder-decoder architectures are central to modern visual processing frameworks. An encoder transforms a high-dimensional visual input (such as RGB images or video frames) into a lower-dimensional latent representation that captures the essential features of the input. This process, often referred to as feature extraction or embedding, is critical for reducing computational complexity while preserving task-relevant information. As demonstrated by O'Shea and Nash [25], effective encoders should be capable of extracting hierarchical features ranging from low-level patterns (edges and textures) to high-level semantic concepts (object parts, complete objects and spatial relationships).

Complementary to the encoder, a decoder maps the compressed latent representation back to a higher-dimensional space by either reconstructing the original input or generating new visual content. In the context of Reinforcement Learning from visual demonstrations, decoders can be used for various purposes, including action and future state predictions and reward estimation. The encoder-decoder paradigm enables the development of efficient and effective visual representation learning techniques that form the backbone of image-based IRL.

As previously discussed, IRL from visual demonstrations involves two key phases: representation learning and policy optimisation. The effectiveness of the representation learning phase directly affects the quality of the inferred reward function, which, in turn, determines the success of the subsequent Reinforcement Learning phase. By examining various visual processing architectures, this chapter provides the necessary background for understanding how visual demonstrations can be transformed into meaningful representations that capture task-relevant features and relationships, ultimately enabling the effective transfer of skills from human demonstrators to robotic systems.

2.3.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a specialised class of deep neural networks that have revolutionised the field of computer vision and, by extension, visual-based Reinforcement Learning. As described by Li et al. [24], CNNs are designed to automatically and adaptively learn the spatial hierarchies of features from input images using a series of specialised layers. This architecture is inspired by the organisation of the animal visual cortex, where individual neurons respond to stimuli only in a restricted region of the visual

field known as the receptive field.

The fundamental building block of a CNN is the convolutional layer, which applies a set of learnable filters (or kernels) to input data. Each filter is convolved across the width and height of the input volume, computing the dot products between the filter entries and the input at each position, resulting in a feature map that responds to specific patterns or features in the input. O'Shea and Nash [25] explained that these filters typically detect features such as edges, corners, and textures in the early layers, whereas deeper layers capture more complex, abstract features such as object parts or complete objects.

A typical CNN architecture incorporates several components beyond the convolutional layers. Pooling layers perform downsampling operations to reduce the spatial dimensions of feature maps, thereby decreasing the computational requirements and providing translation invariance. Activation functions, typically Rectified Linear Units (ReLU), introduce non-linearity into the network, enabling it to learn complex patterns. Fully connected layers, often positioned at the end of the network, integrate the extracted features for final classification or regression tasks.

The hierarchical feature extraction capability of CNNs makes them particularly well-suited for processing visual inputs in the context of Reinforcement Learning. Mnih et al. [26] demonstrated this in their work on Deep Q-Networks (DQN), where they successfully applied CNNs to learn control policies directly from high-dimensional visual input in Atari games. Their approach processes raw pixels from the game screen through several convolutional layers to extract relevant features, which are then used to approximate the action-value function for Reinforcement Learning. This study established that CNNs can effectively bridge the gap between raw sensory inputs and the abstract representations required for decision-making in complex environments.

In the specific context of imitation and inverse Reinforcement Learning from visual demonstrations, CNNs serve as powerful feature extractors during the representation learning phase. Claessens et al. [27] demonstrated that CNNs can automatically extract state-time features from raw sensory data, eliminating the need for manual feature engineering. This capability is crucial for image-based IRL, where the goal is to learn a reward function directly from visual observations of expert demonstrations.

CNNs have proven particularly valuable for robotic manipulation tasks because they can process visual information about the robot environment, including object positions, orientations, and spatial relationships, without requiring explicit state representations. This aligns with the end-to-end learning paradigm, in which a single network learns to map directly from raw sensory inputs to task-relevant outputs, whether they are action predictions or reward estimates.

Despite their strengths, CNNs have certain limitations that are relevant to visual Reinforcement Learning applications. They typically require large amounts of data for effective training, which can be challenging to obtain in robotics. Additionally, standard CNNs lack explicit mechanisms for modelling temporal dependencies in sequential data, which are important for understanding demonstrations that occur over time. These limitations have motivated the development of more sophisticated architectures, such as residual networks and transformers, as well as specialised temporal learning approaches, such as Time-Contrastive Networks, which are discussed in the subsequent sections.

2.3.2 Residual Networks

Residual Networks (ResNets) represent a significant architectural improvement in deep learning that addresses a fundamental challenge in training deep neural networks. As comprehensively surveyed by Xu et al. [28], the key innovation of ResNets lies in the introduction of skip connections, also known as residual connections, which allow information to bypass one or more layers of a network. This solution mitigates the vanishing gradient problem, which previously limited the depth of effectively trainable neural networks.

The core building block of a ResNet is the residual unit, which can be formally defined as

$$y = F(x, \{W_i\}) + x \tag{2.20}$$

where x and y are the input and output vectors of the layers considered, $F(x, \{W_i\})$ represents the residual mapping to be learned, and the direct connection performing the identity mapping x is the skip connection. This formulation fundamentally transforms the learning objective: rather than directly fitting a desired underlying mapping $H(x)$, the network learns the residual function $F(x) = H(x) - x$. As Xu et al. [28] explained, this reformulation makes optimisation easier because it is typically simpler to optimise the residual mapping than the original, unreferenced mapping.

The practical implementation of ResNets involves stacking multiple residual blocks, each containing several convolutional layers with batch normalisation and ReLU activation functions. Skip connections periodically bypass these blocks, allowing the gradients to flow more effectively during backpropagation. This design enables the training of substantially deeper networks than previously possible, with state-of-the-art implementations reaching depths of over 100 layers while maintaining or improving the performance.

Two particularly influential architectures in the ResNet family are ResNet-18 and ResNet-50, which have become standard backbones for several computer-vision tasks. ResNet-18 consists of 18 layers organised into four main blocks with two residual units each, offering a good balance between computational efficiency and representational power. ResNet-50, with its 50 layers and more complex bottleneck design, provides enhanced feature extraction capabilities at the cost of increased computational requirements. Both architectures employ downsampling between major blocks to reduce the spatial dimensions while increasing the feature channels, thereby creating a hierarchical representation of visual information.

In the context of visual representation learning for imitation learning, ResNets offer several advantages over standard convolutional neural networks (CNNs). The increased depth enables the extraction of more detailed features from visual demonstrations, which is particularly valuable when learning complex behaviours such as driving. As Xu et al. [28] noted, skip connections also facilitate better gradient flow through the network, resulting in more stable training dynamics and faster convergence, which is especially beneficial when working with the limited demonstration data that are often available in imitation learning scenarios.

Furthermore, the hierarchical nature of ResNet representations aligns well with the requirements of inverse Reinforcement Learning, in which different levels of abstraction

may be relevant for reward inference. Lower-level features may capture immediate visual cues, such as object positions, whereas higher-level features may encode more abstract concepts, such as progress toward task completion or relationships between entities. This multi-scale representation capability makes ResNets particularly suitable for extracting meaningful reward-relevant features from visual demonstrations in robotics applications such as Reinforcement Learning.

2.3.3 Vision Transformers

Vision Transformers (ViTs) represent a paradigm shift in computer vision architectures, adapting the transformer model originally designed for natural language processing to visual tasks. As extensively reviewed by Han et al. [29], transformers fundamentally differ from convolutional networks by replacing the local weight-sharing operations of convolution with global self-attention. This architectural distinction enables ViTs to capture long-range dependencies and the global context of images more effectively than CNN-based approaches.

The core innovation of ViT lies in its image processing. Rather than operating directly on pixel-level data, the standard ViT first divides an input image into a sequence of nonoverlapping patches. These patches were linearly projected into an embedding space and augmented with positional encoding to preserve the spatial information. The resulting sequence of patch embeddings was then processed using a series of transformer encoder blocks, each consisting of multi-head self-attention (MHSA) and feed-forward neural network (FFN) layers. The self-attention mechanism can be formally expressed as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.21}$$

where Q, K, and V represent the query, key, and value matrices derived from the input embeddings, respectively, and d_k is a scaling factor equal to the dimensionality of the key vectors. This formulation allows each patch to attend to all other patches, enabling the model to capture the relationships across the entire image regardless of the spatial distance.

A particularly advanced implementation of vision transformers is DINOv2, which was introduced by Oquab et al. [30]. DINOv2 represents a significant advancement in self-supervised visual representation learning and is trained on a diverse dataset of images without human annotation. It employs a teacher-student framework with a knowledge distillation objective, in which the student network learns to match the output distribution of a momentum-updated teacher network. This training approach produces highly transferable visual features that demonstrate strong performance across a wide range of downstream tasks, from image classification to dense prediction tasks such as semantic segmentation.

Compared with ResNets, Vision Transformers offer several distinct advantages. First, the self-attention mechanism provides greater flexibility in capturing the global context and modelling long-range dependencies, which is particularly valuable for understanding complex scenes with multiple interacting objects. Han et al. [29] highlighted that this

capability enables ViTs to better recognise relationships between distant image elements, which traditional CNNs struggle to capture efficiently because of their locality-constrained receptive fields. Second, Vision Transformers demonstrate superior scaling properties. Their performance continues to improve with larger model sizes and more training data, often surpassing the saturation points observed for convolutional architectures.

ViTs also offer computational benefits for certain operations. Unlike CNNs, which process the entire spatial dimension at each layer, transformers process a fixed-length sequence of patch embeddings, making the computational complexity independent of the spatial resolution after the initial patch embedding stage. This property is advantageous when processing high-resolution images or when computational resources must be allocated more flexibly.

However, ViTs have certain limitations compared to ResNets. As noted by Han et al. [29], ViTs typically require larger datasets for training from scratch because of the lack of inductive biases inherent in CNNs, such as translation equivariance and locality. The quadratic complexity of self-attention with respect to the sequence length also poses computational challenges, particularly for high-resolution input or dense prediction tasks. Furthermore, the discrete patch-based representation may not capture fine-grained local structures as effectively as convolutional operations, potentially limiting performance on tasks requiring detailed local feature extraction.

2.3.4 Temporal Cycle-Consistency (TCC)

Temporal cycle consistency (TCC) is a self-supervised learning framework designed to address the challenge of learning representations from temporal data without explicit supervision. As introduced by Dwibedi et al. [31], TCC approaches this challenge through the concept of cycle consistency across time, that is, the idea that corresponding events in different video sequences should be aligned in a temporally consistent manner, regardless of variations in speed, viewpoint, or embodiment.

The fundamental principle underlying TCC is the establishment of dense and fine-grained correspondences between temporal sequences in the data. Unlike methods that rely on global video-level representations, TCC operates at the frame level and learns embeddings in which the distance between frames reflects their temporal alignment in the underlying activity. This is achieved through a cycle-consistency objective that enforces a powerful constraint: if a frame from one video is mapped to a frame in another video and then mapped back, it should return to a temporally close position in the original sequence. Formally, the cycle-consistency constraint can be expressed as

$$d(f_i, f_{cycle}) < \epsilon \tag{2.22}$$

where f_i is the original frame, f_{cycle} is the frame reached after the cycle mapping process, d is a temporal distance metric (typically the frame index difference), and ϵ is a small threshold. This constraint is implemented using a contrastive learning objective that encourages the embedding space to preserve temporal relationships across sequences.

The TCC architecture consists of two primary components: a feature extraction backbone, which can be any suitable neural network, such as ResNet or Vision Transformer,

and a temporal embedding module that transforms the extracted features into a space in which the Euclidean distances correspond to temporal alignment. The training process does not require explicit labels or annotations of the correspondences. Instead, it leverages the natural temporal structure of the video data.

When implemented with different backbone architectures, TCC demonstrates interesting trade-offs. When paired with ResNet architectures, the TCC benefits from the strong spatial feature extraction capabilities and inductive biases of CNNs, resulting in stable training and good performance in tasks where spatial features are predominant. As noted by Dwibedi et al. [31], the hierarchical nature of the ResNet features provides a multiscale representation that can capture both fine-grained spatial details and higher-level semantic concepts, which is valuable for establishing correspondences across visually diverse sequences.

Alternatively, when combined with ViT architectures, such as DINOv2, TCC can leverage the global contextual understanding and long-range dependency modelling capabilities of transformers. This configuration excels in scenarios involving complex scenes with multiple interacting elements, in which the global context is crucial for understanding the temporal progression of activities.

However, each configuration presents its own specific challenges. ResNet-based implementations may struggle with long-range temporal dependencies and scenarios that require a global context understanding. Transformer-based implementations, while more flexible in modelling complex relationships, typically require more computational resources and larger datasets for effective training. They may also lack the strong spatial inductive biases that make CNNs effective for visual tasks with limited amounts of data.

Chapter 3

Related Work

In this chapter, we review the existing literature on imitation learning, Procedural Learning and Intrinsic Reward and their applications in robotics. In detail, we focus on the key methodologies, findings, and challenges identified in the literature and highlight the most relevant works in each area. The aim is to provide a comprehensive overview of the state-of-the-art in these fields and to identify potential avenues for future research.

3.1 Imitation Learning in Robotics

The evolution of imitation learning in robotics has been driven by the necessity to overcome several critical limitations of traditional Reinforcement Learning approaches. As discussed in Section 2.2, the challenge of reward engineering becomes particularly relevant in robotic manipulation tasks, where the desired behaviours are often easier to demonstrate than specify. Furthermore, the high cost and potential safety risks associated with exploratory interactions in physical systems make learning from demonstrations an attractive alternative to trial and error learning.

The application of imitation learning in robotics faces several fundamental challenges that have shaped the development of the field. The use of images and videos in inverse Reinforcement Learning introduces complications caused by variations in the viewpoint, brightness changes, background clutter, and occlusions. These factors can significantly complicate the learning process and require sophisticated algorithms to effectively utilise visual information. Additionally, the diverse nature of videos and embodiments introduces substantial variability in the demonstrations that learning algorithms must handle robustly.

The methodology has evolved in several key directions, each addressing different aspects of these fundamental challenges. Cross-embodiment learning has emerged as a critical capability that enables robots to learn from demonstrations performed by different embodiments, including humans. This advancement addresses the practical need for scalable solutions that can leverage demonstrations from various sources without requiring an embodiment-specific data collection. Zakka et al. [22] introduced Cross-Embodiment Inverse Reinforcement Learning (XIRL), which addresses the fundamental problem of

learning reward functions from demonstrations across different embodiments without requiring action labels or correspondence annotations. Their approach leverages temporal cycle consistency to align the corresponding task phases across demonstration videos, regardless of the demonstrator's morphology. The key innovation lies in learning embeddings, where temporal distances in the embedding space correspond to the semantic alignment between different sequences. The method demonstrates significant improvements in cross-embodiment transfer, allowing robots to learn manipulation skills from human demonstrations without requiring robot-specific training data.

