Language Grounding in Deep Reinforcement Learning for Dynamic Goal-Oriented Robotics

Vom Promotionsausschuss der
Technischen Universität Hamburg

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigte Dissertation (Monografie)

von
Frank Röder

aus
Kaduna, Nigeria

2026

Chair of the Examination Board:
Prof. Dr. rer. nat. Christian Lüthje

Reviewers:
Prof. Dr. rer. nat. Nihat Ay
Prof. Dr. Pierre-Alexandre Murena

Date of Oral Examination:
10. February 2026

DOI:
https://doi.org/10.15480/882.17098

ORCID:
https://orcid.org/0009-0008-9524-752X

In memory of my father, Norbert Heinrich Röder.

Abstract

Researchers have long attempted to teach robots and other embodied artificial agents to follow instructions, approaching language as the primary medium for communication, knowledge transfer, and cognition. While toddlers excel at language acquisition and utilizing it for problem-solving, robots and voice-based assistants struggle to achieve a grounded and robust understanding of natural language due to conversational noise, such as disfluencies and polysemy. This thesis investigates the limitations in language grounding that currently hinder the development of intelligent agents to comprehend and execute lingual goals, as well as their capacity to revise misinterpretations arising from underspecified or ambiguous instructions. We utilize a sparse reward-driven language-conditioned reinforcement learning setup and leverage insights from cognitive science and developmental psychology, presented in the following two pillars.

The first pillar explores the utilization of linguistic feedback and egocentric speech as mechanisms for learning from unsuccessful outcomes, by implementing a synthetic caretaker that provides feedback when the agent deviates from the expected course of actions. Unintended deviations may prove beneficial as alternative goal specifications, potentially satisfying different objectives. For instance, a robot might be assigned to prepare a cup of tea, but ends up brewing coffee instead, thereby accomplishing an unintended objective, in this case a different goal. In the case of egocentric speech, our research focuses on developing a multimodal translation model, designed to generate appropriate goal specifications based on observed behaviors. The model retrospectively predicts suitable goal commands that align with the observed actions, used for learning in hindsight. Both approaches of linguistic feedback and egocentric speech aim to emulate aspects of language development in young children and significantly enhance sample efficiency in robotic reinforcement learning.

The second pillar addresses the challenge of action correction, specifically targeting erroneous behaviors stemming from misinterpretations of goal specifications. We identify three distinct categories of misunderstanding: ambiguities arising from underspecified statements, unintentional miscommunications (*e.g.*, erroneously conveyed intentions), and discrepancies in common ground between the instructor and the robotic agent. Instead of learning with a different goal specification in hindsight, like in the first pillar, we aim to correct the misunderstanding through further verbal input from the operator. This provides an additional challenge for the agent, which needs to reconsider the original language goal given the new context and the returned action correction. By implementing a novel approach that incorporates the uncertainty about the actual goal and utilizing our methods from the first pillar, we demonstrate that egocentric speech significantly improves learning by generating action corrections in hindsight. We highlight this context-sensitive hindsight approach as the first in this domain to enhance the resolution of misunderstandings.

Acknowledgments

Writing this thesis has been a long, challenging, and formative journey. I could not have reached this point without the support, trust, friendship, and love of many people.

Above all, I am grateful to my supervisor, Dr. Manfred Eppe, whose intuition and support shaped my path into research during my master's studies. He gave me the opportunity to publish early in my research career and later supported me in starting my doctoral studies. Throughout this journey, he combined valuable feedback with the freedom and flexibility I needed to grow as a researcher. I am deeply grateful for this life-changing opportunity.

I am also grateful to Prof. Stefan Wermter, head of the Knowledge Technology Research Group, for his support during the first year of my doctoral studies. I thank the entire group for providing such a welcoming environment, especially my fellow doctoral students, with whom I began this journey and shared many valuable experiences.

I am very grateful to Prof. Nihat Ay for his support when I joined the Institute for Data Science Foundations and shifted the focus of my research. He gave me the opportunity to complete my doctoral studies in a stimulating environment with wonderful colleagues, including Jan Benad, Jesse van Oostrum, Carlotta Langer, Pradeep Kr. Banerjee, Adwait Datar, Leon Sierau, and Alexander Klemps.

I am equally grateful to my friends outside academia, especially my bouldering and music friends, who shared many enjoyable moments with me away from work. In particular, I would like to thank Momme, Marcus, Ray, Stephan, Martin, and many others.

Finally, I would like to thank my family for their generous and steady support. I am especially grateful to my wife, Hanna, for her patience, encouragement, and love; to my daughters, Mathea, Wilma, and Hilda; to my son, Jorit; to my sister, Dorit; and to my mother, Mitaire-Bright. My thanks also go to Sonja and Henning for their support throughout the years.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 3
- 1.2 Symbol Grounding . . . 5
- 1.3 Embodiment . . . 8
  - 1.3.1 Embodied Language . . . 8
  - 1.3.2 Embodied Communication . . . 10
- 1.4 Developmental Approach . . . 11
- 1.5 Research Objectives . . . 17
- 1.6 Summary of source code . . . 18
- 1.7 Short Outline . . . 18
- 1.8 Main Results and Outline . . . 18

2 Preliminaries and Background **34**
- 2.1 Deep Learning . . . 35
  - 2.1.1 Classification . . . 40
  - 2.1.2 Regression . . . 40
  - 2.1.3 Regularization and Normalization . . . 41
- 2.2 Reinforcement Learning . . . 44
  - 2.2.1 Markov Decision Process . . . 45
  - 2.2.2 Value Learning . . . 48
  - 2.2.3 Policy Gradient . . . 51
  - 2.2.4 Actor Critic Method . . . 51
  - 2.2.5 Goal Extension of the MDP . . . 52
  - 2.2.6 Current Reinforcement Learning Methods . . . 56
  - 2.2.7 Soft Actor-Critic . . . 58

3 Learning Language in Reinforcement Learning **63**
- 3.1 Related Work . . . 64
- 3.2 Background . . . 67
  - 3.2.1 Goal-Conditioned RL . . . 67
  - 3.2.2 Language-Conditioned RL . . . 68

3.2.3 Language Processing . . . 69
3.2.4 Representing the Instructions . . . 75
3.3 Language-Conditioned Soft Actor-Critic . . . 82
3.4 Language Robotics Environment . . . 86
3.4.1 Language Tasks . . . 86
3.4.2 Formal Grammar Description of Goal Instructions . . . 90
3.4.3 Pixel-based Observation . . . 90
3.5 Experiments . . . 92
3.5.1 Initial Experiments . . . 92
3.5.2 Full Experiments . . . 94
3.6 Summary . . . 98

4 Hindsight Language Learning **99**
4.1 Related Work . . . 103
4.2 Background Hindsight Learning . . . 106
4.2.1 Goals as State . . . 106
4.2.2 Language Goals . . . 108
4.3 Methods . . . 109
4.4 Expert Feedback . . . 110
4.5 Replay Strategies . . . 111
4.6 Hindsight Bias . . . 115
4.7 Hindsight Instruction Prediction from State Sequences . . . 116
4.7.1 Sequence-to-Sequence Modeling . . . 117
4.7.2 Transformer Encoder-Decoder Architecture . . . 121
4.7.3 Learning to Predict Hindsight Instructions . . . 128
4.8 Hindsight Language Learning Methods Overview . . . 129
4.9 Experiments . . . 132
4.9.1 Results Replay Strategies . . . 132
4.9.2 Grounding Hindsight Instructions . . . 133
4.9.3 New Replay Strategy Results . . . 135
4.9.4 Full Hindsight Learning Experiments . . . 137
4.9.5 Compositional Generalization . . . 141
4.10 Summary . . . 144

5 Action Correction for Language-Conditioned Reinforcement Learning **145**
5.1 Related Work . . . 148
5.1.1 Reinforcement Learning and Dialogs . . . 148
5.1.2 Conversational Corrections . . . 149
5.2 Situated Misunderstandings . . . 150
5.2.1 Ambiguity . . . 150
5.2.2 Lack of Common Ground . . . 151
5.2.3 Instruction Correction . . . 152
5.3 Description of Action Corrections . . . 153
5.4 **LANRO** Action Correction Extension . . . 156

5.5 Action Correction Baselines . . . 161
5.5.1 Solving uncertainties with uncertain critics . . . 161
5.5.2 Language-Conditioned DroQ . . . 162
5.6 Hindsight Learning with Action Correction . . . 163
5.7 Experiments . . . 167
5.7.1 Critic Ensemble Uncertainty . . . 170
5.7.2 Hindsight Action Correction . . . 173
5.8 Summary . . . 178

6 Conclusions **179**
6.1 Answering the Research Questions . . . 181
6.1.1 RQ1: Language Grounding with Sparse Rewards . . . 181
6.1.2 RQ2: Developmental Approaches for Efficient Language Grounding 181
6.1.3 RQ3: Modeling the Problem of Misunderstandings . . . 182
6.1.4 RQ4: Solving Misunderstandings with Action Corrections . . . 183
6.2 Limitations and Future Directions . . . 184

Appendices **186**

A Algorithm Details **188**
A.1 Language Encoder Details . . . 188
A.2 Language-Conditioned Soft Actor-Critic . . . 189
A.3 Language-Conditioned DroQ . . . 189
A.4 Hindsight Instruction Prediction from State Sequences . . . 190

B Experimental Results **192**
B.1 Language Experiments . . . 192
B.1.1 Setting with Two Objects . . . 193
B.1.2 Setting with Three Objects . . . 195
B.2 Hindsight Instruction Experiments . . . 198
B.2.1 Expert Feedback without Hindsight Bias . . . 198
B.2.2 Hindsight Instruction Prediction Experiments . . . 201
B.3 Action Correction Experiments . . . 203
B.3.1 **LCSAC** and **LCDroQ** Experiments . . . 203
B.3.2 Hindsight Action Correction Prediction . . . 204

General Definitions **208**

List of Symbols **210**

List of Acronyms **212**

Bibliography **214**

Chapter 1

Introduction

An ongoing challenge in artificial intelligence (AI) research is the development of robots and intelligent machines capable of acquiring a robust and grounded understanding of human language (Bischoff and Graefe, 1999; Steels, 2008; Tellex et al., 2020). Natural language is often ambiguous and highly context dependent, therefore making it to this date a challenging topic of current research. Recent advancements in conversational AI, particularly through the use of large language model (LLM) such as ChatGPT (OpenAI, 2023), have substantially accelerated the research progress in language comprehension and human-machine interaction. The integration of LLMs into robotic systems (Ahn et al., 2023; Liang et al., 2023) paved a new area of research and created many novel applications. Those systems appear to bridge the gap between the high-level capabilities of LLMs and pre-learned motor control of robots. Despite their impressive capabilities, these models still struggle with fundamental tasks requiring a common understanding of real-world physics that even young children possess (Baillargeon, 2002). Such models tend to oversimplify and *"hallucinate"* [1] inaccurate information, revealing gaps and limitations in their current capabilities. One significant shortcoming is the lack of grounding – a process of learning language based on additional sensor modalities beyond the language itself, such as visual and proprioceptive inputs (Steels, 2008; Matuszek et al., 2012; Hill et al., 2021; Xu et al., 2022). The language without grounding is also described as *meaning without reference* (Piantadosi and Hill, 2022), as purely textual systems have never experienced the tangible significance of words like *"hot"* or *"heavy"*.

The language processed by LLMs is inherently compositional. Sentences are composed of words, which in turn are formed by letters. Altering a single letter can change the meaning of a word, thereby potentially changing the overall meaning of the sentence. The meaning of a complex expression, like a sentence, is determined by its parts – the words and letters – and their specific arrangement. Current LLMs appear to capture syntactic compositionality, as they seem to comprehend the syntax and the subjects involved in sentences like *"she sneezed the napkin off the table"*, as illustrated in Figure 1.1. However, they cannot make sense of what is happening in the real world, as their meaning of words

[1] The term *"hallucinate"* is frequently used in LLM research, though we refrain from anthropomorphizing such system.

is primarily determined by statistical learning of co-occurrences. These solely textual representations can lead to semantically nonsensical results, such as in the aforementioned *hallucinations*, highlighting the lack of logical and coherent understanding. This issue renders these systems as unreliable and makes them appear to treat the grammatical structure independent of the meaning. The provided explanation hints at missing pieces that enable understanding the corresponding real-world implications involved in Figure 1.1. A solution to this would be a system that captures semantic compositionality. Understanding the meaning at this level requires the integration of information coming from multiple modalities, adhering to the concept of embodiment (Feldman, 2010; Röder et al., 2021).

Figure 1.1: An illustration depicting the sentence, *"she sneezed the napkin off the table"*. **Note**: The prompt to generate this image required considerable effort in tweaking the text to make the system DALL-E (Ramesh et al., 2021) – a model designed to generate detailed visual representations of textual descriptions – successfully produce the desired scene. This emphasizes the aforementioned limitations of current systems in understanding language at the level of actual world dynamics.

Researchers attempt to address this limitation by grounding the LLM afterward (Carta et al., 2023), essentially providing them with *"hands and eyes"* in the form of a robotic body (Ahn et al., 2023). However, this approach fundamentally differs from human language acquisition (Bisk et al., 2020; Röder et al., 2021), as toddlers can solve problems without relying on language in the first place (Mandler, 2004). This raises a critical question: Can we achieve equivalent language capabilities and meaningful representations if the grounding of advanced systems like LLMs is addressed as an afterthought? Resolving the misalignment between internal symbols and external world dynamics remains a persistent challenge, particularly when dealing with current state-of-the-art models that encompass a broad spectrum of knowledge (Radford et al., 2018; Devlin et al., 2019; Radford et al., 2019).

In this thesis, we consider autonomous agents that learn to interact with their world based on language commands and a guiding feedback signal alone. With this, we provide a more natural and grounded setup for language learning (Steels and Loetzsch, 2012), and

unlike language models (Brown et al., 2020), jointly consider language and other sensor inputs in a bottom-up fashion. This setting involves embodied nuances being part of natural language conversations, which we demonstrate with our methods and the corresponding experiments in later sections. Considering insights from cognitive science and developmental psychology, we prove that our grounded approaches to human-robot interaction are crucial for further progress in conversational AI and improved language understanding in robotics. We showcase a grounded compositional approach to language learning, where the learner repurposes past unsuccessful experiences by retrospectively translating them into successful ones. As a surprising core contribution of this work, we present a mechanism to address the problem of misunderstanding by enabling the learner to *"talk to itself"*, which improves resolving misunderstandings significantly.

1.1 Motivation

This section motivates the domain of embodied language learning, building upon our previous work Röder et al. (2021). It demonstrates how language learning for artificial agents and robots is fundamentally rooted in research on developmental psychology and cognitive science. Drawing inspiration from these fields, we identify the essential components and justify their necessity in addressing the challenges central to this thesis within the field of language-conditioned reinforcement learning for robotics.

Natural language understanding and generation has been a central focus in machine learning and robotics for decades (Wermter and Weber, 1997; Bengio et al., 2003; Steels, 2008; Tellex et al., 2011; Narasimhan et al., 2015; Bisk et al., 2016; Shridhar et al., 2022). Recent breakthroughs with large language models (Devlin et al., 2019; Radford et al., 2019) have sparked significant interest, leading to novel applications such as the widely recognized ChatGPT system (OpenAI, 2023).

While LLMs exhibit impressive capabilities in problem-solving, *e.g.*, programming challenges and reading comprehension tasks, they often struggle to understand the nuanced complexities of natural language and real-world context. This limitation becomes particularly evident when faced with tasks requiring lateral thinking, such as riddles and puzzles, where these models may fail to make intuitive leaps necessary for resolution.

However, as the datasets on which the models were trained are not publicly available, it remains unclear to what extent certain responses are basic lookups versus utilizing knowledge stored in the billions of parameters. A similar question of knowledge storage could be asked about the human brain, where we have evidence for distributed representations of language, making it clear that there exist neurons used for multiple purposes, involved in different kinds of cognitive and bodily processes (Rizzolatti and Arbib, 1998; Pulvermüller, 2005).

The next-word-prediction objective leading to such capable systems that are able to solve a wide array of tasks, such as summarization, translation, and classification (Brown et al., 2020), was unexpected. However, this idea aligns with the hypothesis of many researchers that intelligence necessitates predictive capabilities by utilizing, for instance, world models (Schmidhuber, 2010; Hafner et al., 2021; LeCun, 2022). Critics argue that

LLMs merely replicate the statistical patterns of conversations presented in their training corpora, earning them the label *stochastic parrot* (Bender et al., 2021). Conversely, some researchers posit that the language (or token) space of LLMs represents a compressed version of our real world processes that generated the text (Gurnee and Tegmark, 2024).

Unlike human language development, LLMs are trained exclusively on text. While recent research has explored incorporating other modalities (Ma et al., 2023), text remains the primary foundation upon which other modalities build. Despite efforts to address the lack of multimodal inputs in the LLM training, these models remain passive observers, unable to interact with the real world. This is, to what we know, also in stark contrast to language development in humans, which involves active interaction with the environment to learn and internalize – or literally *"grasp"* – the language.

Toddlers demonstrate remarkable problem-solving abilities, including dexterous manipulation, navigation in unfamiliar environments, and sorting of unknown objects. These skills often surpass the capabilities of most, if not all, current contemporary robotic systems. Notably, toddlers achieve this without an extensive vocabulary or even the use of language at all (Mandler, 2004; Röder et al., 2021).

Human language development appears to follow a natural curriculum that initially focuses on motor skill acquisition, driven largely by intrinsic motivation (Oudeyer et al., 2007; Schmidhuber, 2010). Language skills emerge later, followed by advanced reasoning abilities that require hierarchical abstraction (Röder et al., 2021; Eppe et al., 2022). This developmental trajectory highlights the complex interplay between physical interaction, language acquisition, and cognitive development in humans (Mandler, 2004).

Language could be viewed as providing the discrete structure for the continuous and messy real world perceptions. This concept is well illustrated by the act of grasping objects. There exists a wide spectrum of grasping techniques, ranging from full-handed grips to delicate fingertip manipulation, and even approaches with both hands. The diversity of graspable objects further complicates this scenario, as each item may require subtle variations in handling. However, toddlers demonstrate proficiency in these complex motor skills before developing advanced linguistic abilities, such as advanced language comprehension or generation. Furthermore, Paulus (2014) postulate that infants imitate actions before they even fully develop language. This observation strongly suggests the presence of fundamental cognitive and motor capacities upon which language acquisition subsequently builds (Mandler, 2004).

To develop sophisticated natural language generation capabilities for communication, a fundamental prerequisite is the ability to comprehend and formulate appropriate responses. In human language acquisition, the social context plays a crucial role (Gallese, 2008; Achimova et al., 2022). The learner initially acquires language through exposure to expert speakers, interpreting their utterances across various situations, and gradually develops the ability to generate language for communication purposes. This principle also applies to the first iterations of Generative Pre-trained Transformer (GPT) models (Radford et al., 2018; Radford et al., 2019; Brown et al., 2020), upon which the successor model InstructGPT (Ouyang et al., 2022) was built. Rather than merely completing text, incorporating human feedback into the learning process subsequently led to models that were useful for communicating with users, as demonstrated by Stiennon et al. (2020).

For ChatGPT, this approach was further refined through a method known as *reinforcement learning from human feedback* (Christiano et al., 2017; OpenAI, 2023). This technique enables the model to learn about preferences regarding the generated answers in terms of a reward signal that it needs to maximize. Similar to a child learning, the model receives an extrinsic reward for answers that improve communication with the human teachers. The objective is to maximize the conversational reward, which in this case translates to optimizing user satisfaction. Nevertheless, the model lacks the intrinsic motivation to explore its environment, and unlike learning children, relies heavily on instructor feedback for improvement.

However, conversational systems like ChatGPT still offer valuable insights into language processing and generation. Beyond mere word prediction, these systems demonstrate the role of language as a cognitive tool for imagination, planning, abstraction, and reasoning, as postulated by Vygotskij (1985). The extensive research on prompting further underscores this concept, demonstrating how language generation can be harnessed to emulate mechanisms analogous to human cognitive processes (Wei et al., 2024). LLMs provide valuable high-level task decomposition but lack true understanding, operating more as system 1 completion engines rather than engaging in system 2 thinking (Kahneman, 2011). The concept of steering language models to produce useful outputs that align with the programmer's preferences is akin to how humans learn from external feedback provided by their caretakers. This reward-driven learning at the core of human development echoes the essence of reinforcement learning (RL), as outlined by Sutton and Barto (2018). The integration of humans as caretakers for artificial agents has proven beneficial in robotics (Christiano et al., 2017; Faulkner et al., 2020) and has been applied to LLMs (Ouyang et al., 2022). RL has demonstrated success in training intelligent systems to solve challenging board (Schrittwieser et al., 2020) and video games (Mnih et al., 2015; Vinyals et al., 2019). Recent advancements have even extended RL to the application of algorithm design (Fawzi et al., 2022) and datacenter cooling (Luo et al., 2022).

As a trial-and-error approach, RL can be costly and sample inefficient, limiting its use in many real-world settings. The exploration phase, crucial for RL systems to learn, may pose risks to physical robots if not executed in a simulation or controlled environment. This thesis focuses on an end-to-end approach to solving RL tasks in an embodied setup, driven solely by external reward signals within a simulated environment. While the mechanisms by which words acquire meaning remain partially unclear (Feldman and Narayanan, 2004; Bisk et al., 2020; Piantadosi and Hill, 2022), this research aims to investigate essential aspects of language learning. A promising approach to language learning and the acquisition of its semantics is the concept of grounding, as proposed by Bisk et al. (2020).

1.2 Symbol Grounding

The following section describes the idea and origin of symbol grounding and highlights its relevance in the context of robotic language learning considered in this thesis.

The question of how words, symbols, or tokens get their meaning is the topic of symbol grounding. It states that symbolic representations must be acquired through a bottom-up

learning process, allowing symbols to obtain a meaning internal to the system rather than being externally assigned. Cognitive scientist Stevan Harnad introduced this as the *Symbol Grounding Problem* in Harnad (1990), drawing inspiration from the *Chinese Room Argument*. It is a thought experiment presented by John Searle in Searle (1980) to refute the notion of the *"strong AI hypothesis"*. The argument is designed to show that a computer executing a program cannot possess a genuine understanding, regardless of how thoughtfully it was designed by humans. In this thought experiment, one imagines a person who does not comprehend Chinese, situated in a room filled with boxes of Chinese characters and a rule book in English for manipulating these symbols. The person receives questions in Chinese through a slot in the door, uses the rule book to match these questions with appropriate responses, and sends them back through the slot. To an external observer, it appears that the person inside the room understands Chinese and is responding intelligently. The argument is that the person inside the room does not understand Chinese at all; they are merely following syntactic rules to manipulate symbols without any genuine comprehension of their meaning. This scenario illustrates that one can program a computer to simulate understanding without truly understanding. One can now posit that systems like the aforementioned ChatGPT are similarly acting primarily on the syntactic level of understanding. A major effort in AI research has been directed towards addressing this limitation by integrating mechanisms into the models to overcome these constraints. Grounding refers to the process of acquiring language in conjunction with other sensory modalities, such as visual and proprioceptive inputs. It involves establishing connections between internal symbolic representations and direct real-world experiences (Harnad, 1990). This process necessitates that symbols – or in our case, words – are not arbitrarily assigned but follow an attribution with systematicity. Despite the seemingly chaotic property of real-world sensory information, we can often find structures that can be appropriately described using language. Human perception operates in a conceptualized manner (Kiefer and Pulvermüller, 2012), with the mind naturally organizing information into categories, like plants, animals, tools, and so on (Rosch, 1978). Rather than dealing with isolated symbols or words, we utilize sentences as combinations of words arranged in specific formations. Current machine learning models can acquire language solely through statistical frequency and co-occurrences analysis alone (Mikolov et al., 2013), highlighting the unique structural properties of language.

Consider the sentence *"The big brown grizzly roars loudly"* as illustrated in Figure 1.2. We argue that a system incapable of visual and auditory perception cannot truly comprehend this statement or at least has a vastly different internal representation. Understanding the visual representation of *"big"* and the auditory sensation of *"loud"* is fundamental to grasping the meaning of these words. Words can be considered as triggers for mental simulations (Rizzolatti and Arbib, 1998), such as visualizing a large brown bear when hearing or reading about one. Grounding language extends beyond passive observation and should encompass the actions executed by an autonomous system. This concept has been recognized as a crucial component for learning systems, such as robots, and considered by numerous researchers like Steels, 2008; Cangelosi, 2010; Tellex et al., 2020; Lynch and Sermanet, 2021 and others (Wermter et al., 2005; Spranger et al., 2014; Heinrich et al., 2020). Roy (2005) proposes that a first-person perspective of cognitive systems is essential

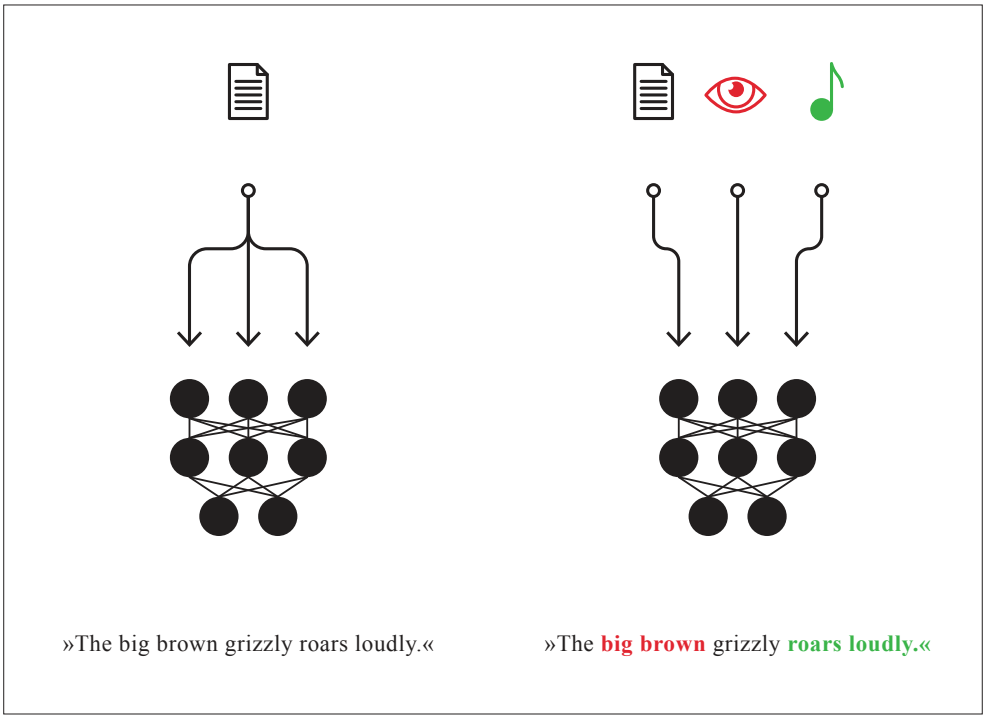


Figure 1.2: We illustrate the relevance of other modalities when processing the sentence *"The big brown grizzly roars loudly"*. To the left, a text-only model has no access to the conceptual appearances related to visual and auditory descriptions. It can only make sense of these on a textual basis. To the right, we illustrate mental concepts that get triggered by hearing key words, such as *"big brown"* and *"roars loudly"*. This figure is taken from our previous work on embodied crossmodal language learning (Röder et al., 2021).

for establishing connections between sensory data, language, and motor actions. Systems that rely only on symbolic representations to ground word meanings are inherently limited and fundamentally different from the author's suggested approach. The latest iteration of ChatGPT, incorporating the GPT-4V model for vision capabilities, serves as an illustrative example (OpenAI, 2023). Despite its visual processing, the representations of GPT-4V still rely on discrete categorical assignments linked to labels, which are ultimately tokens or words rather than abstract conceptual representations (Confalonieri et al., 2016). In contrast to digital agents, robotics provides a physical embodiment with low-level sensory inputs from various modalities, such as touch, vision, and audio. Robots also incorporate output devices – the actuators – enabling them to manipulate objects and execute bodily movements. However, robots exhibit diverse morphologies and actuator types, which determine their functional capabilities and operational methods. The field that investigates

the body as an integral part of intelligence is known as embodied cognition.

1.3 Embodiment

The upcoming section expounds the notion of embodiment and particularly discusses its relevance to language and communication.

Embodied cognition is a concept that proposes that many features of cognition, such as thoughts, feelings, and behaviors, are deeply influenced by aspects of the physical body. It emphasizes the significant influence of the motor and perceptual systems, as well as the situated body, on the mental state. A recent study by Anderson (2003) suggests that thinking is a *situated activity*, highlighting cognition as predominantly influenced by external factors rather than predominantly internal processes, as previously believed. In the context of artificial agents, embodiment refers to the emergence of intelligence through the interplay between the agent's body, its brain, and the environment, as defined by the configuration of the sensorimotor loop according to Der and Martius (2011). Consequently, the resulting behavior is not solely a product of abstract computational processes by the agent but also deeply influenced through its bodily experiences and interactions with the environment. Many studies endorse systems that consider this situatedness and the ability to perceive the physical world in a bottom-up manner (Steels, 2008; Eppe et al., 2022).

In summary, embodied cognition is a phenomenon with various aspects that span several disciplines and have many overlaps. Figure 1.3 provides an overview of the topics to be discussed in the following sections. Cognitive science and cognitive psychology play crucial roles in understanding embodied cognition. Specifically, we will examine language, perception, decision-making, and social cognition, considering their relevance to our artificial intelligence setup for robotics.

1.3.1 Embodied Language

The concept of embodiment plays a major role in the acquisition and comprehension of language. Language appears to be intertwined with various sensory systems. Research by Pulvermüller (2005) and Fischer and Zwaan (2008) demonstrates its active engagement with the motor system. Recent studies suggest that language understanding involves internal simulations facilitated by mirror neurons (Fischer and Zwaan, 2008; Gallese, 2008). Investigations in neuroscience have revealed motor neuron activation when individuals perceive action-related words such as *grasping* or *jumping* (Rizzolatti and Arbib, 1998; Feldman and Narayanan, 2004). This interconnection is further exemplified by the colloquial expression *"to grasp something"*, often used to describe the process of understanding language or comprehending written text (Jirak et al., 2010). These insights offer an alternative perspective on it, suggesting that language need not always be abstract, particularly when words refer to concrete real-world experiences. Recent works on embodied language learning with artificial agents demonstrate the impact of acquiring language alongside sensorimotor experiences (Steels and Loetzsch, 2012; Spranger et al., 2014). More recent investigations (Chaplot et al., 2018; Narasimhan et al., 2018; Hill et al., 2020; Hill et al.,

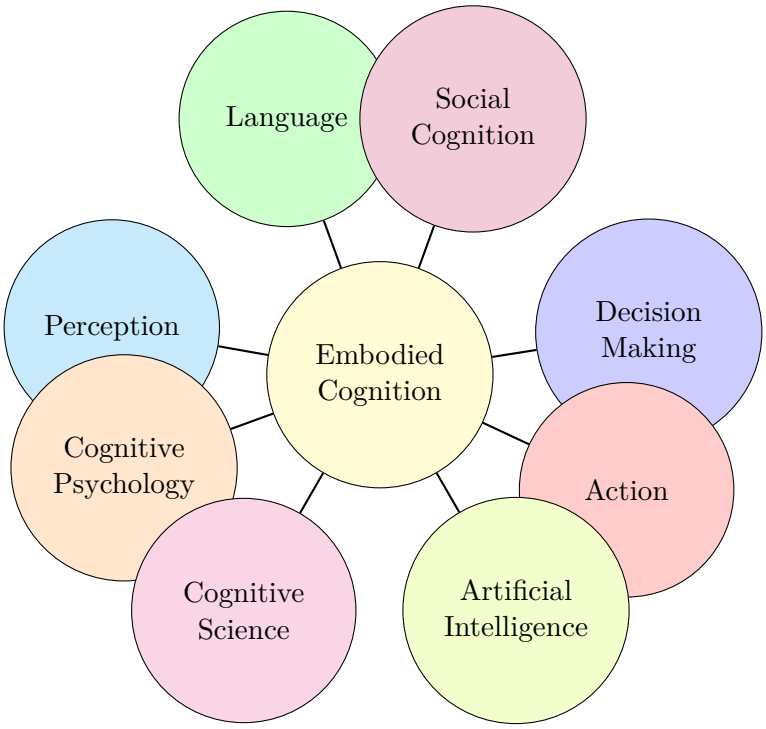


Figure 1.3: Embodied cognition and its connections to other research fields. The grouping of *Decision Making*, *Artificial Intelligence*, and *Action* could be related to the field of robotics and reinforcement learning (Sutton and Barto, 2018), whereas *Perception*, *Cognitive Psychology*, and *Cognitive Science* in this thesis could be assigned to the field of developmental psychology and developmental robotics (Cangelosi and Schlesinger, 2015; Eppe et al., 2021). *Language* and *Social Cognition* (Mead et al., 2000) have special roles in this thesis, as they define the means of goal communication and what the reward actually corresponds to.

2021) highlight methods for grounding language in an end-to-end reinforcement learning setting, driven solely by an extrinsic reward signal. However, these reinforcement learning approaches typically involve a situated agent within an environment, perceiving the world from a first-person perspective and observing language and objects in conjunction with the reward signal that facilitates grounding. Current limitations of these approaches include the necessity for millions of learning steps, resulting in substantial training time and high computational costs. The work of Colas et al. (2020) and Akakzia et al. (2021) indicates that incorporating social factors, such as the presence of a teacher to assist, can significantly enhance the learning process, particularly in scenarios involving human interaction. Nevertheless, setups involving human participation are often cumbersome and challenging to scale. Moreover, providing language feedback can be problematic in certain situations, especially when evaluating the current context is difficult. For instance, consider the complexity of a human providing language feedback for a robot attempting a complex motion like a backflip, which is challenging to explain due to the morphological and actuator differences between humans and robots. This noisiness of natural language

needs to be accompanied by sensor experiences that not all robotic applications consider. Consequently, current AI research is facing limitations by disregarding the factors of embodiment, hindering effective communication between humans and robots.

1.3.2 Embodied Communication

Human communication encompasses both verbal interactions through speech and nonverbal elements like gestures, facial expressions, and body language (Tsironi et al., 2017). The nonverbal components especially play a significant role in human-to-human communication and necessitate the presence of a physical body. Recent work in human-robot interaction suggests that even tiny additions, like a robot mimicking facial expressions with simple LED lights (Kerzel et al., 2017), improve the trustworthiness and atmosphere of the interaction.

Natural language is inherently embodied and grounded in sensorimotor representations, functioning as an integral part of human cognition (Vygotskij, 1985) and plays a major role in social cognition (Mead et al., 2000) (see Figure 1.3). It is often characterized by the appearance of disfluency, polysemy, and ambiguity. A current challenge of contemporary research is to address this linguistic noisiness through methods that account for uncertainties in communication and techniques to query additional information. A real-world example of this challenge is the mental continuation of a conversation that occurred in the past. One might recall a friend discussing recent news events, which raised numerous unanswered questions while listening. Upon reading the newspaper later, and acquiring updated information related to the past conversation, one can still reference moments from that discussion, resolve misunderstandings, and gain a clearer comprehension of what was said afterward. A more common approach, much more related to the work presented in this dissertation, is to address communication uncertainties in real-time (Shi et al., 2024), while the interaction takes place.

In the work of Clark and Brennan (1991), the authors propose that human-to-human communication continually establishes a shared understanding of beliefs, knowledge, and assumptions among participants. As this process requires a mutual interaction, it can be viewed as a group activity (Clark and Brennan, 1991). However, this differs from what current systems like ChatGPT provide, as these lack a real intent and respond passively to user inputs. Communication scenarios may also involve participants where one only responds nonverbally through physical actions, a common occurrence in human-robot interaction. The absence of natural language generation does not necessarily simplify the task; it remains comparably challenging, as the robot must still interpret its actions in relation to the human's utterances. For agents to effectively understand and generate natural language, they must develop rich representations that go beyond purely linguistic information. These representations should encompass both sensory information and environmental context, pointing to the importance of multimodal approaches (Röder et al., 2021) that consider both the physical environment and broader situational context in which communication occurs (Bisk et al., 2020). An illustrative example is the sensory input of an image in the context of the utterance *"hand me the fruit"*. If no fruit is visible to the agent, an appropriate response would be to initiate a search routine by moving the

camera around. This needs to be considered by the operator when instructing the robotic agent and necessitates a unified common conceptual view of the world, shaped by similar multimodal representations for successful communication.

To further elucidate the necessity of embodied communication in AI and robotics, consider the expressions *"seeing the world through rose-colored glasses"* or *"light at the end of the tunnel"*. The first metaphorical phrase conveys optimism or a tendency to view things in a positive light, blending the physical act of seeing with the symbolic meaning of rose-colored glasses. The second metaphor, while not referencing physical light or a literal tunnel, symbolizes hope and the possibility of emerging from difficulty. For humans, deciphering these metaphors involves comprehending both the literal meanings of words and their symbolic significance within a varying context (Feldman and Narayanan, 2004). This process may involve the subconscious visualization of reaching the light at the end of the tunnel while simultaneously considering the literal implications of a tunnel's length and darkness, as well as the light's role as an exit. However, for AI-based systems, deciding whether to interpret these phrases literally or metaphorically – such as contextualizing the act of seeing and colored glasses within the notion of optimism, or associating the light and tunnels with the concept of emergence – remains an unresolved challenge. In human-robot interactions, it is a critical feature to detect when systems fail to understand, rather than producing confidently incorrect responses or, in the worst case, applying unintended real world changes that could cause harm.

Various approaches exist for implementing verbal and physical conversational interactions in this domain. Many of these conversations in human-robot interaction are facilitated through dialog systems (Weston, 2016), which have traditionally been unimodal, with current research efforts focused on expanding their multimodal capabilities (Li et al., 2020b). However, these systems typically do not incorporate low-level physical interactions within simulated or real environments, and often prioritize the application over bottom-up learning approaches. Alternative embodied conversational setups are implemented using RL (Luketina et al., 2019; Colas et al., 2020; Akakzia et al., 2021) or related approaches, such as imitation learning (Lynch and Sermanet, 2021; Mees et al., 2022). In this thesis, we focus on a setup where we equally represent the sensorimotor experiences and the language, facilitating the emergence of a multimodal representation for solving physical decision-making tasks based on external reward signals.

1.4 Developmental Approach

The following section highlights two robotic learning approaches that utilize the notion of embodiment for effective language grounding, as outlined in Röder et al. (2021).

In our review on embodied language grounding (Röder et al., 2021), we highlight the developmental stages of humans and draw parallels with current robotics reinforcement learning systems. Analogous to toddlers, robots and intelligent agents must leverage language as a cognitive tool to improve their capabilities regarding planning, reasoning, abstraction, and imagination (Mirolli and Parisi, 2011). Unlike current AI research, human development could be distinctly separated into two phases: skill learning and language

learning. A large part of infant learning happens independently of language utilization. The vast amount of sensory experiences is subsequently grounded and associated with corresponding concepts from the sensorimotor skill-learning phase (Mandler, 2004). One could posit that language provides a discrete structure to the continuous and complex world of sensorimotor signals, thus serving as a tool for conceptual organization (Waxman and Markow, 1995). A central notion to skill acquisition is the role of play (Vygotsky, 1967). Play is usually driven by intrinsic motivation (Oudeyer et al., 2007), wherein children come up with their own objectives and goals, exploring phenomena that diverge from their preexisting beliefs.

In our review (Röder et al., 2021), we discovered that current research already provides the necessary components for implementing the first (Eysenbach et al., 2019; Lynch et al., 2020) and the second phase (Hanjie et al., 2021; Hill et al., 2021). However, only a limited number of approaches (Lynch and Sermanet, 2021; Akakzia et al., 2021) combine them jointly as suggested by research on language development (Cangelosi, 2010). Currently, AI systems take a different approach to language learning, by first leveraging vast textual datasets and exploiting the efficiency of scaling compute (Devlin et al., 2019; Brown et al., 2020), following the notion prescribed in *The Bitter Lesson* by Richard Sutton[2].

Nevertheless, employing a training methodology that significantly deviates from our understanding of human language development still results in systems with inherent limitations. A major difficulty arises from the lack of grounding that shows up with a weak connection between the language and the sensor space. This deficiency leads to constraints in comprehending physical properties and real-world phenomena. Recent research endeavors to address this shortcoming by attempting to retrospectively ground these models (Ahn et al., 2023; Carta et al., 2023). In the following two paragraphs, we highlight two approaches presented in Röder et al. (2021) that follow the proposed curriculum.

First Approach Akakzia et al. (2021) present the Language-Goal-Behavior (LGB) architecture that addresses the autonomous acquisition of repertoires of skills in a reinforcement learning setup. Their LGB approach implements an autonomous learning routine by decoupling skill learning (goals to behaviors) from the language grounding (language to goals). In the initial exploration phase, the learning is driven by intrinsic motivation to build up essential skills for manipulating objects following self-generated goals. A predefined semantic representation with binary predicates (*e.g.*, A above B, B close to C) is employed to indicate the presence or absence of specific properties and spatial relations among three objects on a table. The agent is rewarded for discovering as many of those semantic goal configurations as possible, especially those with a high learning progress, enabling it to build up a variety of skills. The second phase actually has two parts. First it focuses on language grounding, where the social partner provides descriptions of what changes the agents caused to the world state. This allows learning a mapping from language instructions to behaviors and their corresponding semantic goal configurations, with the possibility to learn multiple semantic goals for a single language goal. Secondly, the instruction-following phase jointly considers competences from the skill learning phase

[2]Visited March 1, 2024, from http://www.incompleteideas.net/IncIdeas/BitterLesson.html.

and the language grounding phase, where the learner is rewarded achieving proposed goal instructions. Their method, inspired by the pre-verbal goal-directed behavior of infants (Mandler, 2004), demonstrates the capacity for autonomous exploration and mastery of spatial configurations through binary predicates. This approach simplifies learning and open-ended skill acquisition, but also constrains the system to a fixed set of objects due to the necessity of predefined semantic representations, which require domain-specific knowledge. Consequently, the scalability to more than three objects is limited, while they bypass a fundamental aspect of language acquisition: the process of transforming continuous, possibly noisy, sensory inputs into discrete, symbolic representations of words and concepts.

Second Approach In the work of Lynch and Sermanet (2021), the authors employ an imitation learning approach based on data generated through unstructured play. To collect the dataset, human operators remotely controlled a robot without a specific task provided. Naturally, the operators exhibited curiosity, exploring the table setting by opening the cabinet and manipulating objects. As their intrinsic motivation decreased, they displayed signs of boredom, resulting in seemingly arbitrary actions that nonetheless generated useful data samples. The collected dataset, comprising 7 hours of robot teleoperation, was subsequently split into windows lasting a few seconds, with the initial frame representing the starting state and the final frame depicting the achieved goal state. The robot was then trained to replicate the behaviors by learning to achieve the visual goal states, starting from the initial window frame, concluding the skill learning phase. Their architectural design incorporates a novel approach to goal representations: goal images are mapped to a latent space that encodes specific goal-conditioned behaviors, so-called latent plans. For instance, opening the cabinet and opening the door are goals represented by very similar latent plans. The next phase exploits the latent representations by learning their corresponding language-conditioned behaviors. For this, the authors employed Amazon Mechanical Turk workers to annotate the windows with language afterward. Some windows captured interesting exploratory behaviors with matching instructions like *"open the cabinet"*, while others reflected more mundane actions describable as *"do nothing"*. Remarkably, annotating less than 1% of the play dataset proved sufficient for the agent to comprehend and successfully follow language instructions. The following training process incorporates samples that provided both visual and language goals. Subsequently, Lynch and Sermanet (2021) demonstrate that by masking the visual goals and only retaining the language goal, the system could extrapolate to novel situations. The structure of the latent space enables the agent to effectively map both modalities to a shared representation, enabling the few language examples to exploit the structure resulting from the training on the rich visual play dataset. Although their method requires additional human effort for collecting the dataset and providing language annotations retrospectively, we posit that these steps could potentially be replaced by purely reward-driven approaches from reinforcement learning.

Takeaway The two discussed RL approaches (Akakzia et al., 2021; Lynch and Sermanet, 2021) present vastly different implementations of the two phases, skill learning and language grounding. We anticipate future work to address their limitations regarding the human annotation labor and the hand-engineered semantic representations. While numerous additional approaches to learning these two phases exist, to the best of our knowledge in Röder et al. (2021), none replicated them as closely as the aforementioned works. In Figure 1.4, we illustrate the different components of skill learning and language grounding, demonstrating how they could be implemented by current RL approaches.

Existing works already provide advanced techniques to replicate the phases. Especially for skill learning, numerous approaches simulate the play phase through intrinsic rewards, which could be derived from prediction errors (Oudeyer et al., 2007; Röder et al., 2020) or information-theoretic measures such as the mutual information (Eysenbach et al., 2019) or the entropy (Haarnoja et al., 2018). Additional methods to enhance skill learning and replicate human cognition include those utilizing learned world models (Ha and Schmidhuber, 2018; Hafner et al., 2021) to accelerate the learning via mental simulations, as depicted in Figure 1.5.

While human development is largely driven by self-supervised exploration, it also incorporates imitation learning. Related works in this domain of learning from demonstrations are also referred to as behavior cloning (Nair et al., 2018a) and associated with offline reinforcement learning (Chen et al., 2021a).

Colas et al. (2022) present the holistic approach of autotelic agents, integrating intrinsically motivated exploration (Colas et al., 2019; Röder et al., 2020) with the ability to set their own goals (Colas et al., 2020). With autotelic agents, Colas et al. (2022) introduce the concept of a social partner into the syllabus, which communicates with the learner by providing language descriptions for observed behaviors (Colas et al., 2020; Röder et al., 2022) or instructs the agent to follow assignments.

In summary, the developmental trajectory of toddlers involves a pre-verbal phase mainly driven by intrinsic motivation and exploration for skill acquisition, prior to the active use of language. Consequently, language grounding takes place after the skill learning phase (Mandler, 2004). However, it is worth noting that these phases may unfold simultaneously or through a gradual transition. This thesis will not implement the entire framework outlined in this section and as detailed in our corresponding publication (Röder et al., 2021). Instead, it considers the relevant aspects of instruction-following by focusing on the relevant parts of early word learning (Plunkett et al., 1992; Hill et al., 2019) as a basis and motivation for our work on improving language grounding in the context of reinforcement learning.

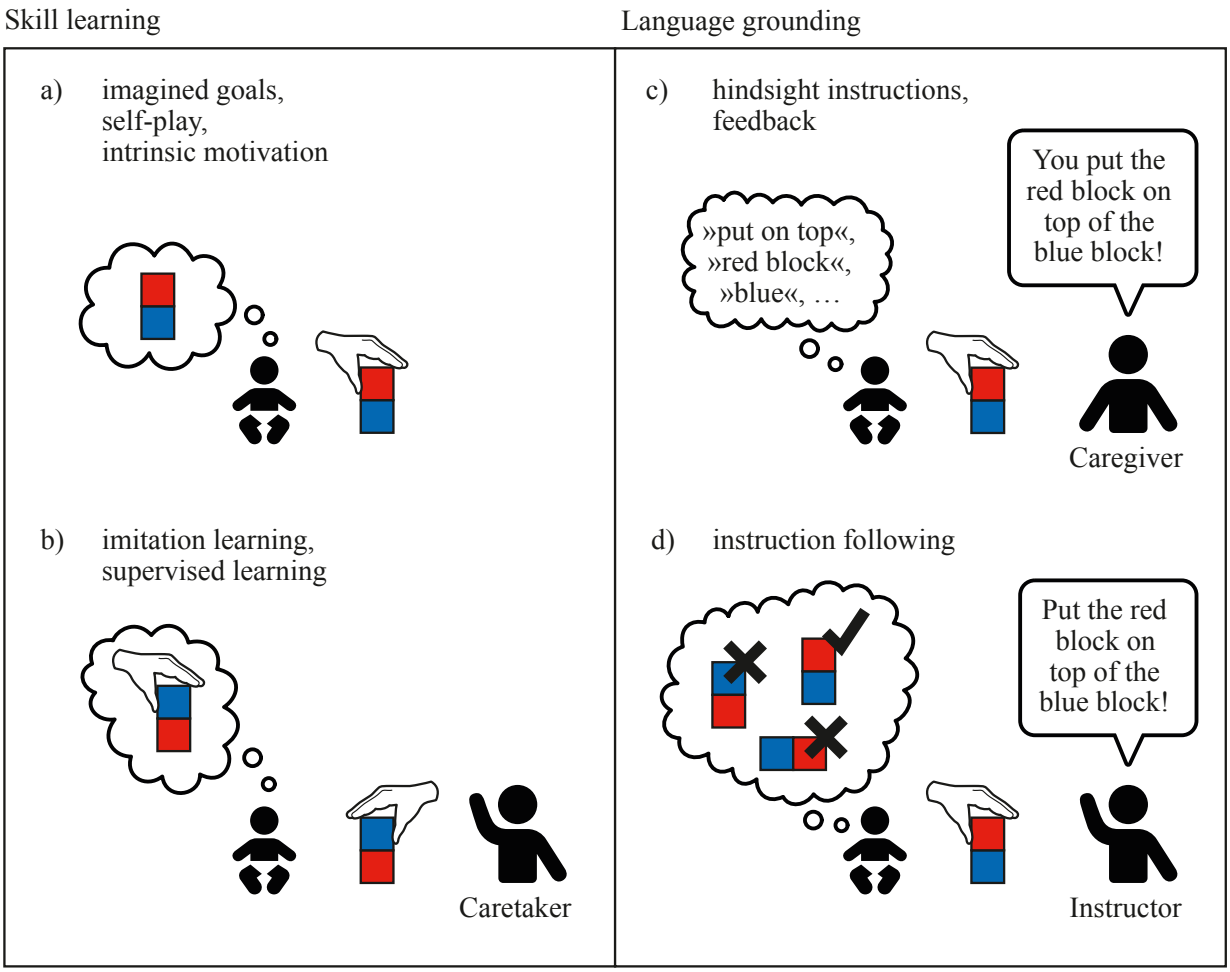


Figure 1.4: The figure illustrates learning approaches, contrasting the phases of skill acquisition (left) and language grounding (right). In the skill learning phase, image a) depicts learning driven by intrinsic motivation, characterized by play and self-generated goal-setting. Image b) showcases learning influenced by external factors, such as caregiver imitation (Paulus, 2014) and supervision. In the language grounding phase, image c) represents the agent learning from external feedback resulting in outcomes based on intrinsic factors. Image d) exemplifies instruction following, wherein the agent aims to follow an instructor's commands to obtain a reward. – Figure adapted from Röder et al. (2021).

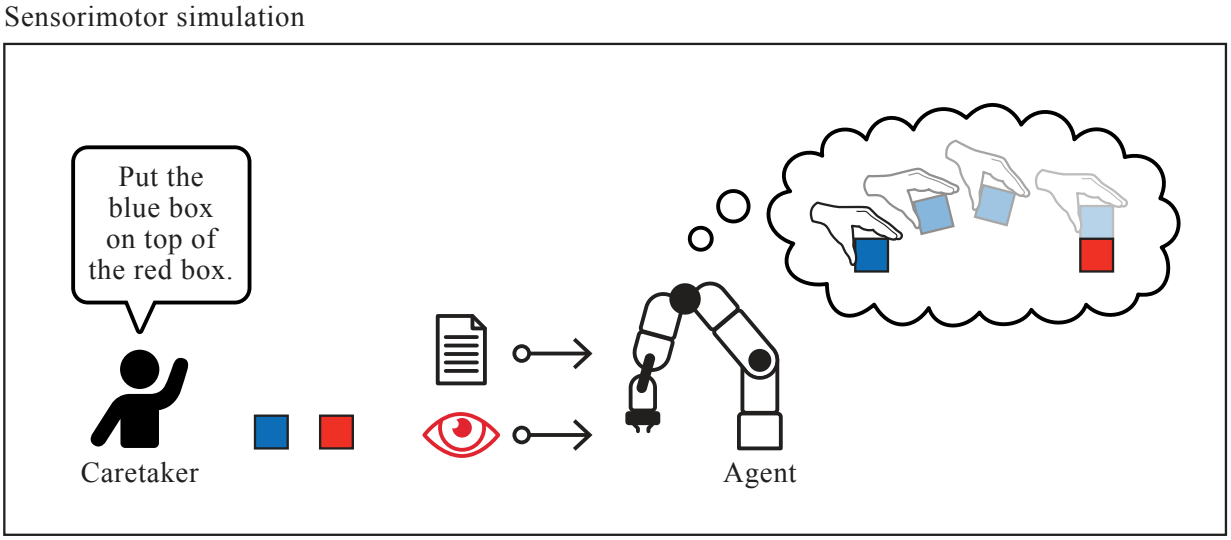


Figure 1.5: The figure illustrates the idea of mental simulation. Harnessing its world model, the agent mentally pictures itself executing the instruction uttered by the caretaker. However, predictions are limited in terms of their accuracy with respect to the number of steps, visualized by the increasing transparency of the blue cube and hand. – Figure adapted from Röder et al. (2021).

1.5 Research Objectives

Human-to-human communication is full of misunderstandings, such as ambiguities, *lapsus linguae*[3], imprecise statements, and a lack of common ground among the participants. Natural language conversations require constantly resolving such disturbances to ensure robust communication for conveying the intended information. This resolution process is crucial not only for verbal exchanges, but also for addressing the consequences previous utterances had on the world state. For instance, a waiter serving a wrong drink not only takes into account the feedback that rectifies the misunderstood order but also replaces the incorrectly placed item by removing it from the table.

In this thesis, we explore the interplay between perception, language, and action, adopting an embodied setting for communication, in contrast to prior text-only approaches, simulating it as a verbal/nonverbal dialog. For this, we employ the concept of language-conditioned reinforcement learning to ground language based on sparse rewards. We implement a synthetic caretaker that instructs the learner to solve tasks, but also offers hints for learning in hindsight. In cases of failure, the teacher provides feedback to resolve issues caused by misunderstandings. Central to our solutions are the concepts of expert feedback and self-speech inspired by the preceding sections on language development that are crucial to improving the sample efficiency of current RL methods.

Consequently, this thesis addresses the following **research questions (RQ)**:

RQ 1 To what degree can language grounding in robotics reinforcement learning tasks be exclusively modeled through sparse rewards?

RQ 2 How can insights from developmental science enhance language grounding and learning efficiency in intelligent robotic agents?

RQ 3 Is it feasible to develop a robust, embodied conversational human-robot interaction model that functions as a standardized benchmark for assessing and quantifying misunderstandings?

RQ 4 How can insights from cognitive science and developmental psychology be leveraged to improve a reinforcement learning approach for resolving misunderstandings?

[3]lapsus linguae is Latin and literally translates to *"slip or fault of tongue"*

1.6 Summary of source code

- The algorithms in Röder et al. (2022) are part of the repository at: `https://github.com/frankroeder/hipss.git`
- All the other algorithms can be found at: `https://github.com/frankroeder/language-conditioned-rl.git`
- The **LANRO** environment used for all of the experiments can be found at: `https://github.com/frankroeder/lanro-gym.git`
- The software suite **Scilab-RL** (Benad et al., 2025) is available at: `https://scilab-rl.github.io/Scilab-RL/`

1.7 Short Outline

The structure of this thesis is organized as follows: Chapter 2 provides the background, including the essential technical and mathematical preliminaries necessary to understand the topics of this dissertation. In Chapter 3, we provide additional information on natural language processing and introduce the basics of language in robotics reinforcement learning. Next, we present our novel benchmark for language grounding along with baseline results. In Chapter 4, we propose our first pillar of contributions, detailing an expert-driven and egocentric speech mechanism for learning from failures, that we demonstrate to be a sample efficient addition for language grounding. Chapter 5 expands on our benchmark by presenting the action correction setup for conversational misunderstandings and empirical results for our methods. This includes an approach incorporating uncertainty into the return estimation, as well as the application of methods developed in Chapter 4. Finally, Chapter 6 summarizes our contributions by answering the proposed research questions in a structured manner (see Section 1.5), highlighting limitations and suggesting directions for future research.

1.8 Main Results and Outline

This extended outline serves as a concise summary of the thesis, providing an overview of the research agenda and the key findings. Our methods along with the core contributions are summarized in Table 1.1.

In Chapter 2, we present the essential technical and mathematical foundations necessary to introduce the core concepts of deep learning (Goodfellow et al., 2016) and reinforcement learning (Sutton and Barto, 2018), as required for the deep reinforcement learning context of this thesis (Achiam, 2018). In addition, we define a special variant of the Markov decision process (Bellman, 1957) that incorporates goal representations (Kaelbling, 1993), specifically in the form of language instructions (Luketina et al., 2019). These language-based goals describe the tasks the robotic agent must complete to obtain rewards. Following this, we provide a comprehensive description of our primary reinforcement learning algorithm

Soft Actor-Critic (SAC) (Haarnoja et al., 2018), which is employed in various algorithmic adaptations throughout this thesis.

Language Grounding in Reinforcement Learning

In Chapter 3, we introduce the natural language processing (NLP) concepts necessary for our language-conditioned RL setting, a special case of goal-conditioned RL that has been extensively explored in numerous recent works (Andrychowicz et al., 2017; Plappert et al., 2018; Nair et al., 2018b; Eppe et al., 2019; Röder et al., 2020; McCallum et al., 2023). We then extend the previously discussed SAC algorithm to a language-conditioned variant, referred to as Language-Conditioned Soft Actor-Critic (LCSAC). This extension focuses on processing language inputs, represented as sequences of words $w_1, w_2, \ldots, w_L$, to obtain meaningful representations for the actor (policy) and critic (Q-function), both implemented using deep neural networks (Goodfellow et al., 2016). To achieve this, we adopt widely used techniques for encoding word sequences into vector representations, following prior work (Hermann et al., 2017; Hill et al., 2021). These include language encoders based on multilayer perceptrons (MLPs), gated recurrent units (Cho et al., 2014, GRU), and the self-attention mechanism (Bahdanau et al., 2015; Vaswani et al., 2017), as described in Subsection 3.2.3.

At the conclusion of Chapter 3, in Section 3.5, we present an extensive empirical study, comparing different approaches to language-based goal representations resulting from the aforementioned language encoders. The analysis in Section 3.5 highlights their relative strengths and weaknesses across diverse task settings.

Main outcome 1 As first outcome of Chapter 3, we present our language-conditioned robotics environment **LANRO** (Röder et al., 2022). It is a novel robotic learning environment that supports a variety of *reaching*, *pushing*, *grasping*, and *lifting* tasks for language grounding based on sparse rewards (see Section 3.4). Built as an extension of *panda-gym* (Gallouédec et al., 2021), it has been adopted by follow-up research such as Sejnova et al. (2024) and our own work Röder and Eppe (2022), as an indicator for being a useful testbed. In Figure 1.6, we provide an illustration of the environment and the simulated Franka Emika Panda robot (Haddadin et al., 2022). The example scene depicts three cubes on a table, arranged from left to right in *cyan*, *purple*, and *orange*. The agent might, for instance, be instructed to manipulate the orange cube, for which it would receive a reward.

LANRO provides language goals to guide the robotic agent, which is controlled through inverse kinematics derived from relative positional changes to the agent's end-effector. To generate instructions, we employ a semantic grammar description, that is inherent to the environment and influences the reward function that regards the properties of the goal object mentioned in the language goal. The grammar follows the Backus-Naur form shown in Figure 3.17 and adheres to the following sentence structure:

<verb><article><property 1><property 2>

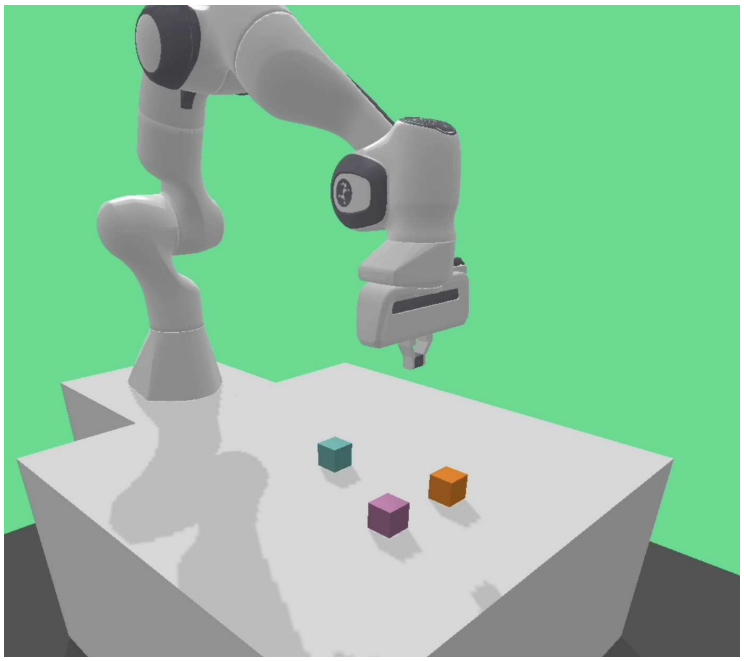


Figure 1.6: A visualization of our **LANRO** environment with the robot facing 3 different colored cubes.

To ensure natural readability, the property order reflects conventional language patterns (*e.g.*, *"red cube"* instead of *"cube red"*). To the best of our knowledge, **LANRO** provides the only setup that supports *tabula rasa* language grounding in a physically realistic robotic RL setting using sparse rewards (Röder et al., 2022; Röder and Eppe, 2022). While relying on a simplified language for instruction generation, similar to Chevalier-Boisvert et al. (2019), no additional assumptions are made about the underlying language representation (Akakzia et al., 2021). **LANRO** distinguishes itself as a unique environment within the robotic RL domain by offering both continuous state-based and pixel-based observations, continuous joint control, and design principles inspired by other language-grounding benchmarks (Hermann et al., 2017; Chevalier-Boisvert et al., 2019; Hill et al., 2021). Throughout the remainder of this work, we adopt the perspective of a synthetic caretaker (Colas et al., 2020; Akakzia et al., 2021), which provides both task instructions and feedback to the agent.

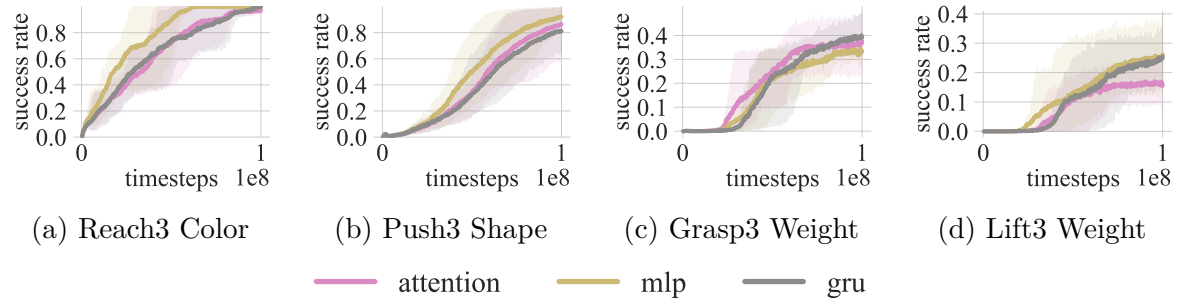
We propose a reinforcement learning benchmark for language grounding in continuous state and action spaces, designed around sparse rewards. To the best of our knowledge, ***LANRO*** *is the only benchmark that provides raw state inputs and language goals for end-to-end learning, without imposing additional assumptions on the representation of the input data.*

Main outcome 2 In Chapter 3, we present a second key outcome based on the Language-Conditioned Soft Actor-Critic (LCSAC)[4] algorithm and the **LANRO** environment. Specifically, we provide initial results demonstrating the agent's ability to solve the outlined tasks with objects of varying properties. This experimental setting offers an end-to-end framework for language-conditioned reinforcement learning, comparable to the work of Akakzia et al. (2021), but distinguishes itself by enabling the agent to ground language from the bottom up, without relying on prior knowledge or manual feature engineering (Colas et al., 2020; Akakzia et al., 2021). To this end, we analyze the impact of different language encoders on the task success rate, as illustrated in Figure 1.7. Each experimental setting includes three potential goal objects whose appearances are extended through variations in color, shape, or weight, depending on the task instructions. This design necessitates for certain properties, such as the weight and shape, the agent to interact with the objects to identify the target object, thereby emphasizing the role of embodiment in the agent's decision-making process. We evaluate the agent's performance via the success rate (ratio of successful episodes achieving the goal, y-axis) across a selection of tasks: *reach*, *push*, *grasp*, and *lift*. For each task, a training budget of 100 million ($1e8$) environment steps is allocated, reflecting the high sample complexity required for language grounding in sparse reward settings.