Liu et al. [32] introduced the Imitation from Observation (IfO) framework, which enables agents to learn behaviors directly from raw video demonstrations without requiring expert action labels. Their method relies on a context translation model that maps expert demonstrations into the learner's environment, effectively bridging domain gaps, such as differences in viewpoints or backgrounds. This approach highlights the potential of purely vision-based imitation learning, particularly in scenarios in which multimodal sensory data or explicit action information are unavailable.

Parallel developments in visual representation learning have focused on extracting meaningful reward functions directly from such demonstrations. Das et al. [33] developed a model-based approach for inverse Reinforcement Learning from visual demonstrations that combines unsupervised keypoint detection with learned visual dynamics models. Their method addresses the challenge of learning stable reward functions from raw visual data without requiring action labels or robot-specific demonstrations. The approach first learns to detect semantically meaningful keypoints in demonstration videos using unsupervised techniques and then constructs a dynamics model that predicts keypoint trajectories over time. The reward function is formulated based on the alignment between the observed and predicted keypoint trajectories, providing a stable learning signal that is less sensitive to visual variations than pixel-level comparisons.

The challenge of distribution shift, fundamental to behaviour cloning approaches, as discussed in 2.2.1, has motivated the development of goal-oriented learning frameworks. Yang et al. [34] proposed a framework for predicting goal-directed human attention using inverse Reinforcement Learning. Unlike methods that directly imitate observed trajectories, their approach infers the latent objectives driving expert behaviour by modelling attention as a reward-guided process. Specifically, they used a hierarchical goal-conditioned model that first identified a human's likely goal from contextual cues and then predicted the attention patterns that would be optimal for achieving that goal. This formulation improves robustness to changes in the initial conditions and environmental variations while also handling scenarios in which multiple trajectories can achieve the same outcome. By focusing on goal-directed allocation of attention rather than the exact action sequence, the method demonstrates stronger generalisation in complex environments.

Recent advances have also addressed the scalability limitations of traditional imitation learning using methods that leverage large-scale, unlabelled human demonstration videos. Frameworks such as HOLD [35] learn goal-conditioned reward functions by modelling functional distances between states, enabling robots to generalise across unseen embodiments and environments. By exploiting the multimodality and diversity of human video datasets, these approaches substantially reduce the supervision and data collection burden typical of imitation learning, establishing new paradigms for reward learning from

raw observation sequences.

These diverse developments demonstrate several important trends that are directly connected to the fundamental concepts discussed in Chapter 2. The progression from single-embodiment to cross-embodiment learning reflects practical scalability needs, whereas the emphasis on visual representation learning directly leverages advances in CNNs, ResNets, and Vision Transformers discussed in 2.3. The temporal consistency objectives employed by several methods are closely related to the temporal cycle-consistency framework discussed in 2.3.4, demonstrating how advances in representation learning enable more effective imitation learning.

3.2 Procedural Learning in Robotics

Procedural Learning is a fundamental paradigm for managing complex, long-horizon tasks by decomposing them into manageable subtasks that can be learned and executed sequentially. This approach relies on the assumption that many complex behaviours can be understood as compositions of simpler atomic actions or skills. This decomposition approach addresses one of the most significant challenges in robotics and artificial intelligence: the curse of dimensionality and temporal complexity that arise when learning policies for tasks with extended temporal horizons.

The motivation for Procedural Learning arises from several practical and theoretical considerations. Long-horizon tasks often exhibit sparse reward signals, making it difficult for traditional Reinforcement Learning algorithms to establish effective credit assignments across extended sequences of actions. Additionally, the exponential growth in the state-action space complexity with task length renders direct policy learning computationally intractable for many real-world applications. Procedural Learning addresses these challenges by creating intermediate objectives that provide more frequent feedback signals and reduce the complexity of the learning problem.

A central challenge in Procedural Learning is determining appropriate task decomposition strategies. The number and nature of the subtasks significantly impact the learning efficiency and final performance. Too few subtasks may fail to adequately simplify the learning problem, while too many subtasks can introduce unnecessary complexity and coordination challenges between components. Kim et al. [36] demonstrated that the optimal number of subtasks is task-dependent and often requires a careful analysis of the underlying task structure and dynamics.

Several distinct approaches have emerged for implementing Procedural Learning in robotic manipulations. Hierarchical decomposition methods break complex tasks into nested hierarchies of subtasks, where higher-level policies coordinate the execution of lower-level skills. Wang et al. [4] proposed MimicPlay, which learns long-horizon manipulation skills through hierarchical imitation learning, demonstrating how complex assembly tasks can be decomposed into sequences of primitive manipulation actions. This approach leverages human demonstrations to identify natural breakpoints in task execution and create a curriculum of increasingly complex subtasks.

Bootstrap learning approaches represent a particularly sophisticated development in Procedural Learning, where agents learn to generate their own training subtasks through

self-supervised task generation. Yang et al. [37] demonstrate how agents can bootstrap increasingly complex skills by generating intermediate tasks that bridge the gap between current capabilities and target behaviors. This self-directed learning capability reduces the dependence on manually designed curricula and enables more autonomous skill acquisition.

Recent advances have focused on addressing the coordination and transition challenges that are inherent to Procedural Learning systems. Ajay et al. [38] demonstrated how independently learned skills can be effectively combined and coordinated to solve novel tasks using compositional learning frameworks. Their approach addresses the critical challenge of skill composition by demonstrating how modular components can be recombined in new ways without requiring complete retraining.

Temporal segmentation of demonstrations presents another significant challenge in Procedural Learning. Pan et al. [39] developed methods for automatically identifying natural breakpoints in human demonstrations, creating meaningful subtask boundaries that align with the human cognitive segmentation of complex activities. This automated segmentation capability is crucial for scaling Procedural Learning approaches to diverse tasks without requiring extensive manual annotation of the demonstration data.

Marzari et al. [40] propose a hierarchical Reinforcement Learning framework for robotic pick-and-place tasks, where complex manipulations are decomposed into low-level subtasks. Each subtask is parameterised as an expert network trained independently via deep Reinforcement Learning, and a high-level choreographer coordinates their sequential execution. This modular design allows the system to adapt to different initial configurations and environmental uncertainties. Both simulation and real-robot experiments demonstrated significant improvements in sample efficiency and robustness compared with traditional learning-from-demonstration approaches. Their work highlights how hierarchical task decomposition can address the challenges of long-horizon manipulation by leveraging modular-policy learning and hierarchical control mechanisms.

The integration of inverse Reinforcement Learning with procedural decomposition has yielded sophisticated approaches for learning from demonstrations of complex behaviour. Sosic et al. [41] presented methods for learning hierarchical reward functions that capture the multilevel structure of expert demonstrations, enabling agents to understand both immediate action selection and higher-level strategic planning. This hierarchical reward learning approach addresses the challenge of extracting meaningful supervisory signals from demonstrations of complex, multi-stage tasks.

Liu et al. [5] developed curricular methods tailored specifically for Inverse Reinforcement Learning that automatically generate sequences of training tasks with progressively increasing difficulty. By structuring training in this manner, their approach facilitates more effective learning of reward functions in complex environments, helping agents acquire robust foundational skills before progressing to challenging tasks. This curriculum-driven IRL framework is particularly valuable for robotic manipulation, where mastering intricate tasks often depends on learning a hierarchy of motor skills and spatial understanding through intermediate subgoals. Their work demonstrated how adaptive curricular subgoals can significantly enhance the stability and generalisation of reward learning in IRL, paving the way for more scalable and efficient imitation from demonstrations.

The integration of Procedural Learning with modern deep learning architectures has

enabled more sophisticated skill representation and composition mechanisms. These approaches leverage the representational power of deep networks to learn abstract skill embeddings that can be flexibly combined and adapted to novel contexts. This development is particularly important for robotic manipulation, where skills must generalise across variations in object properties, environmental conditions, and task specifications.

In inverse Reinforcement Learning, the combination of Procedural Learning and deep learning has led to improved methods for inferring reward functions from demonstrations. By leveraging the hierarchical structure of skills and the representational power of deep networks, agents can learn to predict rewards for complex behaviours more effectively than traditional RL methods. This integration enhances the agent's ability to generalise from limited demonstrations and adapt to new situations.

These developments collectively demonstrate the maturation of Procedural Learning from simple task decomposition to sophisticated frameworks capable of handling complex real-world problems. The field's progression toward adaptive, self-supervised, and hierarchically organised learning systems provides a foundation for the methodological contributions examined in subsequent chapters, particularly in contexts that require complex manipulation skills and long-horizon planning capabilities.

3.3 Intrinsic Reward in Robotics

Intrinsic Reward mechanisms have emerged as a powerful paradigm for enhancing exploration and learning in Reinforcement Learning (RL) agents, particularly in environments where rewards are sparse or delayed. These mechanisms provide internal motivation signals that encourage agents to find novel states, learn diverse skills, and improve their understanding of the environment, thereby addressing some fundamental challenges associated with traditional RL approaches.

Intrinsic Reward mechanisms can be broadly categorised into several approaches based on their underlying principles. Curiosity-driven exploration encourages agents to seek states or transitions that are difficult to predict or understand, thereby promoting the discovery of novel behaviours and environmental dynamics. Novelty-based approaches reward agents for visiting states that differ significantly from previously encountered states, encouraging comprehensive coverage of the state space. Empowerment-based methods focus on maximising the agent's ability to influence its environment, promoting the acquisition of skills that enhance the agent's control capabilities.

The application of Intrinsic Rewards to robotic manipulation tasks has been explored in various studies, demonstrating their potential to improve learning efficiency and policy robustness. In most of these studies, Intrinsic Reward is used in the representation learning phase to encourage the agent to explore and learn diverse behaviours that can be leveraged for downstream tasks or to allow learning in tasks where the reward signal is weak or absent.

Liu and Abbeel [42] introduced the Active Pre-Training (APT) approach, which leverages unsupervised exploration to enable agents to learn meaningful behaviours without explicit reward signals. This method is based on entropy maximisation in an abstract representation space. The use of Intrinsic Rewards in APT encourages agents to explore

diverse states and actions, leading to the acquisition of a wide range of skills that can be fine-tuned for specific tasks.

Building upon this foundation, Liu and Abbeel [43] further developed Active Pre-training with Successor Features (APS), which combines the benefits of unsupervised exploration with structured representation learning. APS employs successor features to encode information about future state visitation patterns, enabling agents to learn representations that capture the long-term consequences of actions. This approach is especially valuable for manipulation tasks in which understanding the temporal dynamics of object interactions is crucial for effective policy learning. In this case, the Intrinsic Reward signal is derived from the novelty of the successor features, which encourages agents to explore behaviours that lead to diverse future states.

A complementary direction was explored by Yarats et al. [44], who introduced prototypical representations for Reinforcement Learning. By clustering similar states into prototypes, their method provides more stable and structured representations that improve exploration efficiency and skill organisation. Such prototypical abstractions are especially relevant for multi-object manipulation, where capturing coherent interaction patterns across diverse spatial configurations is essential for achieving optimal performance.

While the aforementioned approaches focus on enhancing representation learning through intrinsic motivation, Schneider et al. [45] addressed a complementary challenge by incorporating Intrinsic Rewards directly into the Reinforcement Learning phase. They introduced curiosity-driven exploration based on forward dynamics prediction errors, which encouraged the agent to interact with objects in novel ways and improved sample efficiency in manipulation tasks with sparse rewards. Unlike methods that pre-train diverse skills, their approach integrates exploration and policy learning simultaneously, enabling the discovery of manipulation strategies that may not emerge under standard exploration. This makes their work particularly relevant for frameworks such as inverse Reinforcement Learning, where representation and policy learning are often considered separate stages.

The application of Intrinsic Rewards in robotic manipulation offers several advantages. First, these mechanisms can help agents overcome the exploration challenges inherent in high-dimensional continuous control problems, where random exploration is often inefficient and ineffective. Second, Intrinsic Rewards can facilitate the discovery of useful strategies that may not be immediately understood from task-specific rewards.

However, the implementation of Intrinsic Reward systems presents several challenges. The balance between intrinsic and distance rewards requires careful tuning, as excessive emphasis on intrinsic motivation may lead agents to neglect the task-specific objectives. Additionally, the computational overhead associated with calculating Intrinsic Rewards, particularly those based on prediction errors or state novelty, can be significant in high-dimensional robotic systems such as humanoid robots.

Chapter 4

Methodology

This chapter details the methodologies employed in this thesis, encompassing data collection, environment setup, training procedures, and evaluation metrics. Each section provides a comprehensive overview of the specific techniques and configurations used to implement and assess the proposed approaches.

4.1 INtrinsic Exploration via SubTask Inverse Reinforcement Learning (INEST-IRL)

After reviewing the state-of-the-art in imitation learning, Procedural Learning, and Intrinsic Reward, as discussed in Chapter 3, we identified several key challenges that our methodology aims to address. The primary objective of this thesis is to develop a robust framework for learning complex manipulation tasks from visual demonstrations using inverse Reinforcement Learning (IRL). This framework leverages Procedural Learning techniques to decompose long-horizon tasks into manageable subtasks, facilitating more efficient learning and generalisation, and intrinsic motivation strategies to encourage exploration and skill acquisition.

The specific task we focus on is the "Sweep to top in order" task, as described in 4.1. This task is a variant of the standard "Sweep to top" task and requires an agent to manipulate objects within a simulated environment and move them to a designated target area in a given order. In this variant, the order is defined by the colour of the block. In detail, the task focuses on moving three coloured blocks (Red, Blue and Yellow) to a green target area located at the top of the environment in this specific order: first the red block, then the blue block, and finally the yellow block. The agent must learn to perform this task effectively based on visual observations alone, without access to explicit state information or action labels.

This task was selected for several reasons. First, it represents a fundamental manipulation challenge relevant to many real-world applications, such as sorting and organising objects. Second, the complexity of the task arises from the need to coordinate multiple actions over an extended temporal horizon, making it an ideal candidate for Procedural Learning approaches. Finally, the structure of the task allows us to understand how well

the learned reward functions can guide behaviour in a sequential decision-making context.

4.2 Training Procedure

As described in 2.2.3, inverse Reinforcement Learning is a two-step process: first, we train a reward function from demonstrations in the representation learning phase, and then we use this learned reward to train a policy using Reinforcement Learning in the policy learning phase.