The results indicate that the encoders provide very similar results across all the settings. However, in certain cases, such as Figures 1.7a and 1.7b, the MLP outperforms the other encoders. Additionally, the self-attention approach demonstrates to be slightly faster in the early stages of the task in Figure 1.7c, while the GRU reaches the highest success value. For the *lift* task in Figure 1.7d, both the MLP and the GRU show equal performance, whereas the GRU is the method capable of processing the instruction with awareness of word order.

*We provide initial results for our language-conditioned RL benchmark **LANRO**, alongside an analysis of various instruction encoding techniques evaluated in different task settings. The results of our basis algorithm **LCDroQ** validates the environmental setup and confirms the sample complexity inherent to language grounding using sparse rewards.*

[4]In all the following cases, **LCSAC** is replaced by Language-Conditioned DroQ (LCDroQ), an extension introduced in the later Subsection 5.5.2



(a) Reach3 Color (b) Push3 Shape (c) Grasp3 Weight (d) Lift3 Weight

Figure 1.7: Main outcome of Chapter 3 with 4 hand-selected experiments. The figure compares the success rate (y-axis) to the number of environment steps (x-axis) for different types of language encoders, including the multilayer perceptron (mlp), gated-recurrent unit (gru) (Cho et al., 2014), and the scaled dot-product attention (Vaswani et al., 2017) (see Subsection 3.2.3 for more details). Enlisted are the plots for the *reach*, *push*, *grasp*, and *lift* tasks. Each of them involves three objects and varies in difficulty regarding their object attributes, such as the color, shape, and weight, and the requested behavior itself. We report the mean success rate along with the 90% confidence interval (shaded area), calculated across 3 random seeds.

Hindsight Language Grounding

Chapter 4 describes the implementation of the first pillar, which replicates the concept of language learning guided by caretaker feedback and egocentric speech. This approach draws inspiration from the mechanisms of human development discussed in Chapter 1. The expert instruction paradigm, introduced in Section 1.4, emphasizes the critical role of close interaction between a caretaker and a learner in facilitating language grounding (Akakzia et al., 2021). Within this framework, the caretaker provides feedback based on goal-oriented behavioral outcomes. Building on this idea, we propose an architecture that emulates the role of egocentric speech observed during the early stages of children's language development (Vygotskij, 1985; Mandler, 2004). Instead of relying on explicit expert-provided instructions, the agent engages in a self-directed learning process by *"talking to itself"*. This self-directed learning involves generating retrospective goal instructions independently for the purpose of hindsight learning. Specifically, the model assigns a goal instruction $\hat{g}_\ell$ to an experienced state trajectory τ_s, formulated as $p(\hat{g}_\ell \mid \tau_s)$. Unlike expert feedback, which is generally flawless, instructions derived from egocentric speech are initially imperfect, as the self-speech model is trained concurrently with the policy.

Much like how young children narrate their actions while their language skills are still developing, the agent's self-generated language may include syntactic errors. These imperfections are incorporated into the self-speech mechanism, introducing noise to the hindsight learning replay process. This induced epistemic uncertainty has been shown to enhance robustness, ultimately improving the agent's overall learning performance (Cideron et al., 2020; Röder et al., 2022).

Main outcome 3 In our work Röder et al. (2022), we propose Hindsight Expert Instruction Replay (HEIR), a method that leverages the tight interplay between a learner and a caretaker. Drawing inspiration from interactive RL (Cruz et al., 2015; Nguyen et al., 2021), we task the learner with solving an objective defined by a linguistic goal while receiving optional verbal feedback. If the learner fails to follow a given instruction, the caretaker provides a suitable hindsight instruction, enabling the agent to retrospectively learn from the outcome (Jiang et al., 2019; Colas et al., 2020). The fundamental principle underlying this approach is to reinterpret an observed outcome by associating it with an alternative goal distinct from the original one. As illustrated in Figure 1.8, this approach demonstrates how **expert feedback** can be effectively employed in our RL setting as the agent is autonomously collecting experiences (Akakzia et al., 2021).

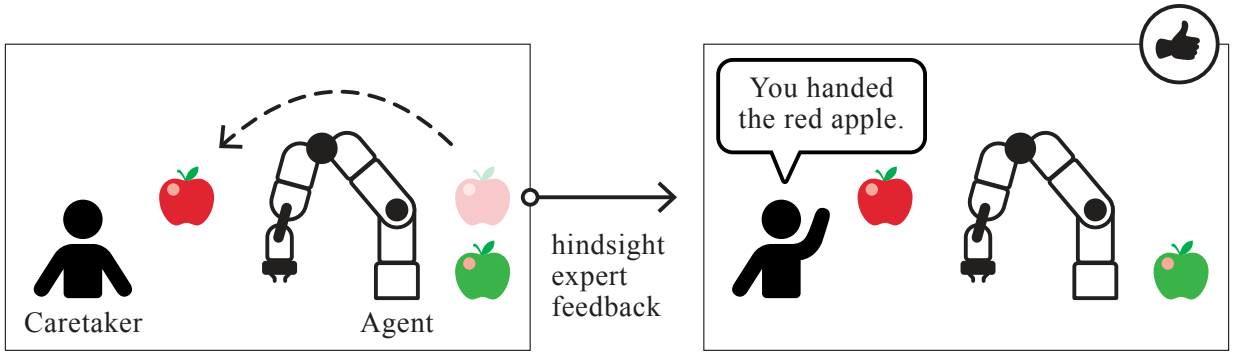
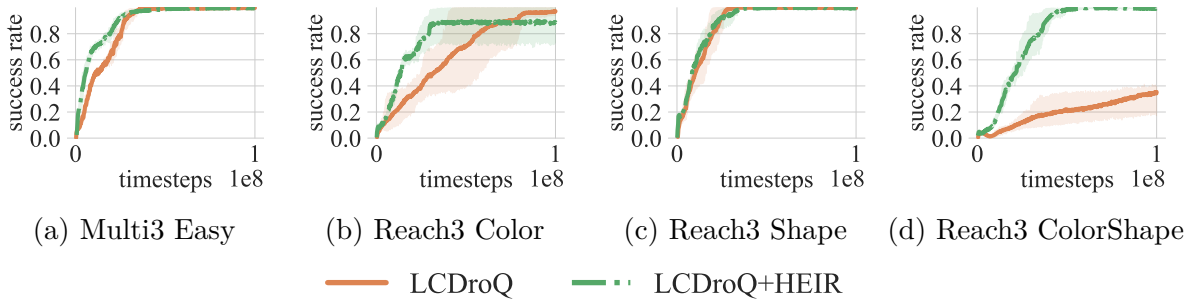


Figure 1.8: We illustrate the procedure of receiving the expert feedback, upon triggering an outcome to which a hindsight language goal such as *"Hand the red apple"* matches.

In the goal-conditioned off-policy setting, analogous to Hindsight Experience Replay (HER) (Andrychowicz et al., 2017), episodes are relabeled using expert feedback, substituting the original goal instructions that led to an unsuccessful outcome with a suitable hindsight instruction that has been achieved. A limitation of this method, presented in Röder et al. (2022), is the impact of hindsight bias (Bai et al., 2021), which arises from a **mismatch** in the likelihood of trajectories generated under the original goal g_ℓ and those conditioned on the hindsight goal g'_ℓ, expressed as $p(\tau \mid g_\ell) \neq p(\tau \mid g'_\ell)$ when $\tau \sim \pi$. In essence, Bai et al. (2021) highlighted the erroneous assumption that the hindsight goal would result in similar behavior, which can adversely affect learning. This issue is particularly significant for suboptimal policies, as the trajectory distribution for certain goals can vary considerably. We addressed this issue of our previous publication (Röder et al., 2022) by restricting the number of hindsight samples during experience replay to mitigate the effects of hindsight bias, recognizing that a poorly trained policy behaves almost randomly in the early stages of training and rarely follows the proposed goal instructions. The updated results, presented in this thesis and depicted in Figure 1.9, include a range of tasks

with variations in object properties, highlighting the superior performance of **LCDroQ** when combined with the **HEIR** replay strategy, which significantly surpasses the base version **LCDroQ**. Furthermore, we provide results from a *multitask* setting that combines the *reach* and the *push* tasks, as well as results for the *reach* task with variations in object colors and shapes. In the scenario shown in Figure 1.9d, **LCDroQ+HEIR** achieves a success rate three times that of **LCDroQ**, effectively addressing object confusion in cases with the most complex color-shape combinations, where distinguishing object properties presents the greatest challenge.



(a) Multi3 Easy (b) Reach3 Color (c) Reach3 Shape (d) Reach3 ColorShape

Figure 1.9: We depict a selection of experiments showing the advantage of incorporating the **HEIR** approach into the base algorithm **LCDroQ**. As shown in Figure 1.9d, **LCDroQ+HEIR** outperforms **LCDroQ** by a factor of 3. The plots show the mean success rate along with the 90% confidence interval as a shaded region on the y-axis, as well as the episode timesteps along the x-axis. Each line represents the results averaged over 5 random seeds.

*Our expert feedback approach, **HEIR**, provides an optimal setting for hindsight instruction replay. By addressing hindsight bias through a simple replay limit, **HEIR** improves sample efficiency compared to our basis **LCDroQ**. This is emphasized by the success rate improvement of up to a factor of 3.*

Main outcome 4 A significant portion of human language learning occurs in an unsupervised or self-supervised manner (Vygotsky, 1967). Once toddlers begin communicating verbally, they are often observed narrating their own play, commenting on what they are currently doing and what they are about to do. This phenomenon is referred to as self-narrating or egocentric speech (Piaget, 1926; Vygotskij, 1985). Building on this concept, we propose Hindsight Instruction Prediction from State Sequences (HIPSS). We implement the mechanism of egocentric speech using a well-established sequence-to-sequence model approach (Sutskever et al., 2014; Cho et al., 2014), in combination with **HIPSS**, as well as the Transformer encoder-decoder architecture (Vaswani et al., 2017), resulting in our implementation Transformer-based Hindsight Instruction Prediction from State Sequences (THIPSS). Both models process a trajectory of states $\tau_s = s_1, s_2, \dots s_T$, to generate corresponding instructions describing them. This is accomplished by encoding the states into an intermediate representation known as the **context**, which captures an abstract understanding of the agent's goal-directed behavior and the changes in the world. This context is subsequently passed to a decoder that generates the hindsight goal $\hat{g}_\ell$. The overall procedure is illustrated in Figure 1.10.

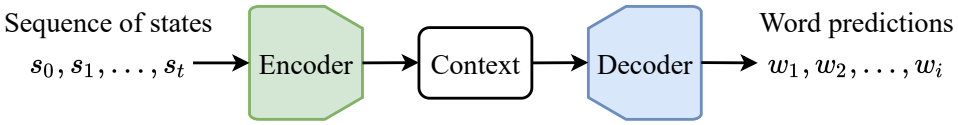


Figure 1.10: This figure illustrates the concept of a sequence-to-sequence model (Sutskever et al., 2014), applied in our setting as a replay mechanism that translates a trajectory into a corresponding language goal, thereby implementing egocentric speech.

To train this mapping function, we utilize successful episodes generated by the learning policy, storing the state trajectories and their corresponding instructions in a dataset for supervised learning. Since we only include rewarding behavior-language pairs autonomously harvested by the agent without the assistance of the caretaker, this approach can be categorized as either unsupervised or self-supervised learning. Our method is inspired by the work on Hindsight Generation for Experience Replay (HIGhER) (Cideron et al., 2020). However, **HIPSS** represents the first model of this kind to function within continuous action and state spaces, taking sequences as inputs to capture complex temporal behaviors. Additionally, it is an open-source implementation published in our work Röder et al. (2022), addressing the limitations identified in Cideron et al. (2020). Similar to human development, and unlike the expert feedback approach employed by **HEIR**, this framework does not require perfect language generation right from the start. In language development, egocentric speech often includes made-up words or reflects an immature vocabulary. In our implementation, incorrect predictions are also present and, as a side effect, enhance learning performance and robustness by introducing noise into the training process. Unlike traditional language modeling (Bengio et al., 2003; Brown et al., 2020), **HIPSS** provides a grounded approach to language generation based on the agent's sensory inputs. Following Mandler's (2004) notion of *"thought before language"*, the agent first learns to perform goal-directed behaviors and subsequently grounds language in hindsight. **HIPSS** might

develop something similar to semantic compositionality (Feldman, 2010) by translating physical behavior patterns into meaningful language descriptions, surpassing the syntactic compositionality of unimodal text models that lack grounding. In that regard, we highlight results on the systematic compositionality of **HIPSS** and **THIPSS** in an additional set of supervised experiments shown in Subsection 4.9.5, where the models successfully assign language instructions to novel situations not encountered during training.
For the RL part, the results in Figure 1.11 compare our egocentric speech approaches to **HEIR** and **LCDroQ**. We highlight the methods across a selection of tasks with different colors and shapes (see Subsection 3.4.1 for additional details). In the *reach* task with an extended range of colors and shapes (see Figure 1.11a), the basis algorithm **LCDroQ** performs the worst (success rate of roughly 0.3), while **LCDroQ** with the hindsight replay extension **THIPSS** achieves a performance that is twice as good. Both **LCDroQ+HIPSS** and **LCDroQ+HEIR** achieve even greater success rates of roughly 0.9 and 1.0, in the latter case outperforming **LCDroQ** by a factor of 3. **LCDroQ+HIPSS** also surpasses the other approaches in the *push* experiments, particularly in the setting with additional colors (see Figure 1.11b), where its performance doubles that of **LCDroQ** and outperforms **LCDroQ+HEIR** by up to 0.4. The combination **LCDroQ+THIPSS** shows very similar results. In Figure 1.11c, **LCDroQ+HIPSS** and **LCDroQ+THIPSS** achieve similar results, while **LCDroQ+HEIR** lags behind. The final experimental setting in Figure 1.11d, similar to the first, contains an extended range of property words and further highlights the advantage of our egocentric speech approaches, as the agent occasionally confuses the goal objects. The differences between **LCDroQ+HEIR** in the most difficult *reach* and the *push* settings (see Figures 1.11a and 1.11d, with additional colors and shapes) can be attributed to the different types of task behaviors involved.

Table 1.1 summarizes the core ideas of the presented methods and their contribution to this thesis. We list the individual algorithm combinations paired with their respective replay approaches.

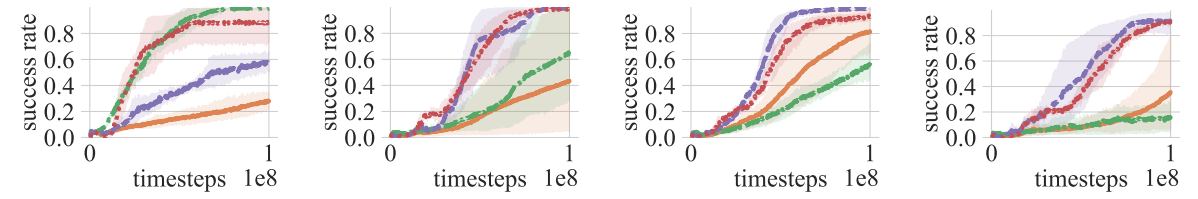


Figure 1.11: Illustrated is a comparison of the former **HEIR** replay mechanism with the novel egocentric speech approaches **HIPSS** and **THIPSS**. For a selection of tasks, along with variations in color and shape, we depict the average success rate for 5 random seeds, with the 90% confidence interval shown as shaded region on the y-axis and the number of environment steps on the x-axis. Comparing Figures 1.11a and 1.11d reveals significant differences with respect to the performance of the expert approach **HEIR**, possibly due to the different *reaching* and *pushing* behaviors involved.

Method	Contribution	Description
Language-Conditioned Soft Actor-Critic (LCSAC)	Our initial RL algorithm as a language-conditioned extension of the Soft Actor-Critic algorithm employed in (Röder et al., 2022).	**LCSAC** serves as our initial baseline implementation following established related work (Jiang et al., 2019; Akakzia et al., 2021).
Language-Conditioned DroQ (LCDroQ)	Our novel language-conditioned RL algorithm incorporating dropout Q-functions (Hiraoka et al., 2022) into **LCSAC**, detailed in Subsection 5.5.2.	**LCDroQ** serves as a strong basis algorithm that injects uncertainty into the return estimation without any replay extensions.
Hindsight Expert Instruction Replay (HEIR)	A hindsight expert replay extension presented in Section 4.4 and Röder et al. (2022).	It forms an integral component of our enhanced algorithm combination with the basis algorithm **LCDroQ+HEIR**.
Hindsight Instruction Prediction from State Sequences (HIPSS)	An egocentric speech replay mechanism utilizing recurrent neural networks in a sequence-to-sequence architecture, introduced in Subsection 4.7.1.	It is implemented as a self-supervised hindsight replay mechanism, integrated into **LCDroQ+HIPSS**.
Transformer-based Hindsight Instruction Prediction from State Sequences (THIPSS)	An advanced egocentric speech approach leveraging the Transformer encoder-decoder architecture, detailed in Subsection 4.7.2.	It enables self-supervised hindsight instruction replay in our Transformer-based implementation **LCDroQ+THIPSS**.

Table 1.1: Overview of the proposed methods and their contributions.

Our self-supervised egocentric speech approach improves the sample efficiency by up to a factor of 3. *Harvesting successful episodes for training **HIPSS** and **THIPSS** allows the model to exploit similarities through systematic generalization. This process enhances policy learning by generating hindsight instructions for failed episodes as well as novel situations.*

Misunderstandings and Action Corrections

Chapter 5 addresses the challenge of conversational failures, which are often caused by communication flaws such as underspecification, disfluency, and polysemy. Existing studies in this domain tend to focus on cases of non-understanding (Hirst et al., 1994), often neglect the more challenging issue of misunderstandings (Bohus and Rudnicky, 2008). In this thesis, we categorize misunderstandings into three distinct types (Röder and Eppe, 2022), which encompass a significant portion of the challenges encountered in human-robot dialogs:

- Ambiguity: The instructor utters an underspecified instruction that cannot be resolved without further contextual input (Achimova et al., 2022)
- Lack of common ground: Words that were not part of the agent's training introduce significant epistemic uncertainty in the model's predictions (Chai et al., 2014)
- Instruction Correction: The instructor unintentionally specifies an incorrect intent by using wrong words, a phenomenon commonly referred to as lapsus linguae (*"slip of the tongue"*)

Participants in a conversation must continually establish common ground by addressing the aforementioned issues through verbal or physical feedback. These same challenges apply to human interactions with intelligent artificial agents or robots (Thierauf et al., 2023). In robotics, resolving misunderstandings is especially critical to avoid potentially harmful outcomes. Figure 1.12 illustrates an example of an ambiguous instruction in which the instructor commands the agent to hand the apple without providing further clarification, despite the presence of two apples. Assume the instructor intended to refer to the green apple. Upon identifying the mismatch between the actual outcome and the intended one, the instructor utters the action correction, *"No, the green one!"*, prompting the agent to adjust its behavior accordingly.

In Chapter 5, we propose a procedure to model misunderstandings by extending the language-conditioned framework of **LANRO** to address challenges that require the repair of conversational breakdowns (Ashktorab et al., 2019) (see Section 5.4). Compared to simple instruction-following, the action correction setting introduces a greater challenge, as the agent must resolve two goals in succession. Initially, the original goal g_ℓ is presented, which possibly leads to one of the three types of misunderstandings. When the operator realizes the agent has misunderstood the instruction, the original goal is complemented by the action correction, resulting in $g_\ell \circ g_{\text{ac}}$. The action correction g_{ac} is programmatically determined by **LANRO**, taking into account the initial instruction, the present goal objects, and what has happened so far within the state space. We incorporate editing terms such as *"actually"* and *"sorry"* to modify the initial instruction, as well as negations beginning with *"not"* to contrast the actual outcome with the desired one. Building on our prior work in Röder and Eppe (2022), this section contributes in two key ways. First, it stands to reason that we reuse hindsight methods. However, to achieve this, we need to improve the base algorithm's performance and ensure the successful generation of episodes that include action-corrected goals. Second, predicting hindsight instructions with action corrections via self-speech requires adapting the generative process. These adjustments must

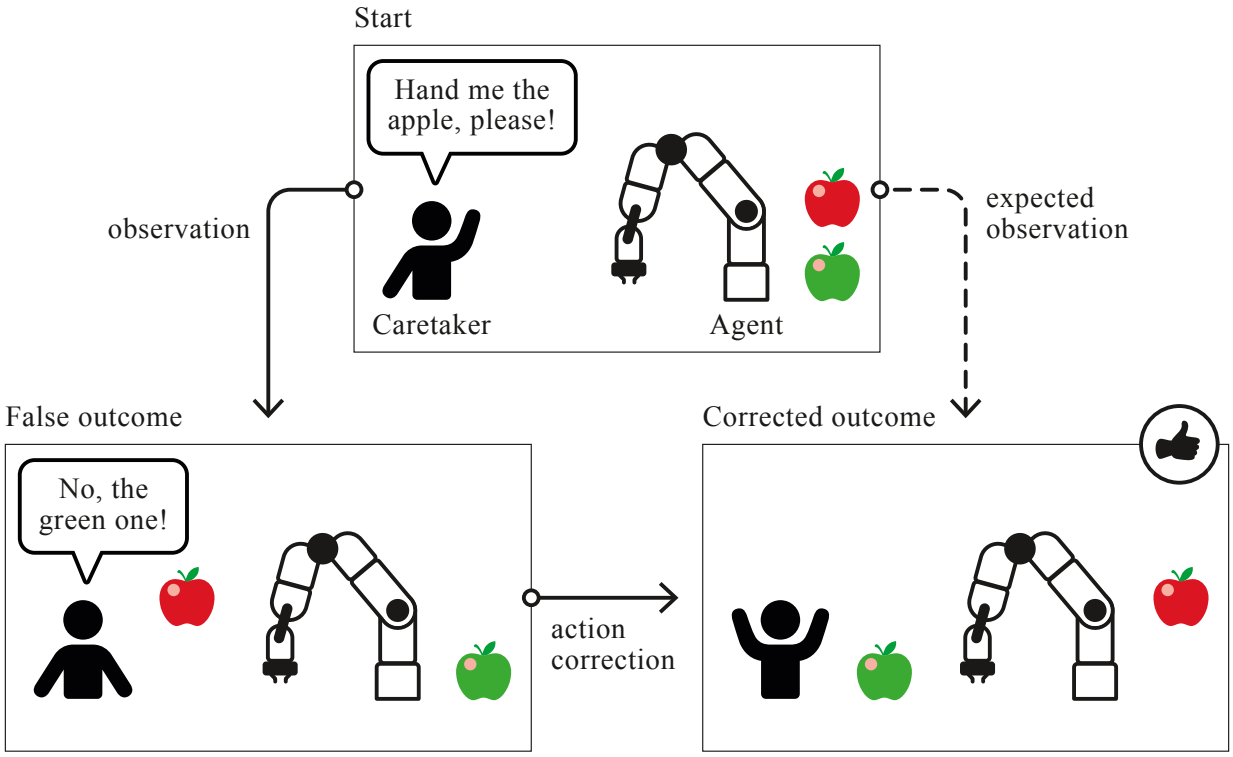


Figure 1.12: We depict a scene with the ambiguous instruction *"Hand me the apple, please!"*. Upon observing the robotic agent handing the red apple (bottom left), the instructor utters the action correction, *"No, the green one!"*. The agent then resolves the misunderstanding by taking back the red apple and handing the green one.

determine whether the agent's behavior results from a simple instruction or an extended instruction that incorporates an action correction.

Main outcome 5 As the first main outcome of Chapter 5, we introduce Language-Conditioned DroQ (LCDroQ) as an extension of **LCSAC** (Röder et al., 2022). **LCDroQ** modifies the critic network to model a distribution using an ensemble of dropout Q-functions (Hiraoka et al., 2022). Each ensemble member incorporates multiple dropout layers (Srivastava et al., 2014), which remain activated during both training and evaluation, thereby representing a proposal distribution and approximating Bayesian inference (Gal and Ghahramani, 2016). The uncertainty in the Q-function helps to address the inherent uncertainty of action correction tasks by approximating $\mathbb{E}_{\bar{\phi}_1,\dots,\bar{\phi}_N}\left[\min_{i=1,\dots,N} Q_{\bar{\phi}_i}(s', a')\right]$ (Hiraoka et al., 2022). Modeling the problem in this way enables the agent to represent multiple possible future modes in its return estimations. Figure 1.13 illustrates the network architecture of a single critic alongside the process of taking the minimum of the target

values for training, which follows the clipped double-Q trick (Fujimoto et al., 2018).

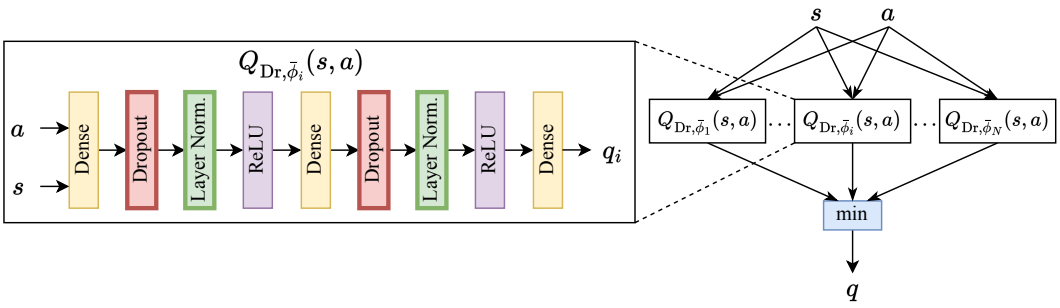
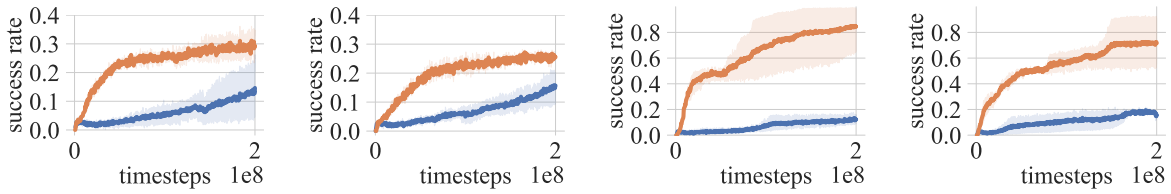


Figure 1.13: The left diagram shows the architecture of a single dropout Q-function with alternating dense, dropout, and normalization layers, paired with ReLU activations. The right illustration depicts the clipped double-Q trick used to calculate the target value for temporal difference learning. Redrawn figure based on Hiraoka et al. (2022).

In Figure 1.14, we highlight the advantages of **LCDroQ** over **LCSAC** across a selection of several action correction tasks. We compare their performance in our *multitask* setting and in the *push* task involving three objects. Specifically, Figures 1.14a and 1.14c present results for the default settings, while Figures 1.14b and 1.14d focus on scenarios including negations. Our findings indicate that **LCDroQ** consistently outperforms **LCSAC** across all task variations by a factor of up to 8 (see Figure 1.14c). This indicates that **LCSAC** struggles to resolve the misunderstandings at all and is further hindered in following the standard instructions. With success rate values falling below 0.5, this behavior aligns with the problem setup, where misunderstandings are deliberately introduced in 50% of the episodes and standard instruction-following occurs in the remaining cases.



(a) Multi3 (b) Multi3 Negations (c) Push3 (d) Push3 Negations

Figure 1.14: The performance of **LCSAC** and the dropout Q-function extension **LCDroQ** is compared in a small selection of *multitask* and *push* action correction scenarios. **LCDroQ** consistently outperforms **LCSAC**, as shown by its higher success rate, which accounts for both the standard instruction-following and the resolution of misunderstandings. The y-axis represents the average success rate with a 90% confidence interval (shaded region), plotted against the environment steps on the x-axis. The results are averaged across 5 random seeds. **LCDroQ** shows performance improvements of up to a factor of 8 (Figure 1.14c).

Incorporating uncertainty into the Q-value estimation of the Q-function, and consequently into the critic of our actor-critic framework, enhances the sample efficiency by accounting for uncertainty in environment rewards of the action correction tasks. This improvement arises from the agent's ability to anticipate potential misunderstandings, thereby preparing for the occurrence of action corrections.

Main outcome 6 As a second contribution to our work on misunderstandings in Chapter 5, we extend the hindsight methods from our previous study on **HIPSS** (Röder et al., 2022), as presented in Chapter 4, to address the more challenging action correction tasks. This is accomplished by implementing an explicit branching mechanism for our hindsight language goal prediction. The generative process of **HIPSS** and **THIPSS** is enhanced by introducing an additional set of tokens/words to support both default instruction prediction and action correction prediction, as illustrated in Figure 1.15. This enables the model to autonomously decide whether to terminate the language generation with an end-of-sequence (`<eos>`) symbol or to continue with the action correction (see Section 5.6 for further details).

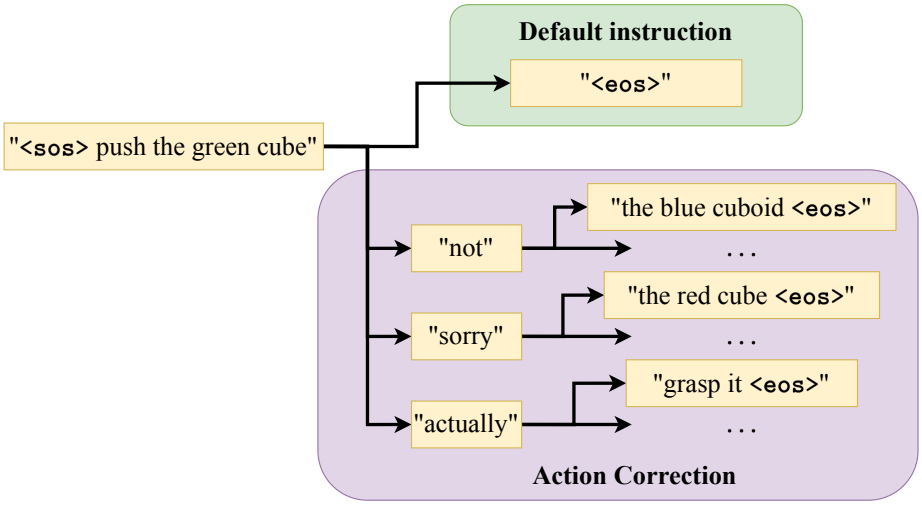


Figure 1.15: We illustrate the branching mechanism of our egocentric speech model, which predicts either the standard instructions from Chapter 4 (green box) or the extended instructions incorporating action corrections from Chapter 5 (purple box).

Referring to the introduction of **HIPSS**, we refine the process of language generation by conditioning on the actual instruction g_ℓ as a prior, allowing the model to generate only the action correction $\hat{g}_{\text{ac}}$ according to $p(\hat{g}_{\text{ac}} \mid \tau_s, g_\ell)$. This process is depicted in Figure 1.16, where the initial instruction *"push the red cube"* (red box) is complemented by the prediction of the action correction *"no, the blue cube"* (blue box), based on the state trajectory τ_s.

In Figure 1.17, we present a selection of tasks to compare **LCDroQ** with the uncertainty extension, our hindsight expert feedback method **HEIR**, and both egocentric speech

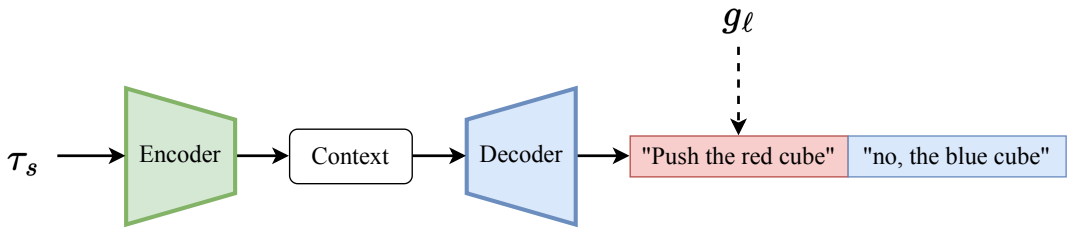


Figure 1.16: We illustrate the sequence-to-sequence approach, which now *teacher forces* the initial instruction g_ℓ = *"push the red cube"* upon the generation of the action correction *"no, the blue cube"*, given the state trajectory τ_s.

approaches, **HIPSS** and **THIPSS**. Although **LCDroQ** demonstrates notable learning progress, it often plateaus or exhibits slow improvements. For example, in Figures 1.17a and 1.17d, it surpasses the critical success rate of 0.5; however, in the latter case, it becomes stuck at a value of 0.7. We further highlight the combination of **LCDroQ+HEIR** as a replay-based approach with perfect hindsight instructions. This method is restricted to default instructions because the caretaker cannot provide action corrections in hindsight. In Figures 1.17a and 1.17d, **LCDroQ+HEIR** plateaus at a success rate of 0.5, successfully addressing the goal instructions but failing to solve the action corrections. These findings validate that improving default instruction-following alone does not accelerate overall learning. While it might be assumed that this curriculum would simplify resolving action corrections subsequently, our results demonstrate otherwise. Both self-narrating approaches, **HIPSS** and **THIPSS**, achieve significant improvements in sample efficiency by learning to predict action corrections in a self-supervised manner, surpassing the performance of the expert feedback method **LCDroQ+HEIR**. Figure 1.17d highlights that both **LCDroQ+HIPSS** and **LCDroQ+THIPSS** achieve success rates approaching 1.0, outperforming the other methods. In the *push* scenario (Figure 1.17b) and the *multitask* setting (Figure 1.17c), **LCDroQ+HIPSS** and **LCDroQ+THIPSS** improve performance by a factor of up to 3. In all settings, our egocentric speech approaches never perform worse than **LCDroQ**; instead, they generally achieve superior results. **LCDroQ+HEIR**, by contrast, exhibits limited performance, as it focuses exclusively on default instruction-following through extended replay, neglecting action corrections entirely.

*Our egocentric speech approaches, **HIPSS** and **THIPSS**, improve sample efficiency for tasks involving action corrections by a factor of up to 3. Notably, self-narration even surpasses the performance of the expert feedback approach **HEIR**, highlighting the advantages of learning to predict linguistic goals and corrections through a grounded, self-supervised process of the agent "talking to itself".*

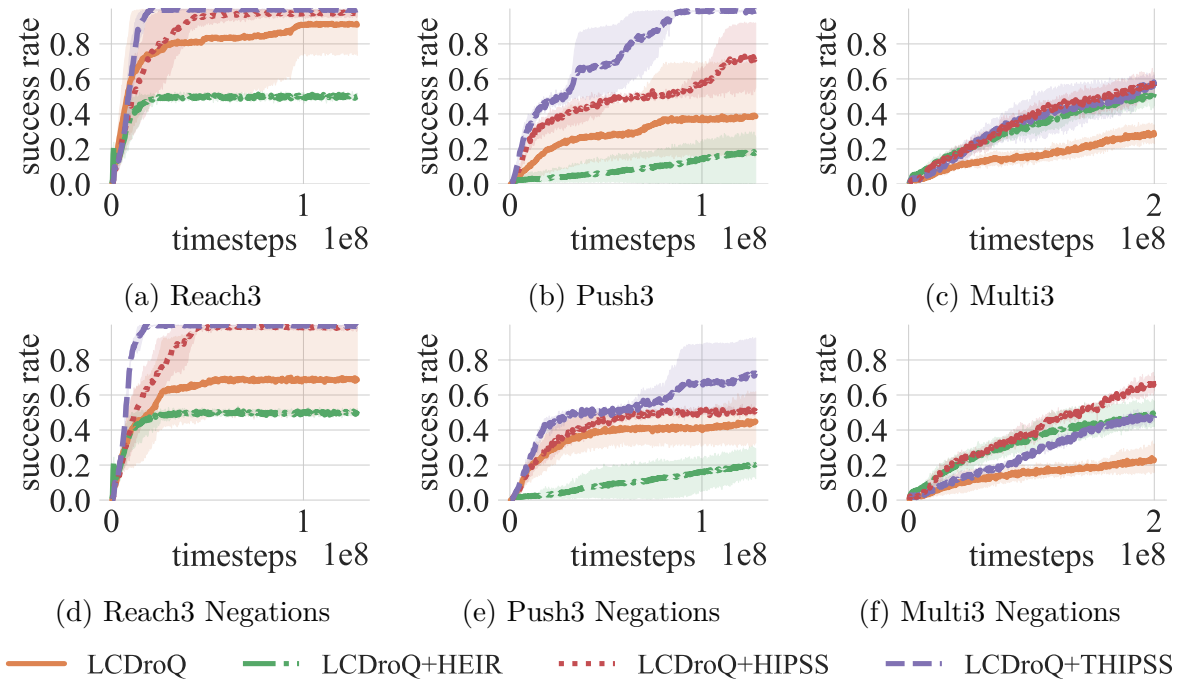


Figure 1.17: We present results for the basis **LCDroQ**, the expert feedback method **HEIR**, and the egocentric speech approaches **HIPSS** and **THIPSS** across various action correction scenarios. These scenarios include *reach*, *push*, and *multitask* tasks, as well as tasks involving negations. The average success rate, computed over 5 random seeds, is displayed on the y-axis, with environment steps plotted on the x-axis. At least one of the egocentric speech approaches consistently leads in performance, demonstrating the benefits of learning to predict action corrections in hindsight. By contrast, the hindsight expert feedback approach **HEIR** focuses exclusively on default instructions and neglects corrective feedback replay, as evident in Figures 1.17a and 1.17d, where it fails to exceed a success rate of 0.5. This result aligns with our configured ratio of episodes containing misunderstandings versus those without (cf. Section 5.4).

Chapter 2

Preliminaries and Background

This chapter presents an overview of the foundational theories and background material relevant to this thesis.

The field of computer science has undergone a major paradigm shift in the past decade due to the vast usage of machine learning (ML). ML, a subfield of artificial intelligence, enables machines to learn from data. Recent advancements in artificial neural networks have demonstrated superhuman performance in certain domains (Mnih et al., 2015), leading to their widespread adoption across many scientific disciplines (Luo et al., 2022). Deep learning (DL), a specialized branch of ML particularly focusing on deep neural networks, has emerged as a valuable area of study. Deep neural networks have proven especially effective in tasks like image recognition (Krizhevsky et al., 2012), natural language processing (Devlin et al., 2019), and speech recognition, enabling breakthroughs in many domains.

The following sections provide the essential background material for this thesis, though not exhaustively covering all aspects of machine learning, deep learning, reinforcement learning (RL), and natural language processing (NLP). However, we provide pointers to appropriate sources and highlight a selection of books that were used throughout the years in studying the topics of this thesis. In the case of reinforcement learning, the book by Sutton and Barto (2018) provides a comprehensive overview and is fundamental to our research, which primarily focuses on deep reinforcement learning (Achiam, 2018), a combination of DL and RL. For in-depth background knowledge on deep learning and natural language processing, we recommend the books by Goodfellow et al. (2016) and Murphy (2022). For a broader understanding of the mathematical foundations of machine learning, we refer to Deisenroth et al. (2020).

2.1 Deep Learning

The upcoming section outlines the fundamental ideas of deep neural networks employed in deep learning, how they are defined, and the way they are optimized to solve various tasks.

Deep learning is a branch of machine learning that exclusively makes use of deep artificial neural network (ANN). Neural networks possess universal approximation capabilities, as demonstrated by Cybenko (1989), and well-defined algorithms exist to adjust their parameters (Amari, 1967).

In the case of supervised learning, our objective is to approximate a function f^* that maps an input x to an output y, expressed as $f^*(x) = y$. The output can be either a continuous value ($y \in \mathbb{R}$), as in regression (see Subsection 2.1.2), or a categorical value ($y \in \{0, 1, 2, \ldots\}$), as in classification (see Subsection 2.1.1). A typical classification problem would involve a model that assigns labels to grayscale images of animals, where $x \in \mathbb{R}^{\text{width} \times \text{height} \times \text{channel}}$ represents the input image, and *cat* and *dog* denote the corresponding animal class labels. The primary objective is to train an ANN to approximate the true function, $f^* \approx f_\theta$, by adjusting the network parameters θ to accurately replicate the underlying input-to-output mapping. As a foundational example of a neural network, we begin with the definition of a linear function parameterized by $\theta = (W, b)$ as shown in Equation 2.1.

$$f(x; \theta) = Wx + b \tag{2.1}$$

The class of linear functions is a well-studied subject in machine learning research, but it has inherent limitations in terms of expressive capacity. To model functions of higher complexity, one approach involves combining multiple functions, each serving different purposes. This concept forms the foundation of deep neural networks, where a series of functions are combined to implement various types of transformations, $\phi : \mathbb{R}^n \to \mathbb{R}^m$, implicitly with respect to their position in the chain.

Consider our prior example: an ANN employs a series of transformations to get from a raw image of a dog to an increasingly abstract representation. In image processing, this begins with the identification of edges, followed by a shape detection and ends with higher level concepts of an animal like the shape of the ears or the number of legs. This process ultimately provides an invariant basis for animal identification.

Before deep learning became feasible to use, which was enabled by the collection of large datasets and the availability of powerful hardware, fields such as computer vision, for instance, relied on manually engineered functions for tasks like edge detection or texture analysis, typically based on manual feature engineering. These traditional methods often proved cumbersome to design and required expert domain knowledge to be implemented effectively. The advent of convolutional neural networks (LeCun et al., 1989), a type of ANN specifically designed for image processing, widely replaced prior approaches by learning the feature extraction directly from data. Figure 2.1 illustrates an example of such levels of abstraction learned by the *Inception* architecture (Molnar, 2022, Section 10.1).

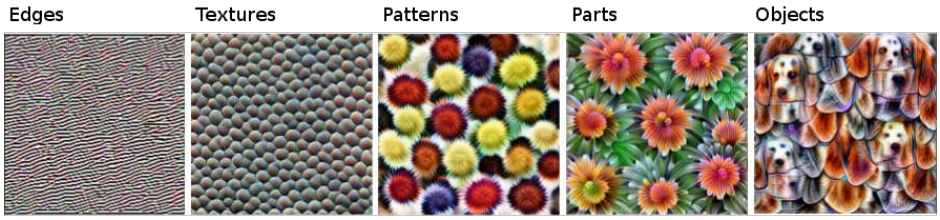


Figure 2.1: Image taken from Molnar (2022, Section 10.1) based on Olah et al. (2017), showing the increasing levels of abstraction (left to right) learned by the *Inception* architecture.

Formally, we denote an intermediate representation as $\phi(x)$, which serves as input to our function, applying a transformation to the raw input x. To utilize such an abstraction, we must combine at least two functions, $f_1 \to f_2$, where $\phi(x) = f_1(x, \theta_1) = W_1 x + b_1$ and $f_2(\phi(x); \theta_2) = W_2\phi(x) + b_2$. Deep learning provides an approach to find the function f_1 via feature learning that derives a suitable intermediate representation to solve a given task solely from data (Goodfellow et al., 2016). Typically, more complex problems require multiple layers of abstraction, and chaining additional functions leads to more capable models. These networks are referred to as multilayer perceptron (MLP), as they contain multiple layers and essentially build upon the idea of the perceptron by Rosenblatt (1958). One could modulate the complexity of the model in two ways: by adjusting the depth of the network (the number of layers) or the individual layer width (the number of parameters within each layer). This controls the degree of abstraction and, therefore, the capacity of the model, which represents the number of functions the ANN could express with parameters $\theta \in \Theta$, where Θ represents the parameter space. Current ML applications that are widely known are predominantly based on deep neural networks (OpenAI et al., 2020; Ramesh et al., 2021; OpenAI, 2023).

A deep neural network with multiple layers can be formally represented by Equation 2.2, where L denotes the total number of layers and the parameters $\theta = (\theta_1, \theta_2, \dots \theta_L)$. Each layer's parameters $\theta_i = (W_i, b_i)$ consist of weights $W_i \in \mathbb{R}^{N_i \times M_i}$ and biases $b_i \in \mathbb{R}^{M_i}$, where N_i is the input dimension and M_i is the output dimension of layer i.

$$f(x; \theta) = f_L(f_{L-1}(\dots f_1(x; \theta_1) \dots; \theta_{L-1}); \theta_L) \tag{2.2}$$

Several constraints apply to the network architecture: For the first layer, the number of input neurons must match the dimensionality of the input vector $|N_0| = |x|$. Consecutive layers must have compatible dimensions such that their output and input, $|M_i| = |N_{i+1}|$, follow simple rules of matrix multiplication (Deisenroth et al., 2020, Section 2). The final layer's output dimension depends on the task: for regression, we usually need a single scalar, $|M_L| = 1$, while for classification, the output dimension equals the number of classes C to which the network assigns confidence scores, $|M_L| = |C|$.

The initial layer f_1 is referred to as the input layer, while the final layer f_L is called the output layer. In the case where $L \geq 3$, the intermediate layers are called hidden layers. The dimensionality of the vectors of the hidden layers, specifically their outputs, is called the

hidden state. This is determined by the output dimension M_1 of the preceding layer, which in this case is the first layer ($L = 1$). Typically, it is desirable to have a hidden size smaller than the input, $|M_1| < |x|$, thereby enforcing the network to perform data compression. Conversely, in cases where the hidden state size exceeds the input dimension, $|M_1| > |x|$, the objective is usually to project the data into a more flexible higher-dimensional space. Each individual layer comprises units known as neurons, with each neuron possessing for every input dimension a single weight w_j and bias term b_j.

This concept dates back to the work on the perceptron, where it was applied to binary inputs and the weighted sum exceeding a threshold was considered as firing; otherwise, it was not (Rosenblatt, 1958).

The output h_j of neuron j can be expressed as the weighted sum of an input x and a vector of weights w_j, as shown in Equation 2.3.

$$h_j = \sum_{i=1}^{n} w_{ij} x_i + b_j \tag{2.3}$$

To provide a more compact notation, this sum can be written as the dot product with the added bias term, leading to $h_j = \langle w_j, x \rangle + b_j$. A single layer comprises input and output neurons, defining a matrix of parameters $W_i \in \mathbb{R}^{N_i \times M_i}$ as previously described. This type of layer formulation is known as a **fully-connected** or **dense layer**, characterized by each neuron of a layer having connections to every neuron of the adjacent layers (predecessor and successor layer), while there are no connections between neurons within the same layer. Figure 2.2 depicts, to the left, a network with one layer consisting of 2 input neurons and a scalar output h_1, hence parameters $W \in \mathbb{R}^{2\times 1}$ and $b \in \mathbb{R}$. To the right of Figure 2.2, we illustrate a network with a single layer containing 2 input and 2 output neurons, therefore $W \in \mathbb{R}^{2\times 2}$ and $b \in \mathbb{R}^2$. The final output vector $y \in \mathbb{R}^2 = [h_1, h_2]$ contains two entries, such as the probabilities of the classes *cat* and *dog*.

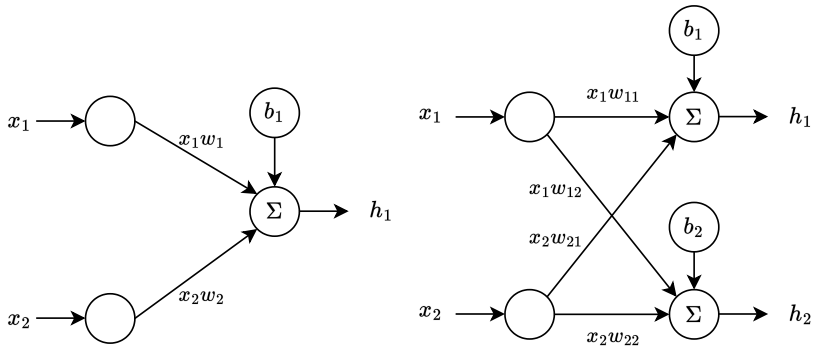


Figure 2.2: This figure illustrates, on the left, a single neuron with two inputs and one output and, on the right, a neuron with two inputs and two outputs. Both examples highlight the computations being applied to the data x with the help of the weights w_{ij} according to Equation 2.3.

So far, we have considered stacking only linear functions, which restricts the neural network to become no more expressive than a single linear function. The introduction of

nonlinearities enables neural networks with multiple layers to approximate various types of complex functions, making them universal function approximators (Hornik et al., 1989). We now integrate a so-called activation function after the weighted sum, resulting in $h_j = \sigma(\langle w_j, x \rangle + b_j)$, applying the nonlinearity to the vector h_j element-wise. An example of such an activation function is the logistic sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$. Other examples include the hyperbolic tangent, $\tanh(x) = \frac{e^x - e^{-x}}{e^{-x} + e^x}$, or the widely-used rectified linear unit, $\text{ReLU}(x) = \max(x, 0) = x\mathbb{1}(x > 0)$, as shown in Figure 2.3.

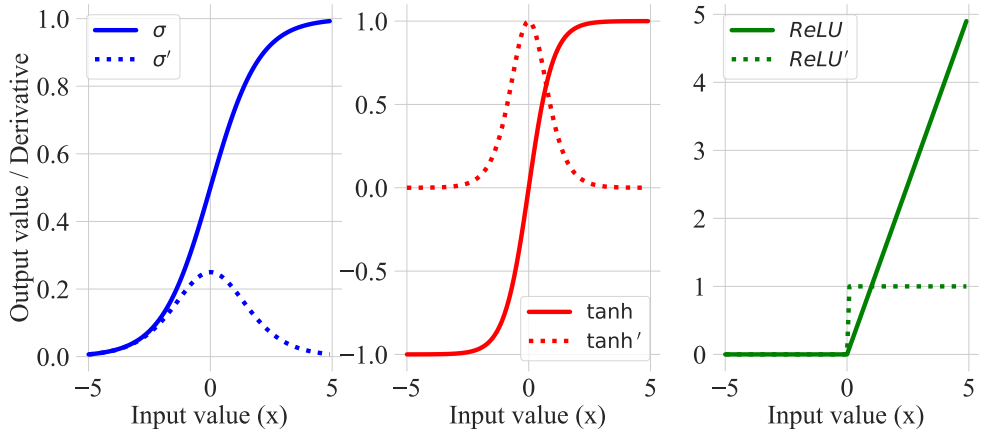


Figure 2.3: Illustration of various activation functions and their gradients: The sigmoid function σ (left) produces vanishing gradients in regions of very low and high input values, limiting effective training of neural networks. The hyperbolic tangent (center) exhibits similar saturation issues but performs better than sigmoid when input values are consistently positive or negative. The rectified linear unit (right), one of the most widely used activation functions in deep learning, behaves linearly for positive inputs while outputting zero for negative values.

In the case of a single output neuron with a sigmoid activation function, the model resembles a linear classifier, where the input x results in an output activation close to 0 or 1, yielding the case of binary logistic regression (Murphy, 2022).

We present an example of an MLP with two layers ($L = 2$) and parameters $\theta = \{\theta_1, \theta_2\}$, as formulated in Equation 2.4 and illustrated in Figure 2.4, now incorporating activation functions.

$$f(x; \theta) = W_2\sigma(W_1 x + b_1) + b_2 \tag{2.4}$$

This architecture provides the cornerstone of current practical deep learning applications, where practitioners simply adjust the number of layers, the individual layer dimensions and select the appropriate nonlinearities. Eventually, solving a task, like predicting which animal an image contains, boils down to adjusting the parameters θ, thereby calculating the extent of change that should be applied to match the true function f^*. This is achieved by employing an algorithm called gradient descent.

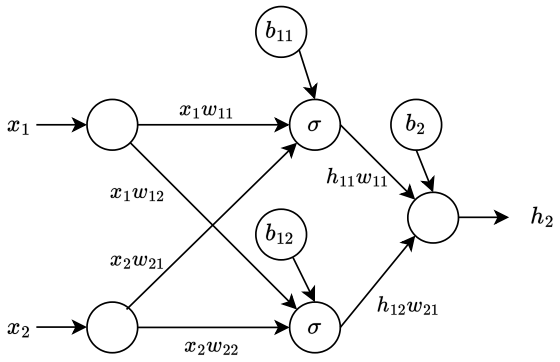


Figure 2.4: An illustration of a multilayer perceptron with two layers, a hidden layer with two hidden nodes and an output layer with one node. The input layer is not considered a layer in the literature.

Learning The composition of functions in deep learning can become highly complex, yet it achieves remarkable performance on challenging tasks such as image classification (Krizhevsky et al., 2012), text-to-image generation (Ramesh et al., 2021), playing video games (Mnih et al., 2015), and controlling a robotic hand (OpenAI et al., 2020). Previously, we outlined a supervised learning setting for classification, specifically mapping images to their corresponding labels. This approach defines a learning problem by showing the system inputs x and their associated labels y, coming from an annotated dataset $\mathcal{D}$. It is important to note that the dataset contains only a subset of samples from the true underlying distribution of animal images, and is therefore referred to as the empirical distribution. The dataset $\mathcal{D}$ is also known as the training dataset because it is utilized for learning, hence adjusting the parameters. To measure the model's performance and guide the adjustment of θ, we employ a differentiable loss as defined in Equation 2.5.

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} L(y^i, f(x^i; \theta)) \tag{2.5}$$

The loss function $\mathcal{L}$ is selected based on the desired task. Common choices include the mean squared error for regression problems or the cross-entropy loss for classification tasks (Murphy, 2022). By comparing the true target values y^i and the model's prediction $\hat{y}^i = f(x^i; \theta)$, one obtains a measure of error. Learning now follows the notion of decreasing this error by changing the parameters using gradient descent. The optimization utilizes the loss to calculate a vector of partial derivatives $\nabla_\theta \mathcal{L}(\theta_m)$ with respect to the parameters. We describe an individual step of this learning procedure in Equation 2.6.

$$\theta_{m+1} \leftarrow \theta_m - \alpha \nabla_\theta \mathcal{L}(\theta_m) \tag{2.6}$$

To assure an incremental improvement with every step m, we define α as the step size, also called the learning rate, that controls the magnitude of the parameter changes. Given that deep neural networks are a bundle of stacked functions, obtaining the gradients of

these interdependent connections requires a method to trace the forward propagation in reverse. This technique of calculating the gradients of the loss with respect to each layer's parameters is known as backpropagation (Rumelhart et al., 1986). The learning procedure follows the idea of the iterative optimization algorithm stochastic gradient descent (SGD). A widely used adaptation of SGD is the Adam optimizer (Kingma and Ba, 2015), which incorporates an adaptive learning rate based on first and second moments of the gradient. Further considerations pertain to the careful selection of the nonlinearities as depicted in Figure 2.3 with their corresponding derivatives.

In the following two paragraphs, we will provide further details on two cases of supervised learning: classification and regression.

2.1.1 Classification

Classification problems involve learning a mapping between input data and their corresponding target labels. In the simplest case with two classes, the target takes on two values $y \in \{0, 1\}$. An objective function typically employed for these binary classification scenarios is the 0-1 loss described in Equation 2.7.

$$\mathcal{L}_{01}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \mathbb{1}(y^i \neq f(x^i; \theta)) \tag{2.7}$$

In scenarios involving more than two classes, where $y \in \{0, 1, 2, \ldots, C\}$, the model needs to predict its confidence across classes as a probability distribution. The target labels y^i can be conceptualized as distributions that assign all of their probability mass to a single class. For such classification problems, the cross entropy loss defined in Equation 2.8 is utilized.

$$\mathcal{L}_{\text{CE}}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \left(y^i \log(\text{softmax}(f(x^i; \theta))) \right) \tag{2.8}$$

Logits, the raw output scores of the neural network, are transformed into a probability distribution using the softmax function (see Equation 2.9).

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.9}$$

In this thesis, classification is used to assign words to behaviors of a robotic agent, as further described in Subsection 4.7.3 of Chapter 4.

2.1.2 Regression

Regression involves training a model to predict scalar values ($y \in \mathbb{R}$). It is similar to classification in the sense that the only parts that differ are the targets and the loss function. A commonly used loss function for regression tasks is the mean squared error, as shown in Equation 2.10.

$$\text{MSE}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(y^i - f(x^i;\theta))^2 \tag{2.10}$$

As an example of a regression task, consider the scenario of predicting a student's grade based on the number of hours spent studying.

2.1.3 Regularization and Normalization

Training a model on the empirical data does not ensure its ability to generate reasonable predictions for novel data instances that were not part of the training dataset. This area of research falls under the topic of generalization and involves additional ideas like the *empirical risk minimization*, *cross-validation*, and regularization (Murphy, 2022), which we will explore in this section due to their relevance to our subsequent approaches.

Deep neural networks, being very powerful at approximation, are prone to overfitting the training data. Overfitting occurs when a model not only learns the essential features to address the task, but also begins capturing the noise within the data. Strategies to mitigate overfitting include expanding the training dataset size or reducing the number of hidden layers or units of the neural network, thereby reducing the model's capacity. However, augmenting the training dataset is not always feasible, and reducing the neural network's expressive power may lead to issues such as underfitting or slower convergence. Moreover, higher-capacity models are often easier to train, as they usually possess more local minima that offer better solutions and are easier to reach with gradient descent than those of small models.

A well-established approach to address overfitting is the introduction of an additional cost term, known as weight decay. With ℓ_1-regularization, also known as lasso[1] regression, we encourage the model to deactivate as many weights as possible by setting their values to zero. This is implemented by incorporating the cost term into the loss, as shown in Equation 2.11.

$$\mathcal{L}(\theta) + \lambda|\theta| \tag{2.11}$$

The hyperparameter λ determines the impact of the regularization on the training. Tikhonov regularization, also known as ridge regression or ℓ_2-regularization, is a method that promotes weight shrinkage, forcing the parameters to take on very small values. It defines the additional cost as the squared sum of weights, as defined in Equation 2.12.

$$\mathcal{L}(\theta) + \lambda||\theta||_2^2 \tag{2.12}$$

Dropout Another effective method to regularize deep neural networks is the dropout technique proposed by Srivastava et al. (2014). This method, applied to specific layers of a neural network, stochastically deactivates neuron connections during training with a

[1] least absolute shrinkage and selection operator

predetermined probability p_o. Figure 2.5 illustrates this concept for a small neural network, visualizing the selective removal of connections. During the training phase, dropout effectively creates and trains multiple distinct subnetworks by excluding different subsets of connections in each forward pass. In the commonly used inverted dropout variant, activations are scaled by a factor of $\frac{1}{(1-p_o)}$ during training, so that no rescaling is needed at test time. The dropout technique essentially trains an implicit ensemble of smaller neural networks. It mitigates co-adaptation by preventing individual neurons from becoming overly dependent on the outputs of other neurons. Consequently, dropout enhances the network's generalization capabilities and overall robustness.

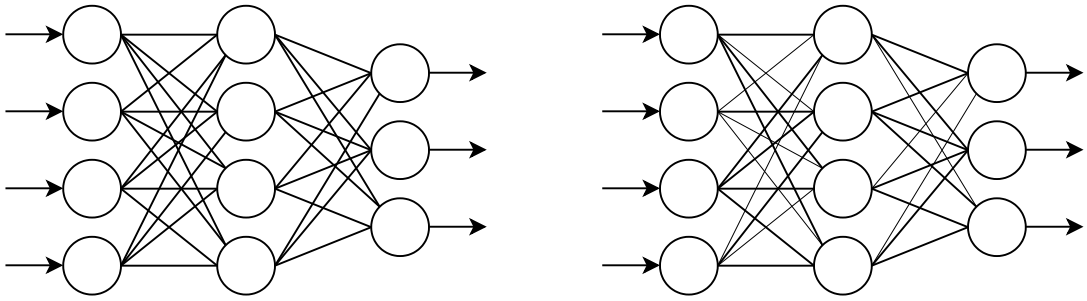


Figure 2.5: Visualization of dropout in a multilayer perceptron with 4 input nodes, 4 hidden nodes, and 3 output nodes. The left side of the figure presents the full network, while the right side demonstrates the effect of dropout, with deactivated connections depicted by the thin gray lines.

We employ the dropout technique in Subsection 4.7.2 and Subsection 5.5.2.

Layer Normalization An alternative method for stabilizing and accelerating neural network training is the technique of normalization. Normalization is frequently combined with dropout to enhance the generalization capabilities of deep neural networks (Radford et al., 2018; Devlin et al., 2019). This paragraph outlines the concept of layer normalization, a method that standardizes activations to ensure a consistent distribution across all feature activations. The approach involves processing data samples $x \in \mathbb{R}^d$, where d represents the dimensionality (corresponding to the number of hidden nodes of that layer), to calculate the mean μ and variance σ^2 across the features as described in Equation 2.13 and Equation 2.14 (Ba et al., 2016)[2].

$$\mu = \frac{1}{d}\sum_{i=1}^{d} x_i \tag{2.13}$$

$$\sigma^2 = \frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2 \tag{2.14}$$

[2]Commonly used biased estimate of the variance.

Following, μ and σ are utilized to scale the activations, resulting in a zero mean and unit variance output, as described in Equation 2.15.

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{2.15}$$

Here, ϵ is a small constant, ensuring numerical stability. The final stage of layer normalization involves scaling and shifting the normalized activations using the learnable parameters $\bar{\gamma}$ and $\bar{\beta}$ as expressed in Equation 2.16.

$$y = \bar{\gamma}\hat{x} + \bar{\beta} \tag{2.16}$$

In summary, the layer normalization technique modifies the scale and offset of normalized activations during training, based on the first and second moments. Unlike other normalization techniques (Ioffe and Szegedy, 2015), the calculations are based on the features of a single data sample. The method of layer normalization is employed in Subsection 4.7.2 and Subsection 5.5.2.

Summary We have provided a concise summary of the fundamental concepts of deep neural networks, including their architecture, training methods, and techniques to improve generalization. For a more comprehensive understanding, further details are available in reference books such as Goodfellow et al. (2016) and Murphy (2022). Due to the many application areas, the selection of a neural network type depends on the specific problem domain. Convolutional neural networks (Krizhevsky et al., 2012) are a suitable choice for image processing tasks (used in Subsection 4.9.5). Recurrent neural networks (Hochreiter and Schmidhuber, 1997; Cho et al., 2014) represent another category of ANNs, specifically designed to handle sequential data such as text (Sutskever et al., 2014) (used in Subsection 3.2.3 and Subsection 4.7.1). One of the more recent and notable advancements in deep neural network architectures is the Transformer (Vaswani et al., 2017) (used in Subsection 4.7.2). It has proven to be a versatile neural network architecture, performing well for various modalities, including text (Devlin et al., 2019), images (Dosovitskiy et al., 2020) and audio (Jaegle et al., 2021).

2.2 Reinforcement Learning

This section examines the core principle employed in this thesis: reinforcement learning. A detailed overview of the theoretical and algorithmic components is provided to introduce the deep reinforcement learning approaches central to this work.

Reinforcement learning (RL) is a framework for modeling autonomous decision-making entities that learn goal-oriented behaviors in complex environments through external feedback signals. Unlike supervised learning, RL operates in environments where the dynamics and the feedback mechanism are generally unknown to the agent. RL offers a unique learning approach based on generated experiences through environmental interactions and behavior learning based on rewards. Essentially, RL does not require ground truth labels, but instead relies on rewards, which are basically scalar feedback signals.

RL has been successfully applied in various scenarios, including human-in-the-loop settings (Christiano et al., 2017) and approaches where information about the underlying structure of the problem is limited. Even in the context of training LLMs to generate appropriate responses, RL can be employed to define preferences for specific outputs, analogous to rewards in a conversational setting (Ouyang et al., 2022). An example of an application without prior knowledge of the underlying problem structure is the work by Mankowitz et al. (2023). The authors aim to predict program code to optimize the runtime of an algorithm using RL, with the execution time serving as the only measurable outcome signal.

The aforementioned cases provide examples where RL is a suitable framework for optimizing objective functions that are not differentiable. RL problems are formulated as sequential decision-making processes, in which the agent must consider uncertainties while interacting with the environment. The agent attempts to maximize a cumulative return or minimize costs, an idea that dates back to the pleasure-pain system proposed by Turing (1948). RL algorithms endeavor to find a policy, denoted by π (often used as a synonym for agent), which is essentially a function mapping the current observed state to an appropriate action. Each RL problem includes a predefined reward function that provides a bounty or penalty based on the combination of the current state, action, and next state. The transition to the following state is determined by the environment's dynamics and is represented by a function of the state-action combination. Unlike the predefined mapping in supervised learning, RL considers a balance between immediate and future rewards with respect to the current actions executed, as they also impact later states and rewards[3]. This trade-off is analogous to the decision of waiting for an apple to grow larger to get a greater reward, rather than consuming a smaller, lower-rewarding apple immediately[4]. Reinforcement learning algorithms enable finding an optimal policy through direct optimization of action selection (see Subsection 2.2.3) or employing a guiding function that acts as a proxy to learn behaviors by estimating the quality of states the agent visits (see Subsection 2.2.2). In both cases, improving the behavior requires exploring the environment.