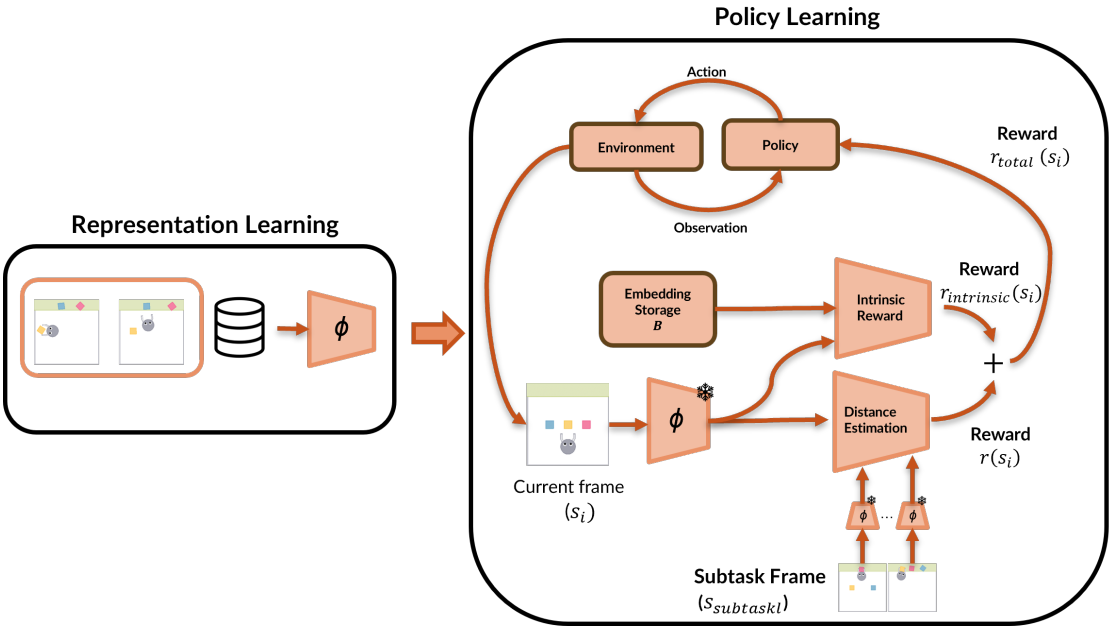


Figure 4.1: Overview of the training pipeline. The process consists of two main phases: representation learning from demonstration videos to learn a reward function, and policy learning using the learned reward to train an agent via Reinforcement Learning.

4.2.1 Representation Learning

The representation learning phase is dedicated to learning a reward function and robust visual representations from demonstration videos, which serve as the foundation for the subsequent policy learning phase. This phase leverages self-supervised learning techniques to extract task-relevant features without requiring manual annotations or explicit supervision.

The core of the representation learning process is an encoder network, which can be instantiated as either a ResNet or a Vision Transformer (ViT), depending on the objective of the model, as discussed in 2.3. For this study, the chosen encoder is a ResNet-18, because it strikes a good balance between complexity and performance for the visual tasks at hand. The encoder is trained using a loss function that guides the learning

process. In this study, the Temporal Cycle-Consistency (TCC) loss was chosen as the loss function, which encourages the network to produce embeddings that are temporally aligned across different demonstration sequences. This approach enables the model to learn representations that are invariant to changes in the viewpoint, embodiment, and environmental conditions, capturing the underlying structure of the manipulation task.

During the representation learning phase, a demonstration dataset was used to sample batches of frames from different videos. In this case, the TCC loss is applied to ensure that temporally corresponding frames are mapped to similar points in the embedding-space. This process facilitates the emergence of features that are sensitive to task progress and subtask boundaries, which are critical for effective reward inference in the policy learning phase.

By the end of the representation learning phase, the encoder produces compact and informative embeddings that summarise the essential aspects of the manipulation task, providing a strong basis for learning reward functions that generalise across diverse scenarios and agent embodiments.

After the representation learning phase, a copy of the embedding representation for each subtask was stored. These are used in the policy learning phase for reward estimation, as described in 4.2.3.

4.2.2 Policy Learning

The policy learning phase focuses on utilising the pre-trained encoder and learned reward function to guide policy optimisation for the task. This phase involves integrating the learned visual representations with Reinforcement Learning algorithms to learn a policy that can effectively complete a task based on inferred rewards.

The policy learning process begins by freezing the weights of the pretrained encoder, ensuring that the learned representations remain stable during policy learning. The learned reward function is then employed to provide feedback to the agent based on its actions and the resulting state transition. This reward signal is crucial for guiding the agent towards behaviours that align with the demonstrated task.

In this study, we employed a Soft-Actor Critic (SAC) algorithm for policy optimisation, as it is well-suited for continuous action spaces and has demonstrated strong performance in various robotic manipulation tasks. The choice of SAC is motivated by its ability to balance exploration and exploitation through entropy maximisation, which is particularly beneficial in environments with complex dynamics and sparse rewards, as reported in Section 2.1.6.

During training, the agent interacts with the environment and collects experience tuples consisting of observations, actions, rewards, and next states. The Reinforcement Learning algorithm uses this experience to update the policy network, optimising it to maximise the cumulative reward over time.

The training process is iterative, with the agent continually refining its policy based on feedback from the learned reward function. This iterative optimisation continues until the agent achieves a satisfactory performance on the task, as measured by the success rates or other relevant metrics.

4.2.3 Reward estimation

The reward estimation process is a critical component of the training phase, as it directly influences the agent's learning trajectory and ultimate performance on the task. The learned reward function, derived from the pretrained encoder, provides a continuous feedback signal that guides the agent's behaviour towards achieving the desired manipulation objectives.

The reward function is designed to capture the essential aspects of task progress, providing higher rewards for actions that bring the agent closer to completing the task and lower rewards for actions that deviate from the desired behaviour. The implementation consists of three main components: embedding distance computation, reward transformation and subtask completion detection.

In this study, the reward was estimated based on the embeddings produced by the frozen encoder. The current observation is passed through an encoder to obtain its embedding. This embedding is then compared to the stored embeddings for each subtask, which are obtained during the representation learning phase. Distance computation is performed using the Euclidean norm:

$$d = ||\mathbf{e}_{current} - \mathbf{e}_{goal}||_2 \tag{4.1}$$

where $\mathbf{e}_{current}$ represents the embedding of the current observation, and $\mathbf{e}_{goal}$ represents the stored embedding for the current subtask. This distance metric quantifies how closely the current state aligns with the target subtask states in the learned embedding space.

The raw Euclidean distance was transformed into a reward signal using a carefully designed function that provided smooth gradients and appropriate reward shaping. The distance-to-reward transformation follows a composite function that combines quadratic and square root terms as follows:

$$r(d) = -\alpha d^2 - \beta\sqrt{d^2 + \gamma} \tag{4.2}$$

where $\alpha = 0.001$, $\beta = 0.01$, and $\gamma = 0.001$ are hyperparameters that control the reward landscape. The quadratic term $-\alpha d^2$ provides strong negative rewards for large distances, encouraging the agent to move towards the target embedding. The square root term $-\beta\sqrt{d^2 + \gamma}$ ensures smoothness near the origin and prevents numerical instability when the distance approaches zero. The small constant γ acts as a regularisation term that maintains differentiability at $d = 0$.

This dual-component design creates a reward landscape that is both informative and stable for gradient-based policy optimisation. The quadratic component dominates for large distances, providing strong directional signals, while the square root component ensures smooth behaviour in the vicinity of the target state.

Procedural Learning Integration

To effectively manage the complexity of long-horizon manipulation tasks, the reward function incorporates a Procedural Learning mechanism that tracks task progress through a

sequence of subtasks. Each subtask corresponds to a specific phase of the manipulation task: picking up and placing each coloured block in the correct order. The system maintains a subtask index that progresses as the agent completes each subtask.

To encourage progression through the task sequence, a bonus reward component was added to the distance-based reward:

$$r_{total} = r(d) + s \cdot c_{subtask} \tag{4.3}$$

where $c_{subtask}$ is a constant subtask cost that provides incrementally higher rewards as the agent advances through the task sequence. This bonus structure ensures that the later subtasks receive higher baseline rewards, preventing the agent from becoming stuck in the early phases of the task.

The system employs a threshold-based mechanism to detect subtask completion, which prevents premature transitions due to noisy reward signals. For each subtask, a specific threshold was defined based on an empirical analysis of the reward distribution during successful task execution. The thresholds vary across the subtasks to account for the different levels of precision required for each manipulation phase.

The completion detection mechanism requires the reward to remain above the threshold for a consecutive number of steps before triggering the subtask transition. This approach, implemented through a counter-based system, ensures robustness against temporary fluctuations in the reward signal and prevents oscillatory behaviour between the subtasks.

When the reward $r(d)$ exceeds the threshold for the current subtask, the counter is incremented. If the counter reaches the required duration, the subtask index is incremented and the counter is reset. If the reward falls below the threshold before reaching the hold duration, the counter is reset to zero, requiring the agent to maintain a consistent performance before progressing.

Upon completion of all subtasks ($s \geq 6$), the reward function transitions to a constant, high-reward state:

$$r_{final} = s \cdot c_{subtask} \tag{4.4}$$

This terminal reward structure ensures that the agent receives the maximum reward upon successful task completion while maintaining the progressive bonus structure established during training.

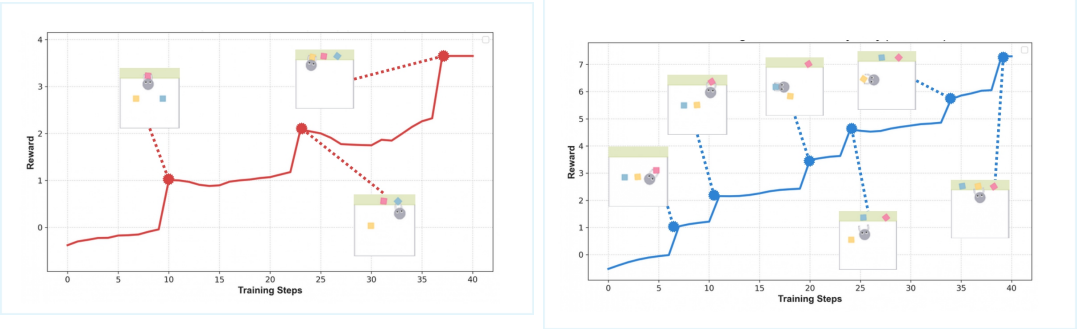


Figure 4.2: Illustration of the reward function evolution through the task sequence. The left panel shows the reward progression with three subtasks, whereas the right panel illustrates the reward structure with six subtasks. Each image in the plot indicates the completion of a subtask phase. The reward function is designed to provide increasing incentives as the agent progresses through the subtasks, culminating in a high reward upon task completion.

Intrinsic Reward Integration

The Intrinsic Reward mechanism implemented encourages the exploration of diverse embedding states during policy learning. This system addresses the fundamental challenge of maintaining adequate exploration throughout the learning process, particularly in long-horizon manipulation tasks, where agents may become trapped in suboptimal local minima or exhibit insufficient behavioural diversity within individual subtasks.

The core of the Intrinsic Reward is based on measuring the novelty of the visited states in the learned embedding space. Unlike traditional state-based novelty metrics that operate directly on raw observations, this approach leverages the structured representation learned during the representation learning phase to compute more meaningful novelty signals. The system maintains separate memory buffers for each subtask, storing the embeddings of the previously visited states within that specific phase of the task.

For each subtask s, a memory buffer $\mathcal{M}_s$ is maintained with a maximum capacity of 5,000 embedding. When a new embedding $\mathbf{e}_{current}$ is observed, it is compared with the stored embeddings in the current subtask's memory buffer. The novelty reward is computed based on the smaller distance between the current embedding and the previously stored embeddings for the same subtask. An higher value corresponds an high exploration.

$$d_B = \|\phi(s_i) - B\|^2 \tag{4.5}$$

$$r_{intrinsic}(s_i) = \|\phi(s_i) - \min(d_B)\|^2 \tag{4.6}$$

This formulation ensures that states similar to previously visited states receive lower novelty rewards, whereas novel states in the embedding space receive higher rewards, encouraging the agent to explore diverse behaviours within each subtask.

The contribution of Intrinsic Rewards to the total reward signal is dynamically adjusted based on task progress and subtask transition. The Intrinsic Reward component is integrated into the total reward as follows:

$$r_{total} = r_{distance} + \alpha_{intrinsic} \cdot r_{intrinsic} \tag{4.7}$$

where $\alpha_{intrinsic}$ is the adaptive scaling factor. Under normal conditions, $\alpha_{intrinsic} = 0.2$, which provides moderate encouragement for exploration. However, immediately following subtask transitions (for n time steps after completion), the scaling factor is increased to $\alpha_{intrinsic} = 0.4$ to encourage more aggressive exploration in the new subtask phase.

$$\alpha_{intrinsic} = \begin{cases} 0.4 & \text{if } t - t_{transition} < n \\ 0.2 & \text{otherwise} \end{cases} \tag{4.8}$$

where t is the current time step, and $t_{transition}$ is the time step of the most recent subtask transition. This adaptive mechanism addresses the exploration challenge that arises when transitioning between subtasks, where the agent must discover new behaviours that are appropriate for the next phase of the task.

To evaluate the effectiveness of the Intrinsic Reward mechanism, the approach includes a comprehensive coverage analysis framework that tracks the exploration patterns across multiple dimensions. Coverage analysis employs two complementary approaches: similarity-preserving grid mapping and hash-based unique state counting.

Similarity-preserving grid mapping projects high-dimensional embeddings onto a 2D grid while preserving the similarity structure of the original embedding space. This is achieved through Principal Component Analysis (PCA) on a buffer of collected embeddings:

$$\mathbf{p}_{2D} = (\mathbf{e} - \boldsymbol{\mu})\mathbf{W}_{PCA} \tag{4.9}$$

where $\boldsymbol{\mu}$ is the mean of the collected embeddings, $\mathbf{W}_{PCA}$ contains the first two principal components, and $\mathbf{p}_{2D}$ is the 2D projection. The projected coordinates were then normalised and discretised to create grid coordinates as follows:

$$\text{grid}_{coords} = \left\lfloor \frac{\mathbf{p}_{2D} - \mathbf{p}_{min}}{\mathbf{p}_{max} - \mathbf{p}_{min}} \cdot (G - 1) \right\rfloor \tag{4.10}$$

where $G = 100$ is the grid size, and $\mathbf{p}_{min}$ and $\mathbf{p}_{max}$ are the bounds of the projected space. This approach creates a coverage grid that maintains the topological relationships in the embedding space while enabling efficient coverage tracking.

In addition to the grid-based coverage analysis, the system maintains a hash-based tracking mechanism for more precise and unique state counting. The embeddings were rounded to three decimal places and converted to hash values.

$$\text{hash}(\mathbf{e}) = \text{hash}(\text{round}(\mathbf{e}, 3)) \tag{4.11}$$

This approach provides an exact count of the unique states visited while remaining computationally efficient. The combination of grid-based and hash-based tracking provides both coarse-grained spatial analysis and fine-grained uniqueness detection.

To ensure computational efficiency during training, the system implements several memory-management strategies. The embedding memory buffers use a first-in-first-out (FIFO) policy when the maximum capacity is reached, ensuring that the most recent experiences are retained while maintaining a bounded memory usage.