[3]See Jaeger and Geiger (2023) for an introduction to RL from a supervised learning perspective

[4]Similar to the Stanford marshmallow experiment by Mischel and Ebbesen (1970) about delayed gratification.

Upon discovering a solution, exploitation of this knowledge may be preferred over further exploration, despite the uncertainty regarding its optimality. This trade-off between exploring new possibilities and exploiting known solutions is termed the exploration-exploitation dilemma (Sutton and Barto, 2018), a fundamental challenge in all reinforcement learning settings.

2.2.1 Markov Decision Process

For a more precise mathematical prescription, problems in RL usually follow the definition of the Markov decision process (MDP) (Bellman, 1957), a stochastic control process discrete in time. It is essentially a mathematical framework to model decision-making under uncertainty, defined by a 6-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, \rho_0 \rangle$. The tuple contains elements, such as the set of states $\mathcal{S} \subseteq \mathbb{R}^n$ that describes all the possible sensory inputs an agent could experience. $\mathcal{A} \subseteq \mathbb{R}^m$ describes the set of all possible actions. The transition probability function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, \infty)$ implements the dynamics of the environment and represents the probability of entering the next state s' at timestep $t + 1$, while being in the current state s and executing action a at timestep t. This is expressed by $\mathcal{T}(s, a, s') := p(s_{t+1} = s' \mid s_t = s, a_t = a)$. The reward function maps state-action-state pairs to scalar reward values according to $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, and is, like the transition function, unknown to the agent in advance. The function $\rho_0 : \mathcal{S} \to [0, \infty)$ specifies the initial state distribution, determining the agent's starting conditions. In Figure 2.6, we describe the MDP as interaction loop accompanied by the graphical model in Figure 2.7, encapsulating the previously introduced components of the tuple.

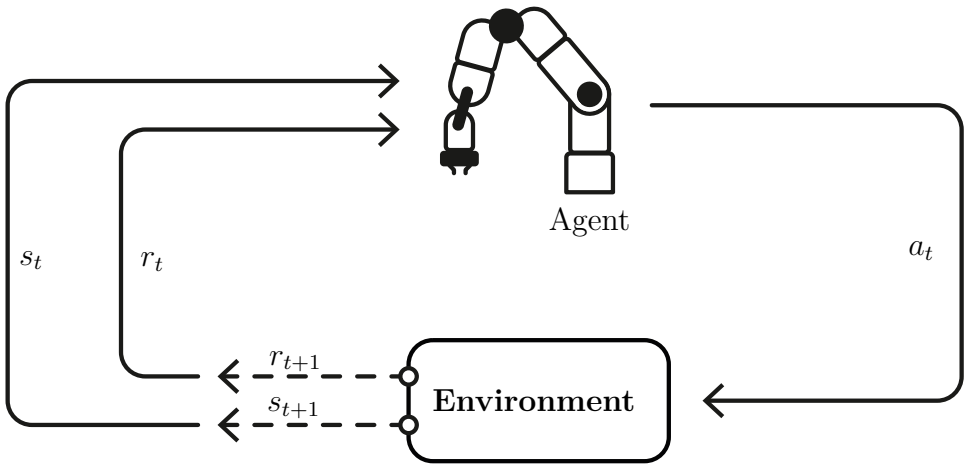


Figure 2.6: Illustration of the reinforcement learning interaction-loop between the agent and the environment. The agent receives the current state s_t as input and, based on this, generates an action a_t. This action transitions the agent into the following state s_{t+1}, accompanied by the corresponding reward r_t in return.

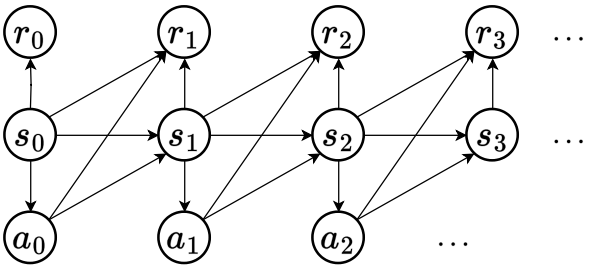


Figure 2.7: Graphical model of the MDP, showing the sequential dependencies between states s_t, actions a_t, and rewards r_t for timesteps $t = 0, 1, 2, 3$. Each state s_t depends on the previous state s_{t-1} and action a_{t-1}, following the transition dynamics, whereas the rewards r_t are determined by the reward function and actions a_t are selected by the policy. An initial state s_0 and reward r_0 are provided by the environment.

Depending on the MDP's underlying task defined by the reward function, certain actions lead to states that yield higher rewards than others. Because the reward function is unknown to the agent, exploring the world is required to find and learn to perform actions that lead to rewarding outcomes. The overall goal of the agent in the MDP is to obtain those rewards $r_t = \mathcal{R}(s_t, a_t, s_{t+1})$ that maximize the sum of rewards $\sum_t r_t$ that we call the return.

Discounted Expected Return and the General Objective Instead of solely maximizing the sum of scalar rewards, RL algorithms usually aim for a discounted version of the return. This discounted expected return, denoted by G_t, describes the anticipated future return at the current timestep t under the consideration of a time-dependent weighting of the rewards. This weighting is implemented by the discount factor γ that defines the extent to which the agent considers future consequences of its current actions. For $\gamma = 0$, the agent becomes myopic (short-sighted) and only cares about immediate rewards. In contrast, values of γ close to 1.0 make the agent farsighted, causing it to care more about rewards far in the future, allowing it to tolerate lower immediate rewards to receive higher rewards in the long term. In theory, to prevent an infinite return calculation, one enforces $\gamma < 1$ for nonzero and constant rewards, limiting the horizon considered for optimization.

In practical applications, the discount factor is a problem-specific hyperparameter typically chosen within the range $\gamma \in [0, 1)$. Equation 2.17 provides the recursive definition of the discounted expected return, which applies exponential weighting to future reward expectations.

$$\begin{aligned} G_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \\ &= r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} + \ldots) \\ &= r_{t+1} + \gamma G_{t+1} \end{aligned} \tag{2.17}$$

To evaluate the performance of our learning system, we need to consider the distribution of trajectories $\tau = (s_0, a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1}, s_T)$ generated by the policy, $\tau \sim \pi$. Equation 2.18 represents the trajectory distribution of a policy as $p_\pi(\tau)$ up to timestep $T-1$.

$$p_\pi(\tau) = \rho_0(s_0) \prod_{t=0}^{T-1} \pi(a_t \mid s_t) \mathcal{T}(s_{t+1} \mid s_t, a_t) \tag{2.18}$$

The expected return for a whole trajectory is calculated employing the summation in Equation 2.19, by essentially evaluating the generated state and action combinations.

$$G(\tau) = \sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1}) = \sum_{t=0}^{T} \gamma^t r_t \tag{2.19}$$

The policy is a function that maps states to actions, defined as $\pi : \mathcal{S} \to \mathcal{A}$. It could be either deterministic $a_t = \pi(s_t)$ or stochastic $a_t \sim \pi(\cdot \mid s_t)$. Measuring the performance $J(\pi)$ of a policy involves calculating the expectation of the discounted return with respect to its T-step trajectories, as shown in Equation 2.20.

$$\begin{aligned} J(\pi) &= \int_\tau p_\pi(\tau) G(\tau) \\ &= \mathbb{E}_{\tau \sim \pi} \left[G(\tau) \right] \\ &= \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{T} \gamma^t r_t \right] \end{aligned} \tag{2.20}$$

Finally, the goal of RL algorithms is to find an optimal policy $\pi^* = \underset{\pi}{\operatorname{argmax}}\, J(\pi)$ that maximizes the objective function in Equation 2.20. This optimal policy π^* achieves the highest expected discounted return from any given state.

Markov Property The Markov property declares that a future state s_{t+1} only depends on the current state s_t and action a_t. This is mathematically formulated with Equation 2.21, which follows the mentioned independence.

$$p(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_1, a_1) = p(s_{t+1} \mid s_t, a_t). \tag{2.21}$$

The transition into the next state s_{t+1} is conditionally independent of all the previous states and actions because the current state contains all the necessary historical information.

Episodic MDP MDPs are usually not assumed to be infinitely long and in our case they are limited by a predefined amount of timesteps T. An episode is defined as a single instance of such a time-limited decision process, lasting at most T timesteps or potentially shorter if a terminal condition is met. An episode consists of a sequence of transitions $\{(s_t, a_t, r_t, s_{t+1})\}_{t=0}^{T-1}$ the agent experiences. The time limit T and the terminal conditions

are determined by the environment designer and are generally problem-dependent. Generally speaking, these decisions often consider relevant references, such as the time required for a human to complete the task or the number of steps required for teleoperating a robot. For theoretical analysis, it is sometimes useful to consider infinite or finite horizons, as well as continuous or discrete time scales, depending on the physical time elapsed after each step. In the robotic setup of this thesis, we employ a discrete time setting where all timesteps t are of equal duration, typically ranging from 20 to 50 Hz. Defining T to be finite additionally provides us with a bounded return that helps with the theoretical analysis (see Equation 2.17) and practical applications.

2.2.2 Value Learning

The concept of value learning revisits the idea introduced at the beginning of this section: estimating the quality of a given state by predicting the reward or expected return, rather than directly predicting actions. This approach follows the supervised learning paradigm of regression, as previously discussed in Subsection 2.1.2. In this context, the loss function is defined as the difference between the predicted return $\hat{G}_t$ and the actual return G_t.

By learning a mapping from the state space $\mathcal{S}$ to a real number $\mathbb{R}$ (the expected return), we can effectively quantify the quality of a given state in terms of its potential to enable the agent to visit other future rewarding states. This state-value function serves as the foundation for inferring an optimal policy, directing the agent to select actions that maximize the value of subsequent states encountered during the decision-making process. There are two primary approaches to value learning that we will describe in the following paragraphs.

Value function Value functions, denoted by $V^\pi(s)$, describe how *good* a state s actually is in terms of the future reward expected from following the corresponding policy π thereafter. This concept is formally expressed in Equation 2.22 as the expected discounted return at timestep t, conditioned on the current state and policy.

$$V^\pi(s) = \mathbb{E}_\pi \left[G_t | s_t = s \right] \tag{2.22}$$

The optimal policy is the one that follows the optimal state-values $V^*(s)$, as shown in Equation 2.23, and can be directly retrieved by following the maximization of state-value estimates.

$$V^*(s) = \max_\pi V^\pi(s) \quad \forall s \in \mathcal{S} \tag{2.23}$$

If a method initially provides a solution in the form of an optimal value function $V^*(s)$, one can derive the policy by selecting actions that maximize the state value of the future step, as formulated in Equation 2.24[5].

[5]Strictly, argmax returns the set $\{a \in \mathcal{A} : f(a) = \max_{a'} f(a')\}$, which may contain multiple elements. In this case, one uniformly samples from this set in the context of RL. We follow the standard convention in RL of writing argmax as returning a single element.

$$\pi(s) = \underset{a}{\operatorname{argmax}}\, \mathbb{E}_{s' \sim \mathcal{T}(\cdot|s,a)} \left[\mathcal{R}(s, a, s') + \gamma V^*(s') \right] \tag{2.24}$$

Figure 2.8 illustrates this concept through a so-called backup diagram, showing states as white circles and the actions as black circles. Selecting an optimal action a in the state s considers the sum of values along all possible paths of subsequent states s'.

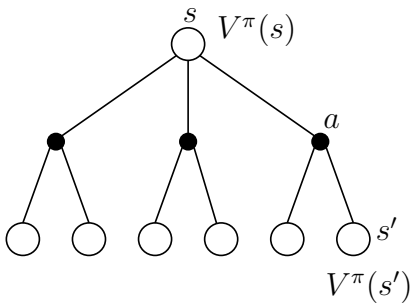


Figure 2.8: Redrawn figure from Sutton and Barto (2018) of a backup diagram for the value function.

For the value function, the recursive Bellman equation is given in Equation 2.25. It represents an average aggregated among all the possible actions and successor states, as denoted in Figure 2.8 by the black dots for the actions (first sum) and the successor states as white circles at the bottom (second sum).

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_{a \sim \pi(\cdot|s), s' \sim \mathcal{T}(\cdot|s,a)} \left[\mathcal{R}(s, a, s') + \gamma V^\pi(s') \right] \\ &= \sum_{a \in \mathcal{A}} \pi(a \mid s) \sum_{s' \in \mathcal{S}} \mathcal{T}(s' \mid s, a) \left[\mathcal{R}(s, a, s') + \gamma V^\pi(s') \right] \end{aligned} \tag{2.25}$$

Q-function Estimating the optimal actions based solely on the value function is limited by the uncertainty regarding the unknown environment dynamics $\mathcal{T}(s' \mid s, a)$. The value function $V(s)$ does not convey information about the actions that led to it, or which actions are particularly advantageous in s. A finer-grained approach involves evaluating state-action pairs (s, a) when estimating the expected future return. The state-action value function, commonly referred to as the Q-function, estimates the values of state-action pairs rather than single states, and is denoted by $Q^\pi(s, a)$. Similar to the value function, the definition of the Q-function is provided in Equation 2.26.

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[G_t | s_t = s, a_t = a \right] \tag{2.26}$$

Figure 2.9 illustrates this concept by a backup diagram, showing how the expected return is calculated by considering a state-action pair (s, a) and evaluating all subsequent paths the policy could take. Inferring the policy from the Q-function works as described in Equation 2.27, which involves selecting the action a in state s that maximizes the state-action value.

$$\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a) \tag{2.27}$$

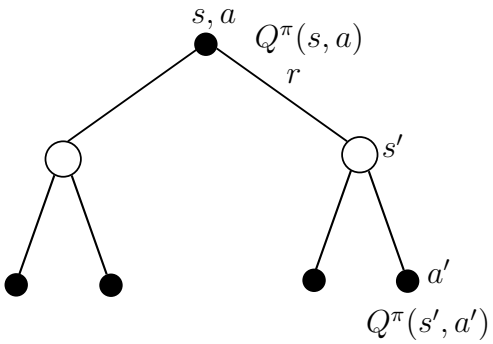


Figure 2.9: Redrawn figure from Sutton and Barto (2018) of a backup diagram for the state-action function.

The Bellman equation of the Q-function is presented in Equation 2.28.

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_{s'\sim\mathcal{T}(\cdot|s,a)}\left[\mathcal{R}(s,a,s') + \gamma \mathbb{E}_{a'\sim\pi(\cdot|s')}\left[Q^\pi(s',a')\right]\right] \\ &= \sum_{s'} p(s' \mid s,a)\left[\mathcal{R}(s,a,s') + \gamma \sum_{a'\in\mathcal{A}} \pi(a' \mid s') Q^\pi(s',a')\right] \end{aligned} \tag{2.28}$$

The value function and Q-function are closely related. The state value $V(s)$ can be derived from the state-action value function $Q(s,a)$ through the expectation outlined in Equation 2.29, which represents an average of all possible action branches for a state-action function (see Figure 2.8).

$$V^\pi(s) = \mathbb{E}_{a\sim\pi}\left[Q^\pi(s,a)\right] \tag{2.29}$$

Q-learning A prominent approach to value learning is the *Q-learning* algorithm (Watkins and Dayan, 1992). It iteratively updates bootstrapped estimates of state-action values using temporal difference (TD) learning (Sutton, 1988). The update rule, described in Equation 2.30, starts with random initializations of the state-action values. Following this, the agent explores the environment, gathering experiences and observing rewards r_t for different state-action pairs (s_t, a_t), from which the discounted expected return is calculated (Section 2.2.1). In this TD learning approach, updates occur at every timestep, following a procedure similar to the supervised setting. A desired value, the TD target, is computed according to $r_{t+1} + \gamma \max_a Q(s_{t+1}, a)$, representing an estimate of the discounted future return plus the current reward when greedily exploiting the action that maximizes the Q-values prediction. The TD-error is declared as $r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)$, guiding the magnitude and direction of the update. The update itself involves a step size parameter α, which dictates the extent of the resulting change for the new estimate $Q'(s_t, a_t)$.

$$Q'(s_t,a_t) = Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right] \tag{2.30}$$

While different variations of TD-learning exist (Sutton and Barto, 2018, Chapter 6), this thesis concentrates solely on the single step approach, known as TD(0). Here, we follow the

Sutton and Barto convention in which r_{t+1} denotes the immediate reward obtained after taking action a_t. In later actor-critic equations, r_t denotes the same transition reward, matching the replay-buffer tuple (s_t, a_t, r_t, s_{t+1}).

2.2.3 Policy Gradient

A different approach to learning involves directly optimizing the action selection through a policy without using any notion of value. These kinds of approaches fall under the umbrella term policy gradient methods, of which one instance is the REINFORCE algorithm (Sutton and Barto, 2018, Chapter 13). The objective, intuitively, is to increase the probability of actions that lead to high rewards, while decreasing those resulting in low rewards. In Equation 2.31, the policy gradient is defined as the expectation of the gradient of the log probability (grad-log-prob) given the policy parameters and a sampled trajectory return.

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log(\pi_\theta(a_t \mid s_t)) G_t\right] \tag{2.31}$$

We optimize the action selection by adjusting the policy parameters θ directly, using gradient ascent (Equation 2.32).

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_{\theta_k}) \tag{2.32}$$

This method results in a policy that outputs actions based on its parameters $\pi_\theta(a \mid s) = p(a_t = a \mid s_t = s)$, maximizing the objective in Equation 2.20.

2.2.4 Actor Critic Method

The actor-critic method combines the strengths of policy gradients with those of value learning. For instance, in the robotics setting with continuous action spaces, policy gradient methods offer a suitable approach as they directly learn the policy. However, policy gradient methods like REINFORCE are well-known for their instability and lack of convergence guarantees. Q-learning, which was already introduced in Section 2.2.2, is a sample-efficient and stable approach that is guaranteed to converge (Watkins and Dayan, 1992). Yet, in environments with continuous actions, Q-learning becomes problematic to employ because it is impossible to evaluate the Q-function for all possible actions in a particular state, thus hindering the selection of the maximizing action according to the predicted Q-values (cf. Equation 2.30). Well-known methods by Lillicrap et al. (2016) and Haarnoja et al. (2018) implement the actor-critic method to address these challenges in current reinforcement learning applications. In the following, we provide the example of a single step optimization with a **deterministic policy**, using TD(0) (Sutton and Barto, 2018)[Chapter 6]. For further details on the topic of actor-critic, we refer to Sutton and Barto (2018, Section 13.5).

The actor-critic architecture primarily consists of two components. The first component is the policy, henceforth also called the **actor**, which is a function parameterized by

θ that maps states to actions, $a_t = \pi_\theta(s_t)$, and is trained using policy gradient methods[6]. The second component, the **critic**, is equivalent to the Q-function, and is a function parameterized by ϕ. It regresses the expected discounted return for a given state-action pair, $Q_\phi(s_t, a_t)$. For the critic, we attempt to minimize the TD-error, the ℓ_2-norm between the predicted return and the target value, hence following the usual idea of value learning with the temporal difference approach. In Equation 2.33, we outline the critic loss function, drawing parallels to the notion of regression from supervised learning as discussed in Subsection 2.1.2.

$$L(\phi) = ||Q_\phi(s_t, a_t) - \delta_t||^2 \tag{2.33}$$

The TD target is the sum of the immediate reward r_t and the expected return of the next state s_{t+1} given the proposed action a_{t+1} by the actor, as described in Equation 2.34. With the loss function actually containing the target as a prediction by the model itself, such methods that base the update on previous estimates are known in the literature as bootstrapping methods, with their idea originating in dynamic programming (Sutton and Barto, 2018, Section 6).

$$\delta_t = r_t + \gamma Q_\phi(s_{t+1}, \pi_\theta(s_{t+1})) \tag{2.34}$$

This is essentially the same as Q-learning from Section 2.2.2, but with the current policy replacing the greedy action selection using the Q-function (cf. Equation 2.30), because the action space is continuous. The actor attempts to optimize the loss over the expected return regressed by the critic, as shown in Equation 2.35.

$$L(\theta) = -Q_\phi(s_t, \pi_\theta(s_t)) \tag{2.35}$$

In essence, the critic evaluates the action selection, while the actor attempts to maximize the expected return estimated by the critic. This process mirrors the action selection in Q-learning, where an action yielding the highest state-action value is chosen. However, instead of evaluating all possible actions in a state, as in Q-learning, we follow a sample-based approach. The critic acts as the leading part of the learning procedure, since we obtain the actor gradients through the estimation of the Q-values, highlighting the synergy between value learning and policy gradient methods in the actor-critic approach.

2.2.5 Goal Extension of the MDP

Our current formulation of the MDP is limited to the settings with a single reward function, thus focusing on a single task. A more versatile approach within the MDP framework involves the introduction of goals, providing additional context to the RL agent regarding desired outcomes within the environment.

The default RL setting could be considered as a special case with only one static goal and a single reward function. However, agents are usually tasked with solving multiple

[6]For probabilistic policies, $a_t \sim \pi_\theta(\cdot \mid s_t)$, the reparameterization trick, as discussed in Subsection 2.2.7, is necessary when using neural networks.

goals, also known as the multi-goal setup (Plappert et al., 2018). The idea regards different realizations of the reward function with respect to the various goals the environment contains. This particular branch of RL is called goal-conditioned RL and has been adopted by various works (Andrychowicz et al., 2017; Röder et al., 2020), especially in robotics, where humans could provide language goals (Akakzia et al., 2021; Röder et al., 2022; Ahn et al., 2023). Goal-conditioned RL dates back to the early work by Kaelbling (1993) on dynamically changing goals. Recent work on *universal value function approximators* by Schaul et al. (2015) formulated a class of value functions that generalize well not only to unseen states, as in the usual RL setting, but to state and goal combinations. Generalizing to many state and goal combinations is challenging as the range of combinations is much larger than when only states are considered. Although only a fraction of state-goal pairs will be seen by the agent, Schaul et al. (2015) highlight the generalization and extrapolation capabilities.

For a precise mathematical formulation, we present the extension of the MDP that involves an additional set of goals $\mathcal{G}$, from which we sample a single instance $g \in \mathcal{G}$ that becomes part of the policy's input. As a result, we get the goal-conditioned policy, $\pi : \mathcal{S} \times \mathcal{G} \to \mathcal{A}$, which generates a distribution over actions conditioned on the state and goal, $\pi(\cdot \mid s_t, g_t)$. Intuitively, conditioning the policy on various goals triggers different types of behaviors to solve diverse problems. This approach strongly relates to learning multiple types of skills (Eysenbach et al., 2019) or, in a hierarchical sense (Eppe et al., 2022), even options (Barto and Mahadevan, 2003) and subgoals (Röder et al., 2020).

The goal g comes from the desired goal distribution $g \sim \rho_g$, ensuring the proposal of achievable goals. For each goal, we have the goal-conditioned reward function $\mathcal{R}(s, a, g)$, which outputs the reward considering the sampled goal g. Goals are often defined as subspace of the state space, $\mathcal{G} \subseteq \mathcal{S}$; for example, it could contain the Cartesian coordinates of a robot gripper and objects in the environments (Andrychowicz et al., 2017). Such a subset would optionally exclude features such as velocities and joint information. To evaluate how well a state contributes to reaching the goal, works like Andrychowicz et al. (2017) employ an oracle function $\Phi : \mathcal{S} \to \mathcal{G}$ that maps states to goals. Referring to the aforementioned example, this function would only return the Cartesian coordinates, excluding features not part of the goal state. With such a function, one could retrieve the achieved goal $\Phi(s)$ and compare it to the proposed desired goal g, which many algorithms utilize for learning (Andrychowicz et al., 2017; Nair et al., 2018b; Colas et al., 2019; Röder et al., 2020).

Goal-conditioned RL typically employs the **dense** or **sparse** reward definition. For instance, consider a scenario where a robot must navigate to a target position, defined by its XY coordinates as the desired goal position g on a map. While the robot's state s_t encompasses more information than necessary, e.g., battery status and other sensor readings, the function ϕ extracts the pertinent details, specifically the agent's current XY-location (achieved goal $\Phi(s_t)$), to perform the reward assessment **in two ways**.

Dense The reward could be modeled as **dense** signal $r_t \in \mathbb{R}$, employing the negative Euclidean distance of the achieved goal and desired goal, $r_t = -\|\Phi(s_t) - g\|$[7]. Such a dense reward function based on distance metrics could lead to policies that get stuck in local optima, misleading the agent and possibly hindering further exploration. For instance, if the agent is tasked to place an object inside a box but first needs to lift it up, thus moving it further away from the goal location, a dense signal would encourage the agent to push the object next to the box, a suboptimal solution.

Sparse We could employ a **sparse** reward function that only yields a positive signal, in this particular case a non-negative value, when a specific condition is satisfied. It could be for example the condition that if the distance to the desired goal is smaller than a predefined threshold ϵ or any other type of logic calculation. Following the idea of the minimum distance that needs to be met, the binary reward signal $r_t \in \{-1, 0\}$ could be defined as $r_t = -\mathbb{1}(\|\Phi(s_t) - g\| > \epsilon)$, where we get a reward of -1 for each step taken and 0 if the goal is achieved. These types of settings with a negative reward are known as a stochastic shortest path problems (Sutton and Barto, 2018), as they encourage the agent to minimize the total amount of costs, hence reaching the goal as quickly as possible. Learning with sparse rewards, compared to dense ones, is usually more challenging and requires additional effort to handle the amount of non-rewarding experiences when not reaching the goal state by chance due to the lack of exploration. However, it has been shown to predominantly work well for complex problems and requires less reward engineering (Andrychowicz et al., 2017; Plappert et al., 2018). For the remainder of this thesis, we consider using the sparse reward definition.

Goal Representations Goals, as previously mentioned, can be represented as part of a state, but they can also take on different forms. Examples for different representations are goal images (Nair et al., 2018b), language-based goals (Jiang et al., 2019), and other semantic goal representations (Arumugam et al., 2019; Akakzia et al., 2021). Since these representations do not directly align with the characteristics of the state space, mapping states to achieved goals now demands an alternative approach.

For instance, in the goal-as-state case where the goal space is equivalent to the state space $\mathcal{S} = \mathcal{G}$, Φ would be the identity function, meaning goals are described by the entire state. In the case of language, a goal g_ℓ tends to be more abstract, and its mapping to a single state is not deterministic and may be ambiguous. For example, the instruction *"pick up the red ball"* satisfies multiple interpretations, including *"pick up the red object"* and *"pick up the ball"*. A solution to this is to consider a whole set of states $g_\ell \in \mathcal{S}_{g_\ell}$, where $\mathcal{S}_{g_\ell} \subset \mathcal{S}$, that satisfy the goal, and

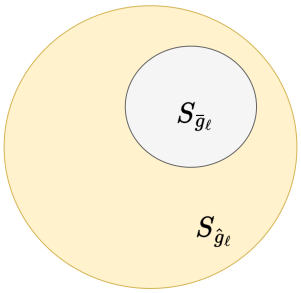


Figure 2.10: We illustrate the language-to-state mapping for the instructions *"pick up the red object"*, mapping to the set $\mathcal{S}_{\hat{g}_\ell}$, and *"pick up the red ball"* mapping to $\mathcal{S}_{\bar{g}_\ell}$.

[7]Written as Euclidean norm

result in a sparse reward signal. For instance, language goals like $\hat{g}_\ell$ = *"pick up the red object"* and $\bar{g}_\ell$ = *"pick up the red ball"* would definitely overlap in terms of their goal states, hence $\mathcal{S}_{\hat{g}_\ell} \cap \mathcal{S}_{\bar{g}_\ell} \neq \emptyset$ and in this case even follows $\mathcal{S}_{\hat{g}_\ell} \supset \mathcal{S}_{\bar{g}_\ell}$ (cf. Figure 2.10). This is because a *"red ball"* is a *"red object"*, and there are many other red objects that would fall under that broader category. The formulation of the MDP with goals is simply an extension to the known setting from Subsection 2.2.1 (Schaul et al., 2015; Röder et al., 2020), and one could consider the goal to be a part of the observed state s_t, in case their modalities align. The goal-conditioned setting mainly changes the inner workings of the MDP-based simulation and requires additional considerations by the environment designer. Apart from that, the previously introduced theory and equations remain largely unchanged; we just need to consider the goal as additional input into the functions that regress the expected returns or conduct the action-selection. We now express the goal-conditioned Bellman equations for the value and Q-function in Equation 2.36 and Equation 2.37, respectively, additionally conditioning on the goal g according to Schaul et al. (2015).

$$V^\pi(s, g) = \mathbb{E}_{a \sim \pi(\cdot|s,g), s' \sim \mathcal{T}(\cdot|s,a)} \left[\mathcal{R}(s, a, g) + \gamma V^\pi(s', g) \right] \tag{2.36}$$

$$Q^\pi(s, a, g) = \mathbb{E}_{s' \sim \mathcal{T}(\cdot|s,a)} \left[\mathcal{R}(s, a, g) + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s',g)} \left[Q^\pi(s', a', g) \right] \right] \tag{2.37}$$

In goal-conditioned RL, it is of additional benefit to consider the episodic MDP setting (Sutton and Barto, 2018, Sec. 3.4) (see Section 2.2.1). Each episode involves a new sample of a goal that remains unchanged throughout the entire episode. For the remainder of this thesis, we will omit the time indicator t of g where it is clear from context that the goal is constant for each episode. In addition to utilizing the reward as a performance metric, one could also calculate the ratio of episodes in which the agent succeeds at reaching the goal state, known as the success rate. In Figure 2.11, we provide the goal-extended MDP as a graphical model.

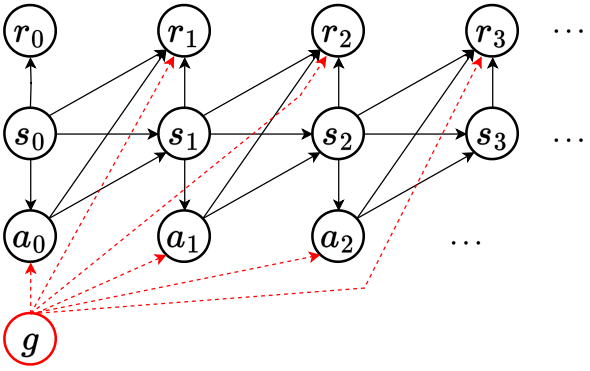


Figure 2.11: Graphical model of the extended MDP, visualizing the sequential dependencies of s_t, a_t, and r_t with $t = 0, 1, 2, 3$. We highlight the goal dependence that influences the action selection by the policy and the resulting rewards. The goal g remains constant for the whole episode. The initial state s_0 and the reward r_0 are provided by the environment, whereas the goal comes from the initial goal distribution, $g \sim \rho_g$.

2.2.6 Current Reinforcement Learning Methods

There is a wide range of reinforcement learning algorithms available to solve various types of problems, such as robotic locomotion (Haarnoja et al., 2019) and manipulation (Nair et al., 2018b; OpenAI et al., 2020), playing video games (Mnih et al., 2015; Vinyals et al., 2019; Hafner et al., 2021), or controlling other complex systems (Luo et al., 2022).

The chosen technique largely depends on the properties of the task's MDP. In MDPs with a small state and action space, where table-based methods are sufficient (Sutton and Barto, 2018, Part I), we can store the policy as a stochastic matrix, where a state corresponds to a row of the matrix that contains the action probabilities. In cases where the state space is continuous or simply too large – for instance, when the inputs are images – we need to employ methods of approximation (Sutton and Barto, 2018, Part II), such as a deep neural network (DNN). Although DNNs cannot store a one-to-one mapping like a stochastic matrix, they essentially learn to encode similar states with comparable representations. It also matters if the action space is discrete (Mnih et al., 2015), where a finite choice of actions follows the paradigm of classification (see Subsection 2.1.1). In the case of continuous actions, such as in robotics, we need to cover the space of all real-valued vectors that could be used to control the robot (Lillicrap et al., 2016; Haarnoja et al., 2018). This requires a technique similar to regression, but with a multidimensional output, different from Subsection 2.1.2.

In cases where both the action and state spaces are continuous or too large, approximation allows us to adopt methods like the aforementioned REINFORCE algorithm (Sutton and Barto, 2018, Part II) directly, or implement different variations of algorithms like Deep Q-Networks (DQN) (Mnih et al., 2015), which are based on Q-learning but employ DNNs. The branch of RL using DNNs is termed deep reinforcement learning (Achiam, 2018; Jaeger and Geiger, 2023), and provides a means to overcome the *curse of dimensionality*, making

it applicable for high-dimensional and real-world problems. Current state-of-the-art RL algorithms in robotics often employ the actor-critic approach (Lillicrap et al., 2016; Schulman et al., 2017), as it combines the strengths of policy optimization and value learning, as outlined in Subsection 2.2.4. Here, both the policy and the value function are implemented by DNNs, where training is conducted using stochastic gradient descent (see Section 2.1). Well-known algorithms stemming from this idea are *Proximal Policy Optimization* (PPO) (Schulman et al., 2017), used for dexterous robotic manipulation (OpenAI et al., 2020), and instruction-following large language models (Ouyang et al., 2022). Furthermore, established algorithms such as *Deep Deterministic Policy Gradients* (DDPG) (Lillicrap et al., 2016), *Twin Delayed Deep Deterministic Policy Gradient* (TD3) (Fujimoto et al., 2018), and the *Soft Actor-Critic* (SAC) (Haarnoja et al., 2018) are commonly used in current research.

2.2.7 Soft Actor-Critic

The Soft Actor-Critic (SAC) is a state-of-the-art reinforcement learning algorithm (Haarnoja et al., 2018) that employs the notion of maximum entropy (Ziebart et al., 2008) to learn robust behaviors with a stochastic policy and soft Q-learning (Haarnoja et al., 2017). Maximum entropy introduces a form of regularization to policy learning, preventing the policy's action distribution $\pi(\cdot \mid s_t)$ from becoming deterministic, thereby potentially avoiding poor local optima. This approach has proven to stabilize and accelerate learning, while improving the sample efficiency and asymptotic performance. Furthermore, SAC has been shown to also work in real-world robotics (Haarnoja et al., 2019) and provides a foundation for many advanced RL implementations (Akakzia et al., 2021; Sodhani et al., 2021; Silva et al., 2022).

Generally, SAC is a model-free, off-policy actor-critic algorithm. Model-free implies that it does not learn a model of the world, i.e., the transition dynamics of the MDP (Hafner et al., 2021). Off-policy describes RL algorithms that store experiences continuously in a buffer, from which mini-batches of past experiences are sampled to update the policy and the Q-function – a technique known as experience replay (Mnih et al., 2015). This method enhances efficiency by reusing collected experiences multiple times and mitigates issues from correlated samples during online learning.

In actor-critic methods, one refers to the policy as the actor and the Q-function as the critic. In Subsection 2.2.4, we have already provided the necessary details on this approach, but now highlight the changes introduced by SAC. Both actor and critic are implemented by deep neural networks as high dimensional nonlinear function approximators, commonly utilized in deep RL. This is particularly beneficial in domains where tabular methods are impractical due to high-dimensional state and action spaces, such as in robotics or other continuous control problems.

Maximizing Entropy We begin by describing how the entropy term is integrated into the objectives of the actor-critic framework. Entropy, in general, quantifies the average amount of information a probability distribution p contains, as described in Equation 2.38.

$$H(p) = \mathbb{E}_{x\sim p}\left[-\log p(x)\right] = \sum_x -\log(p(x))p(x) \tag{2.38}$$

The entropy term in SAC is used to encourage the policy to generate action distributions that are as uniform as possible, while balancing this with the extrinsic reward from the task. This creates a trade-off between maximizing the expected return and the entropy term simultaneously. This approach somewhat resembles the regularization from Subsection 2.1.3, where additional costs are added to the objective, akin to ℓ_1 and ℓ_2-regularization. A more uniform distribution implies higher entropy, leading to increased exploration. Maximum entropy policies have demonstrated robustness against model and estimation errors (Ziebart et al., 2008; Eysenbach and Levine, 2022). Moreover, they can capture multiple equally good solutions, assigning equal probabilities to choosing them.

In Equation 2.39, we provide the maximum entropy RL objective. To prevent the agent from getting obsessed by its own randomness, a temperature value $\alpha > 0$ is chosen as a

hyperparameter to balance the two feedback signals.

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t\Big(\mathcal{R}(s_t, a_t, s_{t+1}) + \alpha H(\pi(\cdot \mid s_t))\Big)\right] \tag{2.39}$$

In addition, the soft Q-function (Equation 2.40) introduces a change to the Bellman equation (Equation 2.41), allowing it to regress the expected entropy-based reward for each timestep t.

$$Q^\pi(s,a) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t\mathcal{R}(s_t, a_t, s_{t+1}) + \alpha\sum_{t=1}^{\infty}\gamma^t H(\pi(\cdot \mid s_t))\Big| s_0 = s, a_0 = a\right] \tag{2.40}$$

$$Q^\pi(s,a) = \mathbb{E}_{\substack{s'\sim\mathcal{T}(\cdot|s,a)\\ a'\sim\pi(\cdot|s')}}\left[\mathcal{R}(s,a,s') + \gamma(Q^\pi(s',a') + \alpha H(\pi(\cdot \mid s')))\right] \tag{2.41}$$

As mentioned, the entropy regularization helps prevent the policy from prematurely converging to a suboptimal or even poor local optimum. Previously, SAC required finding an optimal temperature value α for each environment separately (Haarnoja et al., 2018). In a follow-up work, Haarnoja et al. (2019) proposed temperature tuning via a gradient-based optimization. This approach integrated the temperature adjustment into the learning process, eliminating the need for manual tuning of this hyperparameter. The temperature objective in Equation 2.42 uses the target entropy heuristic $\bar{H} = -\dim(\mathcal{A})$, the negative dimensionality of the action space.

$$L(\alpha) = \mathbb{E}_{a_t\sim\pi_t}\left[-\alpha\log\pi_t(a_t \mid s_t) + \alpha\dim(\mathcal{A})\right] \tag{2.42}$$

As the agent progresses with the task and receives increasing extrinsic rewards, the temperature decreases. The entropy bonus essentially supports the exploration in the early stages of training and later on converges to a small value, leading to an equilibrium between the extrinsic reward and intrinsic entropy bonus.

Experience Replay SAC is an off-policy method that employs the notion of experience replay (Mnih et al., 2015). Such RL methods store the experienced transition tuples $e_t = (s_t, a_t, r_t, s_{t+1})$ in a replay buffer $\mathcal{D}_t = \{e_1, \dots e_t\}$, adhering to the first in - first out (FIFO) principle. When the algorithm switches from its exploration phase, where it collects experiences, to the training phase, it samples $(s_t, a_t, r_t, s_{t+1}) \sim p(\mathcal{D})$ from the replay buffer $\mathcal{D}$ to update the parameters of the actor-critic model. The term off-policy is used because the experiences sampled for training are coming from earlier stages, where the policy might have acted very different in comparison to the current one, hence being *off* the current policy.

Target Networks To account for the continuously changing parameters of the critic when training with its own bootstrapped TD targets of the next step (cf. Equation 2.34), SAC employs target networks whose parameters update less frequently (Mnih et al., 2015).

Having parameters that change more slowly provides temporary stationarity in terms of the targets used for learning the regression of the return. The target critic parameters $\bar{\phi}$ are a slowly changing version of the critic parameters ϕ. Their rate of change is determined by the hyperparameter ρ and are updated according to Equation 2.43, known as Polyak averaging.

$$\bar{\phi} \leftarrow \rho\bar{\phi} + (1-\rho)\phi \tag{2.43}$$

Soft Q-Learning and Double Q-trick SAC employs the double Q-trick to address the overestimation bias in return estimation by consistently selecting the lowest estimate from N (*e.g.*, N=2) return estimates. Essentially, it defines the critic as an ensemble of identically structured but differently initialized Q-functions, each providing their own predictions. This technique was inspired by the work on Double DQN and is utilized in other RL algorithms as well (Fujimoto et al., 2018). It is formally described in Equation 2.44, where it takes the minimum of the proposed return predictions.

$$\min_{i=1,\dots,N} Q^{\pi}_{\bar{\phi}_i}(s', \tilde{a}'), \quad \tilde{a}' \sim \pi_\theta(\cdot \mid s') \tag{2.44}$$

Combining the clipped double Q-trick (Fujimoto et al., 2018) with the soft Q-function results in the new TD target outlined in Equation 2.45.

$$\delta(r, s') = r + \gamma \left(\min_{i=1,\dots,N} Q^{\pi}_{\bar{\phi}_i}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}' \mid s') \right), \quad \tilde{a}' \sim \pi_\theta(\cdot \mid s') \tag{2.45}$$

This leads to the result in Equation 2.46, which provides the objective for training the N critics.

$$L(\phi_i, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(s,a,r,s') \in \mathcal{D}} \left(Q_{\phi_i}(s,a) - \delta(r,s') \right)^2 \tag{2.46}$$

SAC uses the TD-0 approach to learn the return estimations for the critic and action predictions for the actor. Unlike REINFORCE, the actor aims to maximize both the expected future return and the expected future entropy bonus simultaneously (see Equation 2.47).

$$L(\theta, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{s \in \mathcal{D}} \left(\alpha \log \pi_\theta(\tilde{a} \mid s) - \min_{i=1,\dots,N} Q_{\phi_i}(s, \tilde{a}) \right), \quad \tilde{a} \sim \pi_\theta(\cdot \mid s) \tag{2.47}$$

In Algorithm 1, we present the pseudocode for SAC, which systematically outlines the objectives introduced.

Implementation details The policy in SAC parameterizes a Gaussian distribution by outputting two vectors: $\mu_\theta(s)$ and $\sigma_\theta(s)$. Actions are then sampled from this distribution according to $a \sim \mathcal{N}(\mu_\theta(s), \sigma_\theta(s))$. To address the challenge of differentiating through

Algorithm 1 Soft Actor-Critic Algorithm (Haarnoja et al., 2018)

θ	actor parameters	ϕ_i	critic parameters for $i = 1, \ldots, N$
$\bar{\phi}_i \leftarrow \phi_i$	target critic parameters	$\mathcal{D}$	off-policy replay buffer
α_π	actor learning rate	α_Q	critic learning rate
α	entropy temperature	α_{entropy}	temperature learning rate

for episode $0, \ldots, E$ **do**
  Sample initial environment state $s_0 \sim \rho_0$
  for each environment step $t = 0, \ldots, T-1$ **do**
    Sample action $a_t \sim \pi_\theta(\cdot \mid s_t)$
    Take environment transition $s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t)$
    Compute reward $r_t \leftarrow \mathcal{R}(s_t, a_t, s_{t+1})$
    Store transition in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
  end for
  for each gradient update step **do**
    Sample mini-batch of transitions $\mathcal{B} \sim \mathcal{D}$
    for each critic $i = 1, \ldots, N$ **do**
      Update critic Q_{ϕ_i} by taking a step of gradient descent
      $\phi_i \leftarrow \phi_i - \alpha_Q \nabla_{\phi_i} L(\phi_i, \mathcal{B})$ according to Equation 2.46
      Update target networks $\bar{\phi}_i$ following Equation 2.43
    end for
    Update actor π_θ by taking a step of gradient descent
    $\theta \leftarrow \theta - \alpha_\pi \nabla_\theta L(\theta, \mathcal{B})$ according to Equation 2.49 based on Equation 2.47
    Update temperature α by taking a step of gradient descent
    $\alpha \leftarrow \alpha - \alpha_{\text{entropy}} \nabla_\alpha L(\alpha)$ according to Equation 2.42
  end for
end for

stochastic nodes, SAC employs the well-known reparameterization trick, transforming the stochastic nodes into deterministic ones. This is achieved by parameterizing the distribution in terms of a fixed standard normal distribution and the policy parameters. Consequently, sampling actions for training follows Equation 2.48, where $\xi \sim \mathcal{N}(0, I)$ represents the noise sampling from the standard normal and s the current state.

$$\tilde{a}(s, \xi) = \mu(s) + \sigma(s) \odot \xi \tag{2.48}$$

Here, I denotes the identity matrix, matching in dimension to the stochastic term ξ. Since the output of the Gaussian distribution is unbounded while action spaces typically have limits, SAC employs the **tanh** function to squash these outputs (cf. Figure 2.3). This squashing appropriately modifies the actor loss according to Equation 2.49, ensuring that the sampled actions remain within the defined action space boundaries.

$$L(\theta, \mathcal{D}) = \sum_{t=0}^{|\mathcal{D}|} \left(\alpha \log \pi_\theta(\tilde{a}_\theta(s_t, \xi) \mid s_t) - \min_{i=1,2} Q_{\phi_i}(s_t, \tilde{a}_\theta(s_t, \xi)) \right), \quad (s_t, a_t) \sim \mathcal{D} \quad \xi \sim \mathcal{N}(0, I) \tag{2.49}$$

The application of the reparameterization trick and the tanh squashing function enables SAC to maintain a stochastic policy while allowing for efficient gradient-based optimization throughout the network.

Summary

This chapter provided an overview of the background material relevant to this thesis. It introduced the key concepts of machine learning, deep learning, and reinforcement learning. For more in-depth knowledge on these topics, our references include seminal books, such as Sutton and Barto (2018) for reinforcement learning, Goodfellow et al. (2016) and Murphy (2022) for deep learning and natural language processing, and Deisenroth et al. (2020) for the mathematics of machine learning.

The combination of deep learning and reinforcement learning, known as deep reinforcement learning (Mnih et al., 2015; Achiam, 2018), forms a key focus of this thesis and is used in Section 3.3 and Subsection 5.5.2 to implement our core algorithms. The subsequent chapters build upon the foundations laid out in this chapter, and the presented concepts will find use within the methodologies employed in this work.

Chapter 3

Learning Language in Reinforcement Learning

This chapter, serving as the first main part of this thesis, explores the concept of language-conditioned reinforcement learning agents. We demonstrate where this research fits into the broader landscape of reinforcement learning, and its relationship to the widely adopted goal-conditioned paradigm.

Language representations of goals make it very natural for humans to interpret them, and, similar to human-to-human communication, enable effective human-robot interaction (HRI). Teaching robots how to act on language commands has been an important topic of research for several decades (Bischoff and Graefe, 1999) and continues to be a major focus of current studies (Ahn et al., 2023; Cui et al., 2023; Shi et al., 2024). In this context, the topic of language grounding (cf. Section 1.2) – how words relate to other sensorimotor experiences – has been studied thoroughly, as it allows learning of otherwise missing nuances of language (Wermter et al., 2005; Steels, 2008; Bisk et al., 2020). The integration of language and reinforcement learning holds great potential, as language provides the necessary abstraction for generalization across diverse and novel tasks (Silva et al., 2022; Spilsbury and Ilin, 2022). For instance, the motion-related word *"grasp"* could be understood by the robot in any situation, consistently referring to a very similar behavior primitive, namely reaching for something and holding it. Moreover, the desired target object, whether a *"red apple"* or a *"bottle of water"*, should not affect the fundamental understanding of the action. While there are innumerable ways a behavior could be executed or an object manipulated, these can be precisely described using a small set of words. Language is abstract, and while it may be challenging to describe precisely how a robot should execute grasping, understanding a few words can convey a wealth of information.

In contrast to unimodal models trained solely on text, such as GPT (Radford et al., 2019), a robotic agent experiences the world through interaction, exploring it while receiving rewards. The concept of sparse rewards is particularly suitable, as they allow the agent to freely explore many types of behaviors without becoming trapped in suboptimal local minima of a dense feedback signal (cf. Subsection 2.2.5 and Andrychowicz et al. (2017)). Another significant advantage of language, beyond its interpretability, is its compositionality. Words are composed of characters, and sentences are made up of words. An intermediate representation of words could be the concept of tokens, which are multiple

characters. In the remainder of this thesis, we exclude this idea of tokens used by many well-known models in the domain of natural language processing (NLP) (Devlin et al., 2019; Radford et al., 2019), and focus on whole words. Altering a single character in a word can result in a completely different semantic meaning, similarly to changing a word in a sentence. Current research highlights that it is not straightforward to train, especially RL agents, to represent language goals in a compositional manner (Spilsbury and Ilin, 2022). Such systems often struggle with systematicity, mapping entire sentences to specific behaviors without leveraging individual word meanings for generalization. Compositional understanding of words, while effortless for humans, remains a significant challenge for learning machines (Hupkes et al., 2020). It requires sophisticated algorithmic designs that incorporate inductive biases to foster compositional language learning in agents (Oh et al., 2017; Andreas et al., 2017). For instance, understanding that a *"red apple"* is similar to a *"red tomato"* in terms of its color, while recognizing that this similarity does not extend to other properties, presents a notable challenge. In such a case, similarity along one feature dimension does not imply a general similarity. Assuming an apple is much firmer than a tomato, the robot would need to adjust its gripper's forces accordingly when grasping either item. Furthermore, a *"yellow tomato"* remains a tomato and requires the same careful handling regardless of its color, which such a system would need to explore from the ground up, despite the present experiences with the *"red tomato"*. Numerous works explore the notion of language grounding, which enhances generalization ability (Chaplot et al., 2018; Hanjie et al., 2021) and learning performance (Sodhani et al., 2021). A very important problem in this area is the *Symbol Grounding Problem* by Harnad (1990), which addresses how words acquire their meaning, as discussed in Section 1.2. For humans, grounding occurs through various sensory experiences: olfaction, vision, audition, touch, taste, and actions, all in the context of the words we hear or have in our mind. The grounding problem states that meaning cannot be derived solely from words; it must be grounded in other sensory experiences and cannot be, in terms of the meaning, words all the way down (Harnad, 1990). Our reinforcement learning framework provides an appropriate setup for language grounding through sensorimotor experiences driven by a scalar reward signal, as the agent autonomously explores and discovers feedback and word relations. In the following section, we will provide an overview of related works employing language as a type of goal representation for deep RL agents.

3.1 Related Work

The upcoming section highlights current research in language-conditioned reinforcement learning, with a specific focus on language grounding.

Language in RL is a rapidly evolving field of research with increasing relevance. Current works on language in robotics aim to utilize RL as an end-to-end approach for instruction learning, with many prominent robotics researchers publishing in this area (Jiang et al., 2019; Tellex et al., 2020; Lynch and Sermanet, 2021). When not used as an end-to-end approach, concepts such as value functions, reward functions, and Bellman equations are, for instance, applied in imitation learning (Lynch and Sermanet, 2021; Shridhar et al.,

2022). The recent review by Luketina et al. (2019) outlines **two roles** language can play in an RL setting. **First**, language could be used to provide **additional contextual information** and has an assisting role. For example, a robotic agent could be supplied with additional information (metadata) about objects that are more suitable to fulfill specific tasks and which objects to avoid because they are too fragile (Narasimhan et al., 2018; Sodhani et al., 2021). The **second role**, which is the primary focus of this thesis, is **instructive language** (Hermann et al., 2017), describing the overall task the agent needs to complete. Unlike assistive language inputs, instructive language is essential and cannot be ignored by the agent.

A specific area of research, that is inherently related to RL as an embodied line of machine learning research, is the concept of grounding. Grounding language refers to the process of learning corresponding relationships between language and other environment features that the agent can sense. Research on grounded language is crucial for overcoming the limitations of current unimodal systems (Radford et al., 2018; Radford et al., 2019).

In the work of Hermann et al. (2017), the authors instruct an RL agent to solve tasks involving picking up objects within a 3D world called *DeepMind Lab* (Beattie et al., 2016). As the agent observes the world through a camera view, it enables grounding language in visual pixel inputs. However, picking up the task objects involves no real physical interaction, because it is simplified by colliding with the desired object lying on the ground. Their work is one of the first language-conditioned RL settings that showed how an agent could learn to relate concepts of language and perception based on a reward signal only. Subsequent work from DeepMind extended the multimodal capabilities through auxiliary tasks (Hermann et al., 2017; Hill et al., 2019), a dual memory-based approach (Hill et al., 2021), and language pretraining (Hill et al., 2019).

In the work of Chaplot et al. (2018), the authors investigate another method for integrating multiple modalities using a gated-attention mechanism. In short, the attention approach allows their language embedding to, in a way, modify the vision embedding, enhancing the visual perception of task-relevant objects based on the language goal. Furthermore, they demonstrate that specific object-word combinations result in distinct visual attention patterns for different object types, colors, and size attributes, providing a method of multimodal fusion with zero-shot task generalization capabilities.

Another publication closely related to this thesis is the work on *decstr* by Akakzia et al. (2021) (cf. Section 1.4). The authors investigate a particular type of semantic representations that eases the language grounding by mapping the spatial relationships of three objects on a table to a semantic encoding. The encoding represents spatial properties of the objects with binary values, indicating if two objects are close to each other or if object A is on top of object B. Through intrinsic motivation, this mapping is learned by exploring all possible spatial configurations. A social partner, implemented as a feedback mechanism by the environment, provides the language representations for the observed outcomes. A strength of their approach lies in the self-generated semantic goals. Due to the abstract nature of language, there are often multiple object arrangements that satisfy a given language instruction (see Subsection 2.2.5). Their approach enables the agent to generate different semantic representations, based on the external language goal, which the agent then tasks itself to achieve. This allows the agent to attempt various semantic goal

representations, for instance, when the first attempt fails. A limitation of this work is the requirement to manually define the semantic representations with respect to the objects in the world upfront, making it challenging to transfer to novel setups with more objects and a different environment setting.

Another line of research on language-conditioned agents utilizes hierarchical RL approaches (Eppe et al., 2022). In hierarchical RL, problems are subdivided into smaller subproblems which are easier to explore and consequently easier to solve (Eppe et al., 2019; Röder et al., 2020). Jiang et al. (2019) employ a hierarchical approach with two policies: one for the internal goal instruction generation (high level) and another for instruction following (low-level). Language in their case provides an intermediate representation to decompose long and complex instructions into temporally simpler and shorter ones. For example, given the instruction *"remove the dishes and clean the table"* as an overall goal description, a technique of HRL would be to decompose it into two subgoals like *"remove dishes from table"* and *"clean the table"*. Oh et al. (2017) present another hierarchical approach, employing a meta-controller (their high-level policy) to map instructions and visual inputs to subtasks. An internal routine within the meta controller tracks the progress of the proposed subtask and can suggest a new one at any time, interrupting the current task execution. In case of an interruption, an instruction memory keeps track of those tasks that are resumed at a later stage. The actual execution of a task is based on parameterized skills, essentially a multitask policy conditioned on the proposed goal representation of the subtask. Using a hierarchy in RL makes solving complex and long-horizon tasks more feasible. Language in this context is a valuable representation that is both interpretable and inherently compositional (Mirchandani et al., 2021). Compositionality allows reusing a couple of phrases or words for many high-level problem definitions.

A more recent line of research applies RL methods with language through the use of LLMs for robotic control. In the work of *SayCan* by Ahn et al. (2023), the authors use an LLM as a backbone for the high-level planning. Given an instruction and with appropriate prompting, the LLM responds with a structured language directive and lists subgoals that need to be fulfilled in a specific order. The predefined skills of the robot, either preprogrammed or learned via imitation learning, need to be paired with the proposed subgoals and executed. To quantify the appropriateness of certain subgoals and therefore influence the plan proposed by the LLM, the authors integrated a value function that incorporates additional modalities, like vision. This allows the system to assess, *e.g.*, whether a subgoal should involve searching for an item or simply grabbing it if it is already in the robot's field of view. Apart from skill learning, the authors claim the system grounds its generated action plans with the help of the value function's predictions, guided by visual and proprioceptive inputs. Based on the work of *SayCan*, many subsequent works attempt to utilize the semantically rich representations of LLMs to implement instruction-following agents (Huang et al., 2023b; Huang et al., 2023a; Liang et al., 2023).

3.2 Background

This section provides additional background on language-conditioned reinforcement learning as a special case of goal-conditioned reinforcement learning. Furthermore, we introduce fundamental ideas on processing language inputs and integrating them with state information for decision-making.

This section highlights the relationship between goal-conditioned and language-conditioned reinforcement learning, complementing the previous explanation of the MDP extension with goals in Subsection 2.2.5. For this, we present the formalism that we will use throughout the remainder of this thesis and introduce the main components for processing language goals in a reinforcement learning setting.

3.2.1 Goal-Conditioned RL

The reward function of a goal-conditioned MDP is defined as a mapping $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \rightarrow \{-1, 0\}$, which in our case is a binary function defining a sparse reward. The more general setting was already discussed in Subsection 2.2.5. We define Φ as an oracle function, usually provided by the environment, that maps states to goals (see Equation 3.1).

$$\hat{g}_t = \Phi(s_t) \tag{3.1}$$

While the desired goal g is the external goal provided by the environment or a teacher, the achieved goal $\hat{g}_t$ can be retrieved using the aforementioned function. Thus, we can compare g_t to $\hat{g}_t$ to evaluate the success. In the case of a fully observable state where the goal is defined in terms of location information, hence the Cartesian coordinates, we use the Euclidean distance and define a threshold ϵ, so that if the distance is below that threshold, a sparse reward is granted. This is described in Equation 3.2, following the usual reward function definition (Schaul et al., 2015; Andrychowicz et al., 2017), despite not all arguments being used for calculating the reward.

$$r_t = \mathcal{R}(s_t, a_t, g) = \begin{cases} 0, & \text{if } \|\Phi(s_t) - g\|_2 \leq \epsilon \\ -1, & \text{otherwise} \end{cases} \tag{3.2}$$

Implementing a reward function that only identifies the goal state is convenient, as it allows the agent to learn unrestricted behaviors without being misguided by a dense reward signal (Andrychowicz et al., 2017), thereby avoiding poor local optima. A drawback of this approach is the necessity for an oracle that knows how to derive goals from states. Setting the reward to be either -1 or 0 is a common choice in goal-conditioned RL settings and relates to the concept of shortest stochastic path problems (Sutton and Barto, 2018). The penalty of -1 for each step t encourages the agent to reach the sparse rewarding goal configuration as quickly as possible, to minimize operational cost. However, sparse reward problems are considered significantly more difficult to solve than dense ones. They also present a greater challenge in conducting sufficient exploration because the goal must be achieved by chance to receive a positive learning signal.

Defining goals as a subspace of states makes too many assumptions about the problem structure the agent needs to solve. Although they have been shown to work well for locomotion problems (Schulman et al., 2017; Röder et al., 2020), they are mostly impractical for manipulation and other complex tasks. In the next section, we will replace the goals-as-state representations with the more natural language goals.

3.2.2 Language-Conditioned RL

In the previous Section 3.1, we highlighted recent work on language-driven RL settings. Language provides a universal and more abstract representation for goals. However, instructions rarely map to single desired state outcomes, and different language goals can have similar outcomes state-wise. In Section 2.2.5, we already highlighted differences of the goal-conditioned case and other goal representations. For language in particular, there is no mapping function available like in the goal-conditioned setting (see Equation 3.1). Therefore, we cannot consider the achieved goal states for the reward evaluation. For linguistic goals g_ℓ, we replace the threshold based reward calculation with a task-specific condition function, which maps from a set of state-language-goal combinations to a Boolean value, $\mathcal{C} : \mathcal{S} \times \mathcal{A} \times \mathcal{G}_\ell \rightarrow \{true, false\}$, indicating whether the goal condition in Equation 3.3 is satisfied.

$$r_t = \mathcal{R}(s_t, a_t, g_\ell) = \begin{cases} 0, & \text{if } \mathcal{C}(s_t, a_t, g_\ell) \\ -1, & \text{otherwise} \end{cases} \tag{3.3}$$

Consider the example where g_ℓ is *"pick up the red cube"*: Every state where the agent is actually holding the red cube in the air falls within the set of desired goal states $\mathcal{S}_{g_\ell}$. In such instances, the evaluation of the condition yields true, signifying that we have reached a state s_t within the defined goal set (see Figure 2.10 from Section 2.2.5 for a visualization and further details). Implementing the condition function and specifying the desired goal state space $\mathcal{S}_{g_\ell}$ for each possible $g_\ell \in \mathcal{G}_\ell$ seems cumbersome at first glance, but actually boils down to a couple of if-conditions for each desired behavior in our setup. In a way the language goals define restrictions on the state space for which a reward is returned. To identify if the correct object is lifted up into the air, the environment's condition function must measure the height and the contact points of the gripper with respect to the desired goal object. Formally, we can describe the trajectory distribution according to Equation 2.18 as goal-conditioned and therefore language-conditioned, as in Equation 3.4.

$$p_\pi(\tau \mid g_\ell) = \rho_0(s_0) \left(\prod_{t=0}^{T-1} \pi(a_t \mid s_t, g_\ell)\, \mathcal{T}(s_{t+1} \mid s_t, a_t) \right) \tag{3.4}$$

3.2.3 Language Processing

In the previous section, we introduced the notion of language-conditioned RL and referred to the lingual goals as instructions or sentences. At the highest level, such language goals are instructions made of words, following a natural or synthetic grammar (Chevalier-Boisvert et al., 2019). Machine learning approaches and deep neural networks in particular cannot directly process sentences as strings. An additional step is necessary to process them, *e.g.*, by segmenting the sentence into individual words. Subsequently, each word is then mapped to a real-valued vector representation that can be processed by a neural network. Figure 3.1 illustrates this transformation from the text input to a word-based vector representation the neural network can work with.

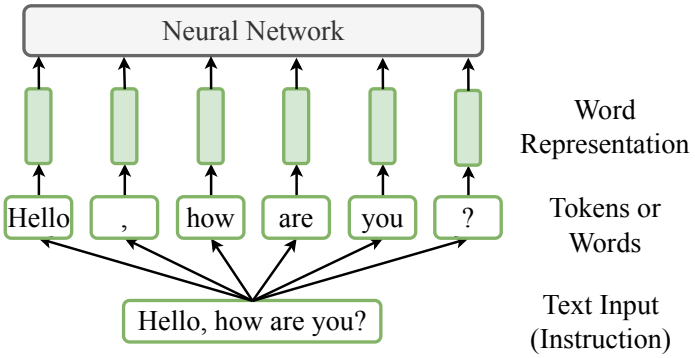


Figure 3.1: We visualize the stages of processing a language instruction: At the bottom (first stage), we have the instruction *"Hello, how are you?"*. In the second stage, the sentence gets split into individual words. Following, words are then mapped to vector representations that serve the neural network as input in the third stage.

The subsequent sections discuss how to define these word-to-vector mappings for our deep reinforcement learning setup.

Vocabulary

To represent our goal instruction, *e.g.*, *"pick up the red cube"* (see Subsection 3.2.2) as vector, we first need to establish an attribution from a set of possible words to integers. Such a mapping from a set of words to unique integer identifiers is known as a vocabulary. The vocabulary is defined as an indexed set of words according to Equation 3.5.

$$V = \{w_1, w_2, \ldots, w_K\} \tag{3.5}$$

The following properties hold true for our defined vocabulary:

- K represents the total number of words in the vocabulary
- w_i represents a distinct word in the natural language, where $1 \leq i \leq K$

These properties ensure the uniqueness of each word through a bijective (one-to-one) mapping, guaranteeing that each word consistently maps to the same integer. The vocabulary

can be readily expanded when new words appear to the agent, as per Equation 3.6, adding the new word to the set and incrementing K.

$$V = \{w_1, w_2, \ldots, w_K\} \cup \{w_{K+1}\} \tag{3.6}$$

For this thesis, we adopt an approach to NLP similar to prior works in language-conditioned RL (Hermann et al., 2017; Chaplot et al., 2018; Röder et al., 2022). Unlike the broader NLP domain, we do not apply additional preprocessing steps. Such steps typically include techniques like stemming or lemmatization, which reduce words to their base or root form, thereby solving issues like mapping *"pushed"* and *"pushing"* to different integer identifiers by instead only using the base form *"push"*. Another common approach in works on LLMs is tokenization, which involves dividing the input words into smaller units called tokens, often words or subword units (Devlin et al., 2019). This prevents allocating huge vocabularies with tens of thousands of words; instead, combinations of tokens can represent a larger vocabulary with fewer subword units. For instance, the tokens *"pu"*, *"sh"*, *"ed"* and *"ing"*, already provide components to represent our three words *"push"*, *"pushed"* and *"pushing"*, while also covering parts of many other word endings, such as *"walking"* or *"grasped"*.

Our RL setup operates with only a subset of the English language, so we do not require such advanced NLP techniques. The following sections present how we obtain the vector representations of the words that we input into our deep neural networks.

Word Embeddings

Once a vocabulary has been established, we can obtain the real-valued vector representation of each word, which we can input into our DNNs. This vector representation, known as a word embedding, captures contextual and semantic information based on the vector space it occupies. Typically, this vector space is high-dimensional, with each dimension potentially representing a latent feature relevant to the language's semantics. For instance, specific dimensions may represent syntactic classes, semantic interpretations, or contextual applications. Mathematically, if w_e represents the word embedding for a word w, it can be expressed as an n-dimensional vector, $w_e \in \mathbb{R}^n$. The dimensionality of this vector space plays a crucial role in determining the effectiveness of the task the DNN is designed to accomplish. Figure 3.2 illustrates how words are assigned to indices and their corresponding word embeddings.

There are two primary approaches to word embeddings that we consider using in the implementations of this thesis: **one-hot encodings** and **learned embeddings**.

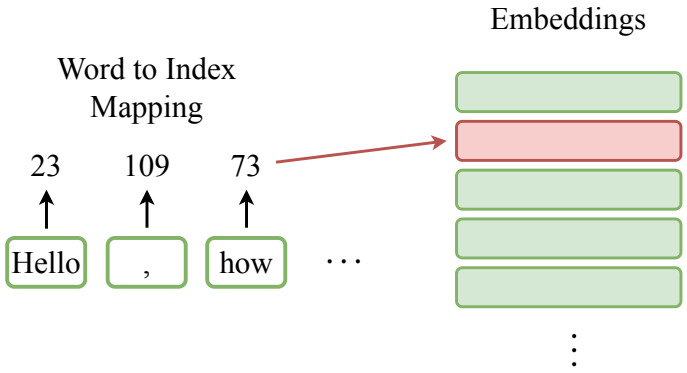


Figure 3.2: We illustrate an example of the mapping of words to their indices corresponding to a particular embedding. The word *"how"* is mapped to index 73, which retrieves its embedding representation highlighted in red.

One-Hot Encoding A one-hot encoding, also referred to as an indicator vector, represents each word in the vocabulary by a unique vector in a K-dimensional space, where $K = |V|$ is the size of the vocabulary. All dimensions are set to zero except for the one corresponding to the word's index (cf. Equation 3.7).

$$v^i = (0, \ldots, 1, \ldots, 0) \in \mathbb{R}^K \quad \text{with} \quad v^i_j = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{otherwise} \end{cases} \quad \forall j \in \{1, \ldots, K\} \tag{3.7}$$

Here, v^i represents the one-hot encoding for the word at index i, and v^i_j is its j-th component. This method, also known as the 1-of-K representation (Deisenroth et al., 2020), is the simplest form of a word embedding. It does not capture semantic similarities among words, as they are all placed equidistant within the vector space. Consequently, two words with very similar meanings are represented no differently than two completely unrelated words. Figure 3.3 provides an example of five words represented as one-hot encoding reduced to only 2 dimensions using the Principal Component Analysis (PCA) technique. Although there are words that could logically be clustered by categories like colors (*"blue"* and *"red"*) or furniture (*"table"* and *"chair"*), this encoding fails to capture these similarities.

An additional downside, aside from the issue of vector comparisons, is that the vector space grows linearly with the size of the vocabulary, making it impractical for very large language corpora.

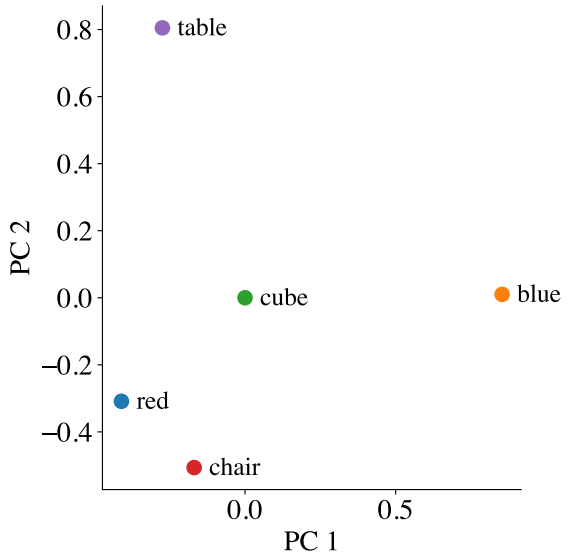


Figure 3.3: We depict the first two principal components as the result of passing the one-hot encodings for the 5 words *"red"*, *"blue"*, *"chair"*, *"table"*, and *"cube"* through the dimensionality reduction. The figure illustrates that the usual word similarities of colors and furniture are not captured. The placement of the words is distorted by the PCA; as it is difficult to visualize a 5-dimensional space in 2D.

Learned Embeddings Trainable embeddings allow for adjusting randomly initialized vector representations with backpropagation. The embeddings are parameters and considered part of the whole model θ, *e.g.*, a deep neural network, described by Equation 3.8.

$$\theta = \{\theta_{\text{network}}, \theta_{\text{embedding}}\} \tag{3.8}$$

As a result, we obtain a $K \times d_{\text{embedding}}$ matrix. Each row represents a dense vector representation of a word. In contrast to the one-hot encodings, the size of the learned embeddings does not grow linearly with the vocabulary size $K = |V|$, as this is entirely determined by the hyperparameter $d_{\text{embedding}}$. In Equation 3.9, we provide the definition, in which we have a distinct mapping for each word i to its embedding $v^i_{\text{embedding}}$.

$$v^i_{\text{embedding}} \in \mathbb{R}^{d_{\text{embedding}}} \quad \text{where} \quad i \in \{1, \ldots, K\} \tag{3.9}$$

They provide a better solution for settings with many words and, in comparisons to the aforementioned one-hot encodings, are able to capture semantic similarities by pushing those vectors in the embedding space closer together during learning. However, such representations only make sense to use when there is a rich training signal in terms of a diverse number of words, and the additional amount of parameters is justifiable to train. Training such embeddings can be challenging, as they may become too general when trained on large language datasets, yet too specialized if limited to a single domain, such as a very specific robotics setup.

In Figure 3.4 we depict the same 5 example words used for the one-hot encoding visualization (cf. Figure 3.3). This time, we encode them using the *GloVe* embeddings (Pennington et al., 2014), which are predetermined word embeddings that were pre-trained on a large text corpus and represent semantic similarities, as seen by the two color related words that form a cluster.

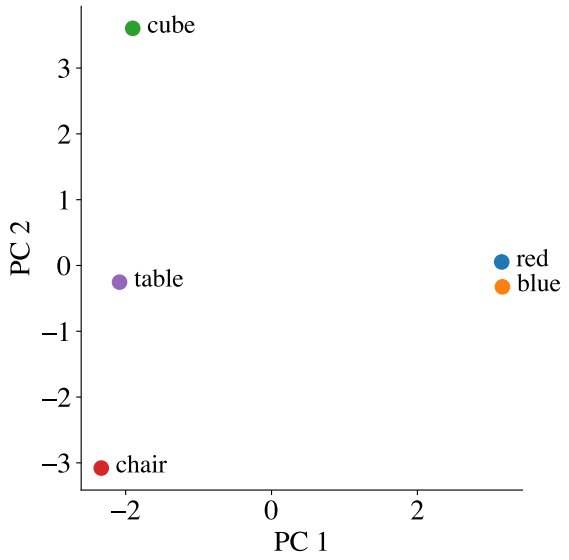


Figure 3.4: This figure depicts the first two principal components of the dense word representations, using the pre-trained *GloVe* embeddings. The 5 words *"red"*, *"blue"*, *"chair"*, *"table"*, and *"cube"* are shown after the dimensionality reduction using PCA. A cluster of colors can be located on the right side, and the furniture-related words are close in space. For this plot, we are using word vectors of size 100 with a vocabulary of 400 000 words.

Pre-Trained Embeddings

Embeddings are real-valued vector-based representations of words, phrases, texts or whole documents. Pre-trained embeddings are general-purpose embeddings resulting from unsupervised or self-supervised training and capture statistical co-occurrences of word semantics and other relationships derived from large text datasets. For example, Word2Vec uses deep neural networks to embed words based on context words, trained either by predicting a word given its context (CBOW: Continuous Bag of Words according to Mikolov et al. (2013)) or by predicting context words based on a target word (Skip-Gram from Mikolov et al. (2013)). Changing the network parameters with respect to the aforementioned tasks and extracting the emergent hidden word state representations provides the resulting embeddings, for which similar words are mapped to close points in the vector space. These representations can capture rich amounts of semantic and syntactic information, resulting in emergent properties that resemble algebraic structures, allowing operations like addition

and subtraction. For this, consider the seminal example with the words *"king"*, *"man"*, *"woman"*, and *"queen"* represented as vectors, as discussed in Mikolov et al. (2013):

$$\vec{king} - \vec{man} + \vec{woman} \approx \vec{queen}$$

Another method called Global Vectors for Word Representation (GloVe) (Pennington et al., 2014), that we already used to illustrate the capabilities of dense embeddings in the previous section (see Section 3.2.3 and Figure 3.4), does not rely on local context information of nearby words, but constructs an explicit co-occurrence matrix for the word statistics across the entire text corpus.

Other methods like Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019), follow a different approach: they generate context-dependent embeddings using the attention mechanism (Bahdanau et al., 2015) (see Section 3.2.4). This allows encoding different word meanings, hence different embeddings, based on the nuances of the surrounding text. The approach utilizes a deep neural network based on the encoder architecture of the Transformer (Vaswani et al., 2017). BERT is trained in a self-supervised manner by predicting missing pieces of a sentence, following a fill-the-blanks approach. To successfully do this, it needs to capture the context information, hence accounting for the differences in word usage within various contexts, as shown in the following example.

We illustrate the context sensitive encoding of BERT for the word *"bank"*:

(1) She walked along the river **bank**, looking for smooth stones.
(2) He went to the **bank** to deposit his paycheck.

However, for this thesis, we are interested in learning language from the bottom up and follow the notion of grounding based on the evidence from developmental psychology and cognitive science outlined in Chapter 1. Adjusting those pre-trained embeddings is often detrimental if one only uses a small fraction of words, as they could cover a vocabulary of tens of thousands (Mikolov et al., 2013) with up to a million words (Pennington et al., 2014).

Furthermore, these vectors are often of high dimensionality, with up to several hundred dimensions, like 768 in the case of the basic version of BERT (Devlin et al., 2019). In the case of our upcoming environment, which we introduce with less than thirty distinct words, this approach is overparameterized and inflated for the setup that we consider. Models like BERT, in their default configuration, are not designed to recognize small nuances in language and might be too general to be used for our robotic learning setups. In the following, we provide an example of two sentences that require a nuanced understanding of the language to identify the goal object and act accordingly:

(1) Push the **red** cube.
(2) Push the **blue** cube.

Based on this idea, in Figure 3.5, we exemplary show the PCA output of 4 sentence representations using BERT, of which two refer to colors and two to specific objects. As expected, the small color differences are not encoded appropriately. This is problematic

for an intelligent robotic agent because instructions including colors are represented more similarly (*"Push the red cube"* and *"Push the blue cube"*) than the instruction without a color (*"Push the cube"*) or even the one referring to a piece of furniture (*"Push the chair"*).

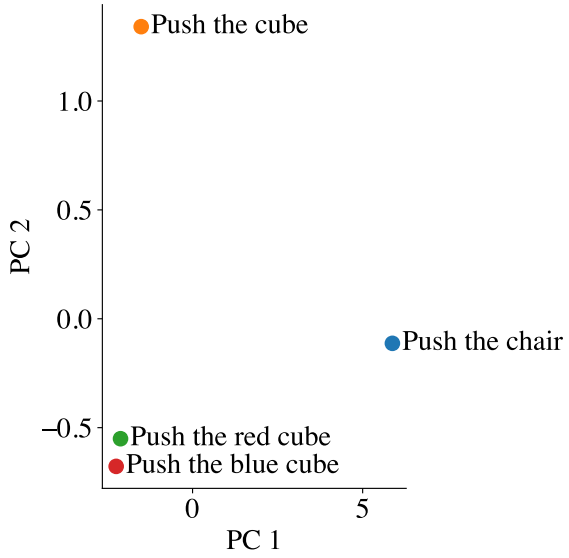


Figure 3.5: This figure illustrates the first two principal components of the word representations, using the base BERT model with 110 million parameters and word embeddings of size 768. The 4 sentences *"Push the cube"*, *"Push the red cube"*, *"Push the blue cube"*, and *"Push the chair"* are visualized after the dimensionality reduction using PCA. As expected, BERT is not able to encode the color differences appropriately, which could make it challenging for an RL agent to differentiate these small nuances, which might be crucial for obtaining the sparse reward.

However, we are aware of pre-trained embeddings that have been shown to work well for instruction-following (Lynch and Sermanet, 2021; Sodhani et al., 2021), but they often require additional algorithmic effort and fine-tuning. Overall, we are aiming for *cognitively* plausible language learning that considers the language modality right from the start to achieve a grounded understanding. Pre-trained language embeddings are in stark contrast to what we know about language development and are often too general, considering corpora vastly different from what appears in robotic problems.

3.2.4 Representing the Instructions

Deep neural networks require the inputs to be real-valued vectors. In the previous sections, we explained how to translate words to indices of a vocabulary and convert those indices into vector representations. We arrive at representing words as either one-hot encodings (Jiang et al., 2019) or dense vectors, while the latter could be trained simultaneously with the model parameters or provided as pre-trained embeddings (Hill et al., 2021). For

now, only BERT considers vector representations for entire sentences, which is a suitable approach for our goal instructions but comes with downsides. To remain flexible and allow the model to exploit the compositional word structure of a lingual goal, we require methods to process sequences of word vectors. In Section 2.1, we introduced the notion of input nodes and how their dimension needs to be equal to that of the input data. In the case of a sentence of length L, the one-hot encodings yield a matrix $\mathbb{R}^{L \times K}$. For the learned word embedding, this results in a $\mathbb{R}^{L \times d_{\text{embedding}}}$ matrix. Conservative approaches to language processing that sum up the individual word vectors of a sentence or apply average pooling lose a lot of information. For the remainder of this work, we exclude such simple mechanisms and prefer to use techniques that keep the order information and are not dominated by more frequent words. In the following sections, we provide three deep neural network layer approaches to work with sequences that are suitable for our language-conditioned RL setup and the latter action correction setting (see Chapter 5).

Multilayer Perceptron

To process sequential data with a multilayer perceptron (MLP), it requires removing the additional sequential dimension. A common approach is to flatten the $\mathbb{R}^{L \times D}$ sequence into a long vector $\mathbb{R}^{L'}$, where $L' = L \cdot D$. Therefore, we build the MLP in a way that it can process the sentence with L' input nodes. However, flattening the input sentence into a single vector comes with a loss of sequential information, although the activations of individual words influence each other from the first hidden layer onwards. In addition, using MLPs for sequence processing comes with the limitation of a fixed input size and requires proportionally many input units if longer sentences appear during training. For sentences that are shorter than the predefined length L', we require placeholder embeddings that actually carry almost no information and waste computational resources. In Figure 3.6, we illustrate how an MLP processes a sequence of word embeddings, in this case for the goal instruction *"Push the green cube"*. We depict the word embedding as small vectors of size 2, hence requiring 2 input nodes for each word to be processed. The interaction between the words only happens at subsequent layers to the input layer, like the depicted first hidden layer. Otherwise, there is no explicit exchange of information, such as the order or neighboring relations involved.

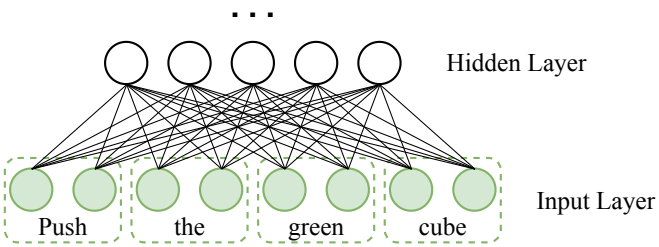


Figure 3.6: Illustration of an MLP processing the flattened sentence as a large vector of the individual word embeddings of size 2. The sentence length is $L = 4$, hence the MLP input requires 8 input nodes. The words exchange information at the level of hidden activations in the subsequent layers, but otherwise lose any type of sequential information.

Recurrent Neural Networks

In this section, we are going to discuss a special type of DNN architecture designed for processing sequential data, known as a recurrent neural network (RNN). RNNs can be considered a special kind of MLP that processes a sequence of inputs, such as words in a sentence or frames in a video. For each step, the network processes an element of the input sequence x_t and generates an output y_t, while passing on its hidden state h_t to the next processing step. That same neural network, essentially a copy with identical parameters, processes the hidden state h_t of the previous step and the next input x_{t+1} of the sequence, to output the following prediction y_{t+1} and hidden state h_{t+1}. Figure 3.7 depicts this with a single neuron (left) containing a loop connection alongside the unrolled version for two illustrated steps (right).

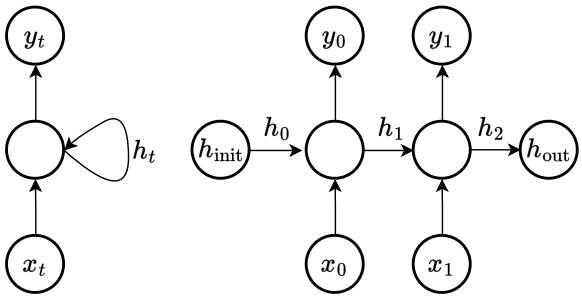


Figure 3.7: To the left, we illustrate the idea of an RNN as a single neuron with a loop connection, transmitting the previous hidden state to the next processing step as additional input. On the right, we show an unrolled version of that neuron, propagating the hidden state to subsequent processing steps.

The memory state, also called the context vector, is of a predefined size, hence $h_t \in \mathbb{R}^d$. The initial hidden state h_{init} is usually initialized with zeros or random values, providing default values for the memory of the RNN to start with. The last hidden state h_{out} provides the output of the network, containing the contextual information for the entire

input sequence. In essence, the hidden state functions as a kind of memory, potentially storing information for numerous steps. This component enables the network to remember dependencies across parts of the input sequence that are necessary for, *e.g.*, translation (Sutskever et al., 2014) and language modeling (Bengio et al., 2003). Processing a sentence word by word allows the network to capture order information while also keeping track of more specific information, such as the current gender of the subject, to correctly output a translation. Unlike the MLP used for language processing, the hidden activations of words in RNNs directly influence each other in a sequential manner. Furthermore, RNNs do not have a limited input sequence size and are computationally efficient for processing sequences of varying lengths. We revisit our previous example sentence *"Push the green cube"*, processed by an RNN as illustrated in Figure 3.8. Each word is represented by an embedding of size 2. The illustrated RNN processes each word individually, starting with the initial memory state h_0 and ultimately outputs the final state h_4, which contains a compressed representation of the sentence.

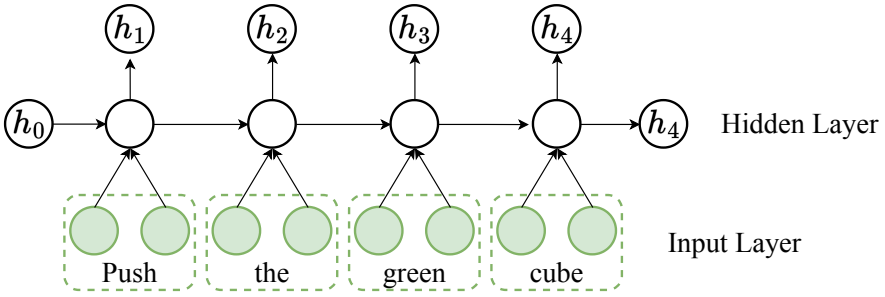


Figure 3.8: This figure demonstrates the information flow of a recurrent neural network, that processes the instruction *"Push the green cube"* word by word. The intermediate connections depict the unrolled network in time, propagating the previous hidden state until the final state h_4 is returned.

Compared to MLPs, RNNs are slower in processing time series, as they sequentially have to process one input element at a time. However, they offer greater flexibility in terms of input length, as current frameworks implement smart mechanisms to computationally efficiently process many sequences of different lengths. A challenge that arises when unrolling an RNN to apply a specific type of backpropagation, called *backpropagation through time* (Goodfellow et al., 2016, Section 10.2), is that gradients tend to vanish or explode. Due to the sequential repetition and derivative operations, the magnitude of the gradients can change drastically. In the following paragraph, we describe a specific RNN architecture that mitigates the problem of vanishing and exploding gradients. Furthermore, it is the primary architecture that we will utilize throughout the rest of this thesis because of its established use in NLP (Cho et al., 2014) and RL (Hafner et al., 2021), and due to its extended and efficient memorization capabilities.

Gated-Recurrent Unit A particular type of RNN that addresses the problem of vanishing gradients is the Gated-Recurrent Unit (GRU) (Cho et al., 2014). It employs gating mechanisms that enable more efficient control over the flow of information through the network. The concept of gating itself was introduced by the work on the Long-Short Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997). The LSTM introduced *input*, *forget*, and *output* gates to allow the network to learn when and how to update information within the hidden state. This enables the unit to capture dependencies of different time scales. The GRU essentially implements a more efficient version of the LSTM by replacing the *input* and *forget* gates with a single *reset* gate and an *update* gate. Compared to the LSTM, it has fewer parameters and performs similarly well on most tasks.

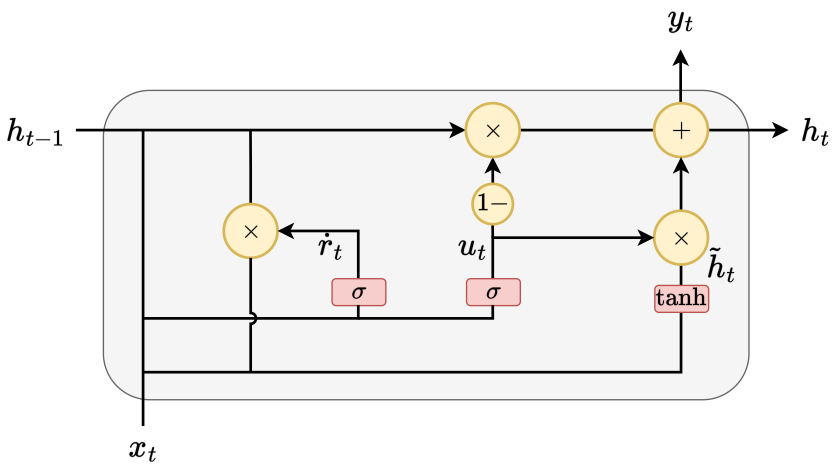


Figure 3.9: Diagram of a Gated Recurrent Unit (Cho et al., 2014), showing the update gate u_t, reset gate $\dot{r}_t$, and current memory content interaction, going from h_{t-1} to h_t. The activation functions are sigmoid functions visualized by σ and the hyperbolic tangent, shown as tanh (see Figure 2.3). The illustration is based on Olah (2015).

Figure 3.9 illustrates the structure of a GRU. It implements a reset mechanism that determines which part of previous hidden state activation h_{t-1} to remove. The forget vector $\dot{r}_t$ is the result of the dot-product between the reset weights W_r, and the concatenation of the past hidden state and the current input $[h_{t-1}, x_t]$. The full reset gate computation is described in Equation 3.10.

$$\dot{r}_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right) \tag{3.10}$$

Utilizing the sigmoid function, the reset gate vector $\dot{r}_t$ determines whether a feature of the last hidden state h_{t-1} should be forgotten, (output dimension is close to 0) or kept (values close to 1). We calculate the result by the element-wise product $\dot{r}_t \odot h_{t-1}$. With the update gate, the GRU decides which features of the last hidden state h_{t-1} should be forwarded to the next hidden state h_t and which features of the new input should be incorporated. Similar to the reset gate, the update gate employs weights W_u, for which we provide the description in Equation 3.11.

$$u_t = \sigma\left(W_u \cdot [h_{t-1}, x_t]\right) \tag{3.11}$$

According to Figure 3.9, Equation 3.12 involves the parameters W_h and tanh, determining the candidate hidden state $\tilde{h}_t$, given the influence of the reset gate and the current input. Like the reset gate, the sigmoid output determines the influence of the resulting vector u_t, as considered in Equation 3.13. The output y_t is identical to the newly computed hidden state h_t (cf. Figure 3.7).

$$\tilde{h}_t = \tanh\left(W_h \cdot [\dot{r}_t \odot h_{t-1}, x_t]\right) \tag{3.12}$$

$$h_t = (1 - u_t) \odot h_{t-1} + u_t \odot \tilde{h}_t \tag{3.13}$$

The overall structure of the GRU allows it to effectively capture both short-term and long-term dependencies, using fewer parameters and therefore being more computationally efficient than the LSTM.

Self-Attention

A limitation of RNNs, like the GRU, is that they process inputs sequentially and need to compress all relevant information into a single hidden state h_t. There is no guarantee of retaining information about words mentioned earlier in the sequence. Furthermore, the meaning of the individual word embeddings is fixed and does not directly take into account the presence of other words. However, different parts of a sentence are typically more useful than others, and realizing word correlations enhances understanding of the sentence as a whole.

An approach that improves on that is the attention mechanism, which was first used in the work of Bahdanau et al. (2015). Attention implements a neural network layer that allows different parts of the input to be considered at different points in time. The type of attention that we regard in the remainder of this thesis is the self-attention mechanism, also known as intra-attention. The work of Vaswani et al. (2017) introduced it as an integral part of the well-known Transformer architecture. Self-attention operates by calculating attention scores as the relevance of word-to-word dependencies within a sentence. These scores are used to determine how to represent words in the context of other word appearances. This results in word representations generated on the fly as a linear combination of contextualized word embedding information. We previously mentioned these types of contextual embeddings in Section 3.2.3 with the method of BERT (Devlin et al., 2019). For instance, a sentence containing the words *"river"* and *"bank"* should have a very different meaning than a sentence with the words *"bank"* and *"money"*. This was previously not possible to achieve with RNNs, because they cannot alter the fixed meaning of a word, and as a result, this had a great impact on the research field of NLP (Vaswani et al., 2017).

In the following, we provide an example, utilizing the dense word embeddings of Section 3.2.3. The scaled dot-product attention (Vaswani et al., 2017) maps with the help of parameters $W^Q \in \mathbb{R}^{d_{\text{embedding}} \times d_q}$, the individual word embeddings w_i to so-called query vectors $q_i \in \mathbb{R}^{d_q}$, where $i, j \in \{0, \ldots, L\}$. The index i defines our current word w_i that is

fixed, and j is the index for all the other words w_j that we compare to. In addition to this query vector, we map the embeddings to the key $k_j \in \mathbb{R}^{d_k}$ and value $v_j \in \mathbb{R}^{d_v}$ representations by using weights $W^K \in \mathbb{R}^{d_{\text{embedding}} \times d_k}$ and $W^V \in \mathbb{R}^{d_{\text{embedding}} \times d_v}$, respectively. Usually, d_q and d_k are chosen to have the same size.

$$q_i = w_i W^Q \quad k_j = w_j W^K \quad v_j = w_j W^V \tag{3.14}$$

With the results of these linear transformations outlined in Equation 3.14, we calculate the dot-product between the queries and the keys to identify a word correspondence, which we scale by dividing them by $\sqrt{d_k}$ to obtain the scaled, unnormalized attention, as shown in Equation 3.15.

$$e_{ij} = \frac{q_i \cdot k_j}{\sqrt{d_k}} \tag{3.15}$$

Applying the softmax (see Equation 2.9) to the scaled attention results in our scaled and normalized attention scores, as described in Equation 3.16.

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{l=0}^{L} \exp(e_{il})} \tag{3.16}$$

Finally, we weight the different values v_j to read out the mixture of information as an attention-weighted sum of the individual value projections. In Equation 3.17, the weighted output is calculated for each word i, hence receiving one contextualized word representation $y_i \in \mathbb{R}^{d_v}$ per input word as output.

$$y_i = \sum_{j=0}^{L} a_{ij} v_j \tag{3.17}$$

Figure 3.10 illustrates the scaled dot-product attention based on the version using the softmax function (Bahdanau et al., 2015). We depict, as in the previous examples in Figures 3.6 and 3.8, the word embeddings as a vector of size 2 for the example sentence *"Push the green cube"*. We highlight only how the **first contextualized representation** y_0, hence $i = 0$, of the word *"push"* is calculated.

The resulting contextualized vector, $\mathbb{R}^{L \times d_v}$, still needs to be flattened as with the input in the case of the MLP.

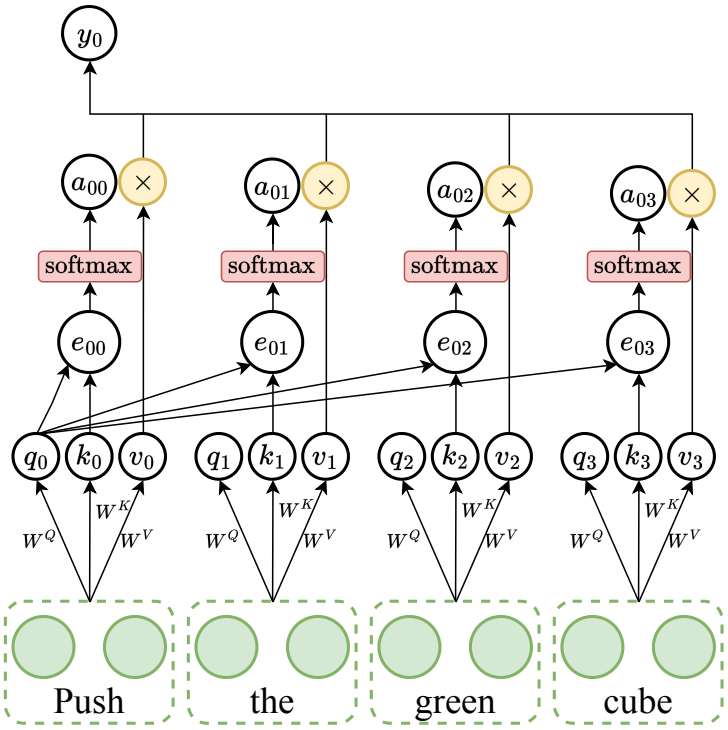


Figure 3.10: We illustrate the calculation flow of the scaled dot-product attention layer. This example considers only the first word, with $i = 0$. The instruction *"Push the green cube"* is provided, along with word embeddings of size 2. The illustrated calculations of e_{ij}, a_{ij}, and y_i follow those outlined in Equations (3.15) to (3.17), respectively.

3.3 Language-Conditioned Soft Actor-Critic

The ensuing section describes the thesis's core algorithm, previously introduced in Röder et al. (2022).

Central to the experiments of this thesis is the previously introduced SAC algorithm (Subsection 2.2.7). We presented the general goal-conditioned setup in Subsection 2.2.5 and Subsection 3.2.1 as an approach to precisely specify what an agent should achieve. This was followed by the concept of language-conditioned RL in Subsection 3.2.2. Extending the algorithm to consider the additional language context is straightforward to implement. In this section, we introduce our baseline model, termed Language-Conditioned Soft Actor-Critic (LCSAC), which serves as the foundational architecture for the remainder of this thesis. Similar formulations have been employed in Akakzia et al. (2021), Sodhani et al. (2021), and Silva et al. (2022). With Equation 3.18, we present the objective that now incorporates the episodic language goal g_ℓ, sampled from the initial language goal distribution, $g_\ell \sim \rho_{g_\ell}$.

$$J(\pi) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t, g_\ell), g_\ell \sim \rho_{g_\ell}} \left[\sum_{t=0}^{T} \gamma^t \left(\mathcal{R}(s_t, a_t, g_\ell) - \alpha \log(\pi(a_t \mid s_t, g_\ell)) \right) \right] \tag{3.18}$$

Both the actor and the critic have an additional neural network module, a language encoder, added to their architecture. The language encoder could be any of the proposed methods for processing a sequence of word embeddings introduced in the prior Subsection 3.2.3. Previously, the actor or critic only worked with state vectors, and in the case of goal-conditioned RL, with a very similar kind of state-based goal vector that was simply concatenated. Combining the state vector and the output of the language encoder can be considered a multimodal input, with data coming from two different types of modalities. Let us consider utilizing a state encoder, which is actually an MLP that outputs a hidden representation of a state h_s. Additionally, the language encoder outputs a hidden representation of the goal instruction h_ℓ. There are numerous ways to process different kinds of modalities with DNNs. A straightforward approach is to concatenate both representations $[h_\ell, h_s]$, which then serve as input to a subsequent fully connected layer, as done by many recent works (Hermann et al., 2017; Narasimhan et al., 2018; Hill et al., 2021; Röder et al., 2022). There are many techniques to implement multimodal fusion, such as the Hadamard product (Chai et al., 2014), gated-attention (Chaplot et al., 2018), or the cross-attention approach, which we will not explore in this thesis since they are more commonly used in vision-based RL or other supervised learning domains. Figure 3.11 illustrates how both actor and critic process the language goal, the state information of the environment, and, in case of the critic the action, to generate an action or Q-value as output.

In Algorithm 2, we provide the pseudocode for **LCSAC**. As already mentioned, we continue to use the equations of SAC defined in the previous Subsection 2.2.7, as the only part that changes is the additional goal-conditioning of the policy and the Q-function, which one can simply assume in the general case to be part of the observed state.

Algorithm 2 Language-Conditioned Soft Actor-Critic (Röder et al., 2022). Equations from SAC (Subsection 2.2.7) are reused since the goal can be considered part of the state.

θ	actor parameters	ϕ_i	critic parameters for $i = 1, \ldots, N$
$\bar{\phi}_i \leftarrow \phi_i$	target critic parameters	$\mathcal{D}$	off-policy replay buffer
α_π	actor learning rate	α_Q	critic learning rate
α	entropy temperature	α_{entropy}	temperature learning rate

for episode $= 0, \ldots, E$ **do**
  Sample initial state $s_0 \sim \rho_0$ and language goal $g_\ell \sim \rho_{g_\ell}$
  for each environment step $t = 0, \ldots, T-1$ **do**
    Sample action $a_t \sim \pi_\theta(\cdot \mid s_t, g_\ell)$
    Take environment transition $s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t)$
    Compute language-conditioned reward $r_t \leftarrow \mathcal{R}(s_t, a_t, g_\ell)$
    Store transition in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1}, g_\ell)\}$
  end for
  for each gradient update step **do**
    Sample mini-batch of transitions $\mathcal{B} \sim \mathcal{D}$
    for each critic $i = 1, \ldots, N$ **do**
      Update language-conditioned critic Q_{ϕ_i} by taking a step of gradient descent
      $\phi_i \leftarrow \phi_i - \alpha_Q \nabla_{\phi_i} L(\phi_i, \mathcal{B})$ according to Equation 2.46
      Update target networks $\bar{\phi}_i$ following Equation 2.43
    end for
    Update language-conditioned actor π_θ by taking a step of gradient descent
    $\theta \leftarrow \theta - \alpha_\pi \nabla_\theta L(\theta, \mathcal{B})$ according to Equation 2.49 based on Equation 2.47
    Update temperature α by taking a step of gradient descent
    $\alpha \leftarrow \alpha - \alpha_{\text{entropy}} \nabla_\alpha L(\alpha)$ according to Equation 2.42
  end for
end for

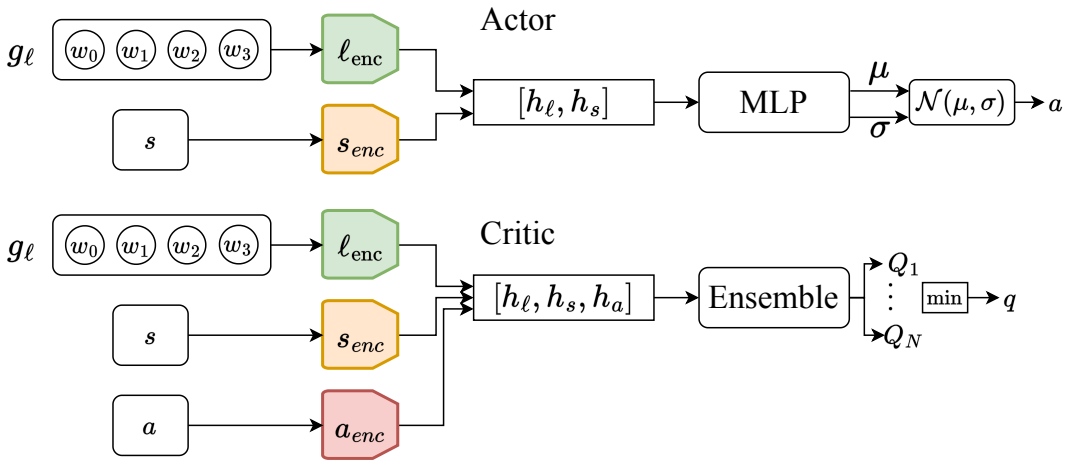
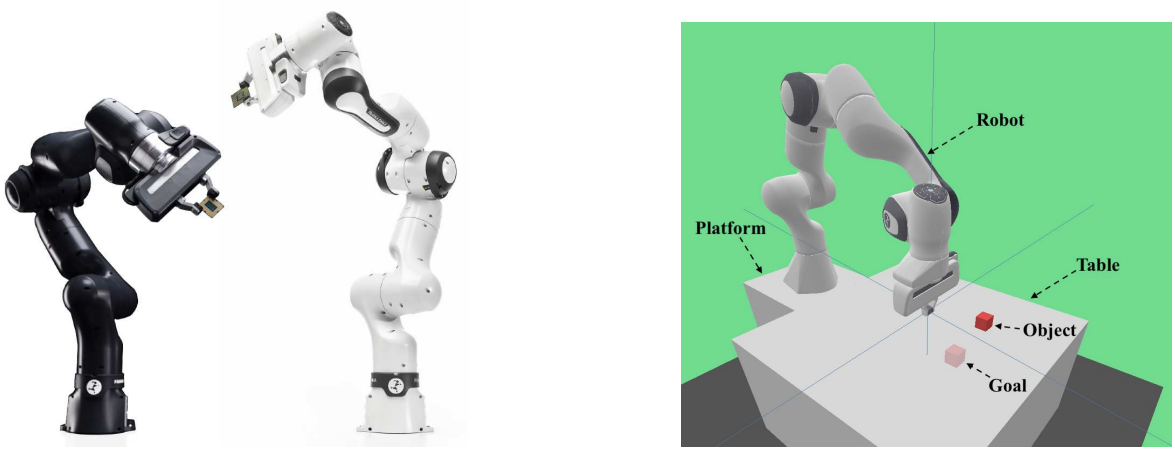


Figure 3.11: Illustration of our language-conditioned soft actor-critic architecture, receiving as input a sequence of words $g_\ell = (w_0, w_1, w_2, w_3)$, the state of the world s, and, in the case of the critic, also the action a. All of these inputs are encoded by a language encoder ℓ_{enc}, a state encoder s_{enc}, and an action encoder a_{enc}. At the top, the actor is illustrated concatenating the hidden representations of both modalities $[h_\ell, h_s]$, based on which it parameterizes a Gaussian distribution $N(\mu, \sigma)$, from which an action a is sampled. The critic architecture additionally encodes the current action a to concatenate the three hidden states $[h_\ell, h_s, h_a]$, based on which it regresses the state-action value q. An ensemble of Q-functions Q_i is employed, of which the minimum is taken as actual output.

3.4 Language Robotics Environment

This section introduces our benchmark for language-conditioned reinforcement learning tasks, which are especially designed for instruction following, learning in hindsight, and dynamically changing goals.

In our previous work Röder et al. (2022), we presented the LANguage RObotics (LANRO) environment, to evaluate RL approaches in a challenging robotics setting using language as goal representation. The implementation was inspired by the *panda-gym* benchmark (Gallouédec et al., 2021), and we employed it as a seed for our major language extension. The robot used in our environment is a Franka Emika Panda robot with 7 degrees of freedom (DOF) (Haddadin et al., 2022). It is a simulated version of the real robot, as depicted in Figure 3.12a. We use the original *Unified Robot Description Format* URDF files provided by the Franka Emika GmbH, obtained from GitHub[1]. For simulation purposes, we rely on *PyBullet* (Coumans and Bai, 2016–2021) as a mature and widely-used physics engine. In our environment, the robot is placed in front of a table and mounted on a stationary platform as depicted in Figure 3.12b. The state of the robot contains the end-effector's positional information, its velocity, its roll pitch yaw values, and angular values and velocities of the joints, forming a subset of the MDP state space $\mathcal{S}_{robot} \subseteq \mathcal{S}$.



(a) Franka Emika Panda

(b) Simulated robot in **LANRO**

Figure 3.12: To the left, we have a picture of the real Franka Emika Panda that we use in our virtual **LANRO** benchmark, shown to the right. The real robot image is taken from the official homepage at `https://franka.de` (Visited November 20, 2024).

3.4.1 Language Tasks

All language-based tasks come with a variable number of objects (2 to 3), different colors, various shapes, different sizes, and different masses. The color, shape, and size proper-

[1] `https://github.com/frankaemika/franka_ros`

ties are directly observable by the agent, while the mass needs to be perceived through interaction.

Objects are randomly placed on the table within an area of 30cm × 30cm with a minimum distance between them defined by a threshold $\epsilon_{obj_{xyz}}$. An individual object has a default height of 0.4 cm. We consider all objects on the table $O = \left\{obj^1_{xyz}, obj^2_{xyz}, \dots obj^N_{xyz}\right\}$ and inspect their unique pairs $\binom{O}{2}$, where an object in this case is a Cartesian vector $obj^i_{xyz} \in \mathbb{R}^3$. We measure the distances of each pair and require that all of them are greater than our predefined threshold, *i.e.*, $\forall\{o, o'\} \subset O,\ o \neq o',\ \|o - o'\| \geq \epsilon_{obj_{xyz}}$.

In addition to the robot's state, each object i has state information as a combination of its Cartesian coordinates, the rotation, velocity, and angular velocity: $s_{\text{obj}_i} = (o_{xyz}, o_{rot}, o_{vel}, o_{\alpha\text{vel}})$. Similar to the state of the robot being a subspace of the state space, one could consider a separate object space $\mathcal{S}_{\text{obj}}$ containing all the aforementioned information. Finally, it follows that the union of $\mathcal{S}_{robot}$ and $\mathcal{S}_{\text{obj}}$ results in the state space $\mathcal{S}$ for our fully observable Markov decision process.

Similar to prior works (Plappert et al., 2018; Gallouédec et al., 2021), we set a limit of $T = 50$ timesteps for each episode. In the upcoming paragraphs, we are going to outline the individual tasks of **LANRO** and visualize how the robotic agent might solve them.

Reach In the reach task, the agent must identify and gently touch the designated goal object. All objects, including the goal object, are placed in the 30cm × 30cm area in front of the robot. To introduce variation, we incorporate the verbs *"reach"*, *"touch"*, and *"contact"* as part of the goal instruction, allowing for distinct task recognition. The reward is determined by a condition function that utilizes the simulation's internal mechanisms to measure the contact between the robot's end-effector and the desired goal object. We implement an additional constraint to maintain the positions of non-goal objects, allowing them to move no more than 2.5cm from their initial locations. Figure 3.13 illustrates, from left to right, the agent's successful execution of reaching the green goal object.

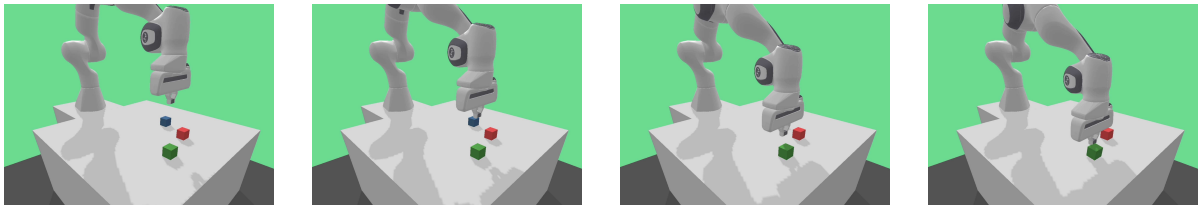


Figure 3.13: We illustrate the reach behavior (from left to right). The agent is tasked with reaching the *green* object.

Push The push task implements the reward condition to move the goal object a certain distance. For a selected goal object, we randomly determine how much it should be moved by uniformly sampling a distance from $U(2.5\text{cm}, 7.5\text{cm})$, encompassing both small and large push motions. The agent has the freedom to move an object in any direction. However, we include an additional constraint to ensure the object remains on the table by maintaining its height (z-coordinate) close to the initial value, preventing the agent from pushing the

object off the table or lifting it. Furthermore, we enforce that non-goal objects stay in place, and therefore not be moved more than our predefined threshold like in the reach task. The action verbs used in this task are *"push"*, *"move"*, and *"shift"*.

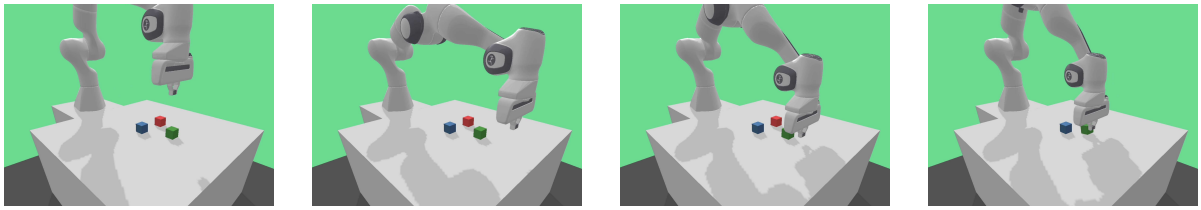


Figure 3.14: The figure demonstrates the push behavior, showing the agent randomly selecting a direction to move the *green* goal object (from left to right).

Grasp In the grasp task, the agent must position the goal object within its gripper and maintain a hold. Technically, we detect successful grasping by verifying contact between the object and both gripper fingers. While this does not prevent the agent from lifting the object, it is not a required condition, simplifying the problem for the agent to solve. For the non-goal objects, similar restrictions hold as mentioned in the reach and push setting. The verbs used in this task are *"grasp"*, *"grip"*, and *"grab"*.

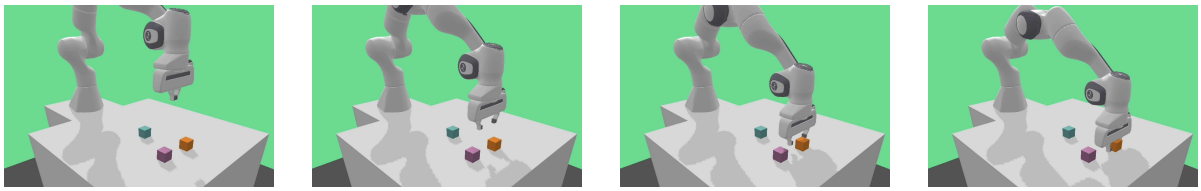


Figure 3.15: The grasp behavior visualized from left to right. The agent is reaching for the *orange* object and grasping it.

Lift For the lift task, the agent needs to place the goal object between its fingers and raise it to a predefined target height sampled from $U(0\text{cm}, 10\text{cm})$. Following previous work on robotics task design (Plappert et al., 2018), in one-third of the cases, the agent does not need to lift the object, facilitating easier exploration. The instructions for this task contain the verbs *"lift"*, *"raise"*, and *"hoist"*.

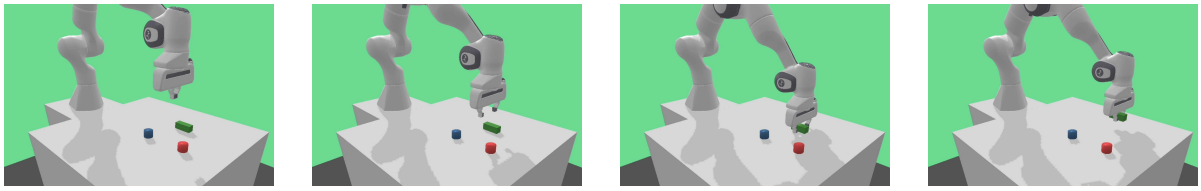


Figure 3.16: Demonstration of the agent successfully lifting the *green cuboid* while avoiding contact with other objects.

Multitask Setup We additionally implement a multitask setup that combines the previously listed tasks into a joint setting with various levels of difficulty:

- *Easy*: reach and push
- *Medium*: reach, push, and grasp
- *Hard*: reach, push, grasp, and lift

No explicit task identifiers are incorporated into the environment state. Tasks are distinguishable solely by their corresponding action verbs used in the instructions. For each new episode, a task is uniformly selected from the distribution of tasks.

Task Options We offer various options to create task variations. In all settings, there are 2 to 3 objects on the table that need to be manipulated according to the task. Object properties can be modified, expanding the vocabulary and the combinatorial space of possible instructions. The four features and their possible values are summarized in following listing:

- color = {red, green, blue, yellow, purple, orange, pink, cyan, brown}
- shape = {cube, cuboid, cylinder}
- weight = {light, heavy}
- size = {small, medium, big}

We depict the implemented options as property combinations in Table 3.1, listing the amount of individual words within the resulting vocabulary and the unique number of instructions considering 3 task-specific action verbs. Default properties are assigned if no specific option is selected, always including three colors {red, green, blue}, the cube shape, the light weight, and the medium object size.

Combination	Colors	Sizes	Shapes	Weights	#instructions	#words
Default					9	9
Color	✓				27	15
Shape			✓		45	12
Weight				✓	33	11
Size		✓			45	12
ColorShape	✓		✓		117	18
WeightShape			✓	✓	87	14
SizeShape		✓	✓		108	15
ColorShapeSize	✓	✓	✓		234	21
ColorShapeWeight	✓		✓	✓	195	20
ColorShapeSizeWeight	✓	✓	✓	✓	312	23

Table 3.1: Overview: Single Task Options

Individual properties are provided as one-hot encodings concatenated to the overall MDP state information of an object (see beginning of Subsection 3.4.1) in the following way: $s^{+}_{\text{obj}} = (s_{\text{obj}}, o_{\text{color}}, o_{\text{shape}}, o_{\text{size}})$, where $o_{\text{color}} \in \{0,1\}^{|\text{color}|}$, $o_{\text{shape}} \in \{0,1\}^{|\text{shape}|}$, and $o_{\text{size}} \in$

$\{0,1\}^{|\text{size}|}$. Again, we place specific emphasis on the missing o_{weight}, that we purposefully exclude to require active exploration by the agent in order to infer the mass of the objects. All of the tasks can be initiated with one of the combinations from Table 3.1, where one additionally specifies the number of objects on the table.

Different task options influence the number of words in the vocabulary, and thus the number of unique instructions the agent observes. However, some properties are more demanding to learn because they require direct interaction with the objects and cannot be observed (e.g., *Weight*), are difficult to ground due to the sheer number of instruction combinations (e.g., *Color*), making it easy to misinterpret the instruction (e.g., *ColorShape*), or require special treatment of the sensor signals as they are too subtle to sense and too insignificant to see (e.g., *SizeShape*). The multitask setting implements task combinations by simply summing the number of action words with the instructions from Table 3.1.

3.4.2 Formal Grammar Description of Goal Instructions

Inspired by Chevalier-Boisvert et al. (2019), we generate goal instructions using a semantic grammar with fixed sentence positions for the action verb, as well as the primary and secondary attributes. Properties are ordered to maintain language correctness or closest colloquial usage. For instance, it is more common to say *"Lift the blue cube"* than *"Lift the cube blue"*. Table 3.2 lists the primary and secondary word sets for each combination. We omit the listing of threefold combinations, as they are composed of twofold combination pairs. For example, the option *ColorShapeSize* is made up of *ColorShape* and *SizeShape*.

Combination	Primary	Secondary
Default	{red, green, blue}	{cube}
Color	color	{cube}
Shape	{red, green, blue}	shape
Weight	weight	{red, green, blue}
Size	size	{red, green, blue}
ColorShape	color	shape
WeightShape	weight	shape
SizeShape	size	shape

Table 3.2: Primary and secondary properties for the task combinations. We provide a fixed order to retain natural-sounding instructions. Enlisted are the default sets (in brackets) and the placeholders for the other property sets.

In Figure 3.17, we depict the Backus-Naur Form (BNF) exemplary for the *"push"* and *"reach"* tasks, with the *Color* and *Shape* options listed.

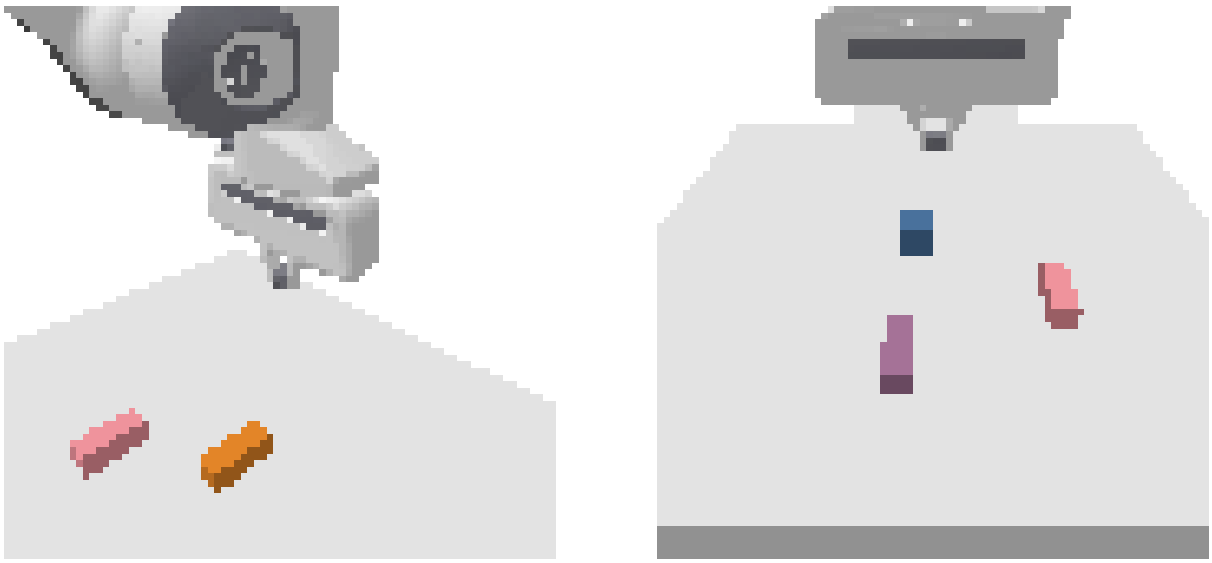
3.4.3 Pixel-based Observation

LANRO also provides a partially observable MDP (POMDP) setting with pixel-based observations. In this setup, the agent receives limited world state information through a

⟨INSTRUCTION⟩ ⊨ ⟨TASKVERB⟩ ⟨ARTICLE⟩ ⟨OBJECT⟩
⟨OBJECT⟩ ⊨ ⟨COLOR⟩ ⟨SHAPE⟩ | ⟨SHAPE⟩ `object` | ⟨COLOR⟩ `object`
⟨SHAPE⟩ ⊨ `cube` | `cuboid` | `cylinder`
⟨COLOR⟩ ⊨ `red` | `green` | `blue` | `yellow` | `purple` | `orange` | `pink` | `cyan` | `brown`
⟨TASKVERB⟩ ⊨ (`reach` | `touch` | `contact`) | (`push` | `move` | `shift`)
⟨ARTICLE⟩ ⊨ `the`

Figure 3.17: Backus-Naur Form for our instruction set inspired by Chevalier-Boisvert et al. (2019) – adopted from our prior work Röder and Eppe (2022).

camera, making the aforementioned explicit encoding of object properties unnecessary, as the agent can directly observe the color, shape, and size properties. The vision input is an RGB image observation $o_t \in \mathbb{R}^{W \times H \times C}$, where $W = 84$ is the width, $H = 84$ is the height, and $C = 3$ are the color channels, yielding $o_t \in \mathbb{R}^{84 \times 84 \times 3}$. This observation is provided by either a virtual camera mounted to the agent for an egocentric perspective (see Figure 3.18a) or a static camera view for an overall scene view (see Figure 3.18b). While vision inputs create a more realistic setting, they might also increase the task's



(a) Egocentric (b) Static

Figure 3.18: Examples from our POMDP setting in **LANRO**, with the egocentric view to the left (a) and static perspective to the right (b).

difficulty. The majority of this thesis focuses on the fully observable MDP setup, unless noted otherwise; for instance, in Subsection 4.9.5.

3.5 Experiments

*The following section provides the initial experimental results for our former and new baseline algorithms in a selection of tasks proposed in our **LANRO** environment. In addition, we highlight the different language encoders involved to process the lingual goals.*

In this section, we provide the initial results first published in our work Röder et al. (2022), and more recent ones from our language-conditioned baseline Language-Conditioned Soft Actor-Critic (LCSAC), coming with this thesis. The former experiments exclusively included results for the *reach* task and a limited amount of task option variations. In the corresponding Subsection 3.5.1, we employ the **LCSAC** using the GRU as a language encoder in combination with the dense word embeddings. In the following, we will exclude further examination of the one-hot encodings (see Section 3.2.3) as we demonstrated those to be inferior to the dense embeddings for our setting in Röder et al. (2022).

With Subsection 3.5.2, we present the latest results we obtained after the aforementioned publication. This section provides a comprehensive empirical study on a vast selection of tasks, including many variations, employing the algorithm Language-Conditioned DroQ (LCDroQ), which replaces the former **LCSAC** baseline and will be presented in detail in Subsection 5.5.2. We highlight all the introduced language encoders from Subsection 3.2.4, employing the dense embeddings from Section 3.2.3. These results are additionally complemented by further findings provided in Section B.1. Both sections evaluate the language grounding capabilities across a selection of tasks to demonstrate the limitations and challenges of learning instruction-following from sparse rewards.

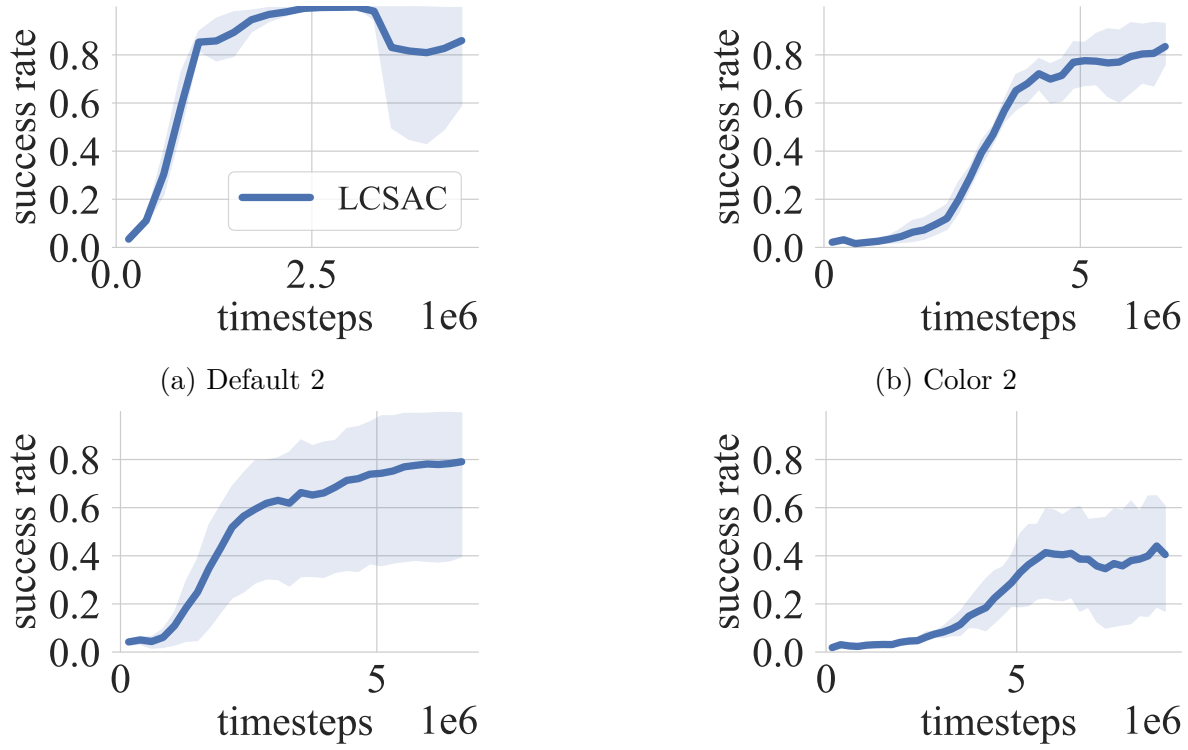
3.5.1 Initial Experiments

This section presents the experimental results of our prior work in Röder et al. (2022). It uses an older version of **LANRO**[2] and our code base[3]. For all the experiments, we exclusively employ the GRU (see Section 3.2.4) as the language encoder and train our own word embeddings from scratch (see Section 3.2.3). Furthermore, the setup only contains two objects on the table.

Figure 3.19 illustrates the *reach* task, where we display the average success rate (y-axis) of the agent in relation to its number of environment interactions on the x-axis (timesteps). The simplest task, using the *Default* mode, hence the setup with the colors *red*, *green*, and *blue*, the shape *cube*, the size *medium*, and the weight *light* (see Section 3.4.1 for the full list of task options), achieves an almost perfect success rate after $1e6$ environment steps. As the task complexity increases, incorporating more colors and shapes (where the *Color* option extends the range of colors by *yellow*, *purple*, *orange*, *pink*, *cyan*, and *brown* and the *Shape* option extends the object shapes by *cuboid* and *cylinder*), it takes longer to observe a performance increase in comparison. Expanding the possible object property combinations not only enlarges the space of object observations but also increases the vocabulary used to refer to the goal object. Figure 3.19d with the additional colors and shapes, where the

[2] https://github.com/frankroeder/lanro-gym/tree/icdl2022
[3] https://github.com/frankroeder/hipss

option *ColorShape* combines the previously mentioned *Color* and *Shape* options, presents the greatest challenge; with its even higher number of object property combinations, our baseline **LCSAC** barely exceeds the success rate of 0.4. This is particularly challenging for



(a) Default 2

(b) Color 2

(c) Shape 2

(d) ColorShape 2

Figure 3.19: We present the results of the *reach* task using the language-conditioned Soft Actor-Critic algorithm (**LCSAC**) from our prior publication (Röder et al., 2022). Each graph shows the number of environment steps (x-axis) and the average success rate (y-axis) for the options *Default*, *Color*, *Shape*, and *ColorShape* (see Table 3.1). The shaded region depicts the 90% confidence interval of 5 random seeds.

state-based representations because the indicator vectors within the observed states that prescribe object properties do not provide any hints on similarities (see Subsection 3.4.1). Consequently, learning word meanings becomes a one-to-one mapping to property words related to property encodings as part of the state observation. In the pixel-based setting, contrasting words that map to similar object representations would yield different results. For example, a *"red"* and *"orange"* object are more similar than a *"red"* and *"blue"* object because their RGB values are also more alike. The state-based setting cannot exploit these regularities of the input data, making it more challenging to learn. The learning curves presented in Figure 3.19 require millions of timesteps, which illustrates the problem of sample inefficiency in grounding language based on sparse rewards. In the following Chapter 4, we study approaches to address this shortcoming (Röder et al., 2022).

3.5.2 Full Experiments

In this section, we provide a comprehensive experimental overview with the latest version of **LANRO** and our latest **LCSAC** extension incorporating DroQ (**LCDroQ**) (Hiraoka et al., 2022), that we will present later in Subsection 5.5.2. Unlike in our paper Röder et al. (2022), we explored two more types of language encoders, namely the aforementioned MLP (see Section 3.2.4) and the self-attention (see Section 3.2.4).

Single Task Experiments The following experiments demonstrate the performance of our new baseline **LCDroQ** (see Subsection 5.5.2) in its default setting with the different language encoders presented in Subsection 3.2.4, and the dense word embeddings of Section 3.2.3. All the tasks contain 3 objects on the table, in comparison to the 2 objects used in the prior setup of Subsection 3.5.1.

In Figure 3.19, we observe that the *Default* settings converge relatively quickly for the *reach* and *push* tasks, while the *grasp* and *lift* tasks achieve an overall lower success rate (1st column with Figures 3.20a, 3.20f, 3.20k and 3.20p). We attribute this to the more challenging nature of the grasping motion, which requires precise placement of the end-effector above the object, opening the gripper to an extent that allows the object to fit between the fingers, and then closing them precisely. Furthermore, lifting an object requires carefully placing the object between the two fingers and making an upward motion while adjusting the forces to avoid dropping it with insufficient strain, or pressing too hard, causing it to slip out of the fingers. As expected, the learning difficulty increases with the number of colors and shapes (2nd and 3rd column in Figure 3.20).