The Intrinsic Reward computation is designed to be sufficiently efficient for real-time policy learning, with the smallest distance search implemented using efficient distance computations and coverage analysis performed asynchronously to avoid affecting the training speed.

Chapter 5

Experiments and Results

This chapter presents the experimental setup, results, and analyses of the proposed method. The experiments were designed to evaluate the effectiveness of the proposed method in completing tasks using the described environment. We conducted a series of experiments to assess the performance of different baseline architectures, ablation studies to understand the impact of various components, and a comprehensive analysis of the results.

5.1 Experiment setup

5.1.1 Environment used and Data Collection

The X-Magical environment, originally introduced by Toyer et al. [46], provides a controlled simulation platform for studying embodied AI challenges, particularly those involving cross-embodiment imitation learning. This environment was selected for our experiments because of its flexibility, controllable complexity, and suitability for evaluating learning approaches, as demonstrated by Zakka et al. [22].

X-Magical consists of a two-dimensional world populated by various objects and agents with different embodiments, all rendered using a top-down perspective. The environment features a physics engine that handles collisions, object interactions, and movement dynamics, thereby creating a realistic yet simplified simulation of physical manipulation tasks. This simplified physics model strikes an effective balance between computational efficiency and behavioural realism, making it ideal for large-scale Reinforcement Learning experiments.

The core tasks in X-Magical involve object manipulation, where agents must move objects to target locations or arrange them in a specific configuration. In this study, we focused on a variant of the "Sweep to top" task, which requires agents to gather scattered objects into a designated target area. This task was selected because it represents a fundamental manipulation challenge that remains consistent across different embodiments, while allowing for varied solution strategies.

In detail, the objects in our version of the environment are three squares with different

colours: Red, Blue and Yellow. The target area is visually distinct owing to its green colour and location at the top of the environment, allowing agents to learn to recognise it from visual observations. In the standard configuration, the success criterion for the task is defined as having all objects within the target area, which provides a clear and measurable objective for the learning algorithms. However, for our experiments, we modified the success criterion because it needed to account for the specific order in which the blocks must be placed in the target area for the task to be considered successfully completed.

One of the key features of X-Magical is its support for multiple agent embodiments with distinct physical characteristics and action space.

- **ShortStick**: A simple short stick that can push objects around the environment.
- **MediumStick**: A longer stick that provides greater reach and leverage for manipulating objects.
- **LongStick**: An even longer stick that allows for more complex manipulation strategies.
- **Gripper**: A two-pronged manipulator that can grasp and release objects.

For our experiments, we used only the gripper embodiment, as it provided the most complex manipulation capabilities and closely resembled real-world robotic manipulators. The ability of a gripper to grasp and release objects introduces additional challenges in learning effective manipulation strategies, making it an ideal testbed for evaluating the proposed methodologies.

For our experimental purposes, we implemented several modifications to the original X-Magical environment.

- **Enhanced Observation Space**: We expanded the observation space to include depth information, block colors, block positions, and flags indicating whether a block is in the target area. This additional information provides a richer context for learning algorithms and supports more effective representation learning.
- **Reward Customization**: Instead of using the default sparse reward structure, we implemented a dense reward function that provides continuous feedback based on the distance of objects to the target area, considering the object order. This reward is used to evaluate the learned reward functions, as described in Section 4.2.3.

The environment provides RGB observations at a resolution of 128×128 pixels, which serve as the primary input to our representation learning models. Observations can be captured from either an egocentric or allocentric viewpoint, with the latter providing a consistent frame of reference across different embodiments of the same concept. For our experiments, both viewpoints were used to evaluate the robustness of the learned representations in both cases.

The action space varies by embodiment: the ShortStick, MediumStick, and LongStick use a 2D continuous action vector representing velocity commands, whereas the Gripper uses a 3D continuous vector for position control and grasping. This diversity in action

spaces represents a significant challenge for cross-embodiment learning approaches because the mapping between perceptual states and effective actions differs substantially across embodiments.

The X-Magical environment was also used to collect demonstration data for training our models. As described in Section 2.2.3, inverse Reinforcement Learning is composed of a two-step process: representation learning and policy learning. In the first step, the importance of the dataset used is crucial, as the quality and diversity of the demonstrations directly impact the effectiveness of the learned reward function. Since the X-Magical environment allows for easy configuration and control, we generated a dataset of demonstration videos using scripted policies that successfully complete the task. These scripted policies were designed to exhibit various strategies for manipulating objects, ensuring that the demonstrations captured a wide range of behaviours and scenarios.

The dataset consists of 1000 demonstration videos, each lasting 4-5 seconds and recorded at 10 frames per second, resulting in a total of 40 frames per video. The demonstrations were collected using the gripper embodiment, as it provides the most complex manipulation capabilities and closely resembles real-world robotic manipulators. The gripper's ability to grasp and release objects introduces additional challenges in learning effective manipulation strategies, making it an ideal testbed for evaluating our proposed methodologies. In each demonstration, the agent successfully completes the task by moving all three objects to the target area in the correct order (Red, Blue, Yellow). In addition, the position of the blocks is randomized at the beginning of each episode, ensuring that the demonstrations cover a diverse range of initial configurations. This diversity is crucial for training robust reward functions that can generalise across different scenarios.

Figure 5.1: Example frames from the demonstration dataset, showcasing both allocentric (left) and egocentric (right) viewpoints. The agent is seen manipulating the colored blocks to move them into the target area at the top of the environment.

The 1000 demonstration videos were split into training and validation sets, with 800 videos used for training the reward functions and 200 videos reserved for validation and hyperparameter tuning. This split ensures that the learned reward functions are evaluated on unseen data, providing a measure of their generalisation capabilities.

Since the demonstrations are generated using scripted policies, they are guaranteed to be successful, providing clear examples of the desired behaviour. This success criterion is essential for training effective reward functions, as it ensures that the learning algorithms

can identify the key features and actions that lead to task completion. All this demonstrations reaches the 100% of success using the custom environment reward, as the agent always complete the task in the given time.

In addition, we created two different versions of the dataset for the viewpoint experiments: one with an egocentric viewpoint and another with an allocentric viewpoint. This allows us to evaluate the robustness of learned representations and reward functions across different perspectives.

Together with the image frame, we have also introduces a textual description for each frame of each video, describing the current subtask. This textual description is used for the CLIP-based model, as described in the next sections. The textual descriptions were generated using a simple template-based approach, ensuring consistency and clarity across the dataset. The chosen textual descriptions are as follows:

- The robot pushes the red block into the green zone
- The robot pushes the blue block into the green zone
- The robot pushes the yellow block into the green zone

5.1.2 Baseline Architectures

In this section, we describe the backbone architectures used in our experiments. Each architecture was selected based on its ability to learn robust visual representations from the demonstration videos, which are crucial for effective reward function learning and policy optimisation. These architectures were used for comparisons with our proposed method, as described in Section 4.1.

XIRL: Cross-Embodiment Inverse Reinforcement Learning

Cross-Embodiment Inverse Reinforcement Learning (XIRL), introduced by Zakka et al. [22], addresses one of the most fundamental challenges in robotic imitation learning: enabling agents to learn from demonstrations performed by different embodiments without requiring action labels or explicit correspondence annotations. This capability is particularly crucial for scaling imitation learning to real-world scenarios in which human demonstrations must be translated into robotic execution.

The core innovation of XIRL lies in its application of temporal cycle consistency (TCC) to the inverse Reinforcement Learning problem. Instead of learning a direct mapping from states to actions or attempting to match state-action distributions, XIRL learns reward functions by establishing temporal correspondences across demonstration videos from different embodiments. The method operates under the assumption that semantically similar phases of task execution should be temporally aligned in a learned embedding space, regardless of the demonstrator's morphology or action space.

The temporal cycle consistency objective forms the foundation of the XIRL learning process. Given two demonstration videos from potentially different embodiments, the method learns embeddings such that if frame f_i from video A corresponds to the most

similar frame in video B (based on the embedding distance), then the reverse mapping should return to a temporally nearby frame in video A.

This objective encourages the model to learn embeddings that preserve the temporal structure across embodiments while being invariant to morphological differences.

XIRL formulates the reward as a function of the embedding similarity to the goal states, typically using a Gaussian kernel around the target embeddings extracted from successful demonstration endings. This approach enables the system to provide dense reward signals that guide Reinforcement Learning agents toward task completion while maintaining embodiment invariance.

The training procedure for XIRL involves sampling batches of frames from demonstration videos and applying cycle consistency loss to encourage proper temporal alignment. The method also incorporates data augmentation techniques, including random cropping and colour jittering, to improve robustness against visual variations that are irrelevant to task progress. This augmentation strategy is particularly important for cross-embodiment learning, where demonstrations can be collected under different lighting conditions or camera angles.

HOLD: Learning Reward Functions by Observing Humans

HOLD (Human Observation Learning for Demonstration), introduced by Alakuijala et al. [35], addresses the fundamental challenge of learning reward functions for robotic manipulation tasks directly from human demonstration video. This approach represents a significant advancement in cross-embodiment learning by enabling robots to acquire manipulation skills from readily available human video data, substantially reducing the data collection burden typical in imitation learning scenarios.

The HOLD framework operates under the assumption that human demonstrators are approximately optimal when performing manipulation tasks and that the reward function can be inferred by understanding the progression toward task completion, as evidenced by state transitions in demonstration videos. This method learns embeddings in which Euclidean distances correspond to functional distances in the task space, thereby enabling the extraction of dense reward signals without explicit reward supervision.

HOLD introduces two distinct architectural variants that address different aspects of the reward learning problem: HOLD-R (HOLD-Regression) and HOLD-C (HOLD-Contrastive). Both variants share a common foundation in learning visual representations that capture task-relevant semantics but differ in their specific learning objectives and architectural implementations.

HOLD-R formulates reward learning as a regression problem, where the model learns to predict the functional distance between states in the demonstration trajectories. The architecture consists of a visual encoder that maps RGB observations to high-dimensional embeddings, followed by a distance prediction network that estimates the number of optimal steps required to transition between states.

The core training objective of HOLD-R is based on temporal distance prediction within demonstration sequences. We assume that the demonstrations are optimal and pose the functional distance learning problem as a supervised regression task:

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} \sum_{t=1}^{T_i} \sum_{\delta=1}^{T_i - t} \|d_\theta(s_t^i, s_{t+\delta}^i) - \delta\|_2^2 \tag{5.1}$$

where s_t^i is the t-th frame of the i-th video, T_i is the length of the i-th video, and d_θ is a function parameterised by θ that is trained to predict δ from the functional distance. The third summation corresponds to data augmentation, allowing any future time step in the video to be considered the goal rather than only the last. This objective encourages the model to learn embeddings in which the predicted distance correlates with the temporal progression of successful demonstrations.

HOLD-C approaches the reward learning problem using a contrastive learning framework, where the model learns to distinguish between state pairs that are temporally close and those that are temporally distant in demonstration sequences. Since directly predicting time intervals is difficult and sensitive to noise, HOLD-C considers learning an embedding space where distances can be defined using a single-view time-contrastive objective.

The contrastive learning objective in HOLD-C is designed to pull together the embeddings of states that are temporally close in successful demonstrations while pushing apart the embeddings of states that are temporally distant. Frames within a small temporal window are encouraged to lie close together in the embedding space, whereas embeddings for frames outside a temporal neighbourhood are pushed apart.

Specifically, if s^p is a positive instance for anchor s, and s^n is a negative instance for all triplets, the objective is

$$||f(s) - f(s^p)||_2^2 + m < ||f(s) - f(s^n)||_2^2 \tag{5.2}$$

where f is the embedding function and margin m is a hyperparameter that controls the separation between positive and negative pairs in the embedding space.

The positive and negative sampling strategies in HOLD-C are crucial for effective representation learning. Positive pairs are constructed from states that are within a small temporal window of each other in the same demonstration sequence, ensuring that they represent similar stages of task progression. Negative pairs are constructed from states that are temporally distant within the same sequence or from different demonstration sequences, providing the necessary contrast to learn discriminative embeddings.

This approach leverages the insight that states with similar functional distances to the goal should have similar representations, whereas states at different stages of task progress should be easily distinguishable in the learned embedding space.

For this study, only the HOLD-R variant was considered, as it represented the most interesting approach for our specific task and because of the absence of negative samples in our dataset, which are essential for the contrastive learning approach used in HOLD-C.

During reward inference, both HOLD variants compute rewards based on the learned distance estimates or embedding similarities to the goal states extracted from successful demonstrations. The reward function is typically formulated as a monotonically decreasing function of the predicted distance or as an increasing function of similarity to goal embeddings, providing dense reward signals that guide Reinforcement Learning agents toward task completion.

REDS: Subtask-Aware Visual Reward Learning

REDS (REward learning from Demonstration with Segmentations), introduced by Kim et al. [36], addresses the challenge of learning reward functions for long-horizon, multi-step robotic manipulation tasks using a subtask-aware framework. Unlike methods that treat complex tasks as monolithic learning problems, REDS explicitly leverages the hierarchical structure inherent in manipulation tasks by segmenting demonstrations into semantically meaningful subtasks and learning specialised reward functions for each of these phases.

REDS architecture employs a multimodal approach that combines visual and textual information to create robust reward representations. Visual inputs are processed using a pre-trained CLIP ViT-B/16 encoder, which provides strong visual representations trained on large-scale image-text datasets. This choice of backbone enables the system to leverage the rich semantic understanding of pretrained models while adapting to specific manipulation tasks.

Subtask information is incorporated through text embeddings generated by CLIP, which are then projected onto specialised subtask embeddings. This design allows the system to condition rewards on explicit subtask indices while enabling generalisation to new tasks, given appropriate textual descriptions of their subtasks.

REDS employs a sophisticated training framework that combines three complementary objectives to ensure effective, reward learning. The first component is an EPIC-based (Equivalent-Policy Invariant Comparison) objective that minimizes the distance between the learned reward function and ground-truth subtask segmentation signals.

The second component is a progressive reward regulariser that enforces monotonic progress within the subtask by ensuring that rewards increase as the agent advances through each subtask.

The third component is a contrastive alignment loss that pulls video segment representations closer to their corresponding subtask embeddings in the learned space. The combination of these three loss components creates a robust training framework that learns both accurate reward functions and reliable subtask classification capabilities.

During the policy learning phase, REDS uses a multimodal approach that combines visual perception with language understanding. Unlike embedding-based approaches that rely solely on visual similarity, REDS dynamically identifies the current subtask from a set of predefined text descriptions and computes the rewards accordingly.

These text descriptions are also used in the training phase, as each frame of the demonstration videos is paired with a textual description of the current subtask, allowing the model to learn the association between visual states and subtask semantics.

The process begins by encoding the current observation into visual features using the CLIP ViT-B/16 backbone. Simultaneously, a collection of task descriptions is encoded into text embeddings.