The *Color* option adds 6 additional colors on top of the *Default* setting, resulting in 9 colors for the agent to manage. As language instructions are uniformly sampled, the learning is challenged by encountering all the colors independently and identically distributed right from the start, making it more difficult to ground them based on rewards only. This is evident from the additional time it takes to reach a success rate close to that of the *Default* setting. For the *reach* tasks, all language encoders achieve a success rate of values greater than 0.8, but only after an order of magnitude more timesteps than the setting in the 1st column (see Figure 3.20b). In the experiments for the *push* task, we observe a similar tendency (cf. Figure 3.20g), although the average highlights the self-attention mechanism as slightly superior to the MLP and GRU. Both the *grasp* and *lift* tasks achieve a much lower average success rate for the *Color* option. In the case of the *grasp* task, the success rate plateaus at 0.25, which is less than half the performance of the *Default* setting (comparing, *e.g.*, Figure 3.20l to Figure 3.20k). Lifting objects yields almost similar results, with the overall performance dropping to approximately a success rate of 0.2, half the performance of the *Default* setting, while the GRU encoder performs worse than the other encoders (cf. Figure 3.20q).

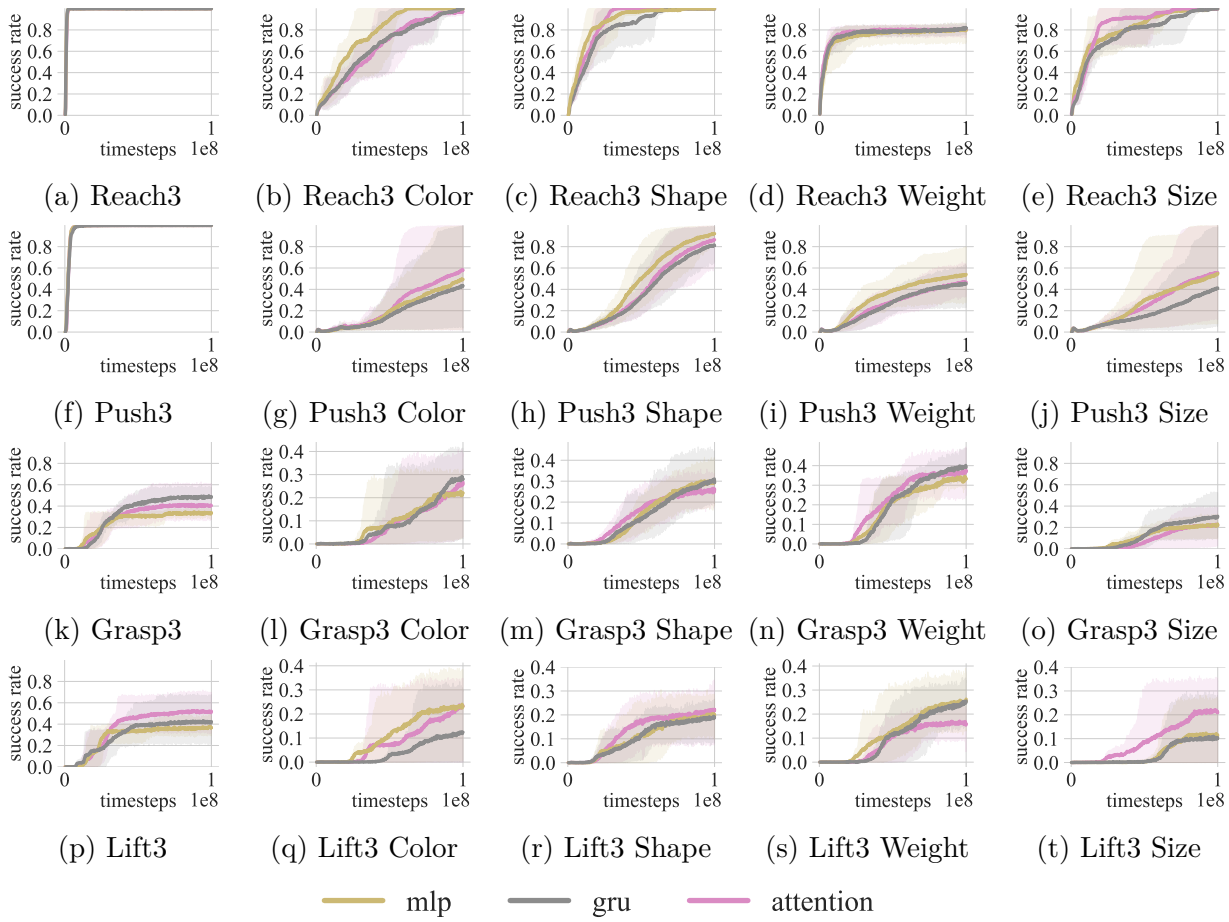
Although the color setting provides more word variations (2nd column of Figure 3.20), making grounding more challenging, the *Shape* option (3rd column) primarily increases the challenge of behavior control with different types of physical object shapes. The *reach* task can converge to the optimal success rate of 1.0 with all encoders, requiring 20 to 60 million timesteps in total to achieve this (see Figure 3.20c). For the *push* task, it takes the

full amount of timesteps reserved for training to achieve a success rate close to 0.8 for all language encoders (see Figure 3.20h). We assume this to be part of the task of pushing different kinds of objects on the table, which all yield different physical dynamics when being moved. Grasping different types of shapes is also more challenging, as shown by the slightly worse success rate of the self-attention encoder, but overall close and better for the MLP and GRU encoders (see Figure 3.20m). While the self-attention language encoder yields marginally better performance in the lift task with the *Shape* option, both the MLP and GRU encoders show slightly inferior results. The self-attention encoder reaches a plateau at a success rate of 0.2, less than half of the performance achieved in the *Default* setting (see Figure 3.20r).

In the case of objects with different weights (4th column, Figures 3.20d, 3.20i, 3.20n and 3.20s), the agent must interact with them to sense the sensorimotor differences, hence the sensor signals and the actions applied. This highlights the importance of embodiment in a closed-loop setup to explore the mass of the present objects to identify the goal object. The *reach* task in Figure 3.20d is a little bit worse than the *Default* setting in Figure 3.20a. We attribute this to the task requiring minimal interaction, and slightly touching an object to sense its weight might be enough, leading to a marginal lower success rate where this weighing of the object occasionally fails. Pushing objects requires greater involvement from the agent, possibly moving the objects too much when trying to sense the weights, resulting in a 0.2 to 0.4 lower success rate when learning, with minimal differences among the language encoders (see Figure 3.20i). In Figure 3.20n, we depict the task of grasping the objects and identifying their weight, which requires a certain amount of lifting. The success rates of all encoders approach similar values to the *Default* setting but are slightly more stable. When tasked with lifting (see Figure 3.20s), detecting mass differences is particularly challenging when objects are lifted too far. The agent may struggle to restore the objects to their initial positions, leading to lower success rates of approximately 0.2 for the GRU and the MLP, and up to 0.35 for self-attention, in comparison to the *Default* setting in Figure 3.20p.

For the tasks with different object sizes (5th column, Figures 3.20e, 3.20j, 3.20o and 3.20t), the behavior required to manipulate the objects changes drastically. While the results in the *reach* task perform quite similar to the case with the varying shapes (see Figures 3.20c and 3.20e), the *push* task seems to provide a harder challenge (see Figures 3.20h and 3.20j). Grasping requires the robot to adjust its gripper according to the mentioned size of the object, resulting in a more difficult task, as indicated by the overall lower success rate of 0.2 (see Figure 3.20o), compared to Figure 3.20m, the setting with different shapes. A similar decline can be observed in the case of the *lift* task depicted in Figure 3.20t, where the GRU and the MLP drop to values of 0.1, while the self-attention encoder drops only to a value of 0.2.

In Section B.1, we provide further experiments on the task with 2 objects that are slightly easier and follow the setup of Röder et al. (2022) as a suitable comparison.



(a) Reach3 (b) Reach3 Color (c) Reach3 Shape (d) Reach3 Weight (e) Reach3 Size

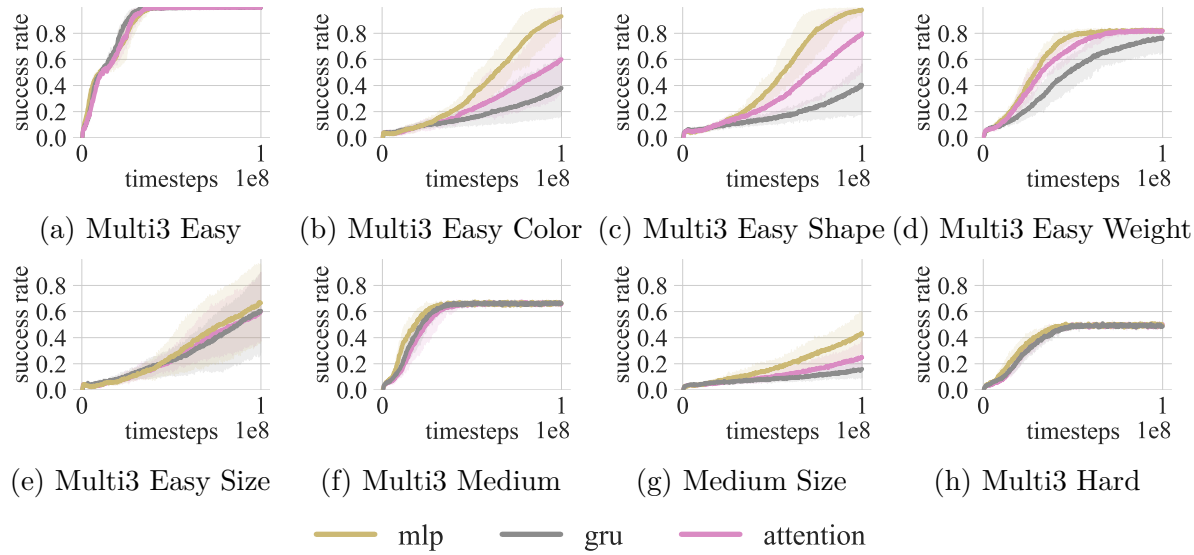
(f) Push3 (g) Push3 Color (h) Push3 Shape (i) Push3 Weight (j) Push3 Size

(k) Grasp3 (l) Grasp3 Color (m) Grasp3 Shape (n) Grasp3 Weight (o) Grasp3 Size

(p) Lift3 (q) Lift3 Color (r) Lift3 Shape (s) Lift3 Weight (t) Lift3 Size

Figure 3.20: The figure compares the different types of language encoders that we use in our language-conditioned RL baseline **LCDroQ**, namely MLP, GRU and self-attention (cf. Subsection 3.2.4). Each graph shows the number of environment steps (x-axis) and the average success rate (y-axis) for 3 trials, with the shaded region being the 90% confidence interval. Illustrated are column-wise, from left to right, the *Default*, *Color*, *Shape*, *Weight*, and *Size* options from Table 3.1, where we have the tasks *reach*, *push*, *grasp*, and *lift* in each row of the figure.

Multitask Experiments Figure 3.21 showcases the average success rate for our language encoders within the multitask setup of our **LANRO** environment. As presented in Subsection 3.4.1, the *easy* settings in Figures 3.21a to 3.21e combine the *reach* and *push* tasks, which are equally likely to be selected at the beginning of each episode. The same holds for the *medium* and *hard* multitask settings in Figures 3.21f and 3.21h, with their respective task distributions (cf. Section 3.4.1). In the case of the *easy* multitask setting, we observe the *Default* and *Weight* options (see Figures 3.21a and 3.21d) converge within our 100 million timestep budget, with success rates of 1.0 and 0.8, respectively. In the *Color* setting (see Figure 3.21b), grounding both tasks seems to be accompanied by intra-task transfer, as learning the *push* task alone does not yield this success result for the case of the MLP encoder (see Figure 3.20g). For the other language encoders, this setting appears to be limited by the difficulty of learning to push, compared to the single-task setup in Figure 3.20. The success rates are very similar, but worse in comparison to the former *reach* performance. For the *Shape* option (see Figure 3.21c), the MLP encoder, closely followed by the self-attention, can exploit intra-task transfer, highlighting the challenge of control rather than coping with an extended vocabulary. For the *Size* option in Figure 3.21e and the *Default* setting in Figure 3.21f, there are almost no success rate differences for the 3 language encoders. However, the *Size* option for the *medium* task in Figure 3.21g reveals a difference of up to 0.25 in terms of the success rate, with the MLP language encoder leading. Figure 3.21f shows that all encoders reach a success rate value of 0.6, which suggests that the agent fails to learn the *grasp* task that is part of this configuration compared to the *easy* setting in Figure 3.21a. The same applies to the *hard* setup in Figure 3.21h, where the agent additionally fails to also solve the *lift* task. We have presented a selection of *multitask* experiments and provide the full range of results in Figure B.5 of Section B.1.



(a) Multi3 Easy (b) Multi3 Easy Color (c) Multi3 Easy Shape (d) Multi3 Easy Weight

(e) Multi3 Easy Size (f) Multi3 Medium (g) Medium Size (h) Multi3 Hard

Figure 3.21: Comparison of our different language encoders across various multitask environments (*easy*, *medium*, and *hard*) with different options (*Default*, *Color*, *Shape*, *Weight*, and *Size*) showing the average success rates (y-axis) over environment steps (x-axis) from 3 random seeds with the shaded area depicting the 90% confidence interval.

3.6 Summary

This chapter introduced our language-conditioned RL benchmark **LANRO**, to investigate the challenge of language grounding with sparse rewards in a robotics setup. It offers a selection of tasks with a wide variety of options to alter the number of object properties and resulting words observed. In addition, we provide results for a language-conditioned Soft Actor-Critic baseline **LCSAC**, based on our previous work (Röder et al., 2022), as a basis for the remainder of the experiments in this thesis. Furthermore, we brought forward results for our novel baseline in Subsection 5.5.2 on the language-conditioned Dropout Q-functions approach, termed **LCDroQ**. With this chapter, we can positively answer **Research Question 1** by modeling a challenging language grounding environment for continuous control in robotics that could be solved by well-established RL algorithms such as SAC and a custom extension of it.

Chapter 4

Hindsight Language Learning

This chapter investigates the concept of learning from failures in hindsight. Failures occur when the learner is unable to achieve a proposed goal. While goal-directed behaviors implement the flow from goals to actions, learning from failures enables the learner to derive the goal from a given behavior, essentially reversing the process. Coming up with appropriate language goals for an observed behavior can be challenging. These goals can be extrinsically assigned by a caretaker as a form of feedback, or they could originate from an internal process of the agent's imagination. Both types of guidance reinforce language grounding through sensorimotor experiences, representing two distinct approaches. In this chapter, we present our implementations of these mechanisms, which are inspired by developmental psychology, and demonstrate how they improve the sample efficiency of a learning robotic agent.

Human communication is noisy and full of pitfalls that lead to misunderstandings, which, when not resolved appropriately, result in undesired outcomes. Humans are proficient at identifying such circumstances by correcting, adapting, and even learning from these unintended situations. To date, such conversational settings present many challenges for intelligent agents, like robots. Although these misconceptions could originate from a lack of grounded language (Bisk et al., 2016), intelligent agents struggle to adapt or learn to proactively prevent these mistakes. In the remainder of this work, we expand upon the concept of language-conditioned reinforcement learning introduced in Chapter 3 and frame this challenge as a verbal/nonverbal dialog between an agent and a human. In our case, a **synthetic caretaker** takes on the role of the **human**, providing language goals verbally, whereas the **robot** attempts to satisfy them through purely **nonverbal physical interaction** with the environment (Trott et al., 2016).

As a primer for how such an interaction looks when the agent gets something wrong and the caretaker possibly intervenes, consider the scenes in Figure 4.1 with the green (🍏) and red apple (🍎). In step **1**, the caretaker instructs the agent to *"Hand the green apple"*, providing the language goal g_ℓ for the agent to follow. Due to the acting policy confusing the colors or simply targeting the closer apple, step **2** results in the agent not receiving the expected reward because of handing the incorrect object. There are now **two possibilities** covered by the approach we present in this chapter. **First**, the lack of

reward leads to step **3a**, where the caretaker recognizes that the behavior diverged from the intended outcome and provides an appropriate language goal retrospectively, which the agent utilizes for learning. In the **second** case (step **3b**), we depict the internal monologue of the agent, where it generates a language goal as a retrospective positive example for learning by itself.

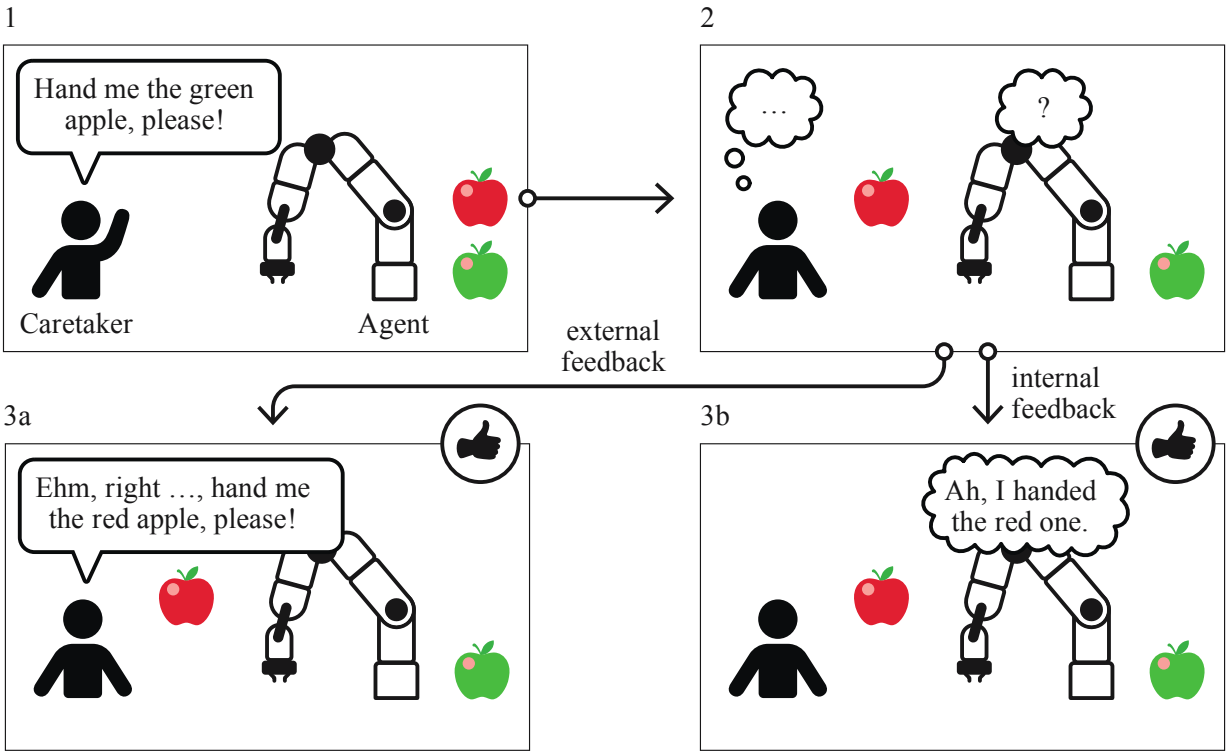


Figure 4.1: We illustrate the example of both the externally and internally prescribed language goals for the undesired outcome. In step **1**, the caretaker utters the instruction *"Hand me the green apple, please!"*. However, in step **2** the agent fails to follow this goal and hands over the *red apple* instead. Wondering why there is no reward, the agent is left with two options to learn from this failure. For step **3a**, the caretaker recognizes this mistake and provides a positive learning experience for the agent by retrospectively changing the goal instruction to one that fits the behavior, in this case, treating it as if it were the intended outcome. Without any external caretaker guidance, in step **3b**, the agent generates an appropriate goal instruction by itself that fits the executed behavior, thus providing a positive learning experience via self-speech.

The two cases highlight the notion of reconsidering an outcome with a different context, hence a different goal. This is beneficial as the agent failed to follow the original instruction of handing the green apple but still ended up performing sensible actions by handing the red

apple instead. An object, which is an *"apple"*, was *"handed"*, matching already large parts of the actual language goal. However, the agent simply failed to identify the correct color *"green"*, essentially a single word of the instruction, possibly due to the lack of grounding.

For the remainder of this work, we assume that the human communicates verbally, while the agent reacts nonverbally through physical actions. Therefore, the human can only discern if the agent misunderstood an instruction when its behavior diverges from the human's expectation. The relationship between the goals, potentially coming from the space of all language phrases, and the resulting behaviors can also be expressed in terms of probabilities. Here we list the cases for the instruction-following and hindsight behavior labeling:

- How likely is the policy to generate a behavior given the goal: $p(\tau \mid g_\ell)$
- The probability of the goal given the behavior $p(g_\ell \mid \tau)$, utilized to search for potentially alternative goals g'_ℓ

We posit that this bidirectional relationship is a result of embodied language understanding, as language in this case always refers to behaviors in the physical world and the other way around. An alternative approach to conceptualize this relationship is through the metric of mutual information $I(X;Y)$, quantifying the mutual dependence of two random variables X and Y. In our case, it intuitively quantifies how much we can infer about the behavior trajectory τ if we know the language goal g_ℓ and vice versa, $I(\tau; g_\ell)$. Hence, we aim for a policy that reliably follows instructions (agent) and a mechanism that assigns with a high probability the correct goal instructions to behaviors (retrospective expert labeling or egocentric speech).

In our review Röder et al. (2021) and Section 1.4, we outlined that large parts of learning in the early phases of childhood are, simply put, primarily driven by goal-directed behaviors (Kaelbling, 1993), imitation (Paulus, 2014), and intrinsic motivation (Schmidhuber, 2010; Colas et al., 2019; Röder et al., 2020). Agents that comprehend the bidirectional relationship between language and behaviors can creatively set their own goals and come up with novel target configurations to explore the world (Colas et al., 2022).

While the aforementioned mechanisms in the earlier developmental stages function without actively exploiting language mentally or verbally to its full potential, the later phases of grounding seem to introduce a kind of structure into the continuous sensorimotor space by assigning symbols. In other words, discrete and concrete word meanings are assigned to specific subspaces of the physical/behavior space. For instance, a child recognizes that grasping is a specific concept that is almost always the same, regardless of what is being grasped. Language aids in the categorization of experiences (Rosch, 1978), facilitating the mind's ability to process and recall information. It also enables and improves cognitive processes through planning, imagination, mental simulation, and reasoning, where language functions as a kind of tool (Vygotskij, 1985; Colas et al., 2022). Language is also inherently compositional by nature. Altering a single word could change the entire meaning of a sentence, as in the previous example with instructions referring to the *"red"* or *"green"* apple (see Figure 4.1). The structure of language allows for grounding the meaning of individual words as part of a more complex sentence, like the following instruction:

"Move the red box to the green area next to the blue box."

Understanding this relies on cognitive capabilities that take into account the physical-world experiences to interpret the compositionality of temporal and spatial relationships between the actions, objects, and locations mentioned. One might assume that if an intelligent agent is able to solve the tasks *"grasp the red cube"* and *"push the blue cube"*, it could also execute the tasks *"push the red cube"* or *"grasp the blue cube"*. The colors should not influence the understanding of which motion to execute; they should result in different representations independent of it. While we simply swap out the verbs for the corresponding goal objects, thus changing the composition of words, intelligent agents still struggle with this type of out-of-distribution (OOD) data samples. Exploiting the compositionality of language in the context of language-conditioned RL requires additional effort (Eppe et al., 2016; McCallum et al., 2023), *e.g.*, by implementing helpful inductive biases or providing manually engineered representations (Spilsbury and Ilin, 2022).

The type of generalization that we are referring to is known as systematicity. Humans are capable of understanding sentences that they have never encountered before by systematically combining known concepts in a compositional manner (Hupkes et al., 2020; Hill et al., 2020). As an example, consider the sentence *"a green elephant with a birthday hat"*; most people could easily comprehend and also visualize it by combining familiar concepts (see Figure 4.2 for an example).

Figure 4.2: We depict *"a green elephant with a birthday hat"*, generated with DALL-E 3 (Ramesh et al., 2021). The model shows the individual parts related to the prompt, hinting at compositional capabilities.

The work of DALL-E (Ramesh et al., 2021) shows that current machine learning systems are able to capture relationships between words and concepts, and can "creatively" combine them. In this case, an elephant as a type of animal with a green-colored skin and a birthday hat is depicted in the style of a comic. The elephant is smiling, wearing a colorful hat with confetti in the background, appearing to enjoy and celebrate its birthday.

Imagination could potentially enhance an agent's language grounding abilities and foster learning from undesired but still educational outcomes, such as the one from our example in Figure 4.1. Educational outcomes are those considered by hindsight learning,

a method to learn from failures by imagining them in a different context. We assume systematicity to be a cornerstone for zero-shot and hindsight learning, as an agent could employ deductive reasoning to solve unseen problems or imagine as many hindsight goals as necessary to improve grounding. Hindsight learning has been a crucial concept in goal-conditioned reinforcement learning (Subsection 3.2.1). The work of Hindsight Experience Replay (HER) by Andrychowicz et al. (2017) enabled the successful completion of various sparse reward robotic tasks, such as those found in the *Gymnasium Robotics* environments (Plappert et al., 2018; de Lazcano et al., 2023). However, applying hindsight learning in the language domain has different requirements that we will reveal in the following sections.

4.1 Related Work

The upcoming section discusses recent work that employs the concept of hindsight learning in sparse reward settings, with a few exceptions considering linguistic goals.

Hindsight learning enables the repurposing of past experiences to essentially learn from failures. This is particularly valuable in robotic applications, where the process of collecting data can be costly or dangerous. Rather than discarding the "unsuccessful" trials, this approach modifies generated behavioral trajectories to extract useful learning signals. In the following, we will highlight publications that utilize hindsight instructions, which are relevant to the language-conditioned setting of this thesis.

In the work from Jiang et al. (2019), the authors propose a hierarchical reinforcement learning approach called **Hierarchical Abstraction with Language** (HAL). HAL employs language to generate subgoals based on the overall environment goal, a complex goal instruction. The approach employs two policies: a high-level policy that decomposes the extrinsic language goal instruction into simpler lingual subgoals, and a low-level policy that solves these subgoals. The low-level policy carries out the environment actions, whereas the high-level policy solely generates subgoals. This approach underpins the idea of compositionality, where language goals are broken into smaller components. In their work, the authors provide instructions that direct the agent to, *e.g.*, rearrange objects based on their color brightness. While this represents an abstract and long-horizon task requiring numerous actions, the high-level policy proposes subgoals to make the agent solve simpler problems that contribute to the overall task instead. For instance, a possible subgoal might instruct the low-level policy to switch the darkest-colored object with the leftmost one, thus initiating a left-to-right sorting process step by step, and more precisely subgoal by subgoal. If the low-level policy fails to achieve a proposed subgoal, a hindsight instruction is used to relabel the actual outcome. For instance, if the low-level policy fails to swap the darkest object's position with the leftmost one, but instead swaps the *"red"* object with the *"blue"* object, this potentially useful subroutine can still be utilized to solve the overall problem. Hindsight instruction replay aids in leveraging these "failures". In essence, the higher-level policy acts as if the achieved outcome was driven by the hindsight goal, *i.e.*, the relabeled subgoal. To summarize, their approach exploits both the compositionality of language and the problem structure itself to employ efficient reinforcement learning that solves deduced easier problems that contribute to solving an overall harder problem.

With the work on **Intrinsic Motivations And Goal INvention for Exploration** (IMAGINE) the authors Colas et al. (2020) also aim to leverage the compositionality of language to self-generate goals that the agent had not encountered during training. Such self-generated goals could include novel color or action combinations that have individually been seen but not in combination. It is an intrinsically motivated approach involving a social peer and crucial design choices such as object-centered representations, gated-attention, and deep sets. IMAGINE attempts to learn how to generate goals and model the goal-achievement reward functions to evaluate behaviors autonomously. A peer supports the exploration of the world by providing feedback in hindsight, generating the necessary experiences for the agent to learn from. By learning how specific outcomes relate to a social peer's lingual feedback, the agent essentially learns to map achieved states to potential goal descriptions. Learning the reward function is essentially modeling a binary classification problem, where the agent predicts whether the social peer would have granted a reward or not. Unlike other hindsight learning approaches, the hindsight instructions in this work (Colas et al., 2020) are essential for learning and generated from the beginning, rather than being a mechanism to repurpose failures, treating it as a path to making the agent even more autonomous. Similar to a young child, the agent can propose its own goals and extrapolate from the samples observed during training with the social peer. To summarize: their main contribution is to show that the architecture is capable of generalizing to unseen lingual goal compositions within an exploratory process where the agent generates and evaluates goals predicted by itself.

Hindsight instructions also appear in other research areas, such as imitation learning (Nair et al., 2018a) and inverse RL (Li et al., 2020a). These approaches generally employ similar notions found in deep RL and provide insights that are also useful in other robotic learning settings. One such work is by Lynch and Sermanet (2021), who utilize a pre-collected robotic teleoperation dataset of unrestricted play behavior (see also Section 1.4). This dataset is harnessed for vision-conditioned imitation learning by splitting the large behavioral trajectory of states into smaller sections, with a start state and an end state as the goal, thus leading to a goal-conditioned setup. Additionally, a language-conditioned subset is collected by employing human annotators to assign descriptions to these short sections in hindsight, rather than at the time the behavior was recorded. This language-conditioned subset makes up less than 1% of the total play dataset. Employing a multi-context imitation setting allows training a single policy that generalizes across different types of task descriptions, such as goal images and natural language instructions. By encoding various task descriptions into a joint latent goal space, the model can efficiently leverage both abundant goal image data and the scarcer language instructions. This reduces the need for extensive language-paired data samples, as the majority of learning is derived from the image goals. Lynch and Sermanet (2021) showcase that a policy trained this way can follow a wide range of instructions, even those not seen during training, and also in different languages using BERT (Devlin et al., 2019). This is achieved by leveraging diverse teleoperated play data and their hindsight instruction pairing method.

The approach that provides the most ideas and inspiration for our work is the architecture Hindsight Generation for Experience Replay (HIGhER) by Cideron et al. (2020). The authors train a model to predict instructions in hindsight given particular state out-

comes, hence learning to describe single state observations. For the training, they have

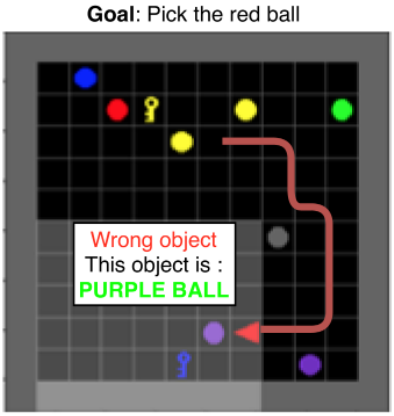


Figure 4.3: Example taken from the paper by Cideron et al. (2020), showing a scene in the BabyAI environment (Chevalier-Boisvert et al., 2019). The agent is instructed to *"Pick the red ball"*. However, it ends up with the wrong object, hence receiving the feedback *"purple ball"*.

to collect rewarding episodes that entail successful pairs of state outcomes and language goals (s_t, g_ℓ). Within their grid world setting, the simplified action and state space allows for relabeling unsuccessful episodes where the agent reached a wrong goal object or ended up at an incorrect location. Those events are identified by standing in a grid cell next to the object or a cell providing the exact location information. Consider Figure 4.3 as an example, where the agent originally was tasked to *"Pick the red ball"* but ended up at the location of the *"purple ball"* instead. The input to their model would now only be the agent in front of the purple ball, for which the prediction *"Pick the purple ball"* is generated and used for hindsight learning. As a limitation, their experiments exclusively consider the grid world environment BabyAI (Chevalier-Boisvert et al., 2019) and single discrete state inputs. Although IMAGINE (Colas et al., 2020) and HIGhER (Cideron et al., 2020) propose useful architectures and models for hindsight learning, they are only applied to environments with simple world dynamics. For robotics, the core domain of this thesis, we require a more advanced model capable of mapping complex behavior patterns to corresponding language instructions. Many behaviors in robotics take several timesteps to unroll and could involve complex motions or interactions with the world that cannot be derived by looking at a single sensory state. Even in the case of the fully observable MDP that satisfies the Markov property (see Section 2.2.1), it would not be possible to track a behavior like grasping an object by only considering a single frame. The act of grasping requires a sequence of states and actions, such as approaching the object, adjusting the gripper orientation, closing the gripper fingers around the object, and lifting it. We assume that it needs a method that determines if an object was grasped by analyzing the transitions of multiple states.

4.2 Background Hindsight Learning

The next section discusses more background on hindsight learning, which stems from the goal-conditioned framework, and its modification for tasks involving language.

In the domain of goal-conditioned RL, hindsight learning plays a crucial role, as demonstrated by numerous prior works (Colas et al., 2020; Lynch and Sermanet, 2021; Röder et al., 2022; Moro et al., 2022). This concept improves the sample efficiency by enhancing the replay mechanism of off-policy algorithms in particular, such as DQN (Mnih et al., 2015) in HIGhER (Cideron et al., 2020), DDPG (Lillicrap et al., 2016) used in HER (Andrychowicz et al., 2017) and CURIOUS (Colas et al., 2019), and SAC (Haarnoja et al., 2018) used in HAL (Jiang et al., 2019) and *decstr* (Akakzia et al., 2021). In complex sparse reward tasks, the agent typically rarely stumbles upon a solution that satisfies the goal condition. This results in the majority of the collected experiences containing unsuccessful trials. Hindsight learning counteracts this imbalance between "good" and "bad" experiences, repurposing the non-rewarding ones by deriving positive learning signals from them. This is achieved by changing the originally proposed goal under which the policy failed to complete the task into one that actually was achieved retrospectively.

4.2.1 Goals as State

The following sections build upon Subsection 2.2.5 and Subsection 3.2.1. To begin with the first part, we assume the goal space to be part of the state space, $G \subseteq S$. Consider Figure 4.4, where the agent should push the puck to the goal location indicated by the red colored marker within the circle. Given the failed attempt to push the puck, Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) allows considering the achieved state in the green circle, the location where the puck actually ended up, as the desired goal state in hindsight. To make the agent believe it made the right decision, it is necessary to also return a sparse reward with the relabeled experience. However, this approach has limitations in its applicability to different robotic settings. HER assumes that we have access to the state-to-goal mapping function from Equation 3.1 and a reward function that we can harness at any time without actually acting in the environment (Andrychowicz et al., 2017). The former means that goals are fully describable by state configurations, and one could derive the underlying goal given the state with the aforementioned mapping function. The latter refers to the hindsight replay, where we need to evaluate the reward function for the virtual goal to replace the old reward. In the cases where we have access to the goal-to-state mapping, we can treat every visited step of a state trajectory $\tau_s = (s_1, \dots s_T)$ as an achieved goal in hindsight, hence as a possible candidate for replay. HER implements different approaches to relabeling, which we will explain in the following paragraph, starting with the so-called **final** strategy. From a sequence of environment transitions in an unsuccessful episode

$$\{(s_0, a_0, r_0, s_1, g), \dots (s_{T-1}, a_{T-1}, r_{T-1}, s_T, g)\}$$

generated by a goal-conditioned policy $\pi(a_t \mid s_t, g)$, following the environment dynamics $\mathcal{T}(s_{t+1} \mid s_t, a_t)$, and the reward function $\mathcal{R}(s_t, a_t, g)$, HER considers the terminal state s_T

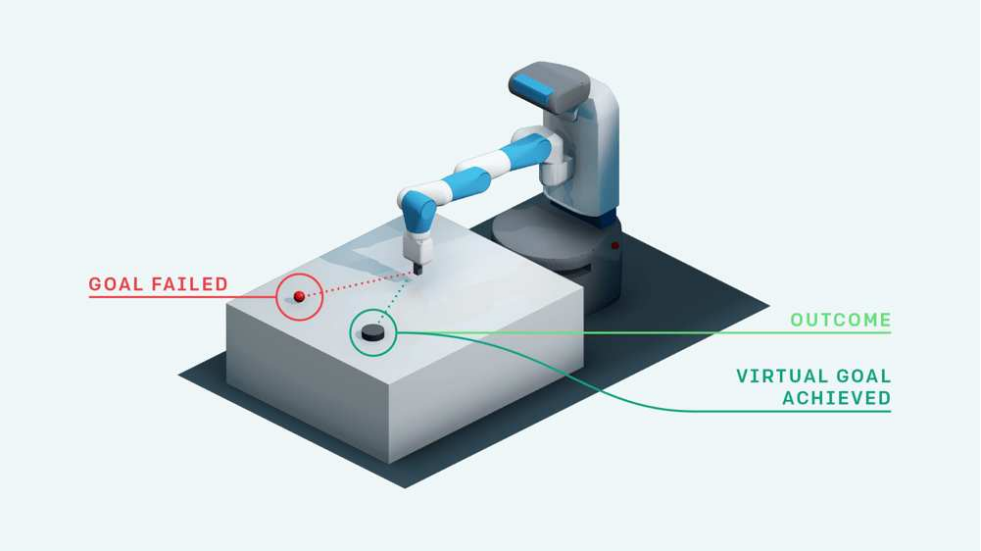


Figure 4.4: Figure taken from the OpenAI blog post *Ingredients of robotics* (Andrychowicz et al., 2017), showing the setting of the Fetch robot pushing the puck that ends up in the wrong location (green circle), instead of the proposed goal location (red circle). For hindsight learning, the actual outcome serves as a virtual goal achieved.

with a reward $r_{T-1} = -1$ as a possible candidate for relabeling. The last environment transition $(s_{T-1}, a_{T-1}, r_{T-1}, s_T, g)$ is changed the following way:

1. Replace the original goal g with $\hat{g}$ by applying the mapping function from Equation 3.1 to get the hindsight goal $\hat{g} = \phi(s_T)$
2. Evaluate the reward function with $\hat{g}$ to get the new reward $\hat{r}_{T-1} = \mathcal{R}(s_{T-1}, a_{T-1}, \hat{g})$
3. Add the relabeled transition $(s_{T-1}, a_{T-1}, \hat{r}_{T-1}, s_T, \hat{g})$ to the agent's replay buffer

Apart from using the last transition, HER employs two additional strategies, called **episode** and **future**. With the **episode** strategy, any timestep t of the episode's state trajectory τ_s could be harnessed as a hindsight goal candidate. The goal is selected by uniformly sampling a time index $i \sim U(0, T)$, to determine the transition for which we replace the state after mapping it to the corresponding goal. Therefore, the goal could be any state, an achieved goal, of that particular episode, s_i where $i \in \{0, \ldots, T\}$. For the **future** strategy, we consider the states achieved after a designated timestep t. This results in states s_i coming from a specific range of the time horizon, where we uniformly sample the time index from that interval, $i \sim U(t, T)$. The boundaries are the current step t and all the future ones until the episode limit of T. We depict the general procedure of HER in Algorithm 3, where one employs an off-policy algorithm like SAC (see Subsection 2.2.7).

Algorithm 3 Hindsight Experience Replay (**future** replay strategy)

π_θ	goal-conditioned policy	E	number of episodes
$\mathcal{D}$	off-policy replay buffer	T	episode horizon
ϕ	state-to-goal mapping function		

for episode $= 0, \ldots, E$ **do**
  Sample initial state $s_0 \sim \rho_0$ and goal $g_0 \sim \rho_g$
  for timestep $t = 0, \ldots, T-1$ **do**
    Sample action $a_t \sim \pi_\theta(\cdot \mid s_t, g_t)$
    Take environment step $s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t)$
    Compute reward $r_t \leftarrow \mathcal{R}(s_t, a_t, g_t)$ (see Equation 3.2)
    Store transition $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1}, g_t)\}$
  end for
  if $r_T == -1$ **then**
    Obtain achieved goal state $\hat{g}_T \leftarrow \phi(s_T)$ using Equation 3.1
    Calculate hindsight reward $\hat{r}_{T-1} \leftarrow \mathcal{R}(s_{T-1}, a_{T-1}, \hat{g}_T)$
    Store relabeled transition $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_{T-1}, a_{T-1}, \hat{r}_{T-1}, s_T, \hat{g}_T)\}$
  end if
  Sample mini-batch $\mathcal{B} \sim \mathcal{D}$
  Update policy π_θ according to chosen off-policy algorithm
end for

4.2.2 Language Goals

A limitation to HER is that it was originally designed for the aforementioned goal representations based on states, making it nontrivial to use for the language-conditioned cases (Röder et al., 2022) and other modalities (Nair et al., 2018b). Furthermore, not every visited state corresponds to a valid language goal $g_\ell \in G_\ell$. To exploit the abstraction and compositionality language provides, recent works implemented different notions of hindsight instruction replay (Jiang et al., 2019; Colas et al., 2020; Lynch and Sermanet, 2021). Another specific challenge of language learning within the reinforcement learning domain is the presence of ambiguities and out-of-distribution references. Under conditions of high uncertainty, an agent might execute incorrect actions triggered by a misinterpretation of the goal instruction. However, this seemingly incorrect behavior might be appropriate in a different context, which could be nearly the same instruction that differs by only one word. Consider Figure 4.5, where the agent is instructed to *"push the blue cube"* but confuses the colors and pushes the *"red cube"* instead. This is a classical example where one could exploit hindsight learning, as the agent actually achieved a sensible outcome, but just for a different goal instruction, such as the ones of the small selection of potential hindsight instructions listed below:

- *"Push the red cube"*
- *"Push the red object"*
- *"Push the object close to you"*

This resembles a distribution of possible alternatives g'_ℓ for the observed trajectory τ, therefore one could attempt maximizing $\text{argmax}_x\, p(g'_\ell = x \mid \tau)$ to get the most likely one. The challenge now becomes how to relabel the observed behaviors with the appropriate task descriptions (Li et al., 2020a).

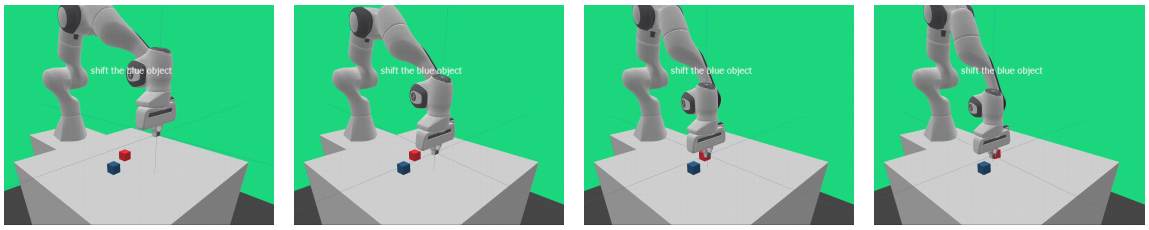


Figure 4.5: We depict an unsuccessful episode of an already useful policy interacting with the world (from left to right). The agent is instructed to push the *"blue cube"*. However, it ends up pushing the *"red cube"* instead, due to a lack of grounding the color property. This scene provides an optimal episode for hindsight language learning, as the experienced episode is a successful one given the instruction *"push the red cube"*, a potential hindsight instruction.

A remaining question is where those hindsight instructions like *"push the red cube"* in Figure 4.5 are coming from. Unlike the goal-conditioned setting, there is usually no function available that maps achieved goal states to appropriate instructions. In what follows, we provide two approaches inspired by developmental psychology that regard caretaker feedback and egocentric speech as sources of hindsight instructions. In the latter case, we even present how it is possible to exploit successful episodes to basically learn a type of state-to-language mapping function without supervision.

4.3 Methods

The following sections present the core contributions of our work Röder et al. (2022). We demonstrate how to improve the sample efficiency of language-conditioned reinforcement learning agents by exploiting concepts of developmental psychology.

Inspired by language development and following the ideas worked out in our previous review article (Röder et al., 2021), we propose two implementations inspired by the concepts of caretaker feedback and egocentric speech. In the former setup, a teacher entity provides guidance through language, like the feedback of what just happened, but only for a small selection of rewarding states. This allows the agent to correlate its sensorimotor experiences with the language provided by the caretaker. The latter approach is inspired by the egocentric speech that children use to comment on or explain to themselves what they are doing or about to do (Piaget, 1926). Using this for learning requires grounded language and some natural language generation capabilities. Although toddlers are not always grammatically correct with their narrative speech and make mistakes, this intrinsic self-supervised process still ameliorates language acquisition for instruction-following.

Both of these examples are illustrated in Figure 4.1, with the caretaker feedback following step **1**, **2**, and **3a** or **3b** for the case of egocentric speech.

Like the prior implementations (Jiang et al., 2019; Akakzia et al., 2021), we consider a policy that receives as input the current state and an episodic language goal (as introduced in Subsection 3.2.2). Initially, the agent is very unlikely to solve the task and needs to explore the environment. One major difference from previous work (Jiang et al., 2019; Akakzia et al., 2021) is that we consider two different origins of the correct instructions in hindsight that would have been the appropriate goal for the unsuccessful episode. Instead of only relying on expert feedback or an oracle function, like the state-to-goal mapping (cf. Equation 3.1), we show, similar to Cideron et al. (2020), how to autonomously predict those instructions.

4.4 Expert Feedback

This section presents the first method of hindsight language learning based on feedback from a synthetic caretaker.

The idea of a social partner or caretaker is very present in current language-conditioned RL papers (Colas et al., 2020; Akakzia et al., 2021). It also relates to works on interactive RL (Cruz et al., 2015; Nguyen et al., 2021; Xu et al., 2022), where a teacher entity provides guidance to the learner. However, interactive reinforcement learning is far more similar to language-assisted than language-conditioned RL, as the teacher's feedback is optional and can also be ignored. The implementation of Jiang et al. (2019) first mentions **hindsight instruction replay** being used for language-conditioned RL, by relabeling a state outcome with an appropriate language goal *a posteriori*. Similarly, in the work of Colas et al. (2020) and Akakzia et al. (2021), they employ a social partner that supports the intrinsically motivated exploration phase of the agent by returning language explanations for achieved outcomes.

Following these ideas from prior work has led us to our first approach, coined Hindsight Expert Instruction Replay (HEIR) (Röder et al., 2022). Like the aforementioned approaches, **HEIR** exploits a function akin to the state to goal mapping, provided by the environment, that returns language goals for achieved goal states. For our instruction setting, this could be described as a probability distribution of equally likely language goals given a trajectory, $p(g'_\ell = x \mid \tau)$, where x is an instance coming from all the trajectory-related language instructions generated according to the grammar in Figure 3.17. This can be thought of as a synthetic caretaker embedded into the **LANRO** environment, that, for each object interaction or other rewarding state outcomes, samples a possible hindsight goal instruction $g'_\ell \sim p(\cdot \mid \tau)$ and returns it to the agent. This is illustrated in Figure 4.6, where the caretaker is describing what the agent did by providing the feedback in hindsight.

However, it requires additional effort to implement such a teacher-like entity and the mechanism to trigger the feedback. We assume this setting to be the contrived perfect case of hindsight language learning, as those goals are generated by the semantic grammar of the environment itself. Similar to HER, we introduce three replay strategies in the

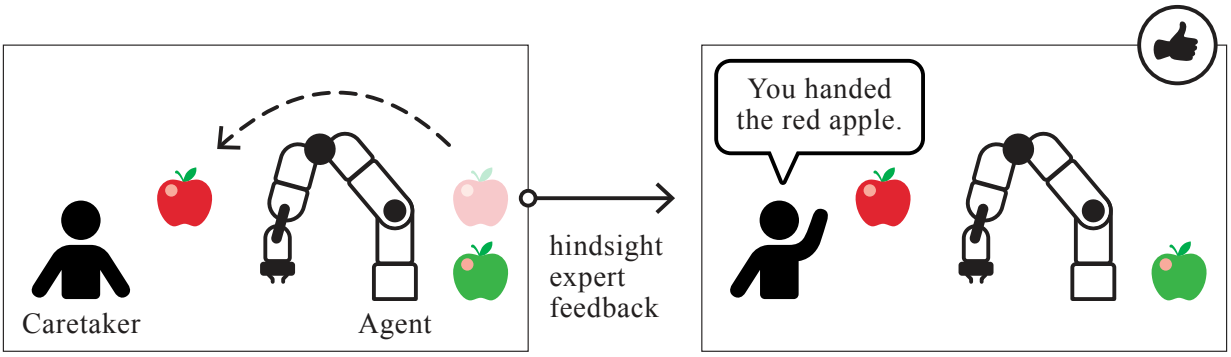


Figure 4.6: Upon handing the red apple to the caretaker (left box), the agent receives the hindsight goal instruction, *e.g.*, *"Hand the red apple"* as feedback (right box). Similar to the idea of a social partner proposed in Colas et al. (2020) and Akakzia et al. (2021).

following Section 4.5 that account for different points in time when the transition should be replayed, compared to when the failure happens.

4.5 Replay Strategies

In the following, we outline various strategies for replaying instructions in hindsight.

In HER (Andrychowicz et al., 2017), the authors proposed several replay strategies one can employ for hindsight learning. They differ in terms of the exact moment one considers a transition to be rewarded. In the following, we provide a description of our three implemented strategies called **future**, **episode**, and **final** for the **language-conditioned** case (Röder et al., 2022), that are different in many cases from those used in HER. These strategies are complemented by a parameter k, the number of transitions one considers replaying per episode. For example, not only the final transition is relevant for the hindsight goal, but also the predecessors that led to it or the successors that also constitute the desired state.

We illustrate this with the help of Figures 4.7 to 4.9, where we depict the original goal object in red and the virtual goals achieved in hindsight as green objects. We consider the *"push"* task, where the agent needs to move the object a certain distance. The agent starts at an initial position depicted by the flag and goes through changes in its sensor state (s_t), exemplified by the gripper, by taking actions. The episode is limited to contain $T = 9$ state transitions.

Future The future strategy considers transitions that happened at the timestep the hindsight instruction was returned, and all thereafter. More precisely, given the hindsight signal at timestep t, we consider sampling hindsight transitions at time index i for which

$t \leq i$, where $t \in \{0, \ldots, T\}$ and thus $t \leq i \leq T$. Intuitively, the timestep that triggered the hindsight transition and all the following ones satisfy the goal condition. In other words, a stack of cubes remains stacked for several steps after the stacking has been completed, hence the transitions where the hindsight goal is already achieved are also those with higher state or Q-values (see Subsection 2.2.2). As we cannot guarantee the stack to hold in place, we only provide a sparse reward of 0 at $i = t$ and a reduced penalty of -0.9 otherwise. In Figure 4.7, we illustrate the trajectory and the **future** hindsight goal sampling. The flag highlights our starting position, with the original goal as a red circle and a green circle as a non-goal object, moved around. Considering the simplified instruction *"push red"*, the agent attempts to achieve this goal but actually ends up at s_5, pushing the wrong object, namely the *"green"* one. As this is not the originally proposed *"red"* object, a mechanism triggers the hindsight relabeling. For the wrongly touched object, a **future** hindsight transition is sampled in the following way. The environment detects the interaction with the wrong object at state s_5. The following states s_6 and s_7 show how the pushing behavior unfolds, while s_7 towards s_9 illustrate the agent turning into a resting position, avoiding further changes to the already achieved state. The **future** method samples from the region i of states as potential hindsight transitions. As depicted by the range spanning from t to T, the transition at timestep t and k additional transitions from the range i are relabeled for learning.

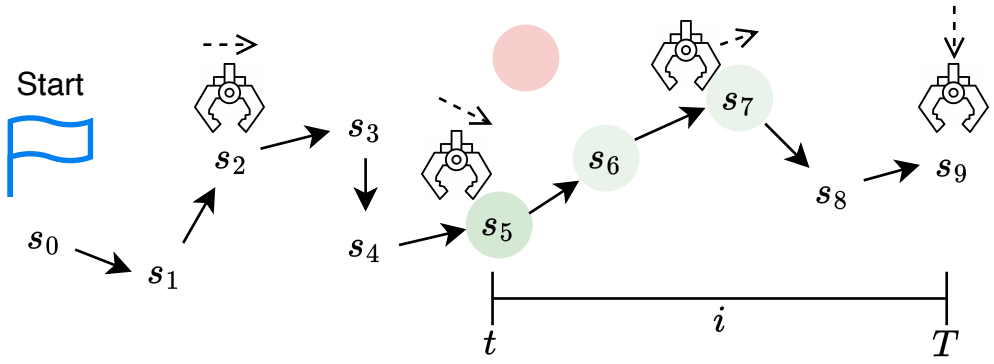


Figure 4.7: This figure illustrates the **future** sampling strategy. We depict the red goal object and the green object that represents the virtual goal. The strategy considers the state s_5 and following future states of the episode $s_{5\ldots9}$, resulting in the region i, from which we draw candidates for hindsight learning.

Episode Our episode strategy prescribes the sampling of hindsight transitions, slightly different from HER[1], as the range i, for $0 \leq i \leq t$ with $t \in \{0, \ldots, T\}$. The main idea is that every step taken up until the transition at timestep t has in a way contributed to the

[1]With the episode strategy, HER considers every state of the episode an achieved hindsight goal.

hindsight success. We reward those transitions with a reduced penalty of -0.9 and the actual hindsight signal at timestep t with a sparse reward of 0.0

Figure 4.8 illustrates the **episode** strategy. Different from the **future** strategy, all the episode transitions from s_0 to s_5 are considered the predecessor states that contributed to ending up at the virtual goal s_5. Intuitively, these states represent the agent approaching the green object, hence a desired behavior for a hindsight goal like g'_ℓ = *"push green"*.

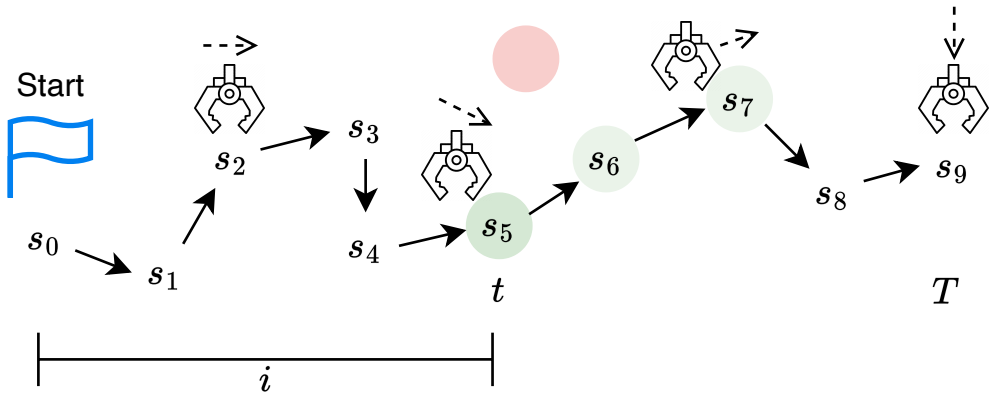


Figure 4.8: The **episode** strategy considers all transitions i that happened before the hindsight feedback was returned, $i \leq t$. The notion is that all the states ($s_{\leq 5}$) contributed in achieving the hindsight goal configuration, hence pushing the green object.

Final With the final strategy, we only consider those transitions where the hindsight signal was triggered, hence $t = i$. For $k > 1$, additional transitions are selected that are direct predecessors of the final transition as far as $t - k$. Those transitions are replayed with a reduced penalty of -0.9, while the actual final step at timestep t is replayed with a reward of 0.0. In Figure 4.9, we illustrate the trajectory and the hindsight goal sampling of this strategy. The state s_5 at timestep $t = 5$ is replayed with the sparse reward, while with respect to the parameter k, the preceding transitions ($s_{5-k\ldots5}$) are included with the reduced penalty. This is different from the replay mechanism used in HER (Andrychowicz et al., 2017), as we aim to provide a complementary version for the **final** strategy and account for the differences in language-conditioned RL.

In Figures 4.18 and 4.20, we provide the experimental results for the proposed replay strategies based on our prior work Röder et al. (2022), to evaluate which one fits our language-conditioned setup.

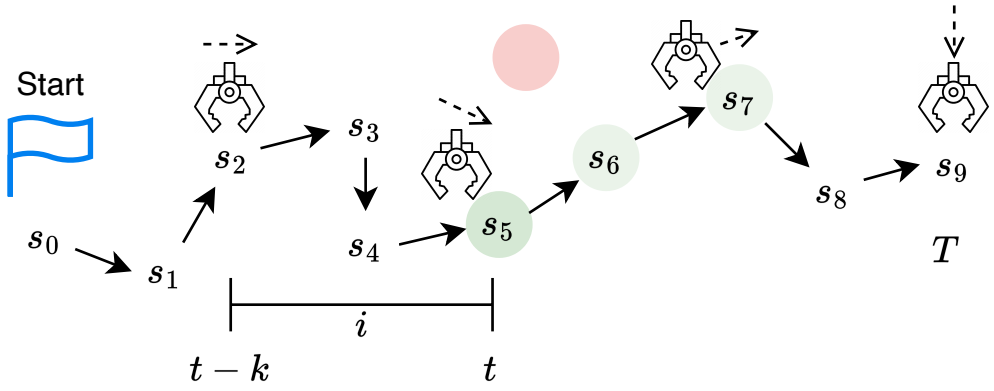


Figure 4.9: With the **final** strategy, we sample transitions that, at first, only correspond to the states where the hindsight mechanism was triggered (s_5) and the predecessor transitions leading to that state, while at most go k steps back in time.

4.6 Hindsight Bias

In the subsequent section, we highlight the problem of hindsight bias.

A shortcoming of hindsight replay and expert feedback, in particular, is that it assumes the hindsight instruction g'_ℓ would have produced a similar behavior τ as the original goal g_ℓ. A fundamental claim of the notion of hindsight bias is that we cannot assume those to lead to a similar or even the same trajectory. This drawback is known as hindsight bias, as recently shown by Bai et al. (2021). Our language-conditioned setup involves a specific kind of hindsight bias, as the hindsight goal cannot be directly derived from every state visited, but only based on those where an interaction with an object occurred. Formally, it describes a mismatch of likelihoods for the original goal instruction and the hindsight goal, by which the agent is conditioned to generate a trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T)$ (see Equation 4.1).

$$p(\tau \mid g'_\ell) \neq p(\tau \mid g_\ell) \tag{4.1}$$

Like Bai et al. (2021), one could measure the bias for a provided trajectory τ, the original goal g_ℓ and the hindsight goal g'_ℓ with Equation 4.2.

$$B(\tau) = \log p(\tau \mid g'_\ell) - \log p(\tau \mid g_\ell) \tag{4.2}$$

A problem of having perfect hindsight instructions right from the start, like in the case of **HEIR**, is that the policy, which is randomly exploring in the early phase of learning, only stumbles upon a correct or a hindsight solution by chance. As the policy is still learning to conduct the mapping from states and instructions to reasonable behaviors, the term in Equation 4.2 is expected to be high (Bai et al., 2021).

When the learning progresses, the policy tends to make different mistakes by mixing up properties during the goal identification, but otherwise has a reasonable behavior to solve the task most of the time. In this case, *"reach the red cube"* and *"reach the blue cube"* trigger a very similar behavior, but may only differ in terms of their target goal location. Being overly optimistic with the hindsight learning in the former case could harm the learning. We addressed this shortcoming from our previous publication (Röder et al., 2022) by limiting the amount of hindsight instructions considered for learning. Furthermore, we now provide a limit that at most $\frac{1}{3}$ of the mini-batch is allowed to be replayed as hindsight transitions (cf. Line 10 in Algorithm 2). This limits the effect of introducing an overestimation bias, while still allowing the agent to benefit from the hindsight transitions for learning.

4.7 Hindsight Instruction Prediction from State Sequences

In this section, we outline the implementation of our egocentric speech mechanism for hindsight language learning.

It has been shown that children narrating their own activities tend to improve their reasoning and problem-solving capabilities (Berk, 1994). In a way, it helps them ground their language in bodily experiences and fosters language learning through repetition in a specific context. The idea of self-narration and egocentric speech dates back to the seminal works by cognitive scientists Piaget (1926) and Vygotskij (1985). A question that arises is:

> *"Does egocentric speech help children to foster language grounding, and can it help intelligent agents to learn language more efficiently?"*

In our review on embodied language understanding, we investigate the hints provided by cognitive science and developmental psychology, and put them into the context of current language-driven RL approaches (Röder et al., 2021). We identified implementations that partly already contain the relevant ingredients for advanced language understanding but have not been combined into one coherent approach. For this thesis, we focus on the ingredients that need to be considered to enhance language grounding when learning to follow instructions. The notion of egocentric speech has connections to self-supervision, where agents set their own goals or describe activities to themselves. This idea has been part of the research on *autotelic* agents (Colas et al., 2022), an approach of self-organization by exploring through intrinsic motivation and proposing self-generated goals.

As already discussed in Section 4.1, we derived the idea of our implementation from the work of Cideron et al. (2020). The authors proposed a method called Hindsight Generation for Experience Replay (HIGhER) which predicts language in hindsight based on a single state observation. In their setup, they utilize the well-known grid world environment called BabyAI (Chevalier-Boisvert et al., 2019) that is part of many works on language grounding (Mirchandani et al., 2021; Spilsbury and Ilin, 2022; Carta et al., 2023). The authors (Cideron et al., 2020) consider a single 7×7 symbolic state representation, where the symbols correspond to obstacles, objects, their colors, their shape and other entities of the world. Apart from the simplistic design of the state representation, the environment provides an advanced procedure for language instruction generation (Chevalier-Boisvert et al., 2019). However, because of the simplified state representation, **HIGhER** lacks the ability to distinguish more complex world interactions. In our case, it cannot discern the subtleties of behaviors like pushing or sliding an object (see Figure 3.14), as it only considers a single state observation. Furthermore, missing real-world physics limits the agent to only learning a very narrow representation of the language, namely grounded in static top-down views of the grid world (see Figure 4.3).

As a novel approach inspired by **HIGhER** and to mitigate the mentioned shortcomings, we propose Hindsight Instruction Prediction from State Sequences (HIPSS) (Röder et al., 2022), an autoregressive generative model that predicts, based on a history of state inputs,

a suitable language instruction for hindsight learning. Our approach is similar to language modeling (Bengio et al., 2003); however, we are not only conditioning on prior language but also on a sequence of state or pixel inputs, thus following the idea of machine translation (Sutskever et al., 2014). It might seem similar to visual question answering, but it mainly focuses on the robot's behavior rather than a static sensor input, accounting for the effect of actions and the physics of the world. With this thesis, we provide two implementations of our **HIPSS** approach to learn to translate a history of low-level state inputs into a hindsight goal instruction.

4.7.1 Sequence-to-Sequence Modeling

Sequence-to-sequence (seq2seq) modeling attempts to process a sequence in the first place, to output another sequence as a result. One of the first works employing such an encoder-decoder architecture dates back to the paper of Sutskever et al. (2014). For the remainder of this section, we focus on utilizing recurrent neural network (RNN) to build seq2seq models, which we already introduced in Section 3.2.4. Main applications for such models are machine translation (Bahdanau et al., 2015), text summarization, speech recognition, video captioning, and many others.

In Figure 4.10, we depict the main structure of a seq2seq model for an example translating German into English. First, the encoder, which is an RNN, processes the German sentence *"Wie geht es dir?"* to output the resulting context vector. One can think of the context vector as a condensed vector representation, encapsulating the abstract meaning of the input sequence. Following this, the decoder, also an RNN, processes the context vector and generates the English translation *"How are you?"*. This is done in an autoregressive manner, hence word by word, generating each output step of the target sentence based on the prior information within the context and the last word predicted.

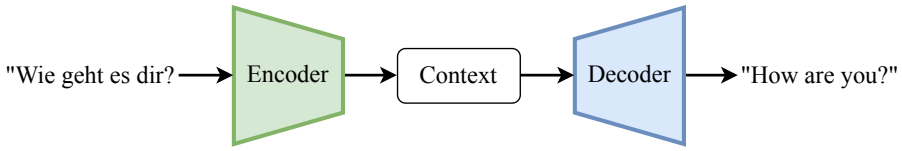


Figure 4.10: We depict a general example of translation, employing a seq2seq model according to the encoder-decoder architecture. As input, the German sentence *"Wie geht es dir?"* is processed by the encoder to output the context vector, which serves as an abstract representation of the input. Following this, the decoder processes the context vector to output the English translation *"How are you?"* word by word.

With this section, we provide the details of our second hindsight learning approach termed Hindsight Instruction Prediction from State Sequences (HIPSS) from our work Röder et al. (2022). It is a model that incorporates multiple modalities to implement the notion of egocentric speech by generating language goals based on sensorimotor experiences. This approach can be conceptualized as translating sensor data into language goals, effectively mapping from the physical space to the language space. We implement this

by first considering an encoder that processes a sequence of states τ_s, our state trajectory, encoding the essential features that are necessary to describe the behavior and world changes. The egocentricity aligns with our original motivation, as the state observations are determined by the sensors and the embodiment of the agent itself (see Section 1.1).

The full encoder procedure involves starting with a dummy hidden state h_0, hence a memory with arbitrary content, and the very first step of the trajectory with state s_0. In Equation 4.3, we depict the formalization for a single step j by the encoder network.

$$\text{enc}(s_t, h_{j-1}) = h_j \tag{4.3}$$

In general, t and j represent different notions of time. The t denotes the environment steps of an episode within the MDP framework. With j, we mark the stage of the processing steps of an RNN and cannot, as in large parts of the literature, reuse the t here. Although in the case of the encoder there is no difference, because it actually processes the environment steps in the same sequential manner as they are executed by the agent. The aforementioned function $\text{enc}(s_t, h_{j-1})$ recursively digests the whole input sequence. After completing this work step, the last hidden state $h_{|\tau_s|}$ provides the context vector z, where $h_j, z \in \mathbb{R}^{d_z}$ and d_z is a hyperparameter to determine the hidden state and context dimensionality. The encoder part can be formally described as in Equation 4.4, mapping the input of length T, like an entire episode, to the last hidden state z.

$$f_{\theta_{\text{enc}}} : \mathbb{R}^{T \times n} \to \mathbb{R}^{d_z} \tag{4.4}$$

Afterward, we employ the decoder described in Equation 4.5, as a function that recursively predicts the goal instruction word by word, considering the initial context vector $z = h_0$ and its own predictions afterward. Here we use the index i to denote the decoder processing steps to differentiate from the previous encoder RNN, but to regard the context involved in both networks. For each word prediction $\hat{w}_i$, the last hidden state h_{i-1} and the last word prediction $\hat{w}_{i-1}$ are considered.

$$\text{dec}(\hat{w}_{i-1}, h_{i-1}) = (\hat{w}_i, h_i) \tag{4.5}$$

We provide the formal description of a target sequence distribution given the context vector according to Equation 4.6. Each of the target vectors in $\mathbb{R}^{|V|}$ represents an unnormalized probability distribution over the words of the vocabulary V (cf. Section 3.2.3). We normalize these logits with the softmax function (see Equation 2.9), to obtain a real probability distribution from which we sample our next-word predictions.

$$f_{\theta_{\text{dec}}} : \mathbb{R}^{d_z} \to \mathbb{R}^{L \times |V|} \tag{4.6}$$

This autoregressive idea of seq2seq models is formalized by the chain rule of probability. In Equation 4.7, we depict this procedure as the probability of the language prediction conditioned on the state trajectory $\tau_s = (s_0, s_1, \ldots, s_T)$, hence the context vector z and the last word prediction w_{i-1}, with the aim to predict the hindsight goal instruction $\hat{g}_\ell = (w_0, w_1, \ldots, w_L)$.

$$p(w_0, \ldots, w_L \mid s_0, \ldots, s_T) = \prod_{i=0}^{L} p(w_i \mid z, w_{<i}) \text{ where } s_t \in \mathbb{R}^n \text{ and } w_i \in \mathbb{R}^{|V|} \tag{4.7}$$

Each conditional probability $p(w_i \mid z, w_{<i})$ is actually the softmax distribution according to the vocabulary size $|V|$, derived from w_i containing the logits.

The implementation of our seq2seq model solely employs the Gated-Recurrent Unit (GRU) introduced in Section 3.2.4. We depict a more detailed architecture description of **HIPSS** in Figure 4.11, which we explain in the following. For each state s_t of the state trajectory τ_s, the current input is processed by an MLP followed by a GRU, that outputs a new hidden state h_j based on the past hidden state h_{j-1}. Similar to Figure 3.7, we visualize this by unrolling the loop connection of a single GRU cell. This hidden state provides both the model's output and the memory for the current processing step t. At the very first step, we need to provide an initial hidden state h_0, that is randomly initialized or filled with zeros. In the next step, the last hidden state h_j and the next state s_{t+1} serve as input for the next MLP and GRU combination. This continues until all instances of the sequence are processed. As a result, the last hidden state h_T, where T corresponds to the length of the trajectory, serves as context vector z. This context is passed on to the decoder, which also consists of a GRU and an MLP. It provides the initial hidden state for the decoding procedure, hence both encoder and decoder require the same hidden state dimensionality, $|h_j| = |h_i|$. For each step, the decoder outputs the logits for the words in our vocabulary $\hat{w}_i$, based on the last hidden decoder state h_{i-1} and the last word logits predictions $\hat{w}_{i-1}$. For the first decoder step $i = 1$, the last hidden state $i = 0$ is the provided context vector by the encoder paired with the previous prediction $\hat{w}_0$, the placeholder symbol `<pad>`. This is a common way to kick-start the decoding routine, but it is also a symbol used to provide the technical required *padding* for shorter sequences, hence filling them up to obtain a mini-batch of inputs with equal length. The `<pad>` symbol is considered to be part of the vocabulary $|V|$.

We summarize the involved vectors and their dimensionality:

- Encoder input $\tau_s \in \mathbb{R}^{T \times n}$, with trajectory length T and state dimensionality n
- Encoder hidden state $h_j \in \mathbb{R}^{d_z}$, where d_z is a hyperparameter
- Context vector $z \in \mathbb{R}^{d_z}$, as last hidden state h_T of the encoder
- Decoder input with placeholder `<pad>` embedding $\hat{w}_0 \in \mathbb{R}^{|V|}$ and encoder context z
- Decoder hidden state $h_i \in \mathbb{R}^{d_z}$, with the same hidden size as the encoder
- Decoder output for each step $\hat{w}_i \in \mathbb{R}^{|V|}$ with $i \in \{0, 1, \ldots, L\}$, representing the word probabilities as logits

Our **HIPSS** model essentially defines a mapping $f : \mathbb{R}^{T \times n} \to \mathbb{R}^{L \times |V|}$ from a state trajectory to a sequence of raw word probabilities (logits) when training the network. For the actual inference, hence when predicting hindsight goal instructions, the output is a sequence of words in $\mathbb{R}^L$, where L is the maximum sentence length. This is the result of transforming the logits into distributions via the softmax function, from which we sample the word predictions.

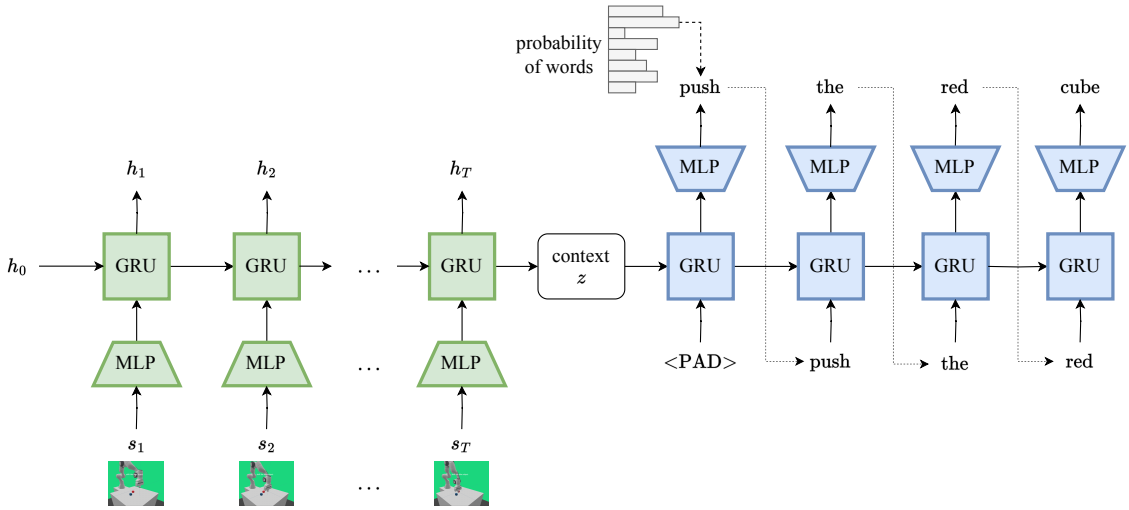


Figure 4.11: In the case of **HIPSS**, the encoder RNN takes as input the past hidden state h_{j-1} and the current Markov state s_t embedded by an MLP, to generate a new hidden state h_j. When all vectors of the sequence are processed, the last hidden state h_T, where $T = |\tau_s|$ is the length of the state trajectory, is now our context vector z, an abstract representation of the whole input sequence. The decoder RNN takes as input a starting symbol `<pad>` and autoregressively generates the sentence *"push the red cube"* word by word, by considering the previous decoder hidden state h_{i-1} and the last word prediction $\hat{w}_{i-1}$. Word predictions, made by the MLPs mapping the hidden state to vectors according to the vocabulary size, refer to logits of categorical probability distributions, that we normalize using the softmax function. We sample from these distributions when generating a sentence. For the first word *"push"*, we illustrate this as a vertical histogram for a small sample vocabulary for illustrative purposes.

In this section, we presented the architecture of **HIPSS**, published in our prior work Röder et al. (2022), which employs the seq2seq model, an encoder-decoder structure (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015), built on recurrent neural networks, as an approach to translate behaviors to linguistic goals. Recurrent neural networks for language processing have recently been dominated by a more advanced deep learning architecture used in current state-of-the-art models such as BERT (Devlin et al., 2019) and GPT (Radford et al., 2018). Those neural networks avoid the RNN memory bottleneck of the hidden state and address the problem of retaining information for several hundreds of processing steps. This resulted in overall greater performance across different types of tasks and improved generalization when training on large datasets. The advanced architecture referred to is called the Transformer (Vaswani et al., 2017) and is explained in the following section.

4.7.2 Transformer Encoder-Decoder Architecture

Another approach for sequence modeling tasks without recurrence is the Transformer architecture by Vaswani et al. (2017). The authors also employ the idea of an encoder-decoder architecture that first encodes a source sentence into a sequence of real-valued vector representations, which then gets consumed by the decoder to generate a target sentence. However, this approach is based on the notion of attention, first introduced in the context of machine translation by Bahdanau et al. (2015). The intuition behind attention is to allow both the encoder and the decoder to focus on different parts of the input sequence while generating the context or the target sequence, respectively. The Transformer architecture has found widespread application in many large-scale systems. One notable example is the Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019), which utilizes only the encoder part to produce context-sensitive embeddings for various NLP applications (previously mentioned in Section 3.2.3). Another variant is the Generative Pre-trained Transformer (GPT) (Radford et al., 2019; Brown et al., 2020; Ouyang et al., 2022; OpenAI, 2023), which only employs the decoder part for language modeling, generating text based on an initial prompt. The Transformer has proven to be a versatile architecture, even finding application in computer vision as a replacement for convolutional neural networks with the *Vision Transformer* (Dosovitskiy et al., 2020) or offline RL as *Decision Transformer* (Chen et al., 2021a).

A compelling reason to utilize Transformers in the context of **HIPSS** is their recent success across nearly all time series tasks, where they significantly outperform recurrent DNN models based on the LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014). This superior performance is attributed to their ability to not only rely on a compressed hidden state representation like RNNs (the context vector from the previous Subsection 4.7.1), but can also attend to hidden representation of each input step of the source sequence and the generated outputs of the target sequence at **all times**. This capability enables Transformers to better capture long-range dependencies and achieve overall superior performance (Vaswani et al., 2017). Furthermore, RNNs require processing the entire sentence to, *e.g.*, comprehend the potentially ambiguous meaning of a word like *"bank"* in a sentence like in the example in Section 3.2.3.

With Figures 4.11 and 4.12, we provide a comparison of both approaches to seq2seq modeling. In the former seq2seq approach from Subsection 4.7.1, the decoder only had access to the context vector as the last hidden state of the encoder, encapsulating the entire input sequence information. This is vastly different from the Transformer, where the decoder has access to all the hidden encoder states at all times via the attention mechanism when generating the target sequence. This is very similar to the aforementioned self-attention from Section 3.2.4, where we calculate the relatedness of words of a single sequence. In the case of the decoding procedure of a Transformer, we refer to it as cross-attention when two different sequences are involved. The target predictions by the decoder regard the sequence of encoder representations and its own predictions generated so far.

The key aspects distinguishing the Transformer architecture from previous approaches are primarily its use of the self-attention mechanisms (see Section 3.2.4), and incorporating a combination of numerous established deep learning components into one coherent

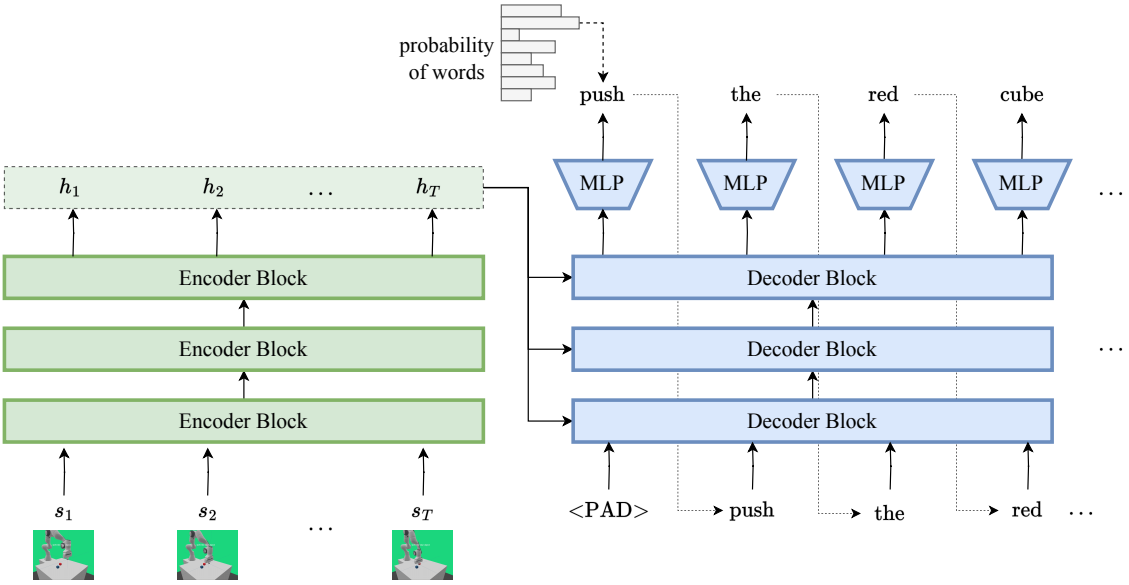


Figure 4.12: With **HIPSS** using a Transformer (Vaswani et al., 2017) instead of RNNs, the encoder, consisting of encoder blocks, processes the whole trajectory of states $s_0, \ldots, s_T$, to generate a sequence of hidden encoder states $h_0, \ldots, h_T$. The decoder, made up of decoder blocks, takes as input the full sequence of encoder states and initial target `<pad>` to generate the first logit vector that gets mapped to a probability distribution employing the softmax function (as illustrated by the small vertical histogram), from which the prediction *"push"* is derived. Actual word predictions are made by the MLPs that map the hidden representations to vectors according to the vocabulary size $|V|$. While decoding, a cross-attention operation is employed to consider the encoder states and the decoder predictions generated so far, indicated by the dashed lines.

architecture. Attention alone would not have determined the success of Transformers; additional parts and layers involved are the usual feed-forward network layers (MLPs), residual connections, dropout layers (Subsection 2.1.3), and layer normalization (Section 2.1.3), to name the most important. The Transformer model (Vaswani et al., 2017) employs these for each sub-layer (e.g., self-attention and feed-forward layers) before the residual connection. The combination of layer normalization and dropout has been widely-used in many deep learning architectures (Radford et al., 2018; Devlin et al., 2019; Radford et al., 2019), stabilizing the training and improving the convergence of the model. We provide a brief introduction to the residual connections in the following paragraph.

Residual Connection Residual connections, also called skip-connections, are commonly used in large deep neural networks (He et al., 2016; Dosovitskiy et al., 2020; Brown et al., 2020) to ease information flow by bypassing layers and directly propagating information. In the following, we describe mathematically what this means for a single connection with the input x, the residual mapping $F(x)$, and the desired mapping $H(x)$. Instead of directly

mapping $H(x)$, we have:

$$F(x) = H(x) - x$$

hence

$$H(x) = F(x) + x$$

achieved through the element-wise addition (cf. Figure 4.13).

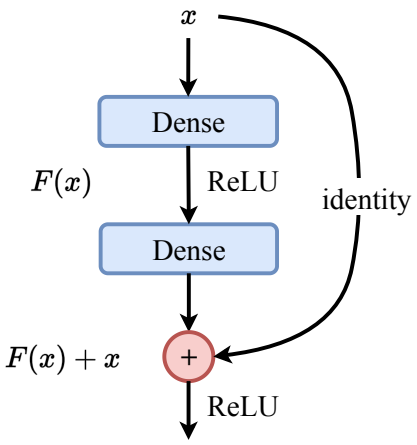


Figure 4.13: Illustration of a residual connection with two dense layers, ReLU activations functions, and the skip-connection (identity).

This formulation encourages the network to learn the identity function $H(x) = x$ by driving the output of $F(x)$ to zero. Residual connections help to address the vanishing and exploding gradient problem, and have been shown to be crucial for training very deep neural networks (He et al., 2016).

Multi-Head Attention The self-attention mechanism was previously presented as scaled dot-product attention based on the work of Vaswani et al. (2017) to encode goal instructions in Section 3.2.4. This formulation was a special case of the originally proposed multi-head attention (MHA) that employs multiple sets of parameters, which we are going to explain in this paragraph.

As a recap of the self-attention, now using matrix notation, encoding the input sequence $X \in \mathbb{R}^{T \times n}$ results in our query, key, and value representation of the input, where $Q \in \mathbb{R}^{T \times d_k}$, $K \in \mathbb{R}^{T \times d_k}$, and $V \in \mathbb{R}^{T \times d_v}$. We can denote this in compact format according to Equation 4.8.

$$\begin{aligned} Q &= XW^Q \\ K &= XW^K \\ V &= XW^V \end{aligned} \tag{4.8}$$

To conduct the linear transformation of X, we define the trainable parameters in terms of the projection matrices $W^Q \in \mathbb{R}^{n \times d_k}$, $W^K \in \mathbb{R}^{n \times d_k}$, and $W^V \in \mathbb{R}^{n \times d_v}$, where n defines the

dimensionality of the input (*e.g.*, $d_{\text{embedding}}$ for word embeddings following Section 3.2.4), T the length of the input sequence, d_k the dimensionality of the query and key projections, and d_v the dimensionality of the value projections, respectively. Intuitively, one can think of the scaled dot-product attention as a soft dictionary lookup (Murphy, 2022, Section 15.4.1). The query defines what we are searching for by moving the high-dimensional input embeddings into a specific direction within the vector space. With the keys, we define the information that we want the query to match; by transforming the input embeddings in a way that they become more sensitive to their corresponding query words. For instance, the query of the word *"bank"* determines how much attention this word pays to other words, like *"finance"* and *"sand"*. Finally, the value matrix determines the amount of input information incorporated into the original input embedding as a linear combination, according to the attention as calculated by $\text{softmax}(\frac{QK^T}{\sqrt{d_k}})$. The resulting function that represents this procedure according to Vaswani et al. (2017) is formulated in Equation 4.9.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4.9}$$

In the aforementioned case of language encoding Section 3.2.4, we use the single-head attention as a special case of the multi-head setting. Employing multiple heads allows the Transformer to generate different contextual representations simultaneously, where each head operates independently with its own set of query, key, and value parameters. We now define a head by a tuple (W_i^Q, W_i^K, W_i^V), where the output of a single head with index i is defined according to Equation 4.10.

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{4.10}$$

The output of a whole attention block is made of the H heads as a concatenation of their individual output matrices. Following this, the output is additionally processed by the so-called output MLP defined by the weights $W^O \in \mathbb{R}^{H \cdot d_v \times d_o}$ (see Equation 4.11), with d_o as the hyperparameter for the dimensionality of the output layer.

$$\text{MultiHeadAttention}(Q, K, V) = [\text{head}_1, \ldots, \text{head}_H]\, W^O \tag{4.11}$$

The purpose of using multiple attention heads is that in the case of language translation, one head might be looking at the subject of the sentence, another head at the object, and one at the gender of the subject. In our setup using **LANRO**, one head might be looking at the motion while another head is focusing on the goal object's color or location. Figure 4.14 illustrates the H heads and how they perform the calculation of the query, key, and value matrices in parallel.

As the multi-head attention is a way of processing sequences by analyzing all words simultaneously, we need a technique for the encoder to ignore placeholder symbols like `<pad>` when the input is shorter than the context length T, such as when the trajectories of a mini-batch have different lengths. In the case of the decoder, we must prevent the model from looking at future words by keeping its input limited to the so-far generated outputs via the autoregressive process through causal masking. We illustrate this part of the decoder process in Figure 4.15. The masking is implemented by setting the appropriate

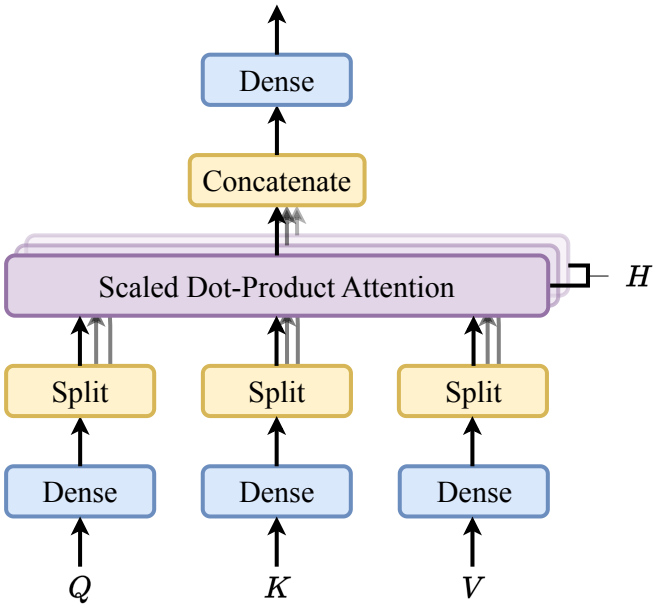


Figure 4.14: The illustration shows the multi-head attention for H heads running in parallel. The dense layers (feed-forward connections) process the query Q, key K, and value V matrices. Those are split among the multiple heads and afterwards combined via concatenation. The last dense layer defines the output with weights W^O. Redrawn figure based on Vaswani et al. (2017).

attention weights to $-\infty$ for the positions that should not be exposed to the decoder before applying the softmax function (cf. Section 3.2.4 in Equation 3.16).

The overall Transformer architecture is illustrated in Figure 4.16 and shows the stacks of N_{enc} individual encoder blocks (left) and N_{dec} decoder blocks (right), encapsulated by the gray boxes. The encoder blocks consist of stacks of multi-head attention layers with normalization layers, followed by a feed-forward neural network layer (MLP). The encoder is complemented by a so-called positional encoding applied to the input embeddings, as there is no notion of order or time when using self-attention. Through this encoding approach, the model gains the ability to detect the sequential structure of the input. Formally, this mechanism establishes the mapping from an input sequence of sensor states to a sequence of hidden encoder states of size d_o (see Equation 4.12).

$$f_{\theta_{\text{enc}}} : \mathbb{R}^{T \times n} \rightarrow \mathbb{R}^{T \times d_o} \tag{4.12}$$

The decoder part employs a multi-head attention layer, of which the first block processes the target sequence generated so far, while the second block conducts the cross-attention that considers both the encoder and the decoder hidden states simultaneously (as depicted by the arrows). This leads to the formal mapping from a sequence of encoder states to the target sequence, which is different from the RNN-based seq2seq model that only had the

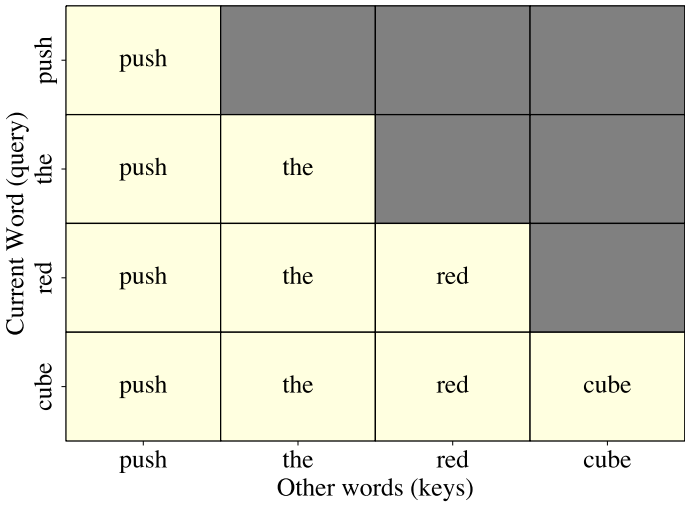


Figure 4.15: We illustrate the causal masking used within the Transformer decoder during training. In the decoder, each word (the query) attends to other words (the keys) based on the computed attention scores according to Equation 3.16. By setting the attention scores of future words to $-\infty$ (depicted by the gray cells), we apply causal attention masking that prevents each position from attending to subsequent words. This is necessary because the attention mechanism processes the entire input sequence in parallel without inherent sequential ordering, and we need to prevent information leakage from future words. For instance, when processing the word *"red"*, the model is allowed to attend only to previous words like *"push"*, *"the"*, and *"red"* itself, but not to future words like *"cube"*.

single context vector (see Equation 4.13).

$$f_{\theta_{\text{dec}}} : \mathbb{R}^{T \times d_o} \to \mathbb{R}^{L \times |V|} \tag{4.13}$$

Ultimately, the output is sent through a feed-forward network layer, before a linear layer and the softmax function are applied. Both blocks and the feed-forward layer are followed by a residual connection paired with layer normalization. The final output is a vector of size $L \times |V|$, the length of the sentence times the total number of words in the vocabulary. From this categorical distribution, one samples sentences either greedily, or via a beam search algorithm, according to the chain rule of probabilities (cf. Equation 4.7).

In the following, we explain how we learn to translate MDP states into language goals, which we use for hindsight learning in the sense of egocentric speech. The architecture of **HIPSS**, for which we employ the Transformer encoder-decoder architecture, is now referred to as Transformer-based Hindsight Instruction Prediction from State Sequences (THIPSS).

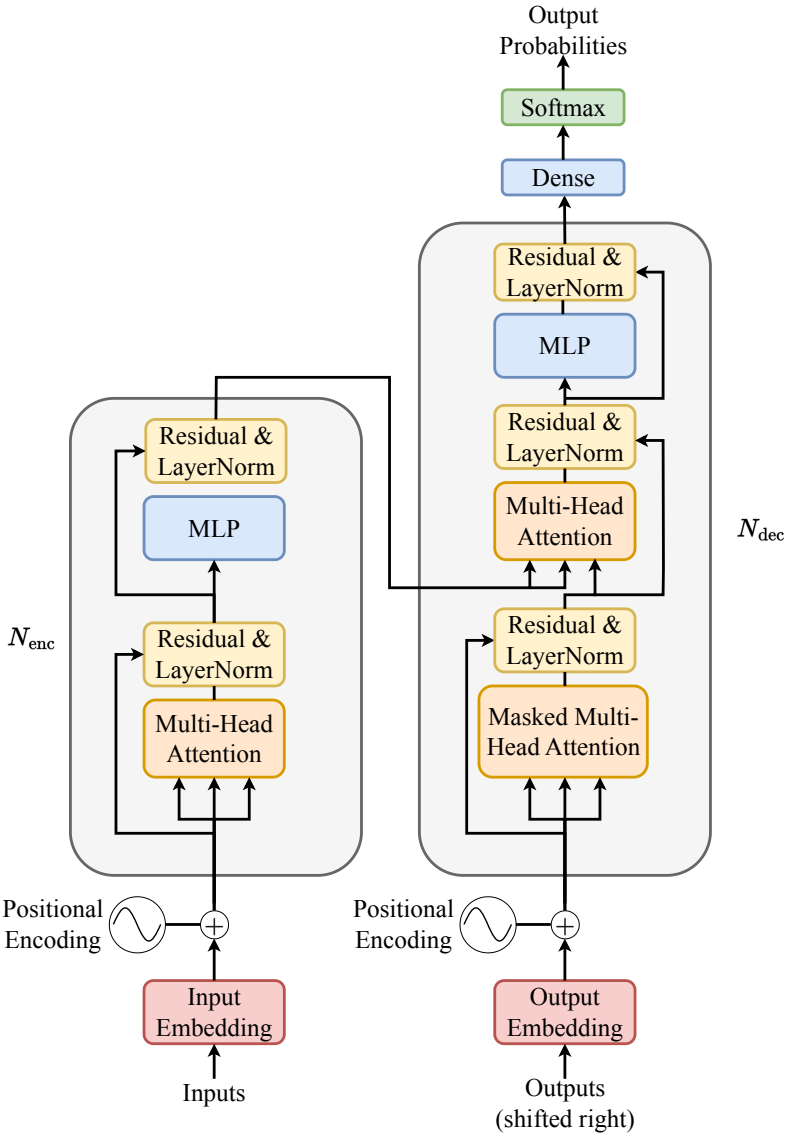


Figure 4.16: Illustration of the Transformer architecture, with N_{enc} encoder blocks on the left and N_{dec} decoder blocks to the right. At the bottom, we have the input and the output embeddings to which we apply the positional encoding. Afterwards, the query, key, and value matrices are provided as input to the multi-head attention (MHA) layers (cf. Figure 4.14). In case of the decoder (right side), the aforementioned causal masking is applied (not shown) to prevent looking into the future. Arrows skipping the masked MHA are the residual connections from Figure 4.13, that we jointly illustrate with the layer normalization from Section 2.1.3. Following, we have MLPs processing the output of the MHA. The decoder employs another MHA layer that implements the cross-attention, by receiving the query and key inputs from the encoder's output. Following, the decoder outputs a logit vector (dense layer), that gets normalized by the softmax function to obtain the output probabilities. Redrawn figure based on Vaswani et al. (2017).

4.7.3 Learning to Predict Hindsight Instructions

In both cases of our original seq2seq model **HIPSS** (Röder et al., 2022) and the Transformer encoder-decoder architecture **THIPSS**, we need to collect episode trajectories that provide training examples of the policy generating rewarding language-conditioned behaviors. Our setup, for the most part, replicates a machine translation setting, but here we translate low-level sensorimotor state encodings of a particular behavior into a sentence describing them. We can express this as a conditional probability over all possible descriptions $p(g_\ell \mid \tau_s) = p(w_0, w_1, \ldots, w_N \mid \tau_s)$, where the words w_i are coming from our vocabulary (see Section 3.2.3). The autoregressive procedure becomes clearer when we break the joint probability of words into a product of conditional probabilities, as in Equation 4.14.

$$p(g_\ell \mid \tau_s) = p(w_0 \mid \tau_s)p(w_1 \mid \tau_s, w_0)p(w_2 \mid \tau_s, w_0, w_1) \ldots p(w_N \mid \tau_s, w_0, \ldots, w_{N-1}) \tag{4.14}$$

We train **HIPSS** and **THIPSS** next to the baseline algorithm, therefore they do not interfere with any gradient computations of the RL objective. The overall optimization attempts to find language goals that maximize the conditional probability of the words according to Equation 4.15.

$$\underset{g_\ell}{\operatorname{argmax}}\, p(g_\ell \mid \tau_s) \tag{4.15}$$

We employ our so-called state trajectory τ_s to emphasize that it only consists of the states visited, excluding the actions that are part of the usual RL trajectory τ. For the training, we are using a variant of the cross entropy loss. For numerical stability, the optimization employs the log-softmax function to convert the logits of the network outputs into log probabilities. In Equation 4.16, we provide the loss, where ϑ represents our model parameters and $y^{(i)} \in \mathbb{R}^{L \times |V|}$ the categorical distribution of the words derived from g_ℓ. We use one-hot encodings for the target words, where all probability mass is assigned to the specific word index, effectively representing the target distributions.

$$L(\vartheta, \mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} y_\ell^{(i)} \log(\text{softmax}(M_\vartheta(\tau_s^{(i)}))) \tag{4.16}$$

The high-level training procedure goes as follows:

- Harvest successful trajectories with policy π_θ for the training and validation dataset $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$
- When training, sample trajectory-instruction tuples from the dataset $(\tau_s, g_\ell) \sim \mathcal{D}_{\text{train}}$
- Update the model according to the loss in Equation 4.16 with gradient descent using the Adam optimizer (Kingma and Ba, 2015)
- Periodically evaluate the validation accuracy with samples from the validation dataset $\mathcal{D}_{\text{val}}$
- If HIPSS reaches a predefined validation accuracy threshold, relabel failures by generating instructions for hindsight replay

To evaluate if the model is good enough to generate instructions for hindsight learning, we track the per word accuracy on the validation dataset.

4.8 Hindsight Language Learning Methods Overview

In the following, we provide an overview of the two different sources of the hindsight instructions.

The hindsight learning approaches **HEIR** and **HIPSS** can be summarized by Figure 4.17. In the case of a wrong object interaction, we either employ **HEIR** to replace the instruction by the hindsight expert feedback or use a variant of **HIPSS** to provide a self-generated instruction for the experienced trajectory. The agent is replaying the altered transition with a positive reward to improve the learning in both cases. For the successful outcomes, we store the state trajectories and corresponding instructions as samples for training. This part is where the self-supervision provides the data and the labels without human intervention.

While we depict both approaches as if they are applied in an online setting, we use them slightly differently in combination with the off-policy learning algorithms based on SAC. Hindsight instructions are generated when sampling data from the replay buffer. The transitions that we sample contain information about wrong object interactions, which we utilize to randomly decide if a transition is relabeled to incorporate the hindsight instruction and the hindsight reward. Using **HEIR** simply involves accessing the information about the wrong object interaction, and the hindsight instruction the caretaker returned. For **HIPSS**, we feed the trajectory of the corresponding episode of the transition into the model to generate the hindsight instruction. In Algorithm 4, we provide the pseudocode of **HIPSS** and **THIPSS**, respectively.

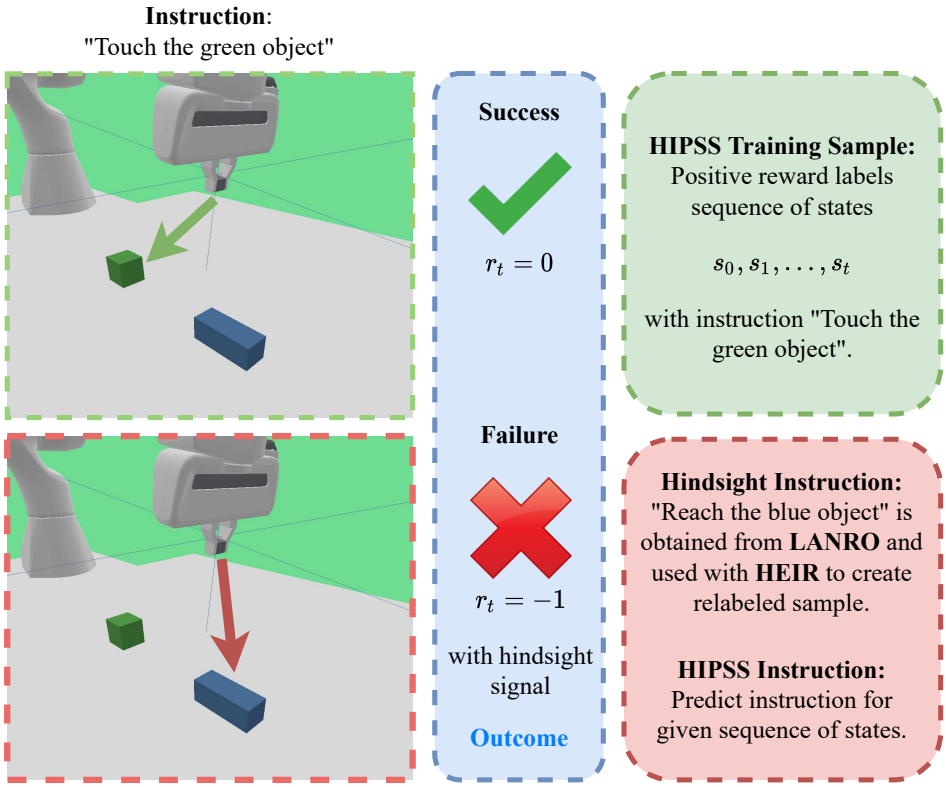


Figure 4.17: To the left, we illustrate the agent interacting with the environment. In the top left, the agent is rewarded by touching the green object as instructed by *"Touch the green object"*. On the bottom left, we present a failure, where the agent touches the wrong object, in this case a blue cuboid, for which a penalty is granted. On the right side of the figure, we illustrate the hindsight procedures that we trigger in each case. In the **successful case**, the outcome serves as an example for training **HIPSS**. In the **case of a failure**, at the bottom right, either **HEIR** or **HIPSS** generate an instruction in hindsight like *"Reach the blue object"* for learning – Figure taken from our work Röder et al., 2022.

Algorithm 4 Hindsight Instruction Prediction from State Sequences

π_θ	policy	M_ϑ	**HIPSS** model
$\mathcal{B}$	replay buffer	$\mathcal{D}_{\text{train}}$	**HIPSS** training dataset
α	**HIPSS** learning rate	$\mathcal{D}_{\text{valid}}$	**HIPSS** validation dataset

for episode $i = 0, \ldots, E$ **do**
  Sample initial state s_0 and language goal $g_\ell^i \sim \rho_{g_\ell}$
  for timestep $t = 0, \ldots, T-1$ **do**
    Sample action $a_t \sim \pi_\theta(\cdot \mid s_t, g_\ell^i)$
    Take environment step $s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t)$
    Compute reward $r_t \leftarrow \mathcal{R}_g(s_t, a_t, g_\ell^i)$
  end for
  if $r_t = 0$ **then**
    With probability 0.2, set $\mathcal{D} \leftarrow \mathcal{D}_{\text{valid}}$, otherwise $\mathcal{D} \leftarrow \mathcal{D}_{\text{train}}$
    Store successful trajectory-goal pair $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_{0:T}, g_\ell^i)\}$
  else if $r_t = -1$ and interplay with wrong object **then**
    Replace g_ℓ^i with predicted goal $\hat{g}_\ell^i \leftarrow M_\vartheta(s_{0:T})$
  end if
  Store transitions $\mathcal{B} \leftarrow \mathcal{B} \cup \left\{(s_t, a_t, r_t, s_{t+1}, g_\ell^i)_{t=0}^{T-1}\right\}$
  Update policy π_θ with mini-batch from $\mathcal{B}$
  for each gradient update step **do**
    Update **HIPSS** model $\vartheta \leftarrow \vartheta - \alpha \nabla_\vartheta L(\vartheta, \mathcal{D}_{\text{train}})$ using loss from Equation 4.16
  end for
end for

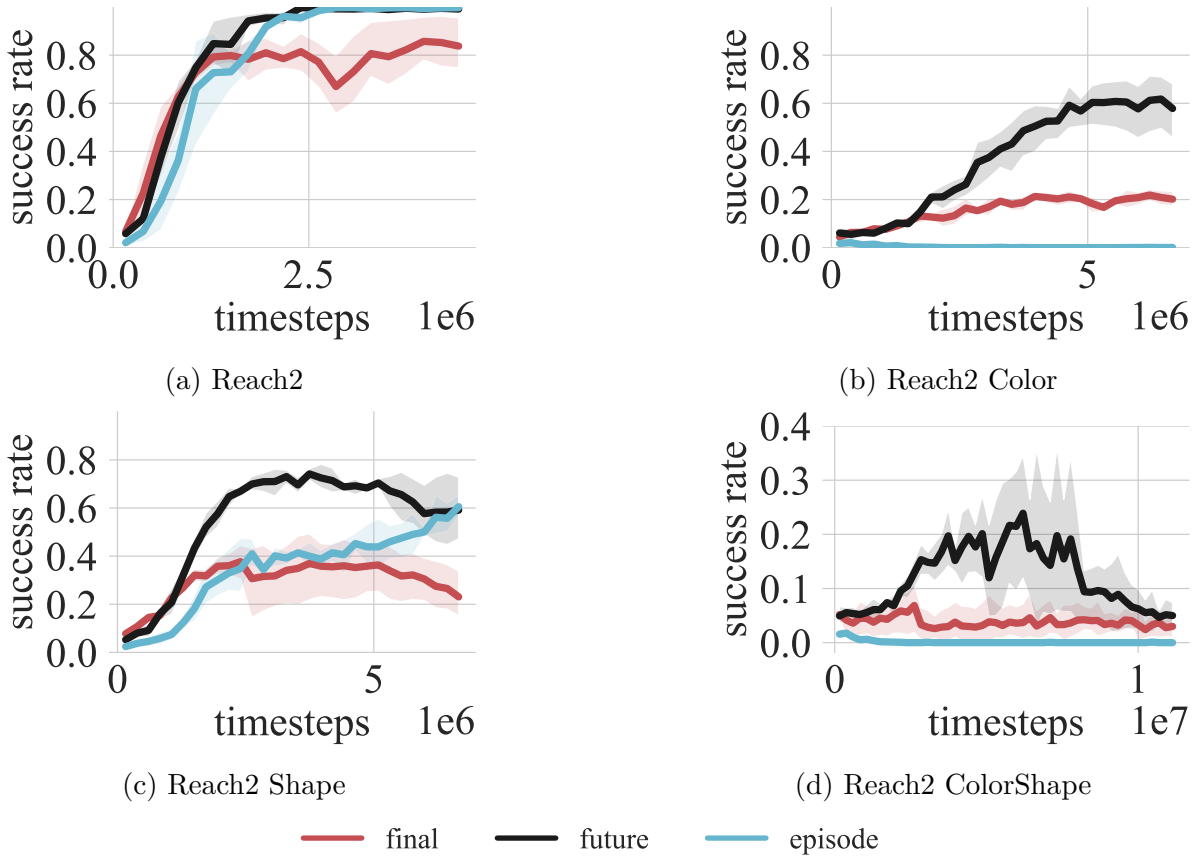
4.9 Experiments

The upcoming subsections highlight the results from our prior publication "Grounding Hindsight Instructions in Multi-Goal Reinforcement Learning for Robotics" (Röder et al., 2022). These results are followed by experiments for their recent adjustments made to address previous limitations. Central to this section are the outcomes of our proposed expert feedback routine, ***HEIR****, as well as the egocentric speech mechanisms, the seq2seq model* ***HIPSS*** *and the Transformer-based version* ***THIPSS****, for hindsight instruction replay.*

4.9.1 Results Replay Strategies

In Section 4.5, we introduced three types of replay strategies that could be used for hindsight instruction replay. Figure 4.18 provides our findings from Röder et al. (2022) for a selection of tasks from **LANRO** with 2 objects on the table, using **LCSAC** with our **HEIR** approach for experience replay. The graphs depict the mean success rates (y-axis) grouped by the different strategies relative to the environment steps taken (x-axis). We depict the 90% confidence interval of 3 random seeds. The **episode** strategy fails to provide reasonable results in the tasks containing more than the default colors (*Color* and *Color-Shape*). Intuitively, replaying any transition of the episode does not guarantee a meaningful interaction with a hindsight goal object, potentially having a detrimental learning effect. The **final** strategy, while not completely hampering the agent's learning, only manages to make modest progress in tasks with more colors and object shapes. With the **future** strategy, we obtain the best results across all levels of difficulty. However, in the case of Figure 4.18d, the success rate drops after 5 million environment steps, possibly caused by the aforementioned effect of the hindsight bias (see Section 4.6). This outcome aligns with the goal-conditioned **HER** approach from Andrychowicz et al. (2017), validating our replay approach implementation.

Remark These results might be influenced by the hindsight bias of **HEIR** identified in Section 4.6. In our recent experiments, we address this issue with new results presented in Subsection 4.9.3.



(a) Reach2

(b) Reach2 Color

(c) Reach2 Shape

(d) Reach2 ColorShape

Figure 4.18: The figures provide the results for our 3 proposed replay strategies **episode**, **future**, and **final**, that we tested across the different *reach* tasks of varying difficulty (*Default*, *Color*, *Shape*, and *ColorShape*). All the experiments were conducted with 2 objects on the table (denoted by *Reach2*), using the former **LCSAC+HEIR** combination (Röder et al., 2022). The figures show the environment steps (x-axis) and the success rate (y-axis). Experiments were run using 3 random seeds to plot the mean success rate and their 90% confidence interval (shaded area). While the *future* strategy yields the best results in almost all experiments, it significantly suffers from the effect of hindsight bias in Figure 4.18d, where the performance drops after 5 million environment steps. The experiments made use of the dense embeddings and the GRU as language encoder.

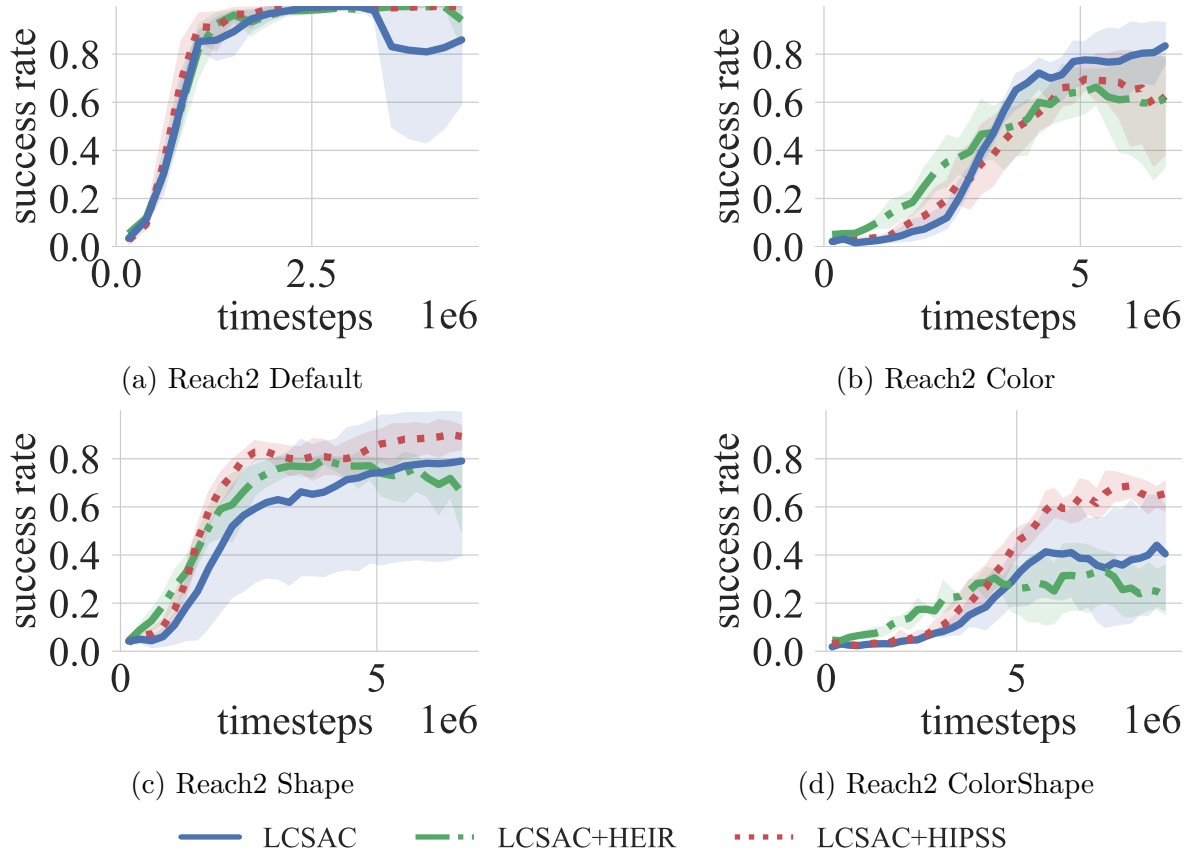
4.9.2 Grounding Hindsight Instructions

This section presents the experimental results for our hindsight expert feedback **HEIR** and the self-supervised approach **HIPSS**, as detailed in our publication (Röder et al., 2022). Figure 4.19 compares our baseline **LCSAC** with **LCSAC+HEIR** and **LCSAC+HIPSS**. All experiments were conducted using only the *reach* task, with the GRU as language encoder, the dense word embeddings, and the **future** replay strategy for hindsight learning. Each algorithm configuration involves 5 random seeds, depicted by the average success rate and its 90% confidence interval, with only 2 objects on the table.

The first experiment in the *Default* setting shows that the replay additions **HEIR** and

HIPSS perform comparably to our baseline **LCSAC**. Notable differences emerge only when task complexity increases by adding more colors or shapes.

In the *Shape* and *Color* configurations, **HEIR** exhibits initial performance increases that are potentially attenuated by a later effect of the hindsight bias (Section 4.6). In the most complex configuration, with additional shape and color combinations, **HIPSS** demonstrates a significant performance improvement, nearly doubling the success rate of **HEIR** and **LCSAC**. The performance of **HIPSS** in this case can be attributed to its ability to predict correct instructions in hindsight while the policy is still learning to ground shape and color words simultaneously, often confusing properties, thus interacting with incorrect objects (see Figure 4.5).



(a) Reach2 Default

(b) Reach2 Color

(c) Reach2 Shape

(d) Reach2 ColorShape

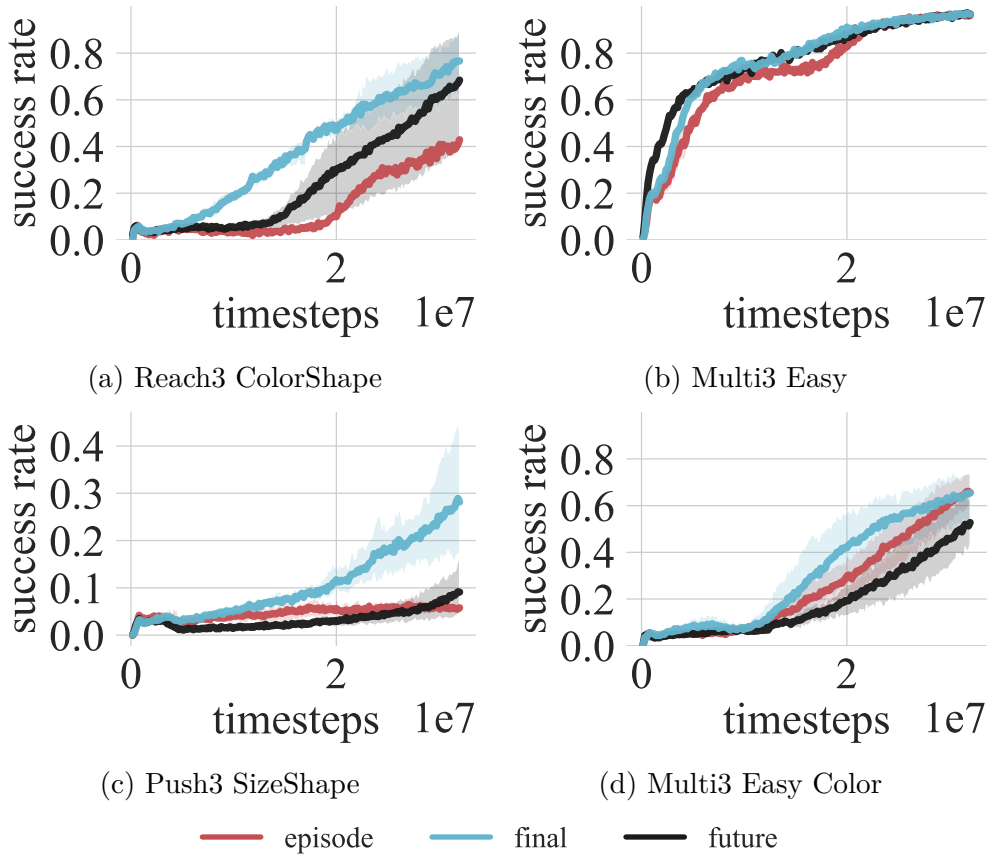
Figure 4.19: Experimental results from our work on grounded hindsight language learning (Röder et al., 2022). We show the performance of the baseline **LCSAC** and the extensions **HEIR** and **HIPSS** in the *reach* task with 2 objects on the table with 4 levels of difficulty (*Default*, *Color*, *Shape*, and *ColorShape*). The performance gain of **HIPSS** increases with the task complexity; hence a higher number of words and object appearances that could potentially be mixed up by the learning policy. We depict the environment steps on the x-axis, with the average success rate and the 90% confidence interval on the y-axis, employing 5 random seeds.

4.9.3 New Replay Strategy Results

In the following, we provide the latest replay strategy results to account for the fix of the hindsight bias using **LCDroQ+HEIR**, complementary to the results of our prior publication experiments shown in Figure 4.18. The experiments in Figure 4.20 were conducted in a small selection of challenging tasks, this time with 3 objects on the table. For the first case, we employ the *reach* task with additional object properties, as shown in Figure 4.20a, with the attention-based language encoder. A similar setting is presented in Figure 4.20c with additional size and shape property combinations. Here, the challenge is to cope with the presence of many property words regarding the color, size, and shape of the goal object that could easily get mixed up by the policy. In both cases, the **future** strategy provides the best performance, aligning with our previous results (Röder et al., 2022), followed by the **final** and **episode** options, in the case of Figure 4.20a, and with way lower performance in Figure 4.20c.

In addition, we also illustrate the results for the *multitask* setup with the *Default* (Figure 4.20b) and *Color* (Figure 4.20d) options. With this setting, we attempt to learn two distinct tasks, namely the *reach* and *push* tasks, simultaneously. While there are synergies through intra-task transfer, it also provides a case useful for hindsight learning, when the agent mixes up the word descriptions related to the task behavior. However, in this particular case, the replay strategies provide similar results, with no clear evidence to suggest which strategy to choose. For the case with many colors (Figure 4.20d), the **future** strategy leads by a small margin, closely followed by the **episode** strategy.

All the experiments comprise 5 random seeds that determine the average success rate and the 90% confidence interval (y-axis), compared to the environment steps (x-axis).



(a) Reach3 ColorShape

(b) Multi3 Easy

(c) Push3 SizeShape

(d) Multi3 Easy Color

Figure 4.20: Replay strategy results for the latest version of **HEIR** paired with **LCDroQ**, showing the mean success rate (y-axis) and the environment steps (x-axis). We are comparing the **episode**, **final**, and **future** strategies within the *reach* task with the *ColorShape* option (see Figure 4.20a), the *push* task with the *SizeShape* option (see Figure 4.20c), and the *multitask easy* setting with and without the *Color* option (see Figures 4.20b and 4.20d). Shown are the results for 5 random seeds with 3 objects on the table, utilizing the self-attention as language encoder. The shaded regions are calculated by the 90% confidence interval.

4.9.4 Full Hindsight Learning Experiments

The following sections expand on the insights from Subsection 4.9.2, based on our paper Röder et al. (2022), in the following ways:

- We present tasks with 3 objects in the scene instead of 2, making the environments more challenging to solve
- Instead of **LCSAC**, we prepone the use of our new baseline algorithm Language-Conditioned DroQ (LCDroQ), providing a detailed explanation in Subsection 5.5.2, with the hyperparameters in Section A.3
- Comparisons with **HEIR** now exclude the effect of hindsight bias, with the expert feedback, as expected, providing the best hindsight learning results

Addressing the Hindsight Bias of HEIR

The following experiments showcase our latest results of **HEIR**, which in our previous work (Röder et al., 2022) suffered from degenerated performance through oversampling the hindsight expert feedback for training.
As expected, Figure 4.21 showcases **LCDroQ+HEIR** as paramount in nearly all tasks, providing the ideal case for hindsight learning. This involves having access to all valid hindsight goal instructions for an encountered failure, where those are always grammatically correct in comparison to those of the learning-based system **HIPSS**. The experiments employ the self-attention as language encoder, while we also supply the results using the GRU and the remaining self-attention experiments in Subsection B.2.1.
Depicted are the *reach* (Figures 4.21a to 4.21d) and *push* tasks (Figures 4.21e to 4.21h) in the first two rows, for which we provide the options *Default*, *Color*, *Shape*, and *ColorShape* within the four columns. In the last row, we depict the *grasp* (Figure 4.21i), *lift* (Figure 4.21j), and two *multitask easy* (Figures 4.21k and 4.21l) challenges. The inscriptions of the figures includes the environment steps on the x-axis in relation to the average success rate on the y-axis. The shaded regions display the 90% confidence interval of 5 random seeds.

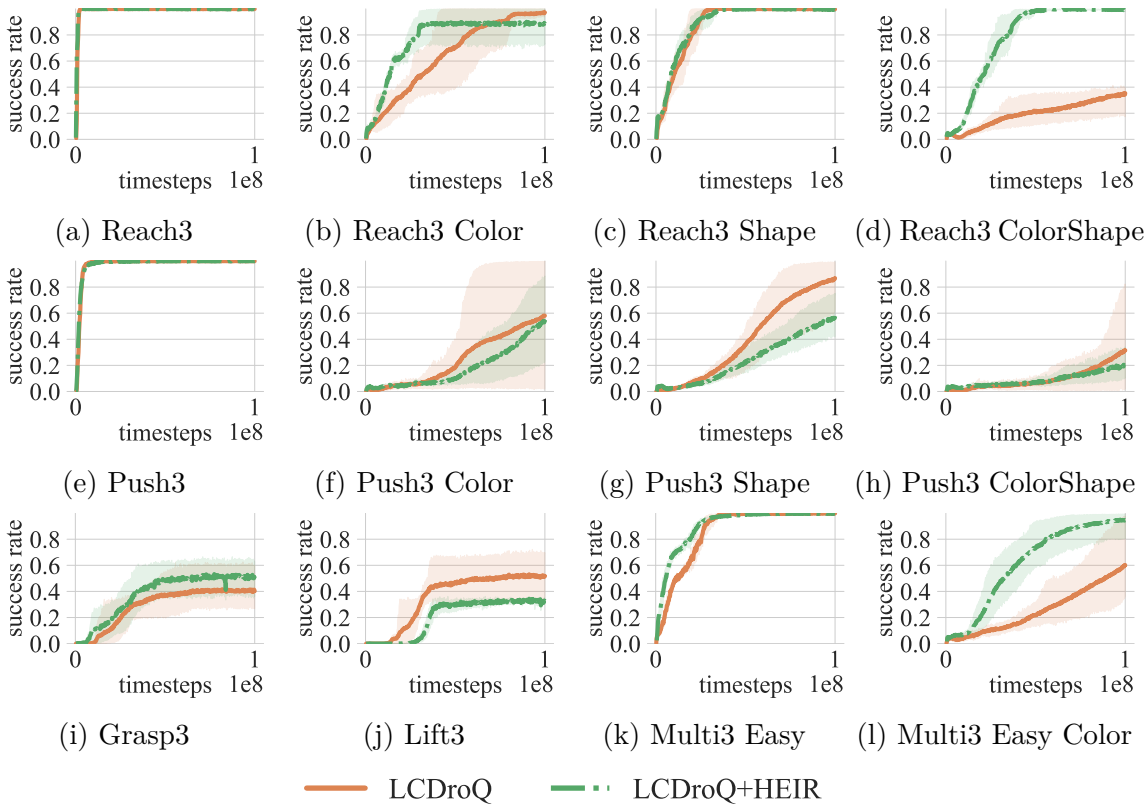
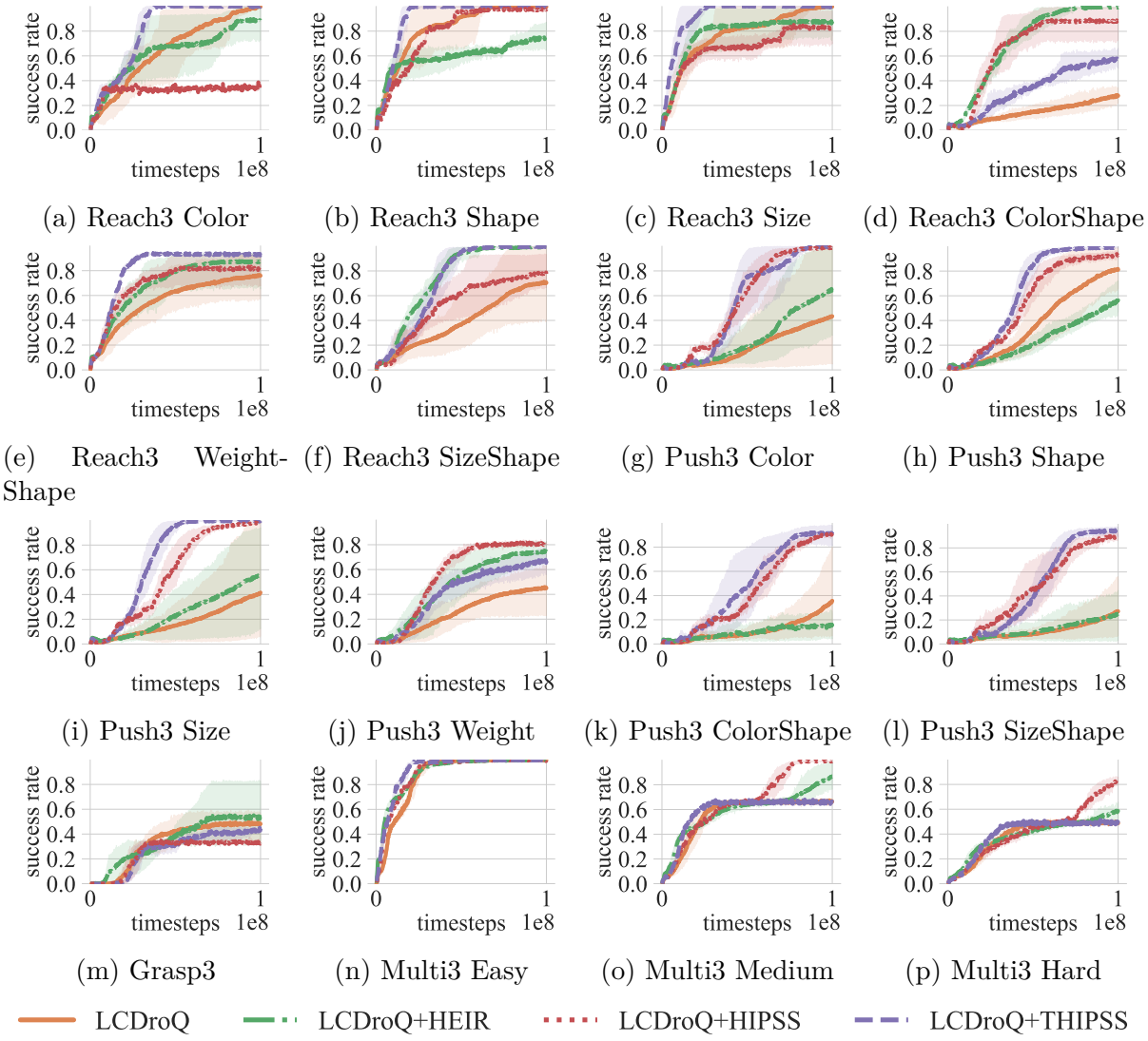


Figure 4.21: We highlight a selection of tasks, showing **LCDroQ+HEIR** outperforming the baseline **LCDroQ** in many cases. Illustrated are the experiments using self-attention as the language encoder for 5 random seeds. Depicted are the environment steps (x-axis) in relation to the average success rate, with the shaded region representing the 90% confidence interval (y-axis). We provide the full set of experiments as a comparison in Subsection B.2.1. Here, **LCDroQ+HEIR** denotes the combination of our baseline **LCDroQ** (cf. Subsection 5.5.2) and the expert feedback replay method **HEIR** (cf. Section 4.4). As expected, the perfect hindsight instructions by the expert show significant performance improvements (*e.g.*, Figures 4.21d and 4.21l).