For each observation, the algorithm computes a contrastive similarity matrix between the visual features and all text descriptions using the cosine similarity. A biased selection mechanism is then applied to determine the most relevant subtask descriptions. Once the most relevant subtask is identified, the corresponding text embedding is extracted and used to compute the reward.

The reward prediction model, which was trained using a three-component loss function (EPIC-based objective, progressive reward regularisation, and contrastive alignment), produced a scalar value that indicated how well the current state aligned with progress toward the identified subtask.

This text-conditioned reward mechanism allows REDS to dynamically adapt to different phases of the task without requiring explicit thresholds or distance calculations, theoretically enabling more flexible task sequencing.

The method also shows strong performance on real-world furniture assembly tasks from FurnitureBench, where the complexity of long-horizon manipulation is particularly challenging. Explicit subtask conditioning enables REDS to provide more informative reward signals than methods that attempt to learn monolithic reward functions for entire task sequences.

However, REDS has some limitations, as reported in the original study. This approach assumes that subtasks can be clearly defined and separated with a text prompt that distinguishes them, which may not hold for all manipulation tasks. Furthermore, while REDS demonstrates strong generalisation capabilities, its performance may still be sensitive to the quality and diversity of the data used for training.

5.1.3 Evaluation Metrics

To evaluate the performance of INEST-IRL and compare it with other baseline architectures, we employed several metrics that provided insights into the effectiveness of the learned reward function and policies.

For all experiments, the number of evaluation episodes was set to 150. This parameter was chosen to provide a sufficient sample size for reliable performance estimation while ensuring that the episodes remained computationally feasible.

The first metric used is the evaluation score, which measures the proportion of episodes in which the agent successfully completed the task within a specified time limit. This metric provides a clear and interpretable measure of an agent's overall performance and ability to achieve the desired objectives. To estimate the evaluation score, we conducted a series of evaluation episodes in which the agent interacted with the environment using a learned policy. Each episode begins with a random initial configuration of objects, and the agent must manipulate them to reach the target area in the correct order. The evaluation score is calculated as the ratio of successful episodes to the total number of episodes, averaged over multiple runs. The score assigned to an episode was proportional to the number of blocks placed in the target area in the correct order. Specifically, a score of 0 was assigned if no blocks were placed in the target area, a score of 0.33 if only the red block was placed in the target area, a score of 0.66 if the red and blue blocks were placed in the target area, and a score of 1 if all three blocks were placed in the target area in the correct order. The maximum score of 1 indicates that the agent successfully completed the task.

The evaluation score plot shows the moving average over a window of five evaluation points, which helps smooth out short-term fluctuations and highlight longer-term trends in performance. The standard deviation is also plotted to provide insights into the variability of the agent's performance across different evaluation episodes.

Together with the analysed models, a dashed line is shown, representing the performance obtained with the environmental reward and providing a benchmark for comparison. This environmental reward is based on the geometric positions of each block and provides a partial positive reward for each subtask completed in the correct order. The episode that completed the task achieved the highest reward. It works as defined in Section 5.1.1 this

In addition to the evaluation score, we considered the success rate, which measures the proportion of episodes in which an agent successfully completed a task. This metric provides a clear and interpretable measure of an agent's overall performance and ability to achieve the desired objectives. To estimate the success rate, we considered the number of episodes in which the agent achieved the highest evaluation score, indicating that it had successfully completed the task. The success rate is calculated as the ratio of successful episodes to the total number of episodes, averaged over multiple runs.

Additional metrics will be used in some ablation studies to provide a more comprehensive evaluation of the learned policies and reward functions, which will be described in the respective sections.

5.2 Ablation Studies

In this section, we present a series of ablation studies designed to investigate the impact of various components and design choices of our proposed method. Each ablation study focused on a specific aspect of the architecture or training procedure, allowing us to isolate its contribution to the overall performance. The results of these studies provide valuable insights into the effectiveness of different design elements and inform future improvements to this method.

Because INEST-IRL evolved through different components and design choices, the ablation studies were conducted sequentially, with each study building upon the findings of the previous one. This approach allows us to systematically evaluate the impact of each component while controlling for confounding factors.

Initially, INEST-IRL was built on an Inverse Reinforcement Learning (IRL) framework based on the XIRL architecture, as described in Section 5.1.2, with the addition of Procedural Learning to better understand the strengths and weaknesses of this model. Therefore, the given name for the model is called Subtask Inverse Reinforcement Learning (ST-IRL). In the third ablation study, we introduced an Intrinsic Reward mechanism based on embedding space coverage, as described in Section 4.2.3, to evaluate its effectiveness in enhancing exploration and improving task performance. Considering the results obtained from the ablation studies, we understood the correct direction to follow to improve the model and develop the final version of the INEST-IRL.

5.2.1 Ablation Study 1: Visual Perspective

The first ablation study focused on the impact of the visual perspective on the performance of the learned policies and reward functions. Specifically, we compared the effectiveness

of using an egocentric viewpoint versus an allocentric viewpoint in the X-Magical environment, as described in Section 5.1.1. The choice of visual perspective can significantly influence an agent's ability to perceive and interpret its surroundings, which in turn affects its learning capabilities.

In this study, we trained two separate instances of the ST-IRL architecture: one using egocentric observations and the other using allocentric observations. To correctly evaluate the impact of the visual perspective, we considered the allocentric dataset and created an egocentric version by changing the camera viewpoint in the environment and maintaining the same action and state sequence. Both models were trained using the same hyperparameters, training procedures, and evaluation metrics to ensure a fair comparison.

To better understand the impact of the visual perspective, we considered ST-IRL without the Intrinsic Reward and a subdivision of the task into three subtasks, as described in Section 4.2.3, to simplify the learning process and focus on the effect of the visual perspective on the learning process. The training was conducted for 8 million environmental steps, with evaluations performed every 50,000 steps to monitor progress and performance over time.

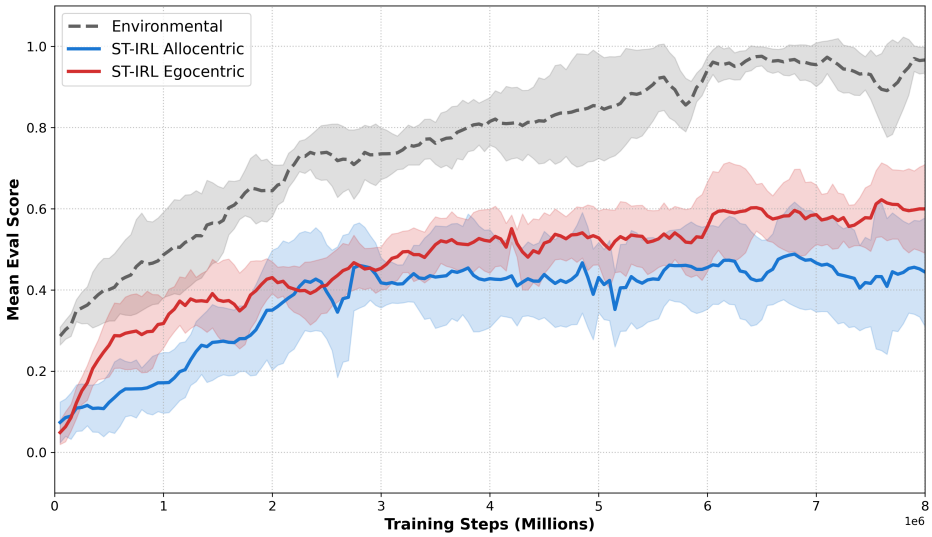


Figure 5.2: Evaluation scores for the egocentric and allocentric models over 150 evaluation episodes. The egocentric model demonstrates a higher average score and lower variance, indicating more consistent performance across different initial configurations.

The results of this ablation study are shown in Figure 5.2. The egocentric model achieved a final average evaluation score of 0.59, whereas the allocentric model achieved an average score of 0.44. The egocentric model also exhibited a lower variance in performance across different evaluation episodes, suggesting that it was more robust to variations in the initial object configurations.

Model	End-of-Episode Subtask Completion (%)		Success Rate (%)
	1 block	2 blocks	3 blocks
ST-IRL Allocentric	11.11	56.00	0.00
ST-IRL Egocentric	10.67	65.11	**13.11**

Table 5.1: Success rates for egocentric and allocentric models. The egocentric model shows a higher success rate for placing two and three blocks correctly.

The table in Figure 5.1 shows the success rates for the egocentric and allocentric models, broken down by the number of blocks that were correctly placed in the target area. The egocentric model demonstrated a higher success rate for placing two blocks (65.11% vs. 56.00%) and three blocks (13.11% vs. 0.00%) correctly, indicating its superior ability to complete the task in the correct order.

From these results, we can conclude that the egocentric viewpoint provides a significant advantage in learning effective manipulation policies in the X-Magical environment. The egocentric perspective offers a more intuitive and direct representation of an agent's interactions with objects, which likely facilitates the learning process. In contrast, the allocentric viewpoint may introduce additional complexity in interpreting spatial relationships, making it more challenging for agents to learn effective strategies.

The embedding learned by the egocentric model demonstrated a clearer understanding of the precise block boundaries and their relative positions, which is crucial for tasks that require precise manipulation. This improved spatial understanding likely contributed to the higher evaluation scores obtained for the egocentric model. On the other hand, the allocentric model can still learn a reasonable representation of the environment, but it may struggle with accurately capturing the nuances of object interactions due to the less direct perspective.

This aspect also affects the agent's learning ability. As shown in. 5.2, the egocentric model showed a more rapid improvement in evaluation scores during the early stages of training, indicating that it could quickly adapt to the task requirements. The allocentric model, while still improving over time, exhibits a slower learning curve, suggesting that it may require more training data or iterations to achieve a performance comparable to that of better understanding the correct block boundaries.

However, it is important to note that the allocentric viewpoint may still be beneficial in certain scenarios, particularly in tasks that require a broader understanding of the environment or when dealing with multiple agents. The choice of visual perspective should be carefully considered based on the specific requirements and constraints of the task.

5.2.2 Ablation Study 2: Subtasks Number

The second ablation study investigates the impact of task subdivision into different numbers of subtasks on the performance of the learned policies and the reward functions. In this study, we evaluate how varying the number of subtasks affects the learning process and overall task performance.

As a previous ablation study demonstrated the superiority of the egocentric viewpoint, we continued to use this perspective in the current study. We trained three separate

instances of the ST-IRL architecture, each with a different number of subtasks (1, 3, and 6). The choice of subtask numbers was motivated by the desire to explore a range of task decompositions, from coarse (1 subtask) to fine-grained (6 subtasks), to understand how this granularity influences learning. The ST-IRL model with 1 subtask is equivalent to the XIRL architecture, as described in Section 5.1.2, in the egocentric view. For this reason, in the results, we refer to it as the XIRL (EGO).

The ST-IRL 1-Subtask treats the entire task as a single, monolithic objective, whereas the ST-IRL 3-Subtasks divides the task into three distinct phases: placing each of the three blocks in the correct order. The ST-IRL 6-Subtasks further refines this decomposition by breaking down each block manipulation into two separate subtasks: picking the block and pushing it into the target area. This finer granularity aims to provide more detailed feedback during learning, potentially facilitating better credit assignment for actions performed at different stages of the task.

All models were trained using the same hyperparameters, training procedures, and evaluation metrics to ensure fair comparison. The training was conducted for 8 million environment steps, with evaluations performed every 50,000 steps to monitor progress and performance over time.

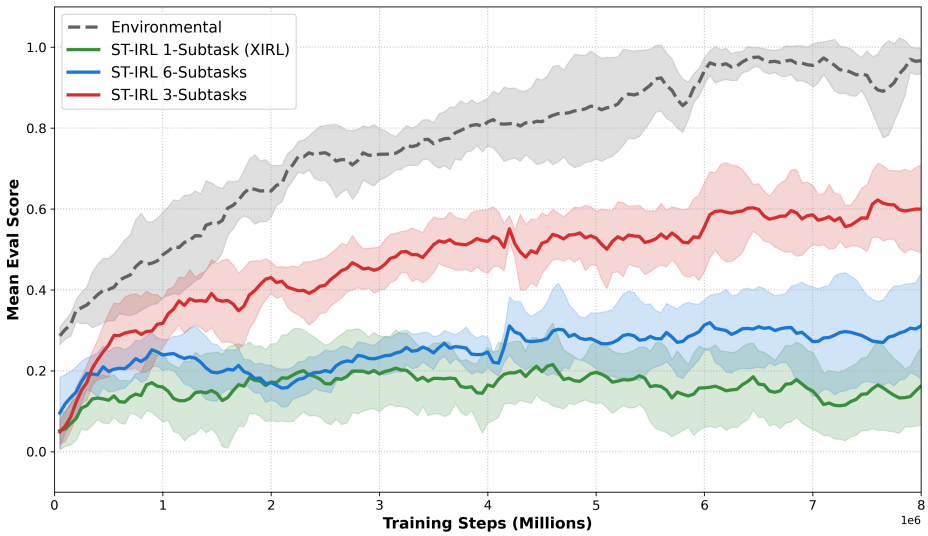


Figure 5.3: Evaluation scores for the 1 Subtask, 3 Subtasks, and 6 Subtasks models over 150 evaluation episodes. The 3-Subtasks model demonstrates the highest average score and lowest variance.

The results of this ablation study are shown in Figure 5.3. The ST-IRL 3-Subtasks achieved the highest final average evaluation score of 0.59, outperforming both the ST-IRL 1-Subtask (0.24) and the ST-IRL 6-Subtasks (0.31). The ST-IRL 3-Subtasks also exhibited the lowest variance in performance across different evaluation episodes, along with the ST-IRL 1-Subtask, suggesting that it was more robust to variations in the initial object configurations.

Model	End-of-Episode Subtask Completion (%)		Success Rate (%)
	1 block	2 blocks	3 blocks
ST-IRL 1-Subtask (XIRL)	54.97	0.00	0.00
ST-IRL 6 Subtasks	34.00	45.33	0.00
ST-IRL 3 Subtasks	10.67	65.11	**13.11**

Table 5.2: End-of-episode subtask completion and success rates for the ST-IRL 1-Subtask, ST-IRL 3-Subtasks, and ST-IRL 6-Subtasks models. The ST-IRL 3-Subtasks model achieves the highest success in placing two and three blocks correctly.

The table in Figure 5.2 shows the success rates for the three models, broken down by the number of blocks correctly placed in the target area. The ST-IRL 3-Subtasks model demonstrated a higher success rate for placing two blocks (65.11% vs. 0.00% and 45.33%) and three blocks (13.11% vs. 0.00% and 0.00%) correctly, indicating its superior ability to complete the task in the correct order. As can be seen, the ST-IRL 3-Subtasks model is the only model that can place the three blocks correctly.

The superior performance of the ST-IRL 3-Subtasks model can be attributed to its balanced approach to task decomposition. By dividing the task into three distinct phases, the model receives more targeted feedback during learning, allowing it to focus on mastering each phase. This structured approach likely facilitates better credit assignment for actions taken at different stages of a task, leading to more effective learning.

The ST-IRL 1-Subtask model, while simpler in its approach, may struggle with the complexity of the task as it attempts to learn a single policy that encompasses all aspects of the manipulation sequence. This monolithic approach can lead to challenges in credit assignment because the model must discern which actions contribute to success across the entire task. In particular, the model may find it difficult to learn effective strategies for intermediate steps, such as picking up blocks, without explicit guidance because the task is too complex to be learned in a single step.

The ST-IRL 6-Subtasks, while providing a finer granularity of feedback, may introduce additional complexity that hinders learning. An increased number of subtasks can lead to fragmentation of the learning signal, making it more challenging for the model to integrate information across subtasks and develop a coherent overall strategy. Additionally, the model may struggle to effectively transition between subtasks, as an increased number of phases requires more precise timing and coordination. This fragmentation can lead to slower learning and lower overall performance, as observed in the evaluation scores. In addition, in the 6 Subtasks decomposition, the picking and placing subtasks are treated as separate entities, but the time distance between them is small, making it difficult for the model to learn distinct representations for each subtask. This close temporal proximity can lead to confusion in the learning process, as the model may struggle to differentiate between the two actions and their respective contributions to task success, as the agent sees a continuously increasing reward. This confusion can result in suboptimal policies that fail to effectively execute the required actions for each subtask, ultimately leading to lower performance compared to the more balanced 3 Subtasks approach.

5.2.3 Ablation Study 3: Intrinsic Reward

The third ablation study explored the impact of incorporating an Intrinsic Reward mechanism based on embedding space coverage, as described in Section 4.2.3, on the performance of the learned policies and reward functions. This study aimed to evaluate whether the addition of intrinsic motivation could enhance exploration and improve task performance.

For this study, we trained three separate instances of the ST-IRL architecture using the egocentric viewpoint and subdivided the task into three subtasks, as these configurations demonstrated superior performance in previous ablation studies. One model was trained with the Intrinsic Reward mechanism enabled, while the other model was trained without it. Both models were trained using the same hyperparameters, training procedures, and evaluation metrics to ensure a fair comparison.

The third model is the state-based Intrinsic Reward model, a variant of the embedding-based Intrinsic Reward model. In this model, the Intrinsic Reward is computed based on the entire observation provided by the environment instead of the embedding space. This model is used to understand if the Intrinsic Reward is effective because of the embedding space or if it is effective in general. The working process of the state-based Intrinsic Reward is the same as that of the embedding-based version. The training was conducted for 8 million environment steps, with evaluations performed every 50,000 steps to monitor progress and performance over time.

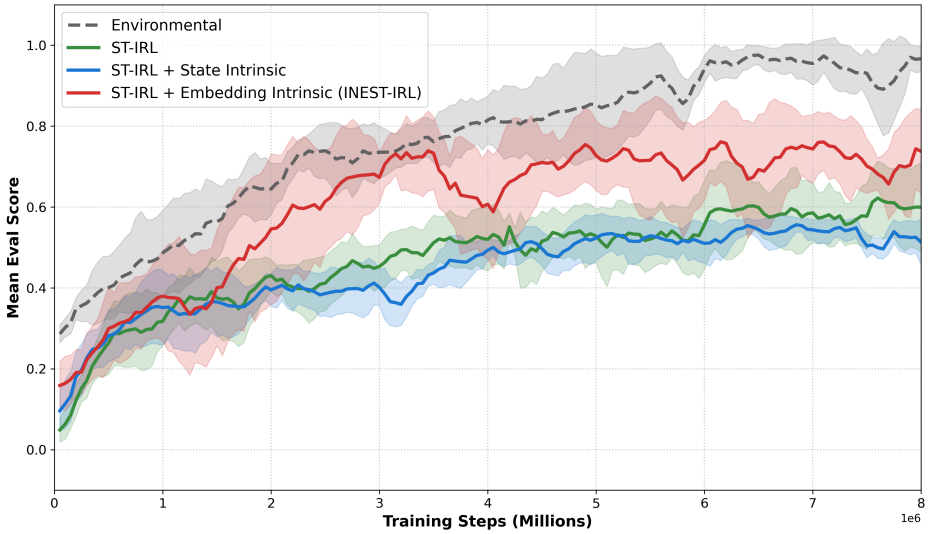


Figure 5.4: Evaluation scores over 150 evaluation episodes. The embedding-based Intrinsic Reward model demonstrated the highest average score and lowest variance.

The results of this ablation study are shown in Figure 5.4. The embedding-based Intrinsic Reward model achieved the highest final average evaluation score of 0.73, outperforming both the state-based Intrinsic Reward model (0.51) and the no-intrinsic-reward model (0.59). The state-based Intrinsic Reward model exhibited the highest variance in performance across different evaluation episodes, suggesting that it was less robust to

variations in the initial object configurations.

The superior performance of the embedding-based Intrinsic Reward model can be attributed to its ability to encourage exploration in a manner closely aligned with the learned representation of the task. By incentivising the agent to visit novel states in the embedding space, the model promotes a more structured exploration strategy that facilitates the discovery of effective manipulation strategies. This targeted exploration is likely to lead to more efficient learning and better overall performance.

Although the state-based Intrinsic Reward model provides an additional exploration incentive, it may not be as effective in guiding the agent toward task-relevant states. The Intrinsic Reward in this model is based on the raw observation space, which can be high-dimensional and noisy, making it challenging for the agent to discern which states are novel and relevant to task completion. This lack of structure in the exploration strategy may lead to less efficient learning and lower overall performance levels.

The no Intrinsic Reward model, while still capable of learning effective policies, may struggle to explore complex environments where the reward signal is sparse or delayed. Without the additional motivation provided by Intrinsic Rewards, the agent may be less inclined to explore novel states, potentially leading to suboptimal policies that fail to exploit the dynamics of the environment fully.

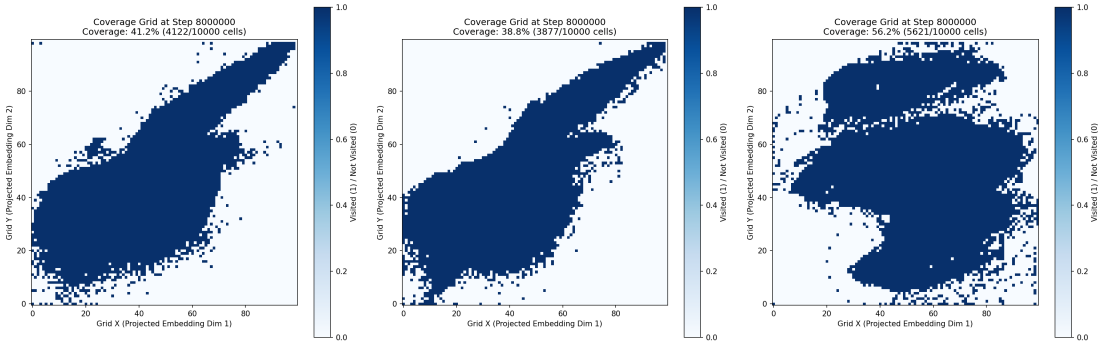
The embedding-based Intrinsic Reward also demonstrated a more rapid improvement in evaluation scores during the early stages of training, indicating that it could quickly adapt to task requirements. The state-based Intrinsic Reward model, while still improving over time, exhibits a slower learning curve, suggesting that it may require more training data or iterations to achieve a performance comparable to that of better understanding the correct block boundaries.

Model	**End-of-Episode Subtask Completion (%)**		**Success Rate (%)**
	1 block	**2 blocks**	**3 blocks**
ST-IRL + State Intrinsic	33.90	66.10	0.00
ST-IRL	10.67	65.11	13.11
ST-IRL + Embedding Intrinsic (INEST-IRL)	0.00	77.55	**22.45**

Table 5.3: End-of-episode subtask completion and success rates for the three models. The embedding-based Intrinsic Reward model achieved the highest success when three blocks were placed correctly.

The table in Figure 5.3 shows the success rates for the three models, broken down by the number of blocks correctly placed in the target region. The embedding-based Intrinsic Reward model demonstrated a higher success rate for placing three blocks (22.45% vs. 0.00% and 13.11%) correctly, indicating its superior ability to complete the task in the correct order. The introduction of the Intrinsic Reward increased the success rate for placing two and three blocks, while reducing it to 0.00% for placing zero and one block. This result suggests that the Intrinsic Reward effectively encourages the agent to explore and learn strategies that lead to successful task completion. In contrast, the state-based Intrinsic Reward model showed a higher success rate for placing one block (33.90% vs. 0.00% and 10.67%), but failed to place three blocks correctly, indicating that while it could learn some aspects of the task, it struggled with the overall sequence required for full task completion.

In addition to this analysis, we evaluated the impact of Intrinsic Rewards on the exploration behaviour of the agent. We measured the coverage of the embedding space during training, which is defined as the proportion of unique embedding states visited by the agent. We considered the unique visited embedding states and plotted them on a grid, where each cell in the grid represented a distinct region of the embedding space. The coverage metric was calculated as the ratio of occupied cells to the total number of cells in the grid.



(a) ST-IRL (41.2%) (b) ST-IRL + State (38.8%) (c) INEST-IRL (56.2%)

Figure 5.5: Coverage of the embedding space during training. The embedding-based Intrinsic Reward model achieves the highest coverage, indicating more effective exploration of the state space.

The three figures in Figure 5.5 illustrate the coverage of the embedding space for the three models considered in this ablation study. The embedding-based Intrinsic Reward model (Figure b) demonstrated a significantly higher coverage rate than both the state-based Intrinsic Reward model (Figure c) and the no Intrinsic Reward model (Figure a). This increased coverage indicates that the embedding-based Intrinsic Reward effectively encourages the agent to explore a broader range of states, which is likely to contribute to its excellent performance. The state-based Intrinsic Reward model showed lower coverage than the model without Intrinsic Reward contribution, suggesting that it may not be as effective in promoting exploration because the observation can include information that is not relevant to increase the agent exploration.

An interesting observation is that the embedding-based Intrinsic Reward model exhibits a more uniform distribution of visited states across the embedding space, indicating that the agent explores a wider variety of regions rather than concentrating on specific regions. The resulting coverage pattern differed from those of the other two models, suggesting that the agent could reach states that were not visited by the others. This broader exploration is beneficial for learning robust policies that generalise better to different initial configurations and task variations.

ST-IRL with embedding-based Intrinsic Reward is the final version of INEST-IRL, as described in Section 1.2.4.

Because our INEST-IRL is complete with the introduction of the embedding-based Intrinsic Reward, we want to understand whether the introduction of this contribution

is the main reason for the improvement of the model or whether the improvement is due to the combination of all the components. For this reason, we compared the embedding-based Intrinsic Reward model with the 3-Subtasks model without Intrinsic Reward, the 1-Subtask model with and without Intrinsic Reward, and the 6-Subtasks model with and without Intrinsic Reward. The results of this comparison are shown in Figure 5.6.

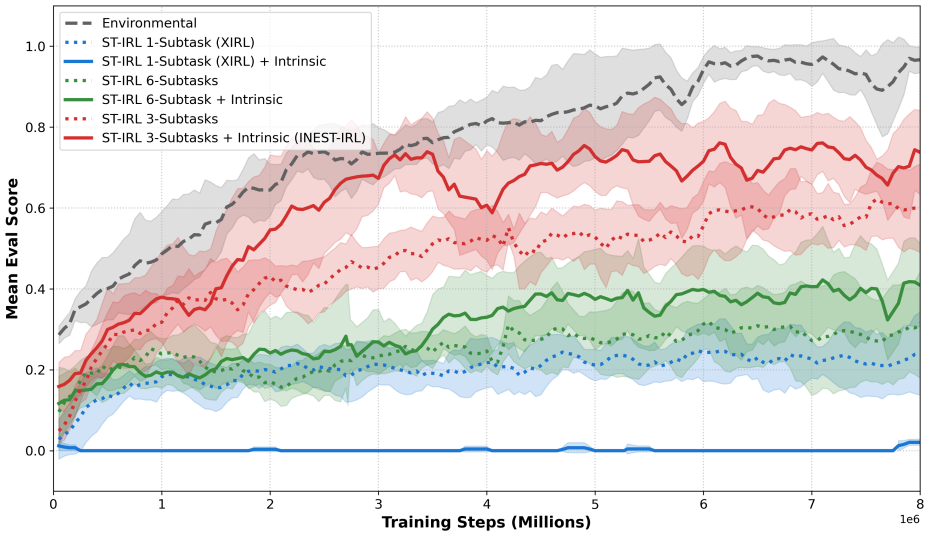


Figure 5.6: Evaluation scores for different subtask models with and without Intrinsic Reward contribution over 150 evaluation episodes. The embedding-based Intrinsic Reward model demonstrates the highest average score and lowest variance, indicating more consistent performance across different initial configurations.

The embedding-based Intrinsic Reward model achieved the highest final average evaluation score of 0.73, outperforming all other models in this study. The 3-Subtasks model without Intrinsic Reward achieved an average score of 0.59, while the 1-Subtask model with Intrinsic Reward achieved a score of 0.02, and without Intrinsic Reward, it achieved a score of 0.24. The 6-Subtasks model with Intrinsic Reward achieved a score of 0.40, whereas without Intrinsic Reward, it achieved a score of 0.31.

These results indicate that the combination of task subdivision into three subtasks and the embedding-based Intrinsic Reward mechanism is crucial for achieving optimal performance. The 3-Subtasks model without Intrinsic Reward still performs reasonably well, suggesting that task decomposition alone provides significant benefits. However, the addition of Intrinsic Rewards further enhanced exploration and learning efficiency, leading to the highest performance.

Notably, the 1-Subtask model with Intrinsic Reward performed poorly, achieving an average score of only 0.02. This suggests that while Intrinsic Rewards can enhance exploration, they may not be sufficient to overcome the challenges associated with learning a monolithic task without task decomposition. The lack of structure in the learning process

likely hinders the agent's ability to effectively utilise the Intrinsic Reward signal. Moreover, the introduction of the Intrinsic Reward reduces the performance, meaning that for the 1-Subtask model, the Intrinsic Reward contribution works as noise that confuses the agent during the learning process.