Extended Hindsight Learning Experiments

In this section, we provide the latest experimental findings on our hindsight methods **HEIR**, **HIPSS**, and **THIPSS**. Figure 4.22 depicts the results for a selection of tasks, especially those with an extended range of object sizes, weights, and color variations (*e.g.*, see Figures 4.22e, 4.22g and 4.22l). Shown are the results for the GRU language encoder with 5 random seeds for each algorithm configuration. Additional results for the self-attention language encoder and other task configurations can be found in Subsection B.2.2. Our **HEIR** approach, with the hindsight bias correction, as expected, performs well in nearly all cases as there is no learning necessary, and the replay mechanism has access to all valid hindsight instructions right from the start. It essentially compares to a version of egocentric speech with perfect predictions but is drastically outperformed by **LCDroQ+HIPSS** and **LCDroQ+THIPSS** in the *push* tasks with the *Size* option activated (Figures 4.22i and 4.22l). In combination with **LCDroQ**, the egocentric speech approaches, similar to a child that is learning to speak, sometimes generate grammatically incorrect instructions. This procedure adds noise into the language grounding, with the positive effects of increased robustness and faster learning. However, there are still architectural differences that often show either **HIPSS** (Figures 4.22j and 4.22o) or **THIPSS** (Figures 4.22c and 4.22i) leading. In the multitask setting *medium* and *hard*, **HIPSS** is the only approach that can progress, hence learning a new task after 50 million steps (Figure 4.22o) and 75 million steps (Figure 4.22p). This highlights how self-generated speech helps with overcoming plateaus and differentiating behavior through verbs. However, **HIPSS** and **THIPSS** sometimes yield quite similar results (*e.g.*, Figures 4.22g, 4.22h and 4.22n) but can also show markedly different performance (*e.g.*, Figures 4.22b to 4.22d). This provides insights into their respective strengths and weaknesses. We hypothesize that **LCDroQ+HIPSS**, utilizing GRUs within its seq2seq model, is more data-efficient (sharp learning progress in Figures 4.22o and 4.22p), but is actually surpassed by the general effectiveness of the Transformer in many cases (*e.g.*, see Figures 4.22a, 4.22c, 4.22e and 4.22i), due to fewer inductive biases (Dosovitskiy et al., 2020).



(a) Reach3 Color (b) Reach3 Shape (c) Reach3 Size (d) Reach3 ColorShape

(e) Reach3 Weight-Shape (f) Reach3 SizeShape (g) Push3 Color (h) Push3 Shape

(i) Push3 Size (j) Push3 Weight (k) Push3 ColorShape (l) Push3 SizeShape

(m) Grasp3 (n) Multi3 Easy (o) Multi3 Medium (p) Multi3 Hard

Figure 4.22: This figure depicts the results for our algorithms **LCDroQ**, **LCDroQ+HEIR**, **LCDroQ+HIPSS**, and **LCDroQ+THIPSS**. We compare them with respect to the amount of environment timesteps (x-axis) it takes to reach a certain success rate, for which we visualize the mean and the 90% confidence interval as shaded area on the y-axis. All experiments employ the GRU as language encoder and contain 5 random seeds. Further results are provided in Subsection B.2.2.

4.9.5 Compositional Generalization

This section explores the compositional generalization capabilities of our **HIPSS** models, focusing on systematic generalization. We highlight to what extent the presented models can predict language instructions for behaviors not encountered during training. For this purpose, we investigate this in our multitask environment with the default setting of 3 objects, 3 colors, and 2 different types of actions, where the two types of actions map to 3 synonyms each. This setting allows examining systematicity not only for the goal object identification, but also considers the desired task behavior.

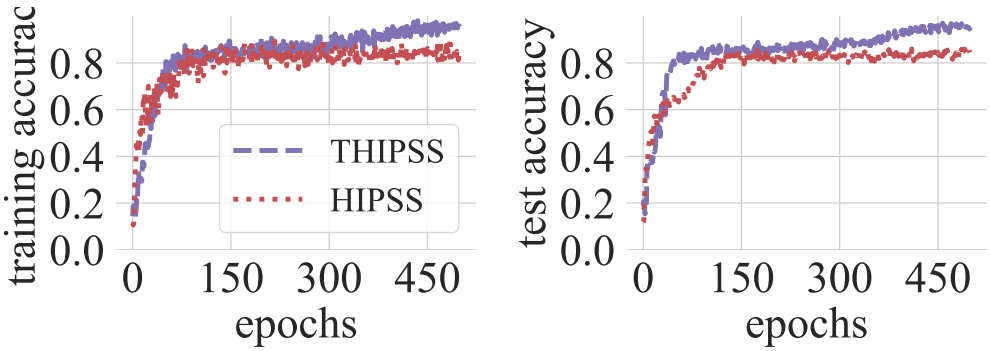
The models are trained in a typical supervised learning setting with a static dataset. For this setting, we employed a trained policy that solves the multitask setting well, generating trajectory-instruction combinations that we store and split into a training and test set. Measuring generalization capabilities involves comparing the training performance to the test set, which contains behavior-instruction combinations that were absent from the training set.

Figures 4.23 and 4.24 compare **HIPSS** and **THIPSS** regarding their training and test accuracy for different out-of-distribution (OOD) split configurations. In comparison to the fully observable state-based setting that we consider in all other parts of this thesis, for these experiments, we use the pixel-based version of **LANRO** as an exceptional case. The predictions of **HIPSS** and **THIPSS** are based on trajectories of egocentric pixel observations (cf. Subsection 3.4.3).

Figure 4.23 depicts the setting where we exclude certain action and object color combinations entirely. The agent does not encounter any combination of the *"push"* task verbs and object colors *"red"* or *"green"*. Furthermore, all the *"reach"* verbs are excluded for the color *"blue"*, but *"red"* and *"green"* references are included.

The table below the two accuracy plots illustrates the color and verb combinations if they are present in the training set (green cells) or absent (red cells). This setting excludes entire verb-color combinations depicted by the red rows of three columns, simplifying the training, as there are fewer combinations to learn. Consequently, both **HIPSS** and **THIPSS** perform well, achieving a training accuracy between 0.9 and 1.0. However, the test accuracy differs, as shown in Figure 4.23 on the right.

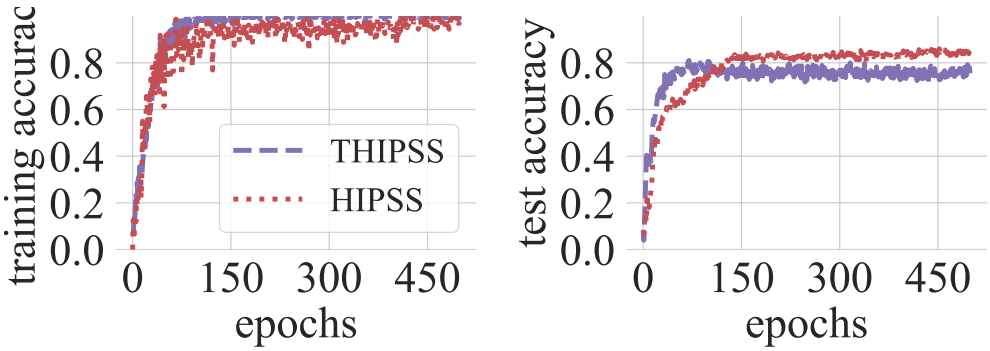
While **HIPSS** struggles to surpass the accuracy of 0.8, **THIPSS** learns faster around 70 epochs and achieves a higher overall accuracy of 0.95 after a sudden jump at 330 epochs. This indicates that **THIPSS** is able to identify behaviors like *"push the green object"*, although it has never seen it and needs to derive this from other behavior-instruction combinations like *"touch the green object"*.



	reach	touch	contact	push	move	shift
red						
green						
blue						

Figure 4.23: We depict an experiment comparing **HIPSS** and **THIPSS** with respect to their systematic generalization (Hupkes et al., 2020). While the training in both cases achieves similar accuracy values for the most part (left figure), **THIPSS** exceeds the performance of **HIPSS** by up to 0.1 accuracy points after the sudden jump at 330 epochs in the test plot to the right. The table at the bottom depicts, as green cells, the word combinations seen during training and, as red cells, the word combinations that were excluded. The models are trained with all instructions containing the verbs *reach*, *touch*, and *contact*, paired with the colors *red* and *green*. In addition, the verbs *push*, *move*, and *shift* are paired with the color *blue*. For testing, the opposite word combinations, shown as red cells, are used. Hence the models have never seen instruction word combinations containing the *reach*-related verbs paired with the color blue or the *push*-related verbs paired with the colors *red* and *green*.

In Figure 4.24, we present a more evenly distributed setup. The table in the lower part of the figure demonstrates how we allow at least one verb-color combination for each task to be part of the training set. This allows both models to reach the nearly perfect training accuracy of values close to 0.95. However, slight differences emerge when evaluating the test accuracy, where **THIPSS** performs up to 0.1 points worse than **HIPSS**. A possible explanation for this is the data that Transformer-based models require for effective learning, as they lack the strong inductive biases provided by their universal architectural design. Nevertheless, this setup validates that **HIPSS**, to a certain extent, also generalizes well to novel samples, highlighting its suitability in online learning settings, such as RL.



	reach	touch	contact	push	move	shift
red						
green						
blue						

Figure 4.24: We examine a case with a more evenly distributed setting of out-of-distribution samples. The training accuracy in the left figure shows very similar results for both **HIPSS** and **THIPSS**, while the latter appears to have a more stable learning curve. On the right, we depict the test accuracy, with **HIPSS** exceeding the performance of **THIPSS** after 100 epochs of training. In the table at the bottom, in-distribution samples used for training are illustrated by the green cells, while the samples that we exclude are depicted by the red cells. More precisely, training the models includes the instructions containing the word combinations *reach* and *red*, *touch* and *green*, *contact* and *blue*, *push* and *green*, *move* and *red*, and *shift* and *blue*. For evaluation, we consider all the other combinations, denoted by the red cells in the table.

In summary, both models exhibit generalization capabilities that are useful for different types of data distributions. **HIPSS** generally provides good results in evenly distributed settings, while **THIPSS** demonstrates the ability to generalize to entirely out-of-distribution samples that were never encountered during the training phase. Both models prove useful for hindsight learning, indicating capabilities of compositional generalization that are particularly important for RL settings, where data is unevenly distributed.

4.10 Summary

This chapter concludes the first pillar of this thesis: hindsight language learning. We demonstrate how concepts from language development and cognitive science provide a valuable foundation for our reinforcement learning algorithm additions. Although our language-conditioned RL setting does not involve a real human in the loop, represented by a synthetic caretaker instead (Cideron et al., 2020; Akakzia et al., 2021), it still offers valuable insights into how such an entity could aid learning. The idea of Hindsight Expert Instruction Replay (HEIR) implements the caretaker, who is always providing the perfect hindsight instruction feedback upon interaction with an unintended object. As expected, it improves learning in comparison to our baseline, but showed declining performance in the midterm training when referring to our initial results in Subsection 4.9.2.
Intuitively, a learner that fails to complete a task but receives the perfect hindsight feedback for an arbitrary behavior that, by chance, achieved a rewarding outcome has a detrimental effect on learning. Another perspective is the straightforward explanation of hindsight bias, as discussed in Section 4.6, which states that the priors resulting in a specific behavior cannot always be paired with the hindsight expert instructions. We address this issue from our prior work by incorporating a simple limit to the number of hindsight expert instructions considered during experience replay. In Section 4.9.4, we showed how this could lead to the expected results of **HEIR**, demonstrating the optimal conditions for hindsight learning.
Our second idea exploits successful episodes as positive training examples to train a model that predicts language instructions in hindsight. We presented Hindsight Instruction Prediction from State Sequences (HIPSS), our seq2seq model approach that relabels failed episodes with language appropriate to the actual outcome. Such a model could be thought of as actually modeling the notion of egocentric speech. For example, in instances where the agent mixes up words and completes the task for a different object, an inner monologue gets triggered to supply an intrinsic hindsight language goal for what actually happened. This, together with a sparse reward, provides a positive learning experience. As the language predictions during the early training stages are not perfect, self-generated instructions tend to improve the robustness of the agent by providing noisy hindsight goals, like a toddler talking to itself. Extending our seq2seq model to incorporate the well-known Transformer architecture (Vaswani et al., 2017) provides additional performance benefits for most of the presented environments. We term this implementation Transformer-based Hindsight Instruction Prediction from State Sequences (THIPSS). An additional brief study on **HIPSS** and **THIPSS** demonstrates their systematic generalization capabilities. Learning an internal monologue model to explain the world in terms of words and extrapolating to novel and unseen setups appears to be a promising approach to robot learning with RL. This affirms our motivation for leveraging compositionality, which our models exploit when generating the language for goal-directed behaviors. In summary, we can positively answer **Research Question 2** with our work on **HEIR**, **HIPSS**, and **THIPSS**. We demonstrate the notion of caretaker feedback and egocentric speech to accelerate the language learning of an artificial agent, increasing the sample efficiency and validating our cognitively inspired reinforcement learning implementations.

Chapter 5

Action Correction for Language-Conditioned Reinforcement Learning

This chapter presents the second part of this thesis, which focuses on action corrections. We introduce a framework that models conversational breakdowns arising from misunderstandings, treating them as reinforcement learning tasks with changing goals and sparse rewards. To address this challenge, we introduce a novel approach within the language-conditioned RL domain, incorporating model uncertainties into the agent's return estimation to mitigate the negative effect caused by misunderstandings during conversations. Additionally, we extend the grounded hindsight learning methods from the previous chapter to this correction framework, tackling the dual challenge of learning to follow instructions while simultaneously resolving misunderstandings.

In the domain of human-robot interaction, the ability of a robot to understand and execute instructions correctly remains a significant challenge, and a major area of research (Tellex et al., 2020; Shao et al., 2020; Ahn et al., 2023). The complexity inherent to natural language arises from the interaction of nonverbal signals and context-dependent subtleties, which can lead to misunderstandings or misinterpretations of utterances. Human communication frequently involves resolving misunderstandings and clarifying uncertainties regarding what was said and what actually happened in the world. Despite these challenges, humans are remarkably proficient at learning and communicating under such circumstances (Kempe and Brooks, 2016).

In robotics, operators commonly assume natural language instructions to be grammatically correct. However, on occasions where instructions are flawed, the focus primarily shifts to the issues of non-understanding (Hirst et al., 1994; Bohus and Rudnicky, 2008). Within these settings, operators either endeavor to reprocess the utterance again or solicit a repetition of the instruction by seeking clarification (Bohus and Rudnicky, 2009; Eppe et al., 2017). Essentially, the aim is to mitigate misunderstandings through additional input data or higher quality data with less lingual or sensor noise (Hirst et al., 1994).

Unlike situations of **non-understanding**, which are easier to detect, **misunderstandings** are hard to spot, a distinction also made by computational linguists (Bohus and Rudnicky, 2008). Compared to non-understandings, misunderstandings are significantly more

costly, as the listener assumes there is no mismatch between the commander's intention and what was understood. Bohus and Rudnicky (2008) emphasize that misunderstandings cause twice as many conversational breakdowns in comparison to the non-understandings. Misunderstandings in human-robot dialogs can arise from various sources, such as inconsistent world models, erroneous perception, or ambiguous instructions (Hirst et al., 1994; Chai et al., 2014).

Consider the scenario in Figure 5.1, where the human utters the command *"Hand me the apple, please!"*, with the intention to have the robot pick up the green apple (🍏). This instruction is ambiguous due to an underspecification, as there are two apples on the table. Subsequently, the robot reaches for the red apple (🍎), which might be closer to its end-effector. Shortly after the human realizes that the instruction was misinterpreted, revealed by the robot's behavior, an intervention is uttered through the action correction command *"No, the green one!"*. The robot then reconsiders the initial ambiguous instruction with respect to the action correction and possible environmental changes, finally approaching the intended goal object by picking up the green apple. Optionally, the robot would also return to the initial world setting, by putting the red apple back (Thierauf et al., 2023).

The aforementioned misunderstanding of Figure 5.1 and other types that we will present in the remainder of this chapter are particularly challenging because the agent acts as if it has correctly understood the instruction. There are several works that address this issue of misunderstanding, but they are limited to verbal interactions only (Bohus and Rudnicky, 2009; Purver et al., 2018). Misinterpretations, in general, can lead to potentially costly, irreversible, and destructive outcomes that may occur in any conversational setting, but especially in those involving a physical or simulated robot. Designing agents to handle noisy conversations should be considered a crucial part of conversational AI. Therefore, we postulate that action corrections should not merely be a mechanism to cope with unpredictable events, but have to be integrated into the language grounding, and consequently, the learning process itself. Natural language instructions by humans can be grammatically incorrect and incomplete, thus containing noise due to disfluency and polysemy. This implies that words can be used inconsistently or may provide incorrect or even multiple references. Such references could be misaligned, meaning the robot's internal representations of the world do not match those of the instructor. From the robot's perspective, unknown words might be mapped to incorrect meanings that lead to false beliefs about what type of actions to execute.

Resolving such seemingly simple misunderstandings initially can only be achieved by incorporating additional modalities beyond the language itself. In this context, it is crucial to consider actions and their effects on the world. The following sections will draw inspiration from approaches modeling conversations as goal-oriented dialogs (Thomason et al., 2015; Ferreira and Lefèvre, 2015; Bordes et al., 2017; Lu et al., 2019) and provide links to our (Röder and Eppe, 2022) and other end-to-end approaches in an embodied language-conditioned RL setting (Ferreira and Lefèvre, 2015). While many works on dialog systems assume the communication to happen only verbally (Bohus and Rudnicky, 2009), corrections aim to adjust the conversation by letting the agent respond in the same manner. In the context of language-conditioned RL, action corrections involve adjusting the agent behavior by extending the lingual goal with the correction. Unlike traditional

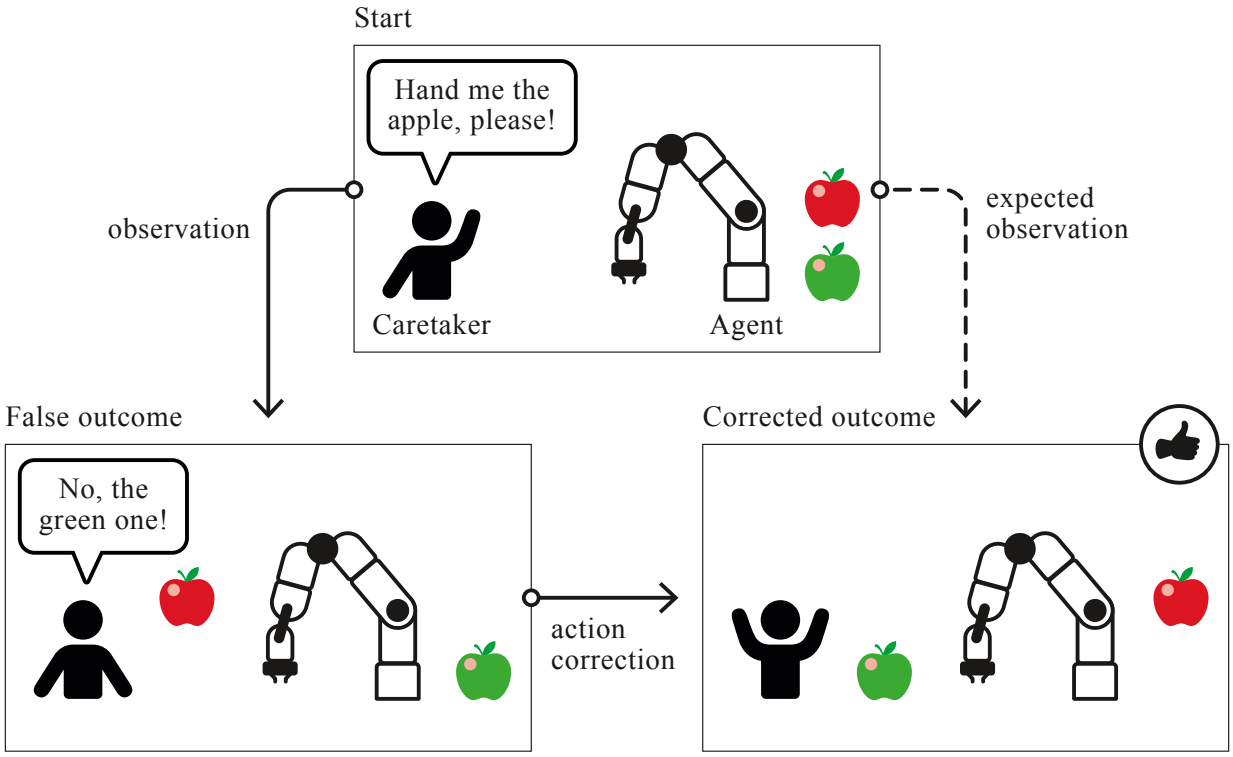


Figure 5.1: An action correction scenario, where a human issues the ambiguous command *"Hand me the apple, please!"* to the agent. However, the operator intends for the robot to pick up the green apple. The robot chooses to reach for the red apple, either by chance or because it is the closer object. When the operator realizes that the actions deviate from the intended outcome, the action correction *"No, the green one!"* is uttered. Then, the robot reevaluates the original instruction given the correction, resolving the underspecification, to successfully hand over the correct apple.

dialog systems that primarily focus on verbal communication, we develop a model that accounts for multimodal inputs. Considering the nonverbal cues of the acting agent, we attempt to create a more realistic and interactive interplay between the synthetic instructor and the agent. Adjusting for action corrections requires an adaptive context-specific representation that incorporates the original instruction and its reinterpretation in the presence of the corrective feedback. This chapter introduces and showcases how to model action correction tasks in a simulated reinforcement learning setting with synthetically generated instructions and corrective feedback within a language-conditioned RL setting with sparse rewards.

5.1 Related Work

The upcoming section provides an overview on dialog systems paired with reinforcement learning and works that employ some sort of corrective feedback to resolve conversational breakdowns and misunderstandings.

5.1.1 Reinforcement Learning and Dialogs

A common approach to model human-robot interactions and conversations are dialog systems (Hirst et al., 1994). In such systems, the robotic agent incorporates many components that solve different parts, such as object identification, motion control, speech recognition, and many more (Eppe et al., 2017). The overall setting of a dialog involves a goal, like completing a flight booking request by the user (Li et al., 2020b) or fetching an object (Eppe et al., 2017). Many of the components rely on pre-trained models that account for gesture recognition, advanced models for language understanding, a dialog manager that tracks the progress of the conversation, and a natural language generation component that produces answers.

For this work, we aim for an end-to-end approach in the context of language-conditioned reinforcement learning based on sparse rewards, where both the task identification and task execution are learned from scratch. However, the work on dialog systems, and especially those using RL (Dodge et al., 2016; Peng et al., 2017), provides useful insights to address our challenge of action correction. Thus, we present a brief summary of some related components involved.

Language processing is central to both approaches, whether following the traditional dialog setup or our verbal/nonverbal RL setting. For language understanding, contemporary dialog systems use semantic parsers (Thomason et al., 2015). Semantic parsers implement procedures to derive a formal meaning of the natural language utterance that resembles interpretable logic relations (Dong and Lapata, 2016). However, they are limited in terms of their capabilities when implemented via slot-filling (Bordes et al., 2017). More recent approaches consider neural semantic parsers, which harness deep neural networks to generate the logical forms (Gardner et al., 2018). These combine semantic parsers and neural networks, where the latter are trained to process text (Eppe et al., 2018). However, semantic parsers are usually designed for domain-specific applications, in which they generally tend to provide better results.

Nowadays, deep neural networks and even LLMs (Zha et al., 2023) are frequently utilized as general-purpose language encoders. Language encoders such as BERT (Devlin et al., 2019) (see Section 3.2.3) represent natural language utterances as high-dimensional vectors, lacking the structural representations of the semantic parsers, but are typically more general due to their training on large language corpora (Dong and Lapata, 2016). In our work, we consider those human-robot dialog settings that are not solely based on verbal interactions but take into account the embodiment of the robot (Eppe et al., 2017), and hence also the nonverbal features.

5.1.2 Conversational Corrections

In the following, we highlight related works that employ techniques to handle different types of misunderstandings and methods of correcting supervision, which we consider in addition to the initial goal instruction. Unlike the language-conditioned setting, corrections typically follow the paradigm of assistive language (Luketina et al., 2019), providing additional context for the overall task. We examine both traditional dialog systems and language-conditioned reinforcement learning approaches that offer insights for our action correction challenge.

Misunderstandings, such as ambiguities, have been shown to provide valuable social information. In the work of Achimova et al. (2022), the authors highlight that observing a listener resolving an ambiguity reveals something about their personal preferences. Furthermore, it provides room for interpretation and creativity, and when applied intentionally, could be used to gain social knowledge. However, different than Achimova et al. (2022), we primarily feature ambiguities as flaws in communication, as we usually have an instructor commanding an agent (Röder and Eppe, 2022). Additionally, we posit that action corrections reveal information about the language capabilities of the agent, adapting to changes in the task context.

Numerous studies investigate misunderstandings and non-understandings by addressing the problems arising from ambiguities (Achimova et al., 2022; Caselles-Dupré et al., 2022). For dialog systems, Ashktorab et al. (2019) reveal valuable insights into different repair strategies to resolve so-called conversational breakdowns, with the focus on designing resilient and more user-friendly conversational agents. Notable strategies involve the agent providing additional options to recover from the breakdown or explaining its limitations that led to it. Generally, many approaches tackle misunderstandings by proposing possible alternative solutions (Sharma et al., 2022; Cui et al., 2023; Zha et al., 2023; Thierauf et al., 2023). While it is necessary to differentiate whether the agent is exposed to corrections during learning or deployment, most works employ an online natural language correction approach, intervening in the agent's action execution at any time (Sharma et al., 2022; Cui et al., 2023; Thierauf et al., 2023; Zha et al., 2023). Bohus and Rudnicky (2009) suggest a solution to consider a confidence metric before an action is executed, mitigating potential harm caused by a misunderstanding. However, their approach is limited to verbal dialogs only. Chai et al. (2014) ensure that both participants share a common perceptual ground by letting the robot inform about its sensation of the environment.

Our correction setting reveals similarities to interactive RL, where language is used to provide an additional supervision signal (Co-Reyes et al., 2019; Faulkner et al., 2020). In the work of Faulkner et al. (2020), the authors employ language feedback for learning by explicitly addressing quality issues, estimating the effectiveness of feedback with respect to the reward. Co-Reyes et al. (2019) investigate the idea of incremental language corrections to guide the policy learning, instead of following a single instruction. This enables the agent to experience different and more natural types of task specifications, allowing for quicker adaptation to new tasks. However, their approach leverages a human user to provide language corrections and is therefore limited.

A crucial decision is determining when and under which circumstances a feedback signal

is returned. While Cui et al. (2023) consider whether the feedback is related to a physical or the overall dialog state, Roh et al. (2020) take into account the context to address instruction mismatches. Li et al. (2020b) provide a multimodal approach to conversational repair, where the agent considers both language and the current state of the dialog GUI (Graphical User Interface) by utilizing their *mutual disambiguation pattern*. Shi and Yu (2018) also propose a multimodal approach, incorporating audio as additional input into their end-to-end dialog system setup. The learning is accompanied by an audio dataset that provides sentiment hints as additional context features. From the hints, they derive a reward signal for their dialog policy optimization, leading to higher success rates compared to the setup without the additional sentiment information. Prior works like Lipton et al. (2018) with learning-based dialog systems employ neural networks to model the slot-filling, but disregard the state of the world. Only a few approaches consider Bayesian inferences to resolve the ambiguities (Achimova et al., 2022; Caselles-Dupré et al., 2022).

In the following, we propose three types of misunderstandings that intelligent agents observe in our unique language-conditioned RL setup for action corrections based on sparse rewards.

5.2 Situated Misunderstandings

In the following, we highlight our types of misunderstandings, which capture an important set of conversational failures.

Unlike previous conversational approaches (Thierauf et al., 2023; McCallum et al., 2023), our action correction is embedded into the language-conditioned learning setting. There is no explicit distinction made between the original goal instruction and the correction itself. The original instruction is not edited but extended by the action correction. This enforces the agent to learn an instruction representation that directly addresses the situated uncertainty. For example, a scene with two red objects and an instruction just referring to a *"red object"* should treat both as equally probable goals until further evidence resolves the ambiguity.

There are two crucial steps for a successful action correction: identifying the initial goal and reconsidering the altered goal given the additional feedback and the current physical state. It is important to note that corrections can occur at any point in time for any type of misunderstanding. In the example of Figure 5.1, we depicted only one type of misunderstanding, namely that resulting from an underspecification. In the following sections, we distinguish between **three types of misunderstandings**: **ambiguity**, the **lack of common ground**, and the **instruction correction**, as the basis for the remainder of this work.

5.2.1 Ambiguity

In cases of an **underspecified instruction**, the agent cannot identify a unique goal object due to an overlap of named and observed features that hinder the inference of a single meaning. Consider the scene in Figure 5.2, where the robot might be instructed to *"grasp*

the cube". This specification is imprecise, as there are two objects categorized as cubes, creating an ambiguity along the single feature dimension referenced by the shape word in the instruction. This resulting issue prevents the agent from identifying the unique and correct goal object without additional background knowledge.

The system has no means to resolve this ambiguity independently and would need to arbitrarily guess the goal object. Even if the agent were to choose the correct goal by chance, it could not provide an explanation for selecting that particular object over the other. Without proper justification, the robot's behavior appears inconsistent to the human, potentially deteriorating trust and the acceptance of the system.

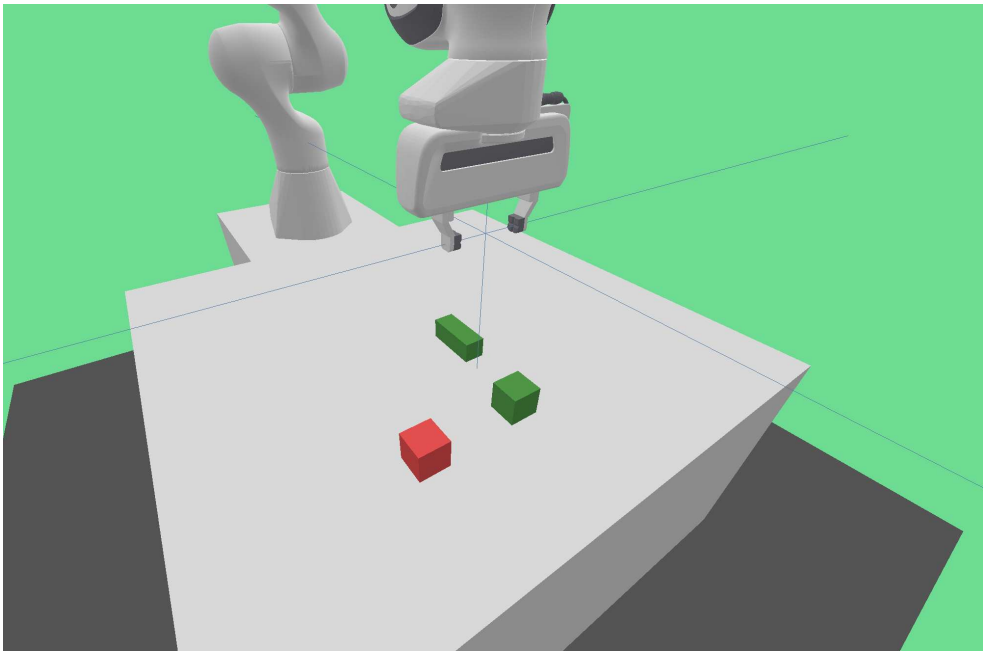


Figure 5.2: A scene from **LANRO**, with a green cube, a green cuboid, and a red cube. Figure taken from Röder and Eppe, 2022.

5.2.2 Lack of Common Ground

A learning agent may encounter an *unfamiliar word* if the instructor uses a term that the agent has rarely or never encountered during training. Unknown words are considered out-of-distribution. However, such settings may still be resolved if the word is not crucial for identifying the goal object or can be complemented by co-appearing words. For instance, the instruction *"grasp the red dice"* could be successfully used in Figure 5.2 to guide the robot without knowledge about the word *"dice"*. Another example would be that the robot wrongly classifies a red apple as a tomato due to a lack of proper grounding, resulting in a mismatch of the language and state abstraction.

The lack of common ground can also manifest in situations where the human instructor and the agent have different conceptual understandings or diverging mental models of the world. While there are many scenarios involving a mismatch in common ground,

we postulate that all of these could be simulated in the same manner, as they lead to comparable outcomes.

The importance of the establishment of a common ground is highlighted by works like Chai et al. (2014), where the authors advocate for the robot to communicate its perceived state of the world to alleviate the chances of misunderstanding.

5.2.3 Instruction Correction

Errors can occur during the production of the instruction itself. This case describes the *instructor's faulty speech* – also known as lapsus linguae – when the error is on the operator's side. In the literature, this is also known as self-repair of speech (Levelt, 1983). Consider the scenario in Figure 5.2, where an instruction like *"pick up the green cube"* is uttered, but the operator intends to have the red cube picked up. Under these circumstances, an immediate or slightly delayed action correction like *"actually, the red one"* resolves the error and guides the agent to execute the correct actions. These situations reflect real-life situations where one immediately recognizes the wrong words produced or only realizes the mistake when the communication partner does something unexpected

Unlike prior works that consider this setting as simply a problem of parsing noisy instructions (Dong and Lapata, 2016; Eppe et al., 2018), we assume this type of correction to appear at any point in time, especially when the agent has already made changes to the world. Therefore, an approach of only denoising through parsing would not consider the modified world state and fail to resolve this setting.

Further Action Correction Implementations For the remainder of this thesis, we focus on our three previously mentioned cases of misunderstanding. There are many additional cases of misunderstanding that present unique challenges, including:

- Misinterpretation of context: The agent cannot understand the instruction *"Pick up the red cube"* when no red cube is available, or all cubes on the table are red
- Temporal misunderstanding: Misunderstanding the order in which tasks should be executed, *e.g.*, *"After you pick up the red cube, place it on the blue cube"* (simple conjugations supported by *decstr* (Akakzia et al., 2021))
- Negating Instruction: *"Pick up a cube that is not red"* – unsolved by recent LLM-based robotics Ahn et al. (2023), but already addressed in language-conditioned RL by Hill et al. (2020)
- Spatial misunderstanding: *"Pick up the object left of the red cube"* (problem of self-reference discussed in our work Röder et al., 2021)

Summary We have introduced three types of misunderstandings that we believe sufficiently cover most action correction scenarios that could arise in human-robot interaction. It is crucial to consider that all the outlined cases could appear at any point in time while the agent is interacting with the world, *i.e.*, at every timestep t of the episodic MDP. Furthermore, the action correction may also occur sooner or later due to delays in recognizing incorrect behavior or changes in the world.

5.3 Description of Action Corrections

The next section summarizes the idea of action corrections as dynamic goals appearing within the episode after an initial goal was proposed.

We define our setup as a dialog between a caretaker and an agent, where the former uses verbal and the latter nonverbal means of communication only. The ongoing dialog is tightly related to the world state, because the agent communicates via physical interactions. We follow the approach of language conditioning (Luketina et al., 2019), where changes in the world reflect the agent's behavioral consequences conditioned by the provided lingual goal, which contains all the necessary information to succeed. Therefore, we model the action correction setup, in the sense of an MDP, as an episode with a changing language goal. Changing this goal could potentially also alter the reward condition, hence one can view this setting as a sequential multitask setup (Röder and Eppe, 2022).

Episodes always start with an initial goal instruction that is optionally extended by the action correction. Figure 5.3 illustrates this general case, supplementing the scenario from Figure 5.1. The agent receives the instruction g_ℓ under which it generates actions according to the policy $\pi(a_t \mid s_t, g_\ell)$ that potentially change the world state, following the environment's transition dynamics $\mathcal{T}(s_{t+1} \mid s_t, a_t)$. The caretaker constantly observes the agent's behavior and, in our case, has access to the full world state s_t. Given this access, a predefined condition function $\mathcal{C}$ (similar to Equation 3.3) is used to detect meaningful attempts or interactions related to non-goal objects. Following this, the caretaker utters the action correction to guide the agent in revising its behavior. The misunderstandings from Section 5.2 are implemented by concatenating the correction to the original goal, $g_\ell \circ g_{\text{ac}}$ (concatenation of strings as chain of symbols).

The idea of the overall action correction setting follows that of changing goals within a single episode (Fang et al., 2019; Fryen et al., 2020). Formally, this can be described as successive trajectories conditioned on different goals, following our formulation of the trajectory distribution in Equation 2.18. In Equation 5.1, we describe this exemplarily for the case of two successive goals, g_ℓ and g_{ac}, with the action correction occurring at time point t_{ac}. However, this could be extended to as many successive goals as necessary by chaining single language-conditioned trajectories as described in Equation 3.4. Furthermore, changing the goal of the extended MDP (see Subsection 2.2.5) also influences the conditioning of the reward function, as described in Equation 3.3.

$$p_\pi(\tau \mid g_\ell, g_{\text{ac}}, t_{ac}) = \rho_0(s_0) \left(\prod_{t=0}^{t_{ac}-1} \pi(a_t \mid s_t, g_\ell)\, \mathcal{T}(s_{t+1} \mid s_t, a_t) \right) \left(\prod_{t=t_{ac}}^{T-1} \pi(a_t \mid s_t, g_\ell \circ g_{\text{ac}})\, \mathcal{T}(s_{t+1} \mid s_t, a_t) \right) \tag{5.1}$$

Before providing the (semantic) grammar definitions to generate the action corrections, we highlight an example with possible corrections that could be involved in the setting of the scene in Figure 5.2. Rather than relying on rephrasing the language goal or replacing it, we extend the instruction to retain the initial information supplemented by the additional

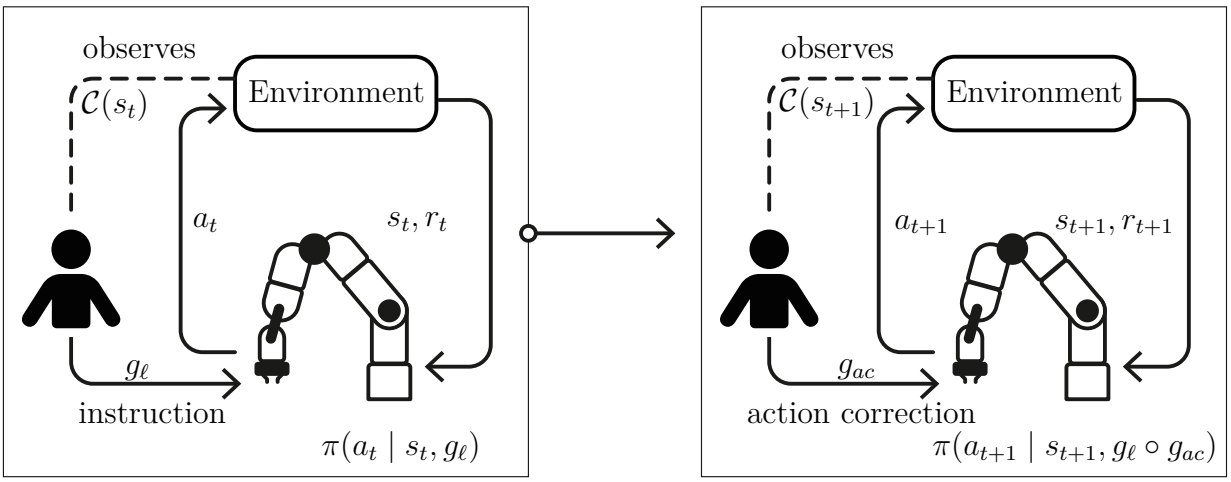


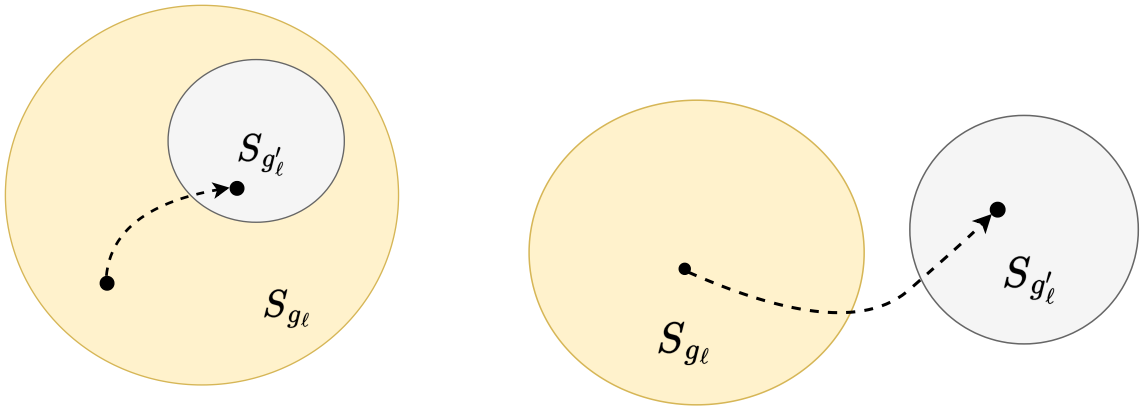
Figure 5.3: The figure describes the general case of an action correction scenario that fits any of the proposed cases of misunderstanding from Section 5.2. The instructor utters a lingual goal g_ℓ that the agent attempts to follow, interacting with the environment depicted by the RL loop (see Figure 2.6). Implemented by the condition function $\mathcal{C}$, the instructor observes the world state s_t for configurations that constitute a misunderstanding. In case of detecting a misunderstanding, an action correction g_{ac} is returned. Figure inspired by our previous work Röder and Eppe (2022).

feedback (Röder and Eppe, 2022). For our work, we utilize linguistic cues, hence editing terms like *"actually"*, *"sorry"*, and *"no"* (Thierauf et al., 2023) instead of colloquial fillers, such as *"ehm"* and *"uh"* (Levelt, 1983). As an additional option, we provide negating corrections that emphasize a property the agent got wrong, by sentences such as *"not the red object"*, when the agent interacts with the *"red cube"* but was tasked to *"touch the green object"* instead.

We initially described language itself as abstract and non-deterministic in terms of the goal states it maps to. In other words, there are many state configurations that satisfy a particular goal, which we introduced in Subsection 2.2.5 as S_{g_ℓ}. Understanding action corrections in the context of these state sets is basically reducing the uncertainty about the potential outcomes that the goal corresponds to. In Figure 5.4, we depict the transition from the original goal g_ℓ to the intended goal g'_ℓ based on the change by the action correction (dashed line).

Figure 5.4a depicts the scenario where the original goal g_ℓ is very similar to the operator's intended goal. For instance, a language goal $g_\ell =$ *"pick up the red object"* corresponds to many red objects the agent could manipulate, resulting in high uncertainty across all the possible objects that could have been meant. This relates to an ambiguity that could

straightforwardly be resolved by an action correction like g_{ac} = *"actually, the cube"* to get $g'_\ell = g_\ell \circ g_{\text{ac}}$. This small semantic difference results in a drastic change regarding the rewarding states involved. As a result, one obtains the new set $S_{g'_\ell}$ that is a subset of the *"red object"* reference set S_{g_ℓ}. In the case of Figure 5.4b, we illustrate the change of the rewarding states in an instruction correction setup (cf. Subsection 5.2.3). The operator unintentionally provides a wrong instruction that shortly after gets corrected, changing the entire meaning of the goal. This represents a leap from the original set of goal states S_{g_ℓ} to the new set $S_{g'_\ell}$, of which both are disjoint. Changing the meaning is actually providing an action correction, visualized by the dashed arrow, which alters the present language goal. Because the agent contemplates what state it should achieve to receive the reward, recognizing the correction also changes this anticipation. As an example, the language goal g_ℓ = *"pick up the red object"* gets corrected by g_{ac} = *"sorry, the blue object"* that corresponds to a different set ($S_{g'_\ell}$) of rewarding states.



(a) Similar (b) Distinct

Figure 5.4: To the left in Figure 5.4a, we illustrate the process of correcting a misunderstanding that is closely related to the corrected instruction; while to the right (Figure 5.4b), we depict the case where correcting the misunderstanding changes the whole meaning. Visualized are the rewarding states that, on the left, overlap and, on the right, are completely disjoint. An example of the first case would be the instruction g_ℓ =*"pick up the red object"* corresponding to the set S_{g_ℓ}, that gets corrected by g_{ac} =*"actually, the cube"*, where $g'_\ell = g_\ell \circ g_{\text{ac}}$ now maps to a smaller set of rewarding states $S_{g'_\ell}$. The same idea could be applied to Figure 5.4b, where the instruction g_ℓ=*"pick up the red object"* gets corrected by g_{ac} =*"sorry, the blue object"*, resulting in a rewarding state mapping $S_{g'_\ell}$, disjoint from the original set of states S_{g_ℓ}.

5.4 LANRO Action Correction Extension

*In the following, we describe how **LANRO** integrates the proposed concept of action corrections into the simulated robotic environment setup.*

With the publication Röder and Eppe (2022), we extended our **LANRO** environment from our previous work Röder et al. (2022) to include the presented lingual action correction feedback. A predefined and task-dependent condition function (Equation 3.3) allows us to detect interactions with correct, but also wrong objects as well. The wrong object interactions trigger the mechanism to return the action correction.

Our implementation generally considers two initial conditions the episodes start with. We illustrate this branching of misunderstandings in Figure 5.5, with three example objects. To the left, we initially have a **valid instruction** *"reach the red cube"*, with which the agent, under normal circumstances, could uniquely identify the goal object. Following this, we have misunderstandings due to the lack of common ground and the false instructions as possible causes. On the right side of Figure 5.5, an **underspecification** *"reach the cube"* provides the situation-dependent instruction that causes the misunderstanding – our case of ambiguity.

Initializing an episode, our instruction generator utilizing a semantic grammar regards the properties of the present objects and references that could be used. For the example in Figure 5.2, we have 3 objects on the table, of which 2 are green. This is a result of additional checks that we employ to avoid unsolvable episodes. Different to our previous work, we enforce feature overlaps in the action correction setting (Röder and Eppe, 2022). To retain settings that could be solved by the agent, we need to ensure that objects have at most one property in common, making them unambiguous if both properties are part of the instruction. For example, referencing the *"red cube"* or *"red"* in the initial instruction and *"cube"* appearing in the action correction. This ensures that the original instruction and correction combination makes the goal object uniquely identifiable.

In the action correction tasks of **LANRO**, we determine that the agent needs to solve a misunderstanding only 50% of the time. This is randomly defined at the beginning of each episode, while the other case is exactly the usual language-conditioned setting from Section 3.4. Apart from providing a natural curriculum of solving simple tasks and the more difficult action correction tasks, conversational repairs also happen frequently but not constantly, making our setting even more realistic. Furthermore, our implementation allows the action correction to technically appear only once per episode. Therefore, the agent needs to solve tasks provided for up to two goals. As this requires more time, we increase the total amount of episode steps to $T = 100$, essentially doubling it compared to the normal tasks from Section 3.4.

In the following paragraphs, we describe the implementations of our three types of misunderstandings and the procedure of generating the action corrections in detail.

Ambiguity To enforce a scenario involving ambiguity, the instruction generator only refers to one property of the goal object randomly, *e.g.*, *"green"* or *"cube"*. For this, we consider a scenario like Figure 5.2 as an example. An instruction like *"reach the green*

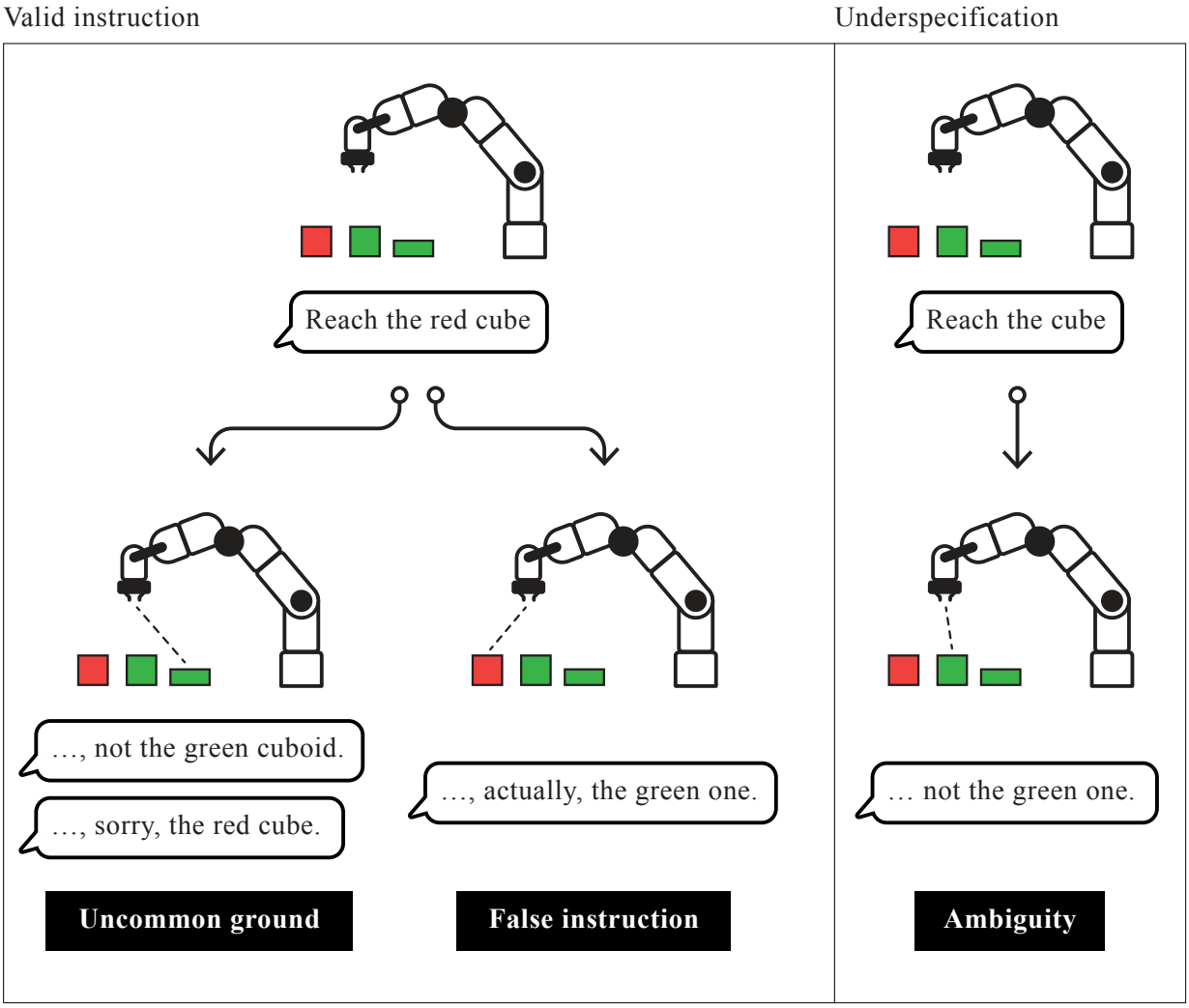


Figure 5.5: This figure illustrates the possible action correction scenarios implemented by **LANRO**. We depict a setting with a red and green cube, and a green cuboid. To the left, we have an initial valid instruction that follows misunderstandings caused by a lack of common ground or an instructor mistake. To the right, the ambiguity is caused by an underspecified statement.

object" as an underspecified statement could only be resolved by an action correction because there are two green objects on the table. Assuming the caretaker does not recognize this flawed instruction, feedback is returned when the agent is interacting with a non-goal object. For this, we include, next to our usual feedback, negations that exclude specific

properties to guide the agent towards the correct object, *e.g.*, on touching the *"red cube"*, uttering the correction *"not the red one"*. This type of misunderstanding directly integrates the embodied nonverbal part of communication (Trott et al., 2016) within our benchmark.

Lack of common ground We have implemented this notion of misunderstanding as negations that are returned when the agent interacts with a wrong object. Like in human-to-human communication, participants provide feedback by saying *"No, not the red one …the green one."*, as they see someone wrongly responding to a command like *"Pick up the green cube"*. This could be achieved by observing the acting partner's generated state trajectory. Getting feedback at a specific point in time enables the agent to consider the immediate states as representing interactions with incorrect objects. Now, referencing an object in the action correction with the same property consequently needs to be further away and not part of the experienced state trajectory. For example, in Figure 5.6, the command *"Touch the green cube"* is followed by an interaction with the green cuboid and the correction *"no, the green cube"* resolves this lack of common ground based on the situatedness and the repetitive feedback alone.

In other words, being in a state where an interaction with a green object happens, while not being rewarded and instead receiving the action correction *"no, the green cube"* hints at the current object being wrong and switching to the other green object, the green cube. This setup coincides with the Markov property and is solvable without a memory that stores the experienced states.

Instruction Correction To simulate the phenomenon of *lapsus linguae*, we incorporate a random hesitation of up to 5 timesteps, emulating a delayed action correction that occurs when the operator recognizes a discrepancy between the desired and executed behavior. This replicates a scenario where a human quickly corrects an erroneous instruction, akin to speaking faster than thinking or simply committing an instructional error. This mechanism allows for a more realistic simulation of human-robot interaction, capturing the inherent imperfections and adaptability present in real-world scenarios. By incorporating such nuances, we aim to develop a more robust and flexible system capable of handling the complexities of human communication and decision-making processes.

Grammar Like the original instructions in **LANRO** (Röder et al., 2022), we have implemented our three types of misunderstandings as represented by the Backus-Naur form in Figure 5.7. We essentially combine the original instructions following the rules from Figure 3.17 with those below, denoted by the extended instruction combination of the original goal <INSTRUCTION> and the action correction <CORRECTION>.

For our **multitask** setting, we propose action corrections for the task verbs that can get corrected. We consider these as **task corrections**, where the caretaker revises the faulty instruction that deviates from the **desired behavior**. For instance, this could involve making the agent switch to the *push* task when currently executing the *reach* task with the task correction *"actually, touch it"* after the initial instruction *"push the red object"*. In Figure 5.8, we provide the Backus-Naur form, in which we depict the possible

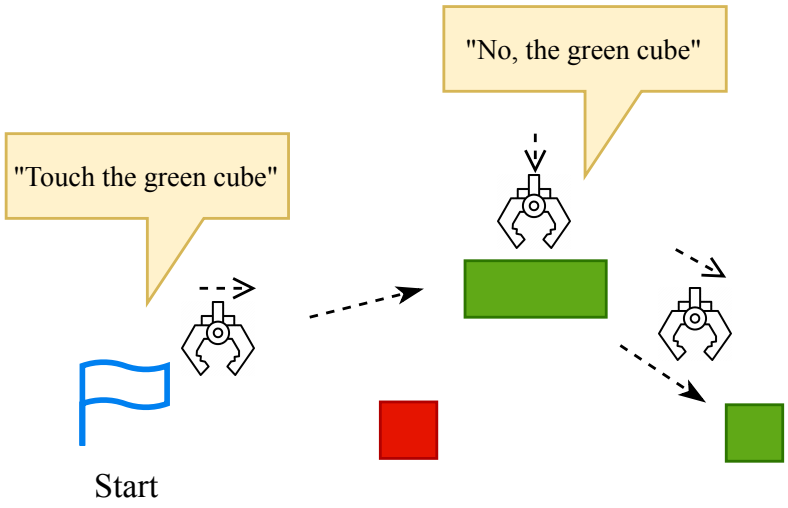


Figure 5.6: We illustrate the lack of common ground where the agent misunderstands the instruction. Instead of reaching the *"green cube"*, it reaches the *"green cuboid"* instead. As we do not know the cause of the misunderstanding, we provide the action correction *"No, the green cube"* to contrast the agent's current state (step 2) and object visited with the actual instruction.

$$
\begin{aligned}
\langle \text{EXTENDED INSTRUCTION} \rangle &\models \langle \text{INSTRUCTION} \rangle \ \langle \text{CORRECTION} \rangle \\
\langle \text{CORRECTION} \rangle &\models \langle \text{BEGINNING} \rangle \ \langle \text{ARTICLE} \rangle \ \langle \text{OBJECT} \rangle \\
\langle \text{BEGINNING} \rangle &\models \langle \text{EXCUSE} \rangle \mid \langle \text{NEGATION} \rangle \mid \langle \text{EDIT} \rangle \\
\langle \text{EXCUSE} \rangle &\models \texttt{sorry} \mid \\
\langle \text{NEGATION} \rangle &\models \texttt{not} \\
\langle \text{EDIT} \rangle &\models \texttt{actually} \mid \texttt{no}
\end{aligned}
$$

Figure 5.7: The extension of our Backus-Naur form grammar from Figure 3.17 for our action corrections, including the negations. The provided definitions are applicable for all language tasks (see Subsection 3.4.1).

action words by the task descriptors <REACH>, <PUSH>, <GRASP> and <LIFT> that contain the respective verbs from Section 3.4. For this subset of corrections, we implemented the *instruction corrections* only (see Subsection 5.2.3), while excluding the negations. Negations in this setting are challenging to implement and would require an expansion of the vocabulary to describe these in a formal manner.

Some task-related behaviors are not possible to change or revert. For instance, *"grasping"* and *"lifting"* are far too similar, as the agent lifting an object also grasps it. Thus, we

$$
\begin{aligned}
\langle \text{TASK CORRECTION} \rangle &\models \langle \text{INSTRUCTION} \rangle\ \langle \text{CORRECTION} \rangle \\
\langle \text{CORRECTION} \rangle &\models \langle \text{BEGINNING} \rangle\ \langle \text{ACTION} \rangle\ \textit{it} \\
\langle \text{ACTION} \rangle &\models \langle \text{REACH} \rangle \mid \langle \text{PUSH} \rangle \mid \langle \text{GRASP} \rangle \mid \langle \text{LIFT} \rangle \\
\langle \text{BEGINNING} \rangle &\models \langle \text{EXCUSE} \rangle \mid \langle \text{EDIT} \rangle \\
\langle \text{EXCUSE} \rangle &\models \texttt{sorry} \mid \\
\langle \text{EDIT} \rangle &\models \texttt{actually} \mid \texttt{no}
\end{aligned}
$$

Figure 5.8: The Backus-Naur form of our task corrections used in the multitask settings of **LANRO**.

excluded those combinations that are too trivial or too difficult to solve. In other words, we offer sequences of tasks that follow from action corrections with respect to the level of difficulty we defined in Section 3.4.1. In Figure 5.9, we provide a selection of example instructions based on the aforementioned grammar definitions in Figure 5.7 and Figure 5.8.

push the red object. sorry, the red cube.

push the green cube. not the green cuboid.

touch the blue cuboid. sorry, the red cuboid.

reach the yellow cuboid. actually, lift it.

Figure 5.9: Example instructions with action corrections used in **LANRO**.

5.5 Action Correction Baselines

This section proposes our custom baseline algorithm with an extension for uncertainty aware return estimation.

Solving action corrections involves reconsidering the current context and the incorporated language feedback jointly. For the experiments that follow in Section 5.7, we consider the fully observable MDP with states containing the full extent of information about the world (see Section 3.4). Returning the action correction g_{ac} at a specific timestep t facilitates correlating the world circumstances with the feedback, as the agent has positional information of its end-effector and the objects in the scene.

We examine the action correction setting in both the single-task and the multitask setting from Section 5.3, as the latter provides a greater challenge for current RL algorithms and a more diverse range of corrections. Unlike the single-task setting, which focuses solely on goal object identification, the multitask setting also provides feedback for behaviors, such as *"reaching"* instead of *"grasping"* an object.

The action correction setting introduces higher uncertainty for the action selection and value estimation than our original setting. Following an instruction, the agent cannot be certain if and when the action correction will occur, thus requiring adaptation of its behavior to account for the feedback signal.

Such challenges of increased difficulty often require specific RL approaches to address the sample inefficiency because of the harder exploration. In the following section, we introduce an extension of our baseline **LCSAC** that incorporates architectural and learning objective adjustments to represent goal uncertainty using an ensemble of deep neural networks (see Goodfellow et al. (2016, Section 7.11)), employing the dropout technique for Bayesian approximation (Gal and Ghahramani, 2016) of the state-action values.

5.5.1 Solving uncertainties with uncertain critics

Improving the sample efficiency of algorithms is an ongoing effort in the RL community (Lillicrap et al., 2016; Schulman et al., 2017; Haarnoja et al., 2018). Just recently, the method Dropout Q-functions (DroQ) proposed by Hiraoka et al. (2022) addressed the issue of overly frequent policy updates, resulting in problems with estimation bias. The main purpose, followed by similar works (Chen et al., 2021b), is to increase the number of training iterations, hence gradient updates, after a fixed amount of environment steps used for collecting training data. This ratio of training iterations compared to collecting data in the environment is known as update-to-data (UTD) ratio. However, longer training runs on less frequently collected experiences have been shown to have a detrimental effect.

The approach **DroQ** has been shown to be sample-efficient with respect to this ratio, but is also computationally efficient because it utilizes a smaller ensemble than previous methods (Chen et al., 2021b). It can be described as an extension to SAC that employs a small ensemble of critics with dropout connections (Srivastava et al., 2014) and layer normalization (see Ba et al. (2016) and Subsection 2.1.3), that on average doubles the training performance (Hiraoka et al., 2022).

DroQ aligns with our idea to incorporate an **awareness of uncertainty** into the action correction learning setting, providing a way to represent multiple future modes in the sense of Bayesian inference. For this, we integrate **DroQ**, as a straightforward extension for off-policy algorithms into our **LCSAC** baseline and highlight how injecting uncertainties into the return estimation improves the performance of solving uncertain linguistic goal specifications.

5.5.2 Language-Conditioned DroQ

We propose our implementation Language-Conditioned DroQ (LCDroQ), which incorporates the components of **DroQ** into our **LCSAC** algorithm (Röder et al., 2022). In the following, we provide the detailed adjustments and the motivation behind this approach. With Subsection 2.2.7, we presented the underlying SAC algorithm on which our approach **LCSAC**, as a language-conditioned modification, builds. As an actor-critic approach, SAC employs an actor network and a critic network as the distinct components for learning. Usually, works focus on improving the return estimation, as the critic guides the learning of the actor and directly influences its gradients. Recent versions of SAC (Haarnoja et al., 2019) extended the algorithm by an ensemble of usually $N = 2$ individual Q-functions, for which the minimum of the target network predictions is used for learning (cf. Subsection 2.2.7). This approach, known as clipped double Q-learning (Fujimoto et al., 2018), alleviates the overestimation bias that typically occurs when employing a single critic. To further reduce the overestimation bias and address the volatile Q-value estimations, methods like Randomized Ensembled Double Q-Learning (REDQ) (Chen et al., 2021b) and SAC-N (An et al., 2021) employ even larger critic ensembles of $N = 3$ to $N = 20$. However, larger critic ensembles make learning less computationally efficient and seem like a brute force approach. Because of this, Hiraoka et al. (2022) propose a more efficient version that improves REDQ by employing a smaller Q-function ensemble. In a sense, **DroQ** implicitly employs a large ensemble by extending the Q-functions with additional dropout layers paired with layer normalization (Ba et al., 2016). Both concepts have been introduced in Subsection 2.1.3 and are a widely-used and efficient combination (Vaswani et al., 2017; Radford et al., 2018; Devlin et al., 2019). To the left of Figure 5.10, we illustrate the architecture of such a Q-function, which alternates dense layers, dropout layers, and layer normalization with ReLU activations. To the right, we depict the ensemble of which we use the minimum as an approximation for $\mathbb{E}_{\bar{\phi}_1,\dots,\bar{\phi}_N}\left[\min_{i=1,\dots,N} Q_{\bar{\phi}_i}(s', a')\right]$, according to (Hiraoka et al., 2022).

Dropout is usually employed as a regularization technique to prevent deep neural networks from overfitting (see Subsection 2.1.3). However, it has been shown to model Bayesian approximation with deep neural networks (Gal and Ghahramani, 2016). This is practically achieved by keeping the dropout enabled at all times, instead of turning it off for evaluation on the validation or test set.

DroQ essentially implements an implicit distribution by which it incorporates model uncertainty into the target approximation. The target approximation follows Equation 5.2. The expected target value is approximated by the proposal distribution q_{dr}.