Model	**End-of-Episode Subtask Completion (%)**		**Success Rate (%)**
	1 block	**2 blocks**	**3 blocks**
ST-IRL 1-Subtask (XIRL)	54.97	0.00	0.00
ST-IRL 1-Subtask (XIRL) + Intrinsic	6.29	0.00	0.00
ST-IRL 6-Subtasks	34.00	45.33	0.00
ST-IRL 6-Subtasks + Intrinsic	33.33	49.33	0.00
ST-IRL 3-Subtasks	10.67	65.11	13.11
ST-IRL 3-Subtasks + Intrinsic (INEST-IRL)	0.00	77.55	**22.45**

Table 5.4: End-of-episode subtask completion and success rates for different subtask models with and without Intrinsic Reward. The embedding-based Intrinsic Reward with 3 subtasks shows the highest rate for placing three blocks correctly.

The table in Figure 5.4 shows the success rates of all the models. The embedding-based Intrinsic Reward model with three subtasks demonstrated the highest success rate for placing three blocks (22.45%), indicating its superior ability to complete the task in the correct order. The 3-Subtasks model without Intrinsic Reward also showed a reasonable success rate for placing two blocks (65.11%) and three blocks (13.11%), while the 6-Subtasks model with and without Intrinsic Reward struggled to place three blocks correctly. The 1-Subtask model with Intrinsic Reward had a very high success rate for placing zero blocks (93.71%), indicating that the agent often failed to progress in the task. Its success rate decreased significantly compared to the 1-Subtask model without Intrinsic Reward, which had a more balanced distribution of success rates across different numbers of blocks placed. This result is consistent with the low evaluation score, indicating that the model struggled to learn effective policies for task completion.

Overall, these findings highlight the importance of Procedural Learning and intrinsic motivation in developing effective learning strategies for complex manipulation tasks. The embedding-based Intrinsic Reward mechanism, when combined with well-structured task decomposition, leads to significant improvements in task performance and learning efficiency.

5.3 Baseline Comparison

In this section, we present a comparative analysis of the different baseline architectures implemented for learning reward functions from demonstration videos using INEST-IRL. The baselines considered in this study included XIRL [22], HOLD-R [35], and REDS [36]. Each baseline represents a distinct approach to learning from demonstrations with unique architectural designs and training objectives.

All models were trained using the same dataset, training procedure, and evaluation metrics to ensure a fair comparison. In this case, we used the same visual perspective because we wanted to compare the baselines with the same dataset. The training was conducted for 8 million environment steps, with evaluations performed every 50,000 steps to monitor progress and performance over time.

Before discussing the training results, we analysed the pretraining results to understand how well each model understood the task. For this purpose, two different metrics were used: the reward function plot, which shows the development of the reward function during the inference of a never-seen demonstration, and a distance metric that compares the cosine similarity Gaussian plot of each model with the ground truth Gaussian plot.

The cosine similarity Gaussian plot is obtained by calculating the cosine similarity between the embeddings of each frame in a demonstration video and the embedding of the subtask frame for models that work with subtask decomposition and the final frame for models that work without subtask decomposition. The cosine similarity values were then plotted against the frame indices, resulting in a Gaussian-like curve that reflects how well the model embeddings align with the subtask or goal state over time. A higher cosine similarity indicates a closer alignment between the current state and the subtask or goal state, suggesting that the model effectively captures progression toward task completion. In detail, we are interested in the shape of the Gaussian curve, which should increase as the agent approaches the subtask or goal state and decrease as it moves away from it. The ideal curve would show a smooth and consistent increase in cosine similarity as the agent progressed through the demonstration, indicating that the model accurately captured the task's temporal structure.

The distance metric reported in Table 5.6 was calculated by comparing the cosine similarity Gaussian plot of each model with the ground truth Gaussian plot, which represents the ideal progression toward task completion. The distance was computed using the mean squared error to quantify the discrepancy between the curves. For models that work with multiple subtasks, the distance is computed for each subtask individually and then averaged. A lower distance value indicates a closer alignment between the model's learned representation and the ground truth, suggesting that the model effectively captures the temporal structure and progression of the task.

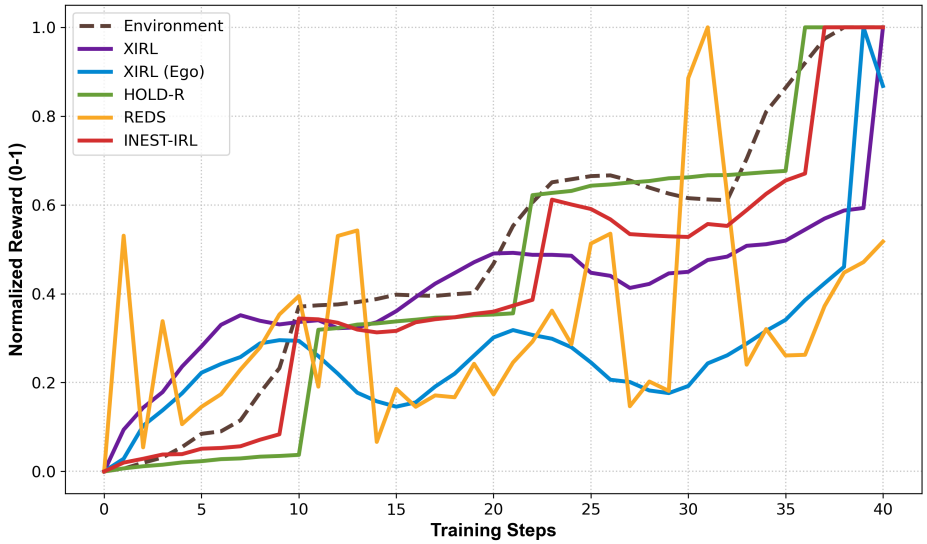


Figure 5.7: Reward function plots for the INEST-IRL, XIRL, HOLD-R, and REDS baselines over a never-seen demonstration. INEST-IRL demonstrated a clear and consistent reward progression, indicating effective learning of the task structure.

The reward function plots for each baseline are presented in Figure 5.7, and the average distances comparing the cosine similarity Gaussian for each model to the ground truth are reported in Table 5.6. The INEST-IRL and HOLD-R baselines demonstrated clear and consistent reward progression, indicating effective learning of the task structure. The XIRL baseline showed a moderately defined reward signal, whereas REDS exhibited erratic performance with some inconsistencies.

Method	**MSE**	**MAE**	**Bias**
XIRL	6.438	2.488	-2.488
XIRL (Ego)	6.565	2.465	-2.465
HOLD-R	1.925	1.139	0.810
REDS	116.864	10.424	-10.424
INEST-IRL	1.130	0.802	0.422

Table 5.5: Performance comparison of different backbone architectures across multiple evaluation metrics. Lower MSE and MAE values indicate better performance, whereas higher Spearman correlation values indicate stronger alignment with the ground truth rankings.

The table in Figure 5.5 summarises the performance of each baseline across multiple evaluation metrics, including the Mean Squared Error (MSE), Mean Absolute Error (MAE), and bias. These metrics are the result of comparisons with the environment reward curve. The INEST-IRL baseline achieved the lowest MSE (1.130) and MAE (0.802) values, indicating superior performance in accurately predicting rewards compared with

the other baselines. HOLD-R also performed well, with an MSE of 1.925 and an MAE of 1.139, suggesting that it effectively captured the task's reward structure. The XIRL baseline had higher error metrics, indicating less accurate reward prediction. However, in this case, the high values are related to the different numbers of subtasks because XIRL relies on one final goal embedding, whereas the environment reward is composed of three subtasks. REDS exhibited significantly higher error values (MSE of 116.864 and MAE of 10.424), reflecting the challenges in learning a consistent reward function.

Backbone	**Cosine Similarity Average Distances**
REDS	3.655451
HOLD-R	2.713410
INEST-IRL	**1.968344**

Table 5.6: Distances comparing the cosine similarity Gaussian for each model and subtask to the ground truth Gaussian plot. Shorter distances indicate better performance in predicting the correct subtasks.

The average distances in Table 5.6 further support these observations. The INEST-IRL baseline is the model with the lowest distance of 1.968344, suggesting that it effectively captures task progression. HOLD-R has a higher distance of 2.713410, indicating some discrepancies in its learned representation, while REDS has the highest distance of 3.655451, suggesting that it struggles to accurately capture the task's temporal structure.

From the cosine similarity distance, we can see that INEST-IRL is the best model for understanding the task because it shows a lower value than the models that work with Procedural Learning HOLD-R and REDS. This result is consistent with the reward function plots, where INEST-IRL shows a clear and consistent reward progression. HOLD-R and REDS exhibited higher distances and more erratic reward signals, indicating challenges in accurately capturing the temporal structure of the task.

Considering all the results, HOLD-R and REDS underperformed in our experimental setting compared to the INEST-IRL. The HOLD-R model can obtain a reward shape similar to that of INEST-IRL but struggles to accurately capture the subtask divisions, as indicated by its higher distance. In contrast, REDS exhibited significant challenges in learning a consistent reward function, as reflected in both the erratic reward signal and high distance metric.

The inferior performance of HOLD-R and REDS in our experimental setting can be attributed to several key factors that misalign with the specific characteristics of our task and dataset.

The architecture of HOLD-R relies heavily on temporal distance prediction and regression-based learning, which tends to focus on low-level frame alignment rather than understanding the procedural and subtask-driven nature of complex manipulation tasks. In our "Sweep to top in order" task, HOLD-R has difficulty capturing the semantic relationships between objects and their sequential manipulation requirements. The model's regression objective, designed to predict temporal distances between states, becomes problematic when dealing with tasks that require an understanding of object identity, colour-based ordering, and multi-stage procedural execution.

As evidenced by the higher distance metric of 2.713410, HOLD-R struggles to understand the precise subtask division, even though the reward progression is partially correct. The model tends to overfit easily predictable visual features, such as the position of the gripper, while failing to capture the critical relationships between coloured blocks and their correct sequential placement in the target area. The generated embedding focuses more on the robot position and not on the object identities and their relationships. This limitation is particularly pronounced in our task, where the reward should reflect not only temporal progression but also the correct understanding of which block should be manipulated at each stage based on the prescribed order (Red, Blue, Yellow).

Although REDS was designed for subtask-aware learning, it exhibited significant performance degradation in our experimental context because of its heavy reliance on high-quality subtask segmentations and robust subtask classification during online interactions. The model's dependence on CLIP-based visual-text embeddings and contrastive learning objectives creates robustness gaps when dealing with the subtle visual transitions characteristic of our manipulation tasks.

The highest distance metric of 3.655451 achieved by REDS, along with the most erratic reward function plots, indicates fundamental challenges in accurately inferring task progress and maintaining consistency across the subtask transitions. In our task, where subtasks involve picking and placing coloured blocks in sequence, REDS struggles to maintain distinct representations for visually similar manipulation phases. The model's contrastive learning framework may misclassify current subtasks when the visual similarity between different phases (e.g. approaching differently coloured blocks) leads to ambiguous embeddings.

As reported in the original REDS paper [36], the model performance is sensitive to the quality of subtask annotations and the clarity of subtask boundaries. In our case, the subtasks were defined by object identity and colour, which may not provide sufficiently distinct visual cues for the REDS to effectively differentiate between them. The model's reliance on textual prompts for subtask identification may also introduce noise, especially if the visual differences between subtasks are subtle or if the prompts do not capture the full context of the manipulation task. In the limitations section of the original REDS paper, the authors noted that the model may struggle with tasks that have similar visual features but small text differences. Unfortunately, our task falls into this category, as the primary visual difference between the subtasks is the colour of the block being manipulated, whereas the overall scene and robot actions remain visually similar.

Furthermore, REDS's architecture assumes clear subtask boundaries and high-quality textual annotations, which may not adequately capture nuanced transitions in our manipulation task. The model's performance deteriorates when dealing with gradual state transitions or when the visual differences between subtasks are subtle, as is often the case in sequential manipulation tasks, where the primary distinguishing factor is the target object rather than the manipulation strategy itself.

The inferior performance of HOLD-R and REDS in our experimental setting highlights the importance of aligning model architectures and training objectives with the specific characteristics of the task and dataset used. In our case, INEST-IRL's design, which emphasises Procedural Learning and intrinsic motivation, proves more effective in capturing the complexities of the "Sweep to top in order" task, leading to superior performance in

both reward function learning and task execution.

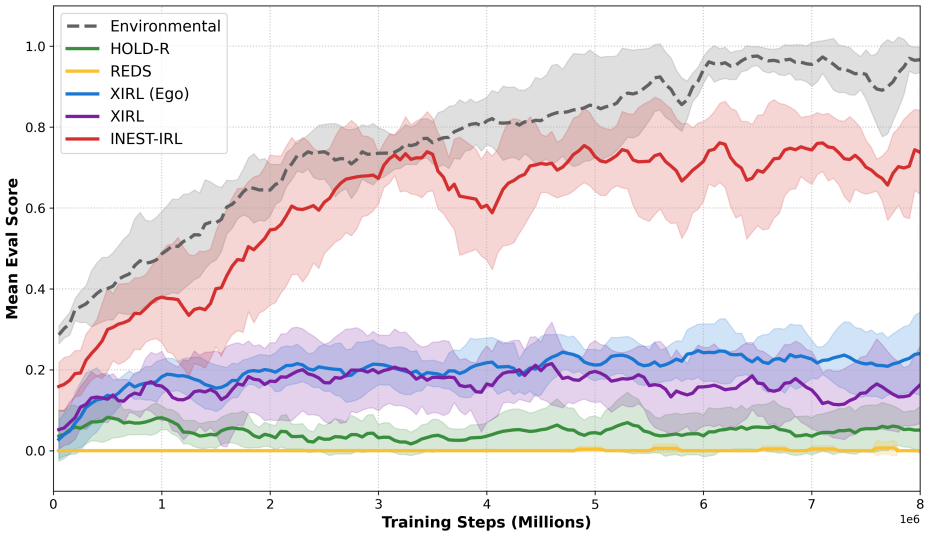


Figure 5.8: Evaluation scores for the INEST-IRL, XIRL (Allo and Ego), HOLD-R, and REDS baselines over 150 evaluation episodes. The INEST-IRL backbone demonstrates the highest average score and lowest variance, indicating more consistent performance across different initial configurations.

As shown in Figure 5.8, INEST-IRL achieved the highest final average evaluation score of 0.73, outperforming the XIRL baseline (0.16), XIRL baseline with the egocentric view (0.24), HOLD-R (0.05), and REDS (0.00). INEST-IRL also exhibited the lowest variance in performance across different evaluation episodes, suggesting that it was more robust to variations in the initial object configuration.

The superior performance of INEST-IRL can be attributed to its effective integration of Procedural Learning and intrinsic motivation, which together facilitate efficient exploration and mastery of the task structure. The model's ability to decompose tasks into meaningful subtasks allows it to focus on mastering each phase sequentially, leading to more effective learning and higher overall performance.

All the issues discussed previously for HOLD-R and REDS are reflected in their training performances. HOLD-R's low evaluation score of 0.05 indicates significant challenges in learning effective manipulation policies, likely because of its focus on low-level frame alignment rather than understanding the procedural nature of the task. The model's struggle to capture the semantic relationships between objects and their sequential manipulation requirements results in suboptimal policies that fail to effectively execute the required actions. Similarly, REDS' evaluation score of 0.00 highlights its inability to learn effective policies in our experimental context. The model's reliance on high-quality subtask segmentations and robust subtask classification proves detrimental when dealing with the subtle visual transitions characteristic of our manipulation task. The model's failure to maintain distinct representations for visually similar manipulation phases leads

to erratic performance and an inability to complete tasks successfully.

Model	End-of-Episode Subtask Completion (%)		Success Rate (%)
	1 block	2 blocks	3 blocks
XIRL	13.33	17.33	0.00
XIRL (Ego)	54.97	0.00	0.00
HOLD-R	17.22	0.00	0.00
REDS	0.00	0.00	0.00
INEST-IRL	0.00	77.55	**22.45**

Table 5.7: End-of-episode subtask completion and success rates for different baseline architectures. INEST-IRL shows the highest success for placing three blocks correctly, indicating its superior ability to complete the task in the correct order.

The table in Figure 5.7 shows the success rates of all the baselines. INEST-IRL demonstrated a higher success rate for placing three blocks (22.45%), indicating its superior ability to complete tasks in the correct order. The XIRL baseline showed a reasonable success rate for placing one block (54.97%) and two blocks (17.33%), whereas HOLD-R struggled to place one block correctly (10.22%) and failed to place two or three blocks. REDS failed to place any blocks correctly, indicating that it struggled to learn effective policies for the task completion.

Chapter 6

Conclusions and Future Work

In this study, we presented INEST-IRL, a novel architecture for learning reward functions from demonstration videos, which is specifically designed to address the challenges of complex manipulation tasks in robotics. Our approach integrates Procedural Learning through task subdivision and intrinsic motivation using an embedding-based Intrinsic Reward mechanism. We demonstrated the effectiveness of INEST-IRL through a series of ablation studies and comparative analyses with existing baseline architectures.

Ablation studies have highlighted the importance of the visual perspective, task subdivision, and intrinsic motivation in improving learning performance. The egocentric viewpoint was superior to the allocentric perspective, providing a more intuitive representation of the agent's interactions with objects. The subdivision of tasks into three subtasks emerged as the optimal granularity, balancing the need for structured learning with task complexity. In general, the number of subtasks depends on the task and should be selected based on the specific requirements of the manipulation task. The embedding-based Intrinsic Reward mechanism significantly improved exploration and learning efficiency, leading to higher evaluation scores and better task-completion rates.

Comparative analyses with existing baselines, including XIRL, HOLD-R, and REDS, demonstrated the superior performance of INEST-IRL in both reward function learning and task execution. The integration of Procedural Learning and intrinsic motivation proved crucial in capturing the complexities of the "Sweep to top in order" task, leading to more effective policies and robust performance across varying initial configurations.

In future work, several avenues can be explored to further enhance the capabilities of the INEST-IRL. One potential direction is to investigate more sophisticated methods for task subdivision, such as adaptive or hierarchical approaches that can dynamically adjust the granularity of the subtasks based on the agent's learning progress. In this study, the subtask division was defined manually. One possible idea is to incorporate better object-detection mechanisms and work with the relationship between the agent and object to automatically understand the task context and correct subtask division. Additionally, exploring alternative Intrinsic Reward mechanisms may provide further insights into how intrinsic motivation can be leveraged to improve learning efficiency.

Another promising direction is to extend the evaluation of INEST-IRL to a broader range of manipulation tasks and environments, including real-world robotic systems and

human environments. This would help to assess the generalizability and robustness of the approach in more complex and dynamic settings. Finally, integrating INEST-IRL with advanced Reinforcement Learning algorithms could further enhance its ability to learn effective policies from demonstration videos, potentially leading to more autonomous and adaptable robotic systems.

Bibliography

[1] Oliver Kroemer, Scott Niekum, and George Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of Machine Learning Research*, 2021. doi: 10.48550/arXiv.1907.03146. URL https://www.jmlr.org/papers/v22/19-804.html.

[2] Aude Billard and Danica Kragic. Trends and challenges in robot manipulation. *Science*, 2019. doi: 10.1126/science.aat8414.

[3] Dong Han, Beni Mulyana, Vladimir Stankovic, and Samuel Cheng. A survey on deep reinforcement learning algorithms for robotic manipulation. *Sensors*, 2023. doi: 10.3390/s23073762. URL https://www.mdpi.com/1424-8220/23/7/3762.

[4] Chen Wang, Linxi Fan, Jiankai Sun, Ruohan Zhang, Li Fei-Fei, Danfei Xu, Yuke Zhu, and Anima Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play, 2023. URL https://arxiv.org/abs/2302.12422.

[5] Shunyu Liu, Yunpeng Qing, Shuqi Xu, Hongyan Wu, Jiangtao Zhang, Jia Cong, Tian Chen, Yang Liu, and Ming Song. Curricular subgoals for inverse reinforcement learning. *arXiv preprint arXiv:2306.08232*, 2023. doi: 10.48550/arXiv.2306.08232.

[6] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2 edition, 2018. ISBN 978-0262039246. URL https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf.

[7] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *IEEE Signal Processing Magazine*, 2017. doi: 10.1109/MSP.2017.2743240.

[8] Leslie P Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 1996. doi: 10.1613/jair.301. URL https://www.jair.org/index.php/jair/article/view/10166.

[9] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 2013. doi: 10.1177/0278364913495721.

[10] Elena Pashenkova, Irina Rish, and Rina Dechter. Value iteration and policy iteration algorithms for markov decision problem. 1996. URL https://api.semanticscholar.org/CorpusID:8013442.

[11] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2016. doi: 10.48550/arXiv.1511.06581.

[12] Jan Peters. Policy gradient methods. *Scholarpedia*, 2010. doi: 10.4249/scholarpedia.3698.

[13] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. doi: 10.48550/arXiv.1707.06347.

[14] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[15] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

[16] Maryam Zare, Parham M. Kebria, Abbas Khosravi, and Saeid Nahavandi. A survey of imitation learning: Algorithms, recent developments, and challenges. *arXiv preprint arXiv:2309.02473*, 2023. doi: 10.48550/arXiv.2309.02473. URL https://arxiv.org/abs/2309.02473.

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 2020. doi: 10.1145/3422622.

[18] Stuart Russell. Learning agents for uncertain environments (extended abstract). In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT)*, pages 101–103. ACM, 1998. doi: 10.1145/279943.279964.

[19] Sateesh Kumar, Jonathan Zamora, Nicklas Hansen, Rishabh Jangir, and Xiaolong Wang. Graph inverse reinforcement learning from diverse videos. *arXiv preprint arXiv:2207.14299*, 2022. doi: 10.48550/arXiv.2207.14299. URL https://arxiv.org/abs/2207.14299.

[20] Ivan Rodin, Antonino Furnari, Kyle Min, Subarna Tripathi, and Giovanni Maria Farinella. Action scene graphs for long-form understanding of egocentric videos. *arXiv preprint arXiv:2312.03391*, 2023. doi: 10.48550/arXiv.2312.03391. URL https://arxiv.org/abs/2312.03391.

[21] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2023. doi: 10.48550/arXiv.2210.00030. ICLR 2023, Notable-Top-25

[22] Kevin Zakka, Andy Zeng, Pete Florence, Jonathan Tompson, Jeannette Bohg, and Debidatta Dwibedi. Xirl: Cross-embodiment inverse reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2021. doi: 10.48550/arXiv.2106.03911. URL https://arxiv.org/abs/2106.03911.

[23] Hanxiao Jiang, Binghao Huang, Ruihai Wu, Zhuoran Li, Shubham Garg, Hooshang Nayyeri, Shenlong Wang, and Yunzhu Li. Roboexp: Action-conditioned scene graph via interactive exploration for robotic manipulation. *arXiv preprint arXiv:2402.15487*, 2024. doi: 10.48550/arXiv.2402.15487. URL https://arxiv.org/abs/2402.15487.

[24] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):6999–7019, 2022. doi: 10.1109/TNNLS.2021.3084827.

[25] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015. doi: 10.48550/arXiv.1511.08458.

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*, 2013. doi: 10.48550/arXiv.1312.5602.

[27] Bert J. Claessens, Peter Vrancx, and Frederik Ruelens. Convolutional neural networks for automatic state-time feature extraction in reinforcement learning applied to residential load control. *IEEE Transactions on Smart Grid*, 9(4):3259–3269, 2018. doi: 10.1109/TSG.2016.2629450.

[28] Guoping Xu, Xiaxia Wang, Xinglong Wu, Xuesong Leng, and Yongchao Xu. Development of skip connection in deep neural networks for computer vision and medical image analysis: A survey. *arXiv preprint arXiv:2405.01725*, 2024. URL https://arxiv.org/abs/2405.01725.

[29] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yiping Xu, Zhaohui Yang, Yiman Zhang, and Dacheng Tao. A survey on visual transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. doi: 10.1109/TPAMI.2022.3152247. URL https://arxiv.org/abs/2012.12556.

[30] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[31] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. doi: 10.48550/arXiv.1904.07846. URL https://arxiv.org/abs/1904.07846.

[32] YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1118–1125, 2018. doi: 10.1109/ICRA.2018.8462901.

[33] Neha Das, Sarah Bechtle, Todor Davchev, Dinesh Jayaraman, Akshara Rai, and Franziska Meier. Model-based inverse reinforcement learning from visual demonstrations. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, Proceedings of Machine Learning Research. PMLR, 2021. URL https://proceedings.mlr.press/v155/das21a.html.

[34] Zhibo Yang, Lihan Huang, Yupei Chen, Zijun Wei, Seoyoung Ahn, Gregory Zelinsky, Dimitris Samaras, and Minh Hoai. Predicting goal-directed human attention using inverse reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[35] Minttu Alakuijala, Gabriel Dulac-Arnold, Julien Mairal, Jean Ponce, and Cordelia Schmid. Learning reward functions for robotic manipulation by observing humans. *arXiv preprint arXiv:2211.09019*, 2022.

[36] Changyeon Kim, Minho Heo, Doohyun Lee, Jinwoo Shin, Honglak Lee, Joseph J. Lim, and Kimin Lee. Subtask-aware visual reward learning from segmented demonstrations. *arXiv preprint arXiv:2502.20630*, 2025. doi: 10.48550/arXiv.2502.20630. URL https://arxiv.org/abs/2502.20630.

[37] Jiaming Yang, Kun Chen, Zhi Li, Shengkai Wu, Yu Zhao, Liangliang Ren, Wen Luo, Chen Shang, Meng Zhi, and Lu Gao. Bootstrapping imitation learning for long-horizon manipulation via hierarchical data collection space. *arXiv preprint arXiv:2505.17389*, 2025. doi: 10.48550/arXiv.2505.17389.

[38] Suyoung Lee, Myungsik Cho, and Youngchul Sung. Parameterizing non-parametric meta-reinforcement learning tasks via subtask decomposition. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 43356–43383. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/86c1fd74fa25bd6be0072937803e0bd1-Paper-Conference.pdf.

[39] Xiaoxiao Pan, Yan Xu, and Kris M. Kitani. Human-interactive subgoal supervision for efficient inverse reinforcement learning. *arXiv preprint arXiv:1806.08479*, 2018. doi: 10.48550/arXiv.1806.08479.

[40] Luca Marzari, Davide Corsi, Enrico Marchesini, and Alessandro Farinelli. Towards hierarchical task decomposition using deep reinforcement learning for pick and place subtasks. *arXiv preprint arXiv:2102.04022*, 2021. doi: 10.48550/arXiv.2102.04022.

[41] Adrian Šošić, Elmar Rueckert, Jan Peters, Abdelhak M. Zoubir, and Heinz Koeppl. Inverse reinforcement learning via nonparametric spatio-temporal subgoal modeling. *arXiv preprint arXiv:1803.00444*, 2018. doi: 10.48550/arXiv.1803.00444.

[42] Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. *Advances in Neural Information Processing Systems*, 34:18459–18473, 2021.

[43] Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pages 6736–6747. PMLR, 2021.

[44] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pages 11920–11931. PMLR, 2021.

[45] Tim Schneider, Boris Belousov, Georgia Chalvatzaki, Diego Romeres, Devesh K. Jha, and Jan Peters. Active exploration for robotic manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022. doi: 10.48550/arXiv.2210.12806. URL https://arxiv.org/abs/2210.12806.

[46] Sam Toyer, Rohin Shah, Andrew Critch, and Stuart Russell. The MAGICAL benchmark for robust imitation. In *Advances in Neural Information Processing Systems*, 2020.

[47] Alessandro Gasparetto and Lorenzo Scalera. A brief history of industrial robotics in the 20th century. *Advances in Historical Studies*, 2019. doi: 10.4236/ahs.2019.81002. URL https://doi.org/10.4236/ahs.2019.81002.

[48] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys*, 2017. doi: 10.1145/3054912. URL https://doi.org/10.1145/3054912.

[49] Muhammad Nasib, Hong Xiao, Dongbin Zhao, and Haiqin Yao. A survey of imitation learning algorithms: Recent developments and challenges. *ACM Computing Surveys*, August 2023. doi: 10.48550/arXiv.2309.02473. URL https://www.researchgate.net/publication/373714865_A_Survey_of_Imitation_Learning_Algorithms_Recent_Developments_and_Challenges.

[50] Sebastian Hofer, Kostas Bekris, Ankur Handa, Juan Camilo Gamboa, Melissa Mozifian, Florian Golemo, Christopher G Atkeson, Joelle Giguere, Heni Ben Amor, Martial Hebert, et al. Sim2real in robotics and automation: Applications and challenges. *IEEE Transactions on Automation Science and Engineering*, 2021. doi: 10.1109/TASE.2021.3064065.

[51] Elisa Salvato, Gianfranco Fenu, Eric Medvet, and Francesco A Pellegrino. Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning. *IEEE Access*, 2021. doi: 10.1109/ACCESS.2021.3126658.

[52] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018. doi: 10.48550/arXiv.1704.06888. URL https://arxiv.org/abs/1704.06888.

[53] Sujoy Paul, Jeroen Vanbaar, and Amit Roy-Chowdhury. Learning from trajectories via subgoal discovery. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/6f518c31f6baa365f55c38d11cc349d1-Paper.pdf.

[54] Pierre Sermanet, Kelvin Xu, and Sergey Levine. Unsupervised perceptual rewards for imitation learning. *arXiv preprint arXiv:1612.06699*, 2016. doi: 10.48550/arXiv.1612.06699. URL https://arxiv.org/abs/1612.06699.

[55] Siddhant Bansal, Chetan Arora, and C.V. Jawahar. My view is the best view: Procedure learning from egocentric videos. In *European Conference on Computer Vision*, 2022. doi: 10.48550/arXiv.2207.10883. URL https://arxiv.org/abs/2207.10883.