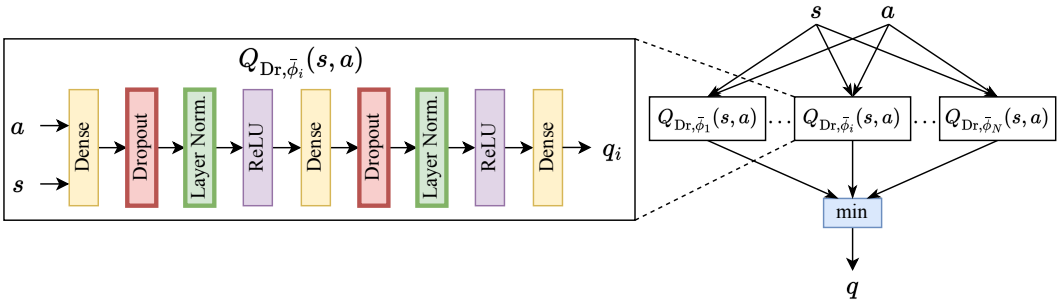


Figure 5.10: To the left, we depict the structure of an individual dropout target Q-function $Q_{\bar{\phi}_i}$ according to Hiraoka et al. (2022). The network consists of dense layers, dropout layers, layer normalization and ReLU functions, and outputs a Q-value q_i. On the right, we depict the ensemble of size N with the min operation to select the actual Q-value target prediction q used for training (see Section 2.2.7). This figure is based on Hiraoka et al. (2022).

$$\mathbb{E}_{\bar{\phi}_1,\ldots,\bar{\phi}_N}\left[\min_{i=1,\ldots,N} Q_{\bar{\phi}_i}(s',a')\right] \approx \int \min_{i=1,\ldots,N} Q_{\bar{\phi}_i}(s',a') \prod_{i=1}^{N} q_{\mathrm{dr}}(\bar{\phi}_i)\, d\bar{\phi}_i \tag{5.2}$$

From Equation 5.2 follows the approximation $\min_{i=1,\ldots,N} Q_{\mathrm{Dr},\bar{\phi}_i}(s',a')$ based on the sample average according to Hiraoka et al. (2022). Employing **LCDroQ** involves changing the prior Q-functions Q_{ϕ_i} to the dropout Q-functions Q_{Dr,ϕ_i}, hence the language-conditioned Q-functions of **LCSAC** and the target functions $Q_{\bar{\phi}_i}$, respectively. The changes concern Equation 2.40, Equation 2.45, Equation 2.47 and Equation 2.46 shown in Subsection 2.2.7.

5.6 Hindsight Learning with Action Correction

Subsequently, we propose the extensions to our previous hindsight learning mechanisms tailored for the action correction setup.

This section provides a continuation of our previous work on **HIPSS**, where wrong object interactions are followed by self-predicted instructions for hindsight learning (see Chapter 4 and Röder et al. (2022)). In this chapter, an action correction is returned, and the agent needs to reconsider the overall task assignment. Initially, the agent is very unlikely to solve the corrections, generating only a few successful episodes. We addressed this issue with **LCDroQ**, as outlined in the experiments of Figure 5.13, which harvests more successful action correction episodes for training **HIPSS**, episodes that are crucial for learning to predict instructions and their corrections in hindsight. Unlike our previous setting, language goal lengths now vary, and the model can no longer rely on filling slots in a semantic grammar with the structure *<verb> <article> <property 1> <property 2>*. Previously, we initiated the language generation process with the placeholder word `<pad>`, but for the dynamic length, we now also extend our seq2seq approach by adding the start-of-sequence (`<sos>`) and end-of-sequence (`<eos>`) symbols (Sutskever et al.,

2014) as part of the vocabulary. Similar to language modeling and machine translation (Cho et al., 2014), **HIPSS** decides whether to finish the sentence generation early with an <eos> prediction or to continue it with an action correction. In practice, this is achieved by assigning higher probabilities to the possible sentence ending <eos> or the beginning of a correction. In the following, we provide two example predictions explaining this procedure:

> *"<sos> push the red cube <eos>"*
> *"<sos> push the red cube, actually the yellow cube <eos>"*

Consider a scenario where the agent pushes the *"green cube"* while tasked to *"push the red cube"*. Figure 5.11 illustrates multiple possibilities for this hindsight instruction prediction to unfold.

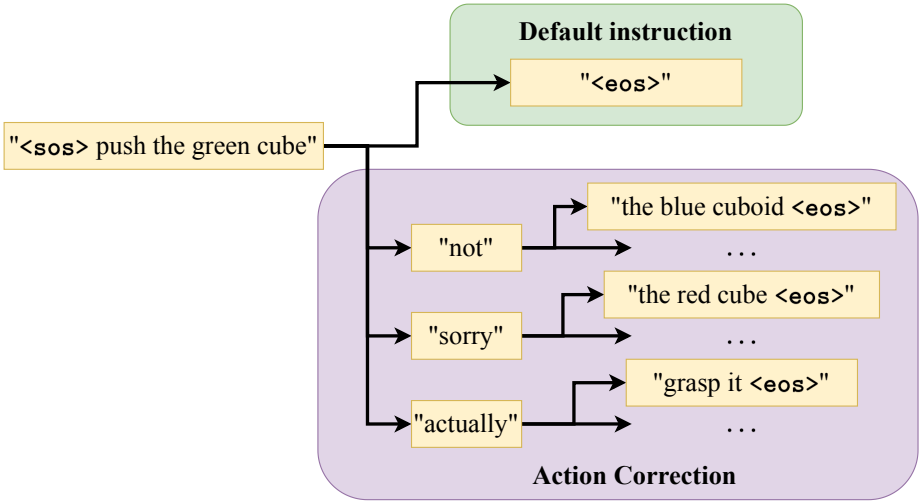


Figure 5.11: The figure illustrates the various ways **HIPSS** can complete the instruction *"<sos> push the green cube"*, considering multiple approaches to hindsight action correction. The initial instruction could be directly terminated with the <eos> symbol, followed by a negation starting with *"not"*, or simply edited with phrases like *"sorry"* or *"actually"*, each potentially followed by a different object description or task behavior.

We depicted the sentence *"push the green cube"* as the initial hindsight prediction, following the same procedure shown in Chapter 4. In this case, the model would end the prediction with the <eos> symbol. However, employing **HIPSS** for action corrections introduces the additional challenge of considering other objects in the scene.

For the first branch, which continues language generation, the negation *"not"* is followed by a description of an object that is not a *"green cube"*, including all other objects in the scene with which the agent may have interacted incorrectly. This demonstrates how the **HIPSS** model needs to be aware of the other goal objects to hypothesize potential interaction failures. In this example, we assume a *"blue cuboid"* is present in the scene and involved in a wrong interaction because the agent confused the colors, potentially

due to a lack of common ground (Chai et al., 2014) (cf. Subsection 5.2.2). The resulting hindsight action correction instruction *"push the green cube, not the blue cuboid"* in a sense simulates the wrong interaction with the *"blue cuboid"*, followed by feedback guiding the agent towards the correct goal object, the *"green cube"*.

The second branch, starting with a *"sorry"*, could indicate an instruction correction (cf. Subsection 5.2.3). Such a sentence might be completed with the action correction *"sorry, the red cube"*, referring to another possible object in the scene. The third branch functions similarly but uses the editing term *"actually"*, followed by an action-related correction *"grasp it"*, that is part of our multitask environment (see Section 3.4.1). In both paths, the initial instruction could refer to any object; we only require the instruction plus the action correction to match a goal configuration met in hindsight.

HIPSS naturally implements all of the aforementioned cases by randomly sampling according to its probability distribution of next word predictions with the highest likelihood. The challenge of learning to predict action corrections lies in the scarcity of the much harder-to-solve conversational repairs. We address this issue with our implementation of **LCDroQ**, which improves upon **LCSAC** as shown in the upcoming section in Figure 5.13. **LCDroQ** generates more successful action correction episodes that serve as training samples for **HIPSS**.

Sampling action corrections remains unlikely as there are only a few of those episodes. To conduct the hindsight learning, we pretend to have solved an action correction on a first sensible object interaction. The difference is that we generate not only simple hindsight instructions but also the ones extended by the action correction. For this, we provide **HIPSS** with the initial goal instruction and let it generate only the correction $\hat{g}_{\text{ac}}$, instead of predicting the entire adjusted language goal $g_\ell \circ g_{\text{ac}}$. Our implementation provides the initial instruction as given, either by teacher forcing it upon the seq2seq model **HIPSS** or adapting the causal masking for our Transformer-based approach **THIPSS**. For hindsight learning, we randomly decide whether to complete an instruction or predict the whole instruction according to Algorithm 5.

Algorithm 5 HIPSS Action Correction Generation

M_ϑ	**HIPSS** model	τ_s	state trajectory
g_ℓ	initial language goal	$\circ$	concatenation operator

Randomly decide to bootstrap action correction generation
if bootstrap is triggered **then**
  Generate action correction based on state trajectory and language goal
  $\hat{g}_{\text{ac}} \leftarrow M_\vartheta(\tau_s, g_\ell)$
  Form corrected goal $\hat{g}_\ell \leftarrow g_\ell \circ \hat{g}_{\text{ac}}$
else
  Generate hindsight instruction according to Chapter 4
  $\hat{g}_\ell \leftarrow M_\vartheta(\tau_s)$
end if
return $\hat{g}_\ell$

We additionally illustrate this in Figure 5.12, which shows the generation of the action

correction based on the state trajectory and the initial instruction, *"push the red cube"*. To predict the action correction word by word, we model the probability of the sequence $\hat{g}_{\text{ac}} = (w_i, w_{i+1}, \ldots, w_L)$ according to Equation 5.3, where τ_s represents the state trajectory, and $g_\ell = (w_0, w_1, \ldots, w_{i-1})$ the sequence of preceding words of the initial instruction.

$$p(\hat{g}_{\text{ac}} \mid \tau_s, g_\ell) = \prod_{j=i}^{L} p(w_j \mid \tau_s, w_0, w_1, \ldots, w_{j-1}) \tag{5.3}$$

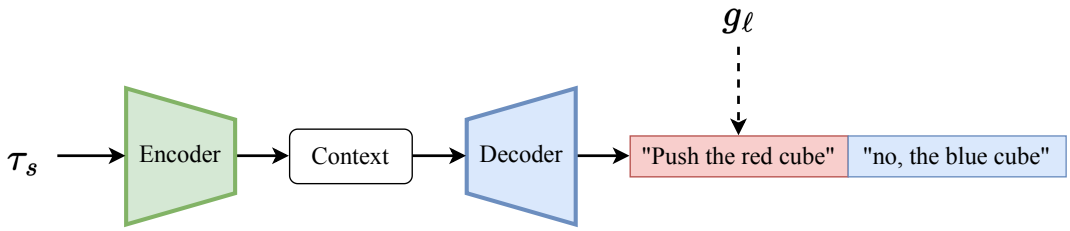


Figure 5.12: The hindsight action correction instruction is generated by completing the provided initial instruction, *"push the red cube"* (red box), with the action correction prediction, *"no, the blue cube"* (blue box).

As action corrections occur whenever the agent interacts with a non-goal object, our hindsight replay strategies from Section 4.5 address these transitions by replaying them within a similar range of timesteps. For instance, the future strategy (Figure 4.7) involves transitions when non-goal object interactions occur and shortly afterward, consulting the action correction delay we introduced for a more realistic training setting.

5.7 Experiments

In what follows, we provide the results for our action correction setting, employing our new uncertainty-aware language-conditioned algorithm, ***LCDroQ****. Next, we highlight results obtained by utilizing our developmental learning approaches,* ***HEIR****,* ***HIPSS****, and* ***THIPSS****, as hindsight replay extensions in the dynamic goal setting involving conversational misunderstandings.*

This section presents the results of our extension to **LCSAC**, termed Language-Conditioned DroQ (LCDroQ). We demonstrate a four-fold improvement in sample efficiency, attributable to the incorporation of model uncertainty through the **DroQ** components, as detailed in Subsection 5.5.2. To highlight the efficacy of our approach, we provide comparative results from a selection of action correction tasks, focusing exclusively on the two methods. Both methods employ identical implementations, with the **DroQ** component being enabled or disabled via a straightforward hyperparameter adjustment. Table 5.1 summarizes the modifications, corresponding to the key components illustrated in Figure 5.10.

Method	Dropout Rate	Layer Normalization
LCSAC	0.0	False
LCDroQ	0.01	True

Table 5.1: Hyperparameter settings for the critic ensemble in **LCSAC** and **LCDroQ**, as recommended by Hiraoka et al. (2022).

Figures 5.13 and 5.14 depict that **LCDroQ** consistently outperforms **LCSAC** across all scenarios, validating its selection as a robust baseline for the subsequent experiments of this thesis. The success rate is shown on the y-axis, and environment steps are shown on the x-axis. For the success metric, we plot the mean with the 90% confidence interval as a shaded region, calculated over 5 random seeds. For the experiments shown, we employ self-attention as the language encoder, with additional results provided in Subsection B.3.1.

Single Task Action Correction In Figures 5.13a and 5.13b, one can observe that **LCDroQ** outperforms **LCSAC** by a factor of 2 and up to 8, respectively. The improved performance of **LCDroQ** can be attributed to the incorporation of uncertainty into the critic architecture. We see that **LCSAC** is unable to surpass the 0.5 mark, *e.g.*, in Figures 5.13a and 5.13e, which potentially involves only solving default instructions and not even the misunderstandings. Only **LCDroQ** exceeds the 0.5 success rate in Figures 5.13a, 5.13b, 5.13e and 5.13f, essentially learning to handle action corrections after mastering the simpler instruction-following task. However, in cases where this is not possible, **LCDroQ** still provides a robust baseline that progresses in comparison to **LCSAC** (Figures 5.13d and 5.13g).

This approach effectively mitigates the inherent uncertainty associated with goal instructions in the action correction setting. In this context, the agent needs to consider potential changes to the current goal and their subsequent impact on the return estimation. We postulate that a deterministic critic does not capture this uncertainty and instead

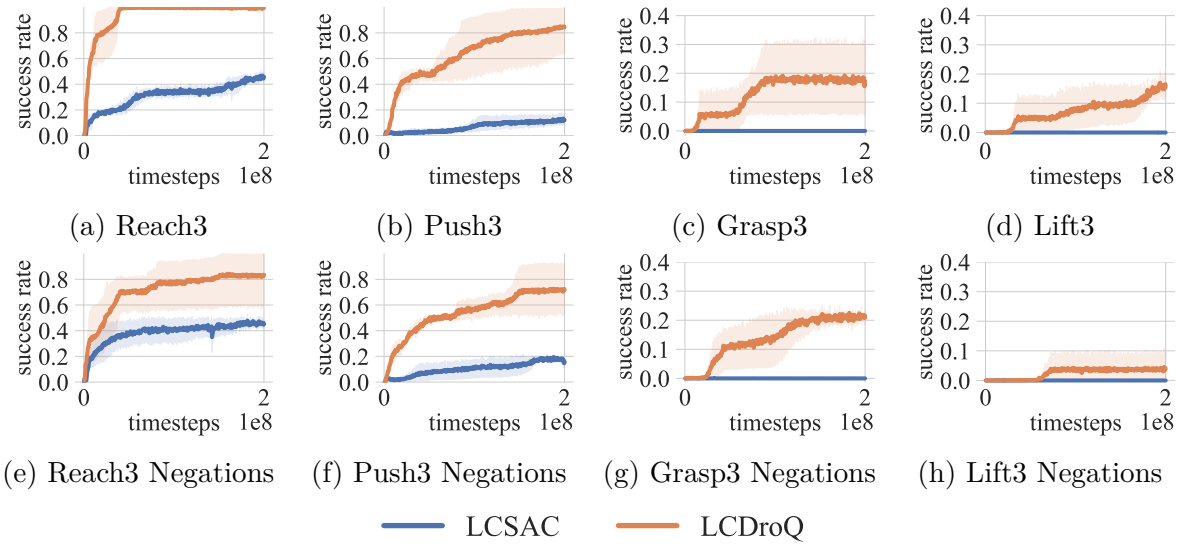


Figure 5.13: This figure compares the performance of our previous **LCSAC** approach with **LCDroQ**. We depict the environment steps taken (x-axis) and success rate (y-axis) of 5 trials, for which we plot the mean with the 90% confidence interval. We show results for self-attention as the language encoder, while further results are provided in Section B.3.1. The upper plots show the success rates for the normal action corrections, while the plots below depict the more challenging setting with negations. The results indicate that negations are generally more difficult to learn. However, **LCDroQ** outperforms the baseline **LCSAC** in all the presented cases.

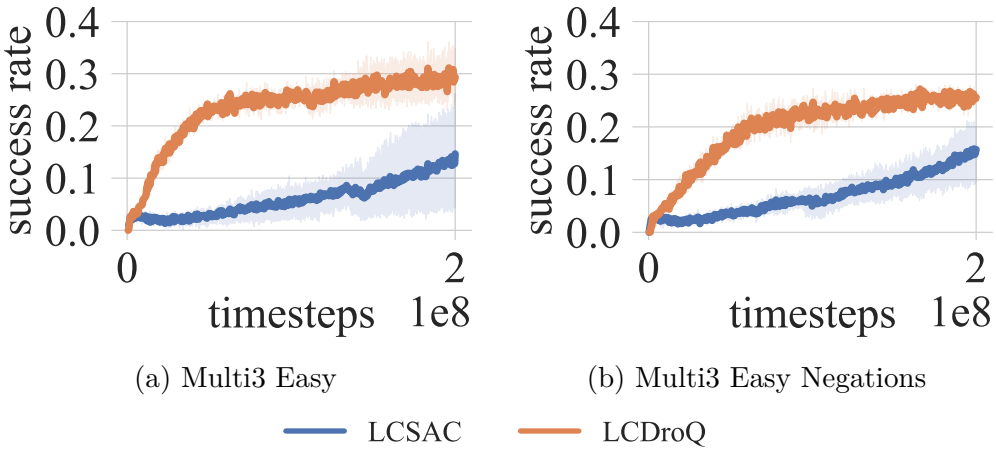
solely focuses on one future outcome, although there are actually two possibilities (cf. Figure 5.5). Consequently, the agent behaves more cautiously, anticipating action corrections, and tends to adopt a more conservative strategy when solving tasks. In other words, the agent preferentially selects environment states that maintain the flexibility to successfully address potential upcoming action corrections. This applies, for instance, to environment states from which it is easier to reverse unintended outcomes in case a different goal via a correction is returned.

For the *reach* and *grasp* tasks (Figures 5.13a, 5.13c, 5.13e and 5.13g), scenarios involving negations present a slightly greater challenge compared to the standard action correction setting. In the *reach* task, the performance of **LCDroQ** in the negation scenario (Figure 5.13e) is, on average, 0.2 points lower than in the default setting (Figure 5.13a). Moreover, it struggles to achieve and maintain the perfect success rate of 1.0 observed in the standard setup from Chapter 3. We attribute this to the difficulty of the negating action corrections, despite this being the easiest environment setting. One can see that **LCDroQ** outperforms **LCSAC**, achieving more than double the success rate.

Similar results are observed for the *push* task in Figures 5.13b and 5.13f. However, the performance of **LCSAC** is lower, reaching a success rate of 0.15 in the default and barely 0.2 in the negations setting. In comparison, **LCDroQ** improves the success rate by a factor of almost 8 and almost 4, respectively. The *grasp* and *lift* tasks demonstrate a more pro-

nounced difference. Figures 5.13c and 5.13g illustrate **LCDroQ** in the *grasp* task, where it plateaus at a success rate of 0.2, while **LCSAC** fails to make any notable progress.
In the *lift* setting, the **LCDroQ** baseline plateaus in the negation scenario at an average success rate of 0.05 (Figure 5.13h), while in the default setup (Figure 5.13d) it continues to improve, surpassing a success rate of 0.1 after 150 million environment steps (three-quarters of the timestep budget). As in the previous task, **LCSAC** also entirely fails to learn.

Multitask Action Correction In the *multitask* setting, the disparity persists, albeit less dramatically. **LCDroQ** achieves a maximum success rate of 0.3 in the default setup (Figure 5.14a) and a value of 0.25 in the negation scenario (Figure 5.14b). Compared to **LCSAC**, **LCDroQ** more than doubles the performance for instruction-following and misunderstandings, and increases it by 0.1 for solving misunderstandings with negations. These results suggest that adapting to a different task behavior has less impact when using negations than in the previous single task cases.



(a) Multi3 Easy (b) Multi3 Easy Negations

Figure 5.14: We provide a comparative analysis of **LCSAC** and **LCDroQ** in the multitask environment. Shown are the environment steps (x-axis) and the mean success rate for 5 trials including the 90% confidence interval (y-axis). To the left, we illustrate the default action correction, while to the right, we show the correction setup incorporating negations. The presented results only employ self-attention as the language encoder.

Summary Our novel algorithm, **LCDroQ**, demonstrates superior performance in the action correction setting compared to our previous method, **LCSAC** (see Figures 5.13 and 5.14). We hypothesize that this improvement arises from the implicit probabilistic representation of return estimates by **LCDroQ**, which contrasts with the single point estimate used in **LCSAC**, associated with the correction feedback. By capturing the uncertainty inherent in correction feedback, **LCDroQ** enables the policy to adjust its action selection more conservatively, targeting environmental states where misunderstandings are easier to resolve. Additionally, the probabilistic distribution of returns allows **LCDroQ**

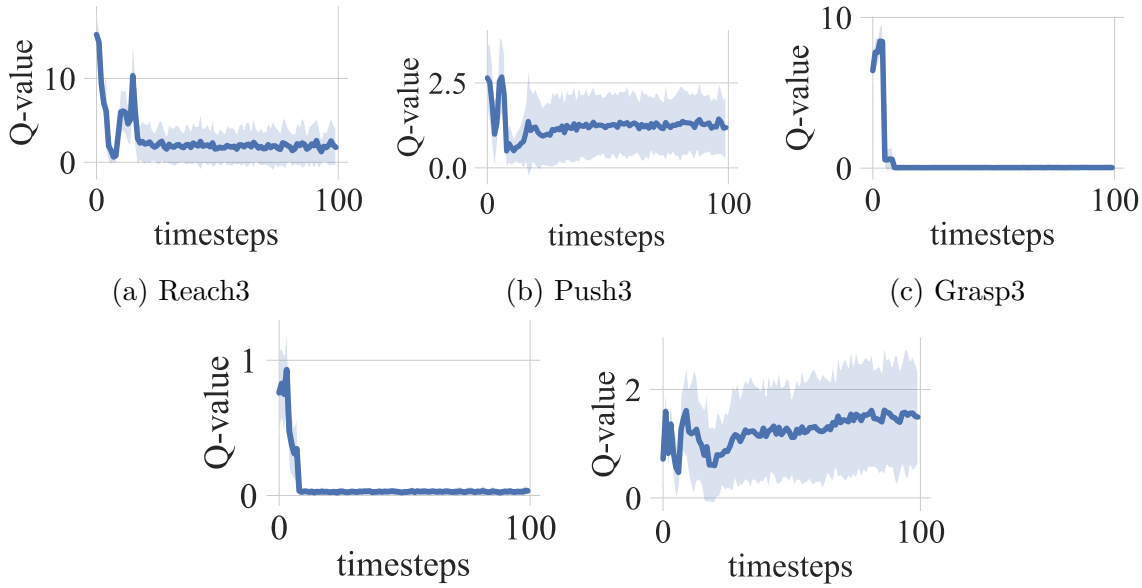
to effectively model two distinct behavioral modes: one for standard instruction-following and another for resolving action corrections. This distinction is particularly evident in the *grasp* and *lift* settings, where **LCSAC** fails to even learn the usual instruction-following, likely due to its inability to balance both scenarios effectively. Further results, including evaluations with the other language encoders, are provided in Section B.3.1.

5.7.1 Critic Ensemble Uncertainty

In this section, we present qualitative results demonstrating the behavior of our critic ensemble and how its uncertainty evolves during an episode when an action correction is returned. Figure 5.15 illustrates the critic ensemble's uncertainty over 80 action correction episodes, each initialized with the same seed, across five scenarios: the individual *reach*, *push*, *grasp*, and *lift* tasks (Figure 5.15a to Figure 5.15d), as well as the multitask setting (Figure 5.15e). The plots depict both the mean uncertainty and its standard deviation (y-axis) over the course of the $T = 100$ episode steps (x-axis). For the four individual tasks, a notable shift in uncertainty is observed at approximately $t = 10$ timesteps, which is the time it takes an optimal policy to arrive at the incorrect object and coincides with the caretaker providing the action correction feedback.

The multitask scenario in Figure 5.15e exhibits an even greater uncertainty, as evidenced by the larger standard deviation. This increased variability effectively captures the compounded uncertainty arising from both the goal object and task identification. Our **LCDroQ** approach, by virtue of its ensemble architecture, is capable of capturing these uncertainties. The method approximates a distribution over return estimations (refer to Equation 5.2), thereby providing a robust mechanism for uncertainty estimation in these settings.

Action Correction Trajectory We demonstrated how correcting an action corresponds to a reduction in uncertainty, as the agent no longer anticipates feedback from the caretaker. Figure 5.16 depicts a sample of the *reach* task with the trajectories before and after the action correction occurs. The illustration depicts the start position (dashed circle) and gripper XY-positions conditioned on the original instruction (represented by the blue dots •). Subsequently, we show the action correction triggered by manipulating the green object's position (indicated by the green × markers). In response, the agent modifies its behavior, approaching and manipulating the blue object (positions marked by the blue × markers). The orange dots • represent the gripper's position following the updated goal, hence conditioned on the initial goal plus the correction. The red object in the top right corner, denoted by the red × marker, remains untouched.



(a) Reach3 (b) Push3 (c) Grasp3

(d) Lift3 (e) Multi3 Easy

Figure 5.15: We illustrate the critic ensemble uncertainty as an average of 80 action correction episodes across 5 tasks. The episode timesteps are shown on the x-axis, and the mean return estimate with the standard deviation as a shaded region is shown on the y-axis. One can see that around $t = 10$ timesteps, the uncertainty drops drastically when the action correction is returned by the caretaker upon interacting with the wrong object.

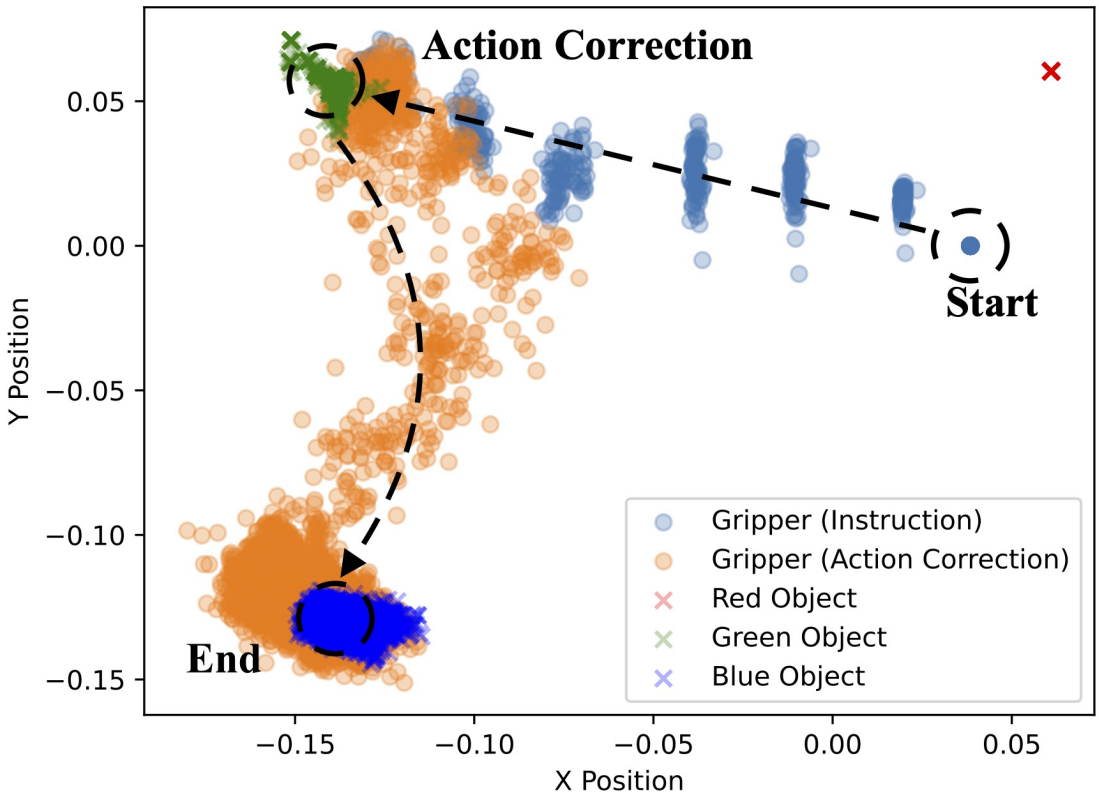
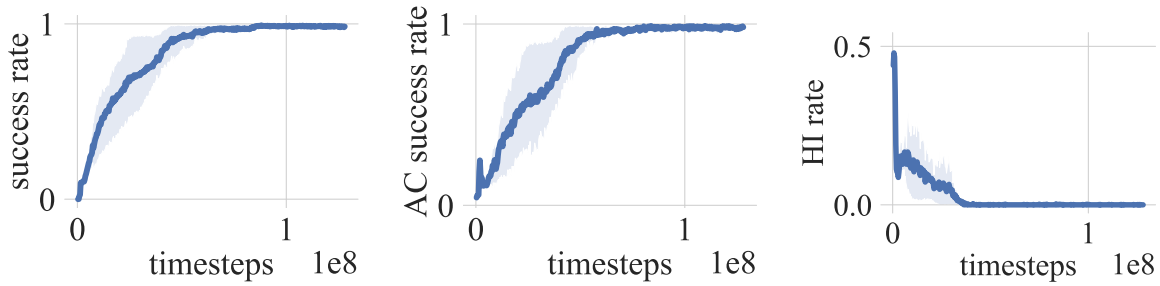


Figure 5.16: We illustrate the position of the agent's gripper conditioned on the initial instruction (blue dots ●) and after the action correction appears (orange dots ●). The red, green, and blue objects are depicted by the × with their corresponding colors. The agent starts at the circled position and approaches the green object, which it manipulates, hence the cloud of green × markers. After receiving the action correction, it changes its trajectory to approach and manipulate the blue object, denoted by the cloud of blue × markers at the end of the episode.

5.7.2 Hindsight Action Correction

In this section, we present the results of our hindsight learning approaches, **HIPSS** and **THIPSS**. Furthermore, we also provide results for **HEIR**, although it cannot replay action corrections in hindsight. However, it serves as another point of reference for the default hindsight instruction replay. In the case of this joint hindsight instruction and action correction setting, we allow half of the episodes to either be considered for hindsight replay or return an action correction (cf. Section 5.4). As described in Section 5.6, we modified our **HIPSS** models to incorporate action correction predictions. Figure 5.17 illustrates the learning progress, demonstrating an increase in both the success rate (left) and the action correction success rate (center), while the hindsight discovery rate (right) decreases. The hindsight discovery rate quantifies the frequency of generating hindsight instructions when the agent interacts with an incorrect object but no action correction is returned. A clear indicator of learning progress is evident in the agent's decreasing number of wrong object interactions reflected by this metric. It is important to note that trajectories stored within the replay buffer (cf. Section 2.2.7) continue to be utilized for hindsight learning as long as they are not overwritten by new experiences. They continue contributing to the ongoing learning process, even after no further hindsight learning signals are triggered.



(a) Success Rate (b) Success Rate Corrections (c) Hindsight Instruction Rate

Figure 5.17: An exemplary illustration of the *reach* task, focusing on the frequency of generating hindsight instructions, with respect to the overall success rate (Figure 5.17a) and the action correction success rate (AC success rate on the y-axis) only (Figure 5.17b). In Figure 5.17c, we depict the hindsight instruction rate (HI rate) caused by wrong object interactions. These interactions are, as expected, high in the beginning and decrease as the agent progresses. The x-axis shows the environment steps, while the y-axis depicts the respective metrics with their 90% confidence interval.

Basic Tasks

In Figure 5.18, we provide our hindsight action correction results for a selection of tasks without any form of *color* or *shape* extension. This serves as an example only, focusing on the challenge of solving two proposed goals during a single episode (cf. Section 5.4).

For the *reach* task (Figures 5.18a and 5.18e), we observe the negation setting to be harder to solve because of the lower success rate of our baseline **LCDroQ**. In the default

action correction setting (Figure 5.18a), **LCDroQ+THIPSS** achieves the highest performance, followed by the baseline **LCDroQ**, to which **LCDroQ+HIPSS** also matches. **LCDroQ+HEIR** performs the worst, mainly by only concentrating on episodes without action corrections, as the replay mechanism does not support them, plateauing at a success rate of 0.5, coinciding with the episode ratio (cf. Section 5.4). The results are vastly different for the task with negations (Figure 5.18e). Both **HIPSS** and **THIPSS** demonstrate a higher success rate, surpassing **LCDroQ** after only a few hundred thousand steps and reaching up to a 0.4 higher average success rate. **LCDroQ**, in this case, yields slightly better results than **LCDroQ+HEIR**, achieving a success rate of 0.6. **LCDroQ+HEIR** again remains at a success rate of 0.5, probably not learning to solve negations at all.

The *push* task (Figures 5.18b and 5.18f) proves harder to solve than the previous *reach* task, as there is more manipulation involved. In both plots, the extension of **LCDroQ** with **THIPSS** outperforms even the one with **HIPSS**, while the performance of **LCDroQ** and **LCDroQ+HEIR** is worse in both settings. In the default setting, depicted in Figure 5.18b, **LCDroQ** achieves a success rate of 0.4 and **LCDroQ+HEIR** a success rate of almost 0.2, whereas **LCDroQ+HIPSS** surpasses the success rate of 0.6. **LCDroQ+THIPSS** outperforms all approaches by reaching a value up to 1.0. In the negation scenario (Figure 5.18f), **HEIR** provides similar results, only barely reaching a success rate of 0.2 at the end of the training. The performance of **LCDroQ** and **HIPSS** is mostly the same, reaching values of up to 0.5. Unexpectedly, **THIPSS** even exceeds the success rate of 0.7, indicating that it seems to be easier for the policy learning to solve action correction negations through **hindsight instruction replay via egocentric speech** than through the usual replay.

The *grasp* task in Figures 5.18c and 5.18g shows similar results for all methods in the default setting, with little variation in the negation setting. In the former, all approaches end up at a success rate of 0.15 to 0.2. In the setup with negations, similar values are achieved, while **LCDroQ** catches up to **LCDroQ+HIPSS**, and **THIPSS** is leading by a small margin, showing how egocentric speech could still exploit the higher variations of action corrections.

Similar to the *grasp* task, the approaches depicted for the *lift* task perform poorly. For the default setting in Figure 5.18d, all the methods achieve success rates of 0.15. In the negations setting in Figure 5.18h, the results are similar. However, the baseline and the expert feedback **HEIR** perform slightly worse, while **THIPSS** shows a small advantage over **HIPSS**.

Summary The results suggest that **HIPSS** and **THIPSS** yield useful results in all of the cases where the policy can make sufficient progress, like the *reach* and *push* tasks, to actually address the challenge of goal identification where language plays a primary role. For the tasks that are already too challenging, such as learning the manipulation in the *grasp* and *lift* tasks, the hindsight methods cannot utilize their full potential. All the presented experiments utilize the GRU as the language encoder, whereas further experiments are included in Subsection B.3.2. For each approach, we plot the average success rate and the 90% confidence interval for 5 random seeds.

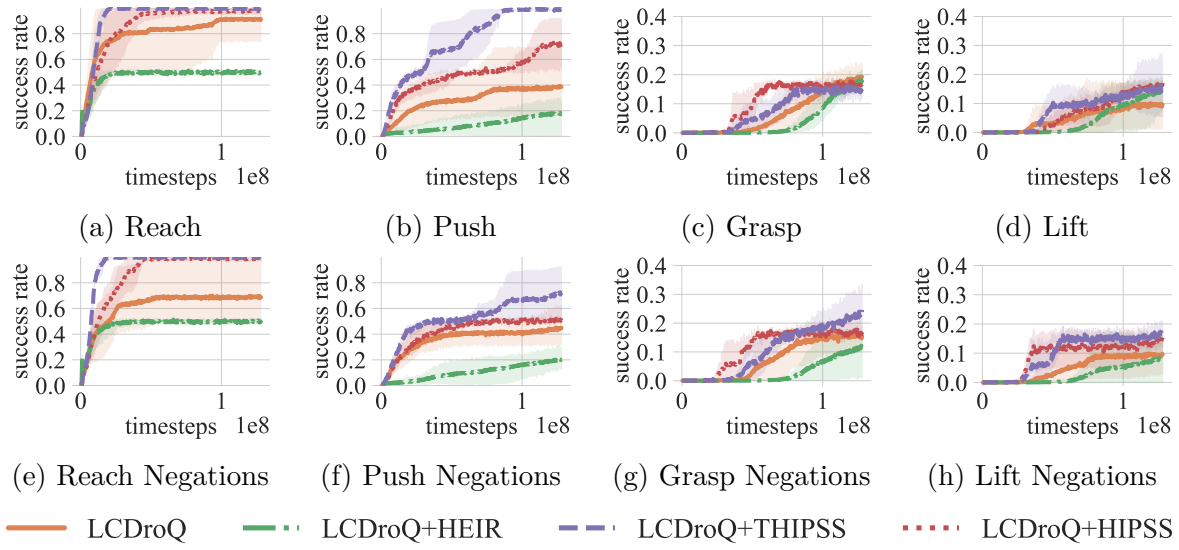


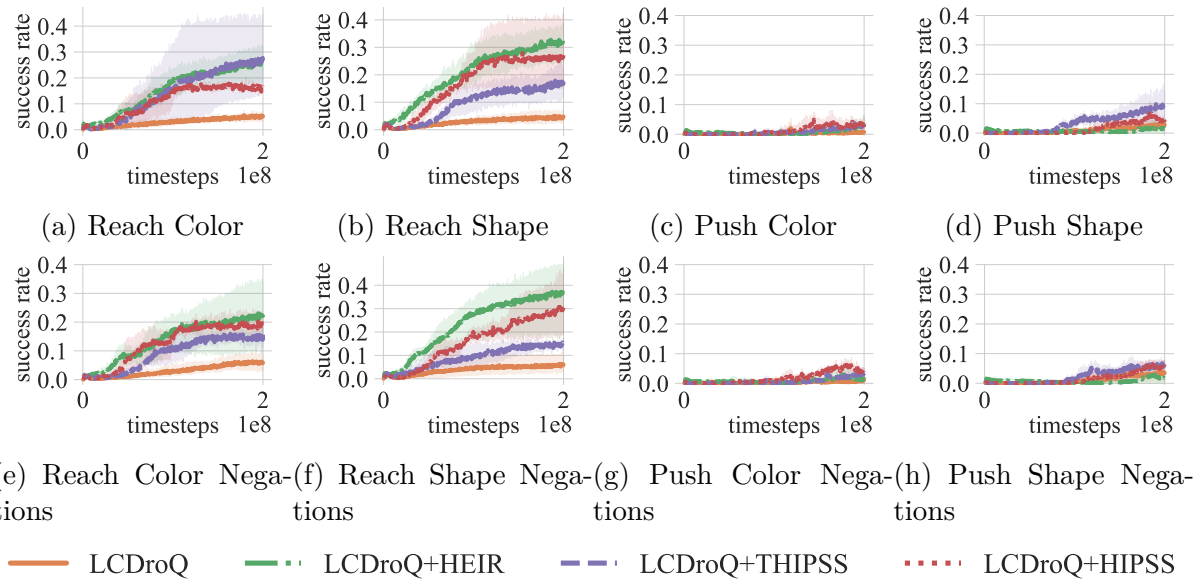
Figure 5.18: We depict our baseline **LCDroQ**, next to **LCDroQ+HEIR** as the baseline extended by the expert feedback, and **LCDroQ+HIPSS** and **LCDroQ+THIPSS**, where **HIPSS** and **THIPSS** incorporate the egocentric speech approaches from Section 4.7. Action correction challenges for the *reach*, *push*, *grasp*, and *lift* tasks without any form of color or shape extension are shown. The default action correction tasks are shown in the first row, while those with negations are depicted in the second row. We show the average success rate and the 90% confidence interval for 5 random seeds (y-axis) with respect to the number of environment steps taken (x-axis). We employ the GRU as a language encoder, while we provide further results in Figure B.16.

Reach and Push Color-Shape Extensions

In the following, we provide results for a selection of more difficult tasks, extending the range of colors and shapes that the agent encounters. This yields an interesting setting for **HEIR**, **HIPSS**, and **THIPSS**, as previously shown, in which a higher diversity of instructions can enhance the effect of hindsight learning (Röder et al., 2022) (see Section 4.9.4).

For the *reach* task with additional colors, we present results for the default setting in Figure 5.19a and the negations in Figure 5.19e. In both settings, **HEIR**, **HIPSS**, and **THIPSS** provide the best success results. **LCDroQ** in this case fails to make any learning progress, only reaching a success rate of 0.05 to 0.1. As **HEIR** and **THIPSS** reach success rates of 0.3 in the default case and **HEIR** and **HIPSS** values of 0.2 in the case of the negations, we expect the methods to only solve the usual instruction following, without even tackling the action corrections, which would be guaranteed upon exceeding the 0.5 mark. Similar insights can be drawn from the *reach* task with the shape extension (Figures 5.19b and 5.19f), where **THIPSS** lags behind, while **HEIR** and **HIPSS** are leading.

For the *push* task with additional colors (Figures 5.19c and 5.19g) and shapes (Figures 5.19d and 5.19h), we do not observe any noteworthy training progress, indicating that these tasks are too challenging for our current approaches, except for **THIPSS** in Figure 5.19d, which shows minimal progress.



(a) Reach Color (b) Reach Shape (c) Push Color (d) Push Shape

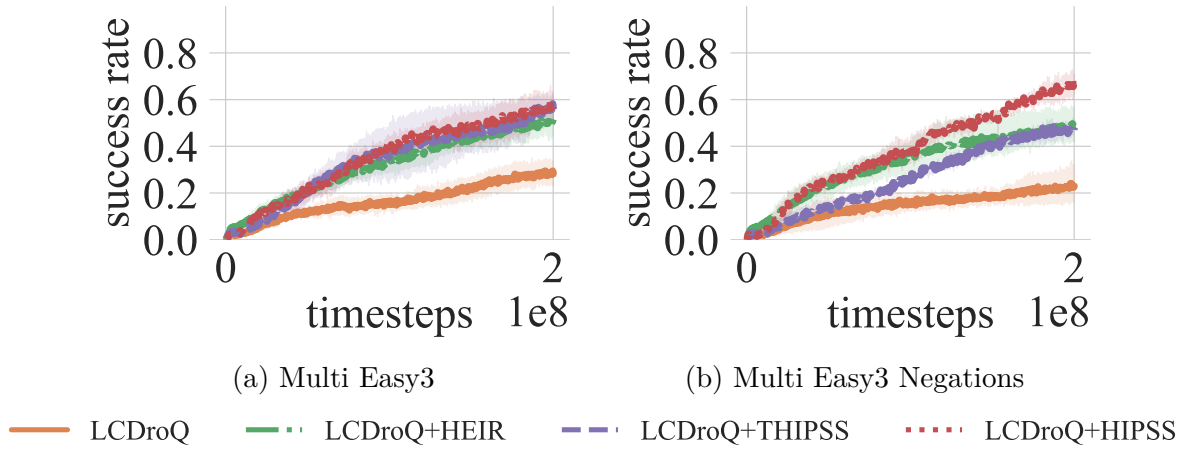
(e) Reach Color Negations (f) Reach Shape Negations (g) Push Color Negations (h) Push Shape Negations

Figure 5.19: We illustrate experimental results for our approaches **LCDroQ**, **LCDroQ+HEIR**, **LCDroQ+HIPSS**, and **LCDroQ+THIPSS** in the *reach* and *push* action correction tasks involving the *color* and *shape* extensions. Depicted are the average success rates, with the shaded regions indicating the 90% confidence interval (y-axis), and the environment steps taken (x-axis). All trials employ the GRU language encoder and contain 5 random seeds.

Summary In all *reach* tasks, only the **HEIR** and **HIPSS** replay extensions are capable of making significant learning progress, although **THIPSS** also performs quite well while lagging slightly behind. **LCDroQ+HEIR**, **LCDroQ+HIPSS** and **LCDroQ+THIPSS** exceed the success rate of **LCDroQ** by up to a factor of 4 (Figures 5.19a and 5.19e) and even reach success rates up to 7 times higher (Figures 5.19b and 5.19f). The performance of the **HIPSS** replay extension can only be explained by the scarcity of successful episodes (the training samples for the seq2seq model) and an uneven distribution of training samples, as hinted at in Subsection 4.9.5. The variations of the *push* task did not provide meaningful insights, suggesting that they are too challenging for our current approaches.

Multitask

Our last batch of hindsight action correction experiments concerns the multitask setting for both the default and negation misunderstanding setup. Unlike the previous challenges, action corrections also incorporate the task behavior that could potentially be corrected, referencing the task verbs. This makes the language setup even more challenging by providing a higher diversity of instruction correction combinations (cf. Section 3.4.1). As shown in Figure 5.20 for the action correction case, our approach **HIPSS** exploits this language diversity and surpasses our baseline **LCDroQ** by a success rate of up to 0.3 in the default and 0.5 in the negations setting. In general, **HIPSS** is closely followed by **HEIR** and **THIPSS**, while in Figure 5.20b it surpasses them with an even larger margin. We provide further results in Section B.3.



(a) Multi Easy3 (b) Multi Easy3 Negations

Figure 5.20: We depict the experimental results for our easy *multitask* settings involving misunderstandings without and with negations. The graphs show the average success rate and the 90% confidence interval on the y-axis with the taken environment steps on the x-axis. Each average is the results of 5 random seeds all using the GRU as language encoder, with more results shown in Section B.3.

Summary Both misunderstanding variations do not indicate significant differences for the most part. However, **LCDroQ+HIPSS** shows to better exploit the action corrections

with negations, as shown by its higher success rate. Similar to the previous experiments in the prior paragraph (see Figure 5.18), **HIPSS** is capable of exceeding the perfect expert instructions of **HEIR**, highlighting it as a valuable approach for language grounding in conversational settings with communication noise induced through the self-speech.

5.8 Summary

This chapter explored the aspects of action corrections within our language-conditioned reinforcement learning benchmark **LANRO**, particularly focusing on a verbal/nonverbal conversation between a synthetic caretaker and a robot. We introduced a fundamental issue of interpreting and executing language instructions by robots, investigating the nuances and challenges of communication, namely misunderstandings caused by ambiguities, the lack of common ground, and the inherent imperfections in human communication, such as disfluencies through the faults of the speaker's tongue. We are among the first to model conversational action corrections within RL using our **LANRO** environment to build a setup that jointly considers verbal and physical features in a sparse reward setting. Modeling conversational misunderstandings could be viewed as a sequential multitask setting. Although we presented this setting for two distinct goals, the initial instruction and the subsequent action correction, future work might consider the case of more than two tasks and even the unlimited continual learning case. With this, we can positively answer **Research Question 3** as we provide a benchmark that needs to jointly consider both the situatedness of the agent and the caretaker instructions. To resolve misunderstandings within this language-conditioned RL setting, we introduced our novel method **LCDroQ** in Section 5.5 that incorporates the notion of uncertainty into the critic function, addressing the uncertainty of the instruction because the agent cannot be sure whether an action correction will follow. The experiments showcase that **LCDroQ** outperforms our prior **LCSAC** approach in all shown experiments by a factor of up to 8 (Figure 5.13b). As a second contribution, we employ our **HIPSS** and **THIPSS** approaches within the action correction setup to support the grounding of noisy instructions and action corrections in hindsight. The experiments showcase the synergy of **LCDroQ** to improve the performance of the agent in order to harvest successful episodes and **HIPSS** to exploit them to learn the action correction prediction, later used for hindsight learning. According to the presented results, especially in the basic *reach* and *push* tasks, as well as the *multitask* setting, we can positively answer **Research Question 4**. Both hindsight prediction approaches, **HIPSS** and **THIPSS**, improve solving the action correction tasks through hindsight instruction and correction predictions, exceeding the performance of our baseline by a factor of 2 in Figure 5.18b and close to 3 in Figure 5.20, respectively. This is in line with our work Röder et al. (2022), where **HIPSS** exploits the mistakes made in difficult tasks with an extended number of instructions. With this chapter, we highlighted the importance of situated language understanding in cases of misunderstandings that could only be resolved by considering the situatedness and the verbal feedback. We expect these results to provide helpful insights for human-robot interaction and other conversational systems.

Chapter 6

Conclusions

The following section summarizes the thesis. We review and answer the proposed research questions from Section 1.5 and conclude the work with thoughts on its limitations and possible future directions.

In this thesis, we introduced mechanisms for sample-efficient language-conditioned reinforcement learning, presenting implementations grounded in concepts from cognitive science and language development. To this end, we introduced our robotics benchmark, **LANRO**, as a testbed for instruction-following and verbal/nonverbal conversation, with various tasks for language grounding based solely on sparse rewards. We employ an extension of an established actor-critic algorithm to ground language, introducing various techniques to process language goals alongside the sensor states of the Markov decision process for action selection and return estimation. Next, we addressed the sample complexity involved in language grounding, which requires millions of data samples to successfully learn a policy that solves a task.

To alleviate the aforementioned sample complexity, we propose two hindsight learning methods derived from goal-conditioned RL for the language-conditioned case, employing the idea of hindsight instruction replay in the context of a developmental RL setup. This setup considers a verbal/nonverbal interaction between a caretaker and a robot, where the former provides goals as language instructions, and the latter communicates via physical actions only. Of the two proposed methods that enhance sample efficiency, the first incorporates feedback from the caretaker, and the second employs a mechanism of egocentric speech, allowing the agent to *"talk to itself"*. As expected, the expert feedback from the caretaker improves the performance of the baseline by a factor of up to 3 (see Figure 4.21d), as the hindsight language instructions always perfectly match the observed behaviors. In contrast, our self-speech approach injects noise into the learning process, similar to a toddler, by sometimes generating syntactic errors. This approach enhances the robustness and stability of learning, improving asymptotic performance and outperforming both the baseline and caretaker feedback by a factor of up to 8 (see Figure 4.22k).

The final set of contributions investigates the concept of conversational breakdowns caused by misunderstandings, of which we distinguish three types: ambiguities, lack of common ground, and instructor mistakes. We address misunderstandings in the same

manner as in language-conditioned setups, by providing additional linguistic goal information that we call **action corrections**. Our extension of **LANRO** implements this conversational setting by allowing the instructor to extend the original language goal. In this way, the agent must now solve two successive goals, presenting an increased challenge because the agent needs to focus not only on the language but also on the state of the world and any changes that might have been caused by the conversational failure. To address this, we introduced a new type of language-conditioned algorithm that accounts for the uncertainty of action corrections, significantly improving language grounding by a factor of up to 8 (see Figure 5.13b). By improving the resolution of action corrections, we generated sufficient number of successful episodes, which we used to train our hindsight instruction replay model, emulating egocentric speech. Compared to the expert feedback, which cannot simulate action corrections for hindsight learning, the agent, through self-speech, improves its ability to resolve noisy conversations by a factor of up to 2 (see Figures 5.18b and 5.18f). Figure 6.1 summarizes this thesis by contrasting the paradigms of hindsight instructions and action corrections, showing that both cases originate from an undesirable outcome. Various factors, such as misunderstanding underspecified instructions, poorly conveyed directives, or a mismatch in shared common understanding, are possible causes. In any of these cases, hindsight learning can directly utilize the outcome, relabeling it to learn as if it were the desired one. Conversely, with action corrections, the operator recognizes the misunderstanding and provides additional contextual feedback.

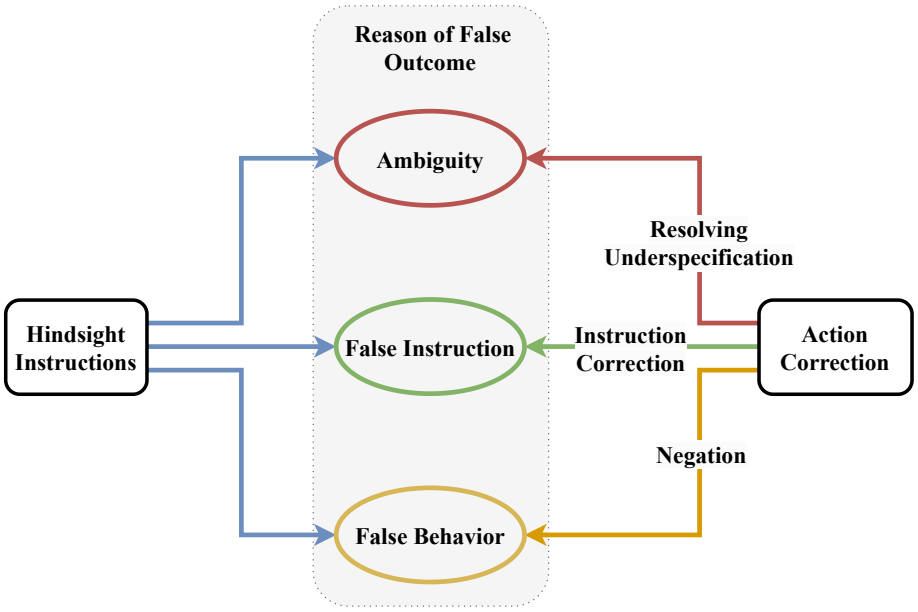


Figure 6.1: This figure contrasts hindsight instructions and action corrections, both of which address undesirable outcomes stemming from various underlying reasons (center, see Section 5.2). In the former case, undesirable outcomes guide retrospective learning, whereas in the latter case, an action correction aims to revise the behavior that led to the unwanted outcome resulting from a misunderstanding.

6.1 Answering the Research Questions

In the following sections, we summarize the core contributions of the presented chapters for the research questions proposed in Section 1.5.

6.1.1 RQ1: Language Grounding with Sparse Rewards

To what degree can language grounding in robotics reinforcement learning tasks be exclusively modeled through sparse rewards?

In Chapter 3, we introduced the concept of language processing for reinforcement learning, particularly the idea of language grounding based on sparse rewards. With Section 3.4, we presented our language-learning robotics benchmark, LANguage RObotics (LANRO), which provides a wide variety of tasks involving the manipulation of objects with different properties. The robotic agent is tasked with following a language goal in the sense of instruction-following, where the goals specify properties that must be parsed to identify the goal object. These properties include various colors, shapes, sizes, and weights, requiring a grounded understanding due to the physical differences involved (see Subsection 3.4.1).

Furthermore, we introduced our baseline algorithm, built on top of the Soft Actor-Critic (SAC) introduced in Subsection 2.2.7, extended by the language extension called Language-Conditioned Soft Actor-Critic (LCSAC). In Section 3.5, we present the results of our initial **LCSAC** approach, introduced in Röder et al. (2022), alongside the derivative Language-Conditioned DroQ (LCDroQ), across a broad selection of tasks, which we discussed in greater detail in Subsection 5.5.2. With these insights, we can positively answer **Research Question 1**, as we provide extensive results in our language benchmark, where the baseline shows learning progress in simple tasks but, as expected, struggles to solve more challenging tasks with greater object property variations. Additionally, we offer comparisons of different types of language encoders introduced in Subsection 3.2.3. However, there are many tasks that the baseline struggles to solve or that require millions of environment interactions, highlighting the difficulty of exploration in sparsely rewarding tasks and the associated sample complexity.

6.1.2 RQ2: Developmental Approaches for Efficient Language Grounding

How can insights from developmental science enhance language grounding and learning efficiency in intelligent robotic agents?

Language grounding still requires an extensive amount of experiences collected in the environment, resulting in our baselines yielding poor sample efficiency. In Chapter 4, we address this problem by investigating the notion of hindsight learning. Hindsight learning, usually applied in the goal-conditioned setting, requires further adaptations to be usable for the language-conditioned setting. Borrowing the notion of a caretaker (Colas et al., 2020; Akakzia et al., 2021), our environment **LANRO** is extended by a mechanism that provides the agent with a correct goal instruction after it fails to achieve the desired outcome. The

actual outcome can still be useful if something valuable happened that could have been correct in a different context. Guided by our previous review (Röder et al., 2021), we highlight two hindsight replay mechanisms that incorporate ideas from language development (cf. Section 1.4). In the first approach, we employ the caretaker feedback that directly relabels failures with suitable goal instructions in hindsight. This approach, called Hindsight Expert Instruction Replay (HEIR), replicates the close interaction between a teacher and a learner. In the second approach, we imitate the concept of egocentric speech. Instead of relying on the caretaker's feedback, the agent actually learns to *"talk to itself"*. In other words, for undesired outcomes, the learner generates hindsight goal instructions autonomously. This mapping is learned in a self-supervised manner, as the agent harnesses previous successful episode outcomes as positive examples for training the egocentric speech model. In Subsection 4.7.3, we describe the approaches called Hindsight Instruction Prediction from State Sequences (HIPSS) and Transformer-based Hindsight Instruction Prediction from State Sequences (THIPSS) as implementations of self-speech. Section 4.9 summarizes the results for the aforementioned hindsight approaches. As expected, the expert feedback outperforms the other methods in many tasks, achieving learning improvements by up to a factor of 2 in comparisons to the baseline (see Figure 4.21). As shown by our prior work (Röder et al., 2022), the egocentric speech approaches **HIPSS** and **THIPSS** achieve performance comparable to the expert feedback of **HEIR**, with even better asymptotic performance (see, *e.g.*, Figures 4.22a to 4.22c). This validates the idea of egocentric speech as an effective improvement for instruction-following, in some cases even outperforming **HEIR** (see Figures 4.22h, 4.22i, 4.22k and 4.22l). This advantage can be attributed to the noise induced by self-speech, as the model, like a toddler, sometimes generates syntactic errors during the continuous learning phase. Further studies on systematic generalization of **HIPSS** and **THIPSS** in Subsection 4.9.5 demonstrate their capability to generate out-of-distribution object property combinations not encountered during training. We hypothesize that our egocentric speech approach is a key method in language-conditioned RL to resemble the phenomenon of the *vocabulary spurt* observed by Plunkett et al. (1992) in early child development. Specifically, it serves as a method to mimic the self-narrating speech of toddlers and fosters language learning and grounding.

These results confirm our hypothesis that mechanisms inspired by language development can improve sample efficiency by utilizing hindsight language grounding. With this, we can positively answer **Research Question 2**.

6.1.3 RQ3: Modeling the Problem of Misunderstandings

Is it feasible to develop a robust, embodied conversational human-robot interaction model that functions as a standardized benchmark for assessing and quantifying misunderstandings?

In Chapter 5, we examine the origin of failures caused by misunderstandings of the language goals in a verbal/nonverbal conversation, where the instructor verbally commands the agent to follow instructions solely through physical actions. Inspired by contemporary work on dialog systems (Ashktorab et al., 2019) and human-robot interaction (Thierauf

et al., 2023) that examine misunderstandings, we present our own interpretation of a conversational reinforcement learning setup. To this end, we present three cases of misunderstanding that we have integrated into our language-conditioned reinforcement learning benchmark **LANRO**, namely: ambiguities caused by underspecifications, a mismatch of common ground, and the instructor mistakes. For each type of misunderstanding, we implement a procedure that detects if the agent deviates from the desired behavior (see Section 5.4). Following this, the caretaker provides the action correction as feedback, based on which the agent attempts to adjust its behavior. In a way, the agent solves two goals sequentially: the initial misunderstood instruction and the one extended by the action correction (see Figure 5.3). Supported by our publication Röder and Eppe (2022), we affirmatively address **Research Question 3** by modeling the verbal/nonverbal dialog containing action corrections in a reinforcement learning setting with two sequential goals and sparse rewards.

6.1.4 RQ4: Solving Misunderstandings with Action Corrections

How can insights from cognitive science and developmental psychology be leveraged to improve a reinforcement learning approach for resolving misunderstandings?

Learning to resolve uncertainties requires the agent to anticipate the possibility of a misunderstanding arising when following simple instructions. In the second part of Chapter 5, we introduced our extension of **LCSAC**, coined Language-Conditioned DroQ (LCDroQ), which incorporates the idea of Dropout Q-functions (DroQ) proposed by Hiraoka et al. (2022). In essence, **LCDroQ** approximates Bayesian inference, helping to account for multiple possible futures when interacting in an action correction setting. In our case, **LCDroQ** is able to capture instructional uncertainty, as shown in Figure 5.15, by employing an ensemble of dropout language-conditioned critic networks. This extension demonstrates its advantages in Section 5.7, where we compare the former **LCSAC** to our novel **LCDroQ** approach, improving learning and, hence, the success rate of resolving misunderstandings by up to a factor of 8 (see Figure 5.13b). The improvement achieved by **LCDroQ** makes it feasible only then to employ the self-supervised replay mechanisms, as they require the agent to harvest enough successful episodes to learn the egocentric speech mapping. In Section 5.6, we reintroduced **HIPSS** and **THIPSS** into the action correction setting, where they attempt to predict either default instructions or those extended by the action corrections. Experiments utilizing self-generated instructions in Section 5.7 showcase an improvement in learning by a factor of up to 3 (see Figures 5.18b and 5.18f). We also employ the hindsight expert instruction replay mechanism **HEIR**, which plateaus at a success rate of 0.5, hence 50%, which aligns with our uniform sampling of episodic instruction-following and action correction tasks (see Section 5.4). **HEIR** is not capable of generating hindsight action corrections like **HIPSS** and **THIPSS** and, therefore, only supports learning the default instruction-following. This highlights the advantages of the self-supervised egocentric speech approach, which enables the agent to *"talk to itself"*, even in the more challenging action correction tasks. Both the dropout ensemble and the self-speech replay mechanism from Chapter 4 provide a positive answer to **Research Question**

4 by significantly increasing the sample efficiency of resolving misunderstandings.

6.2 Limitations and Future Directions

In the following, we provide future pointers based on the limitations of our work.

Similar to previous studies (Hermann et al., 2017; Chevalier-Boisvert et al., 2019; Cideron et al., 2020), our setup involves a simple kind of language and a restricted set of objects. We made this decision when designing **LANRO** due to the difficulty of sparse reward tasks that require learning both language understanding, to identify the goal, and the policy behavior, to manipulate the identified goal object and solve the task from the ground up. Works like Akakzia et al. (2021) consider a more sophisticated language, employing sentences that involve spatial relations and conjugations while utilizing hand-engineered language representations for this purpose. We assume that integrating a more sophisticated language in combination with a more advanced language encoding procedure could drastically improve the presented approaches, making them even more viable for human-robot interaction (Chai et al., 2014; Eppe et al., 2017).

The results of this thesis only consider the fully observable MDP setting, while current robotics applications rely on language and visual inputs (Nair et al., 2018b; Lynch and Sermanet, 2021; Ahn et al., 2023; Ma et al., 2023). Including pixel observations provides not only a more realistic and general setup but also requires fewer assumptions about the structure of the underlying task. Nevertheless, such setups often demand more computational power, and their longer runtime slows down the progress of developing novel algorithmic ideas. Hence, we investigated our approaches in the more computationally efficient setting of state-based inputs and believe that many of the insights generated can transfer to the visuo-lingual domain.

Our action correction setting incorporates only three different types of misunderstandings. However, there are many more types of conversational failures that require different forms of correction, which are not considered in this work. This limitation is mainly linked to the simplified language that we use when instructing the agent. Extending this scope would also introduce different situations that require other kinds of action corrections, as listed in Section 5.2.3. Although our setup only considers one action correction (see Section 5.3), we provided a general formulation that could incorporate an unlimited sequence of action corrections, thus resembling a continual learning setting with constantly changing goals.

The action correction setting does not involve natural language generated by the robot for communication purposes but only utilizes it for hindsight instruction replay (Röder et al., 2022). We posit that, with current insights from language modeling (Brown et al., 2020), generating robot utterances is becoming increasingly feasible. Other works have shown that the agent can communicate its current perceptions to inform the instructor about its limitations (Chai et al., 2014). For instance, the agent might request further information or express its uncertainty regarding upcoming courses of action instead of waiting for an intervention.

Although LLMs, as mentioned in the introduction of this thesis, to a certain extent lack the representations obtained by grounding, one possible use case could be to translate the action corrections and negations of Chapter 5 into simple goal instructions that the agent could directly solve, rather than learning the full resolution process end-to-end. For example, mapping the action-corrected instruction *"push the red cube, sorry, the blue one"* into *"push the blue cube"*. Other approaches consider using LLMs to synthesize routines that the agent should execute (Huang et al., 2023a) or to accompany instruction-following for textual observations (Carta et al., 2023), which is particularly relevant for contemporary dialog systems.

Appendices

Appendix A

Algorithm Details

A.1 Language Encoder Details

Word Embedding For all word embeddings, we use vectors in $\mathbb{R}^{d_{\text{embedding}}}$, where $d_{\text{embedding}}$ is 64.

Multilayer Perceptron The MLP as language encoder flattens the input sentence of length L into a vector of size $\mathbb{R}^{L \cdot d_{\text{embedding}}}$. We set the output dimensionality of the MLP to 256.

Gated Recurrent Unit The recurrent neural network that we employ as language encoder contains a single GRU cell (Cho et al., 2014) with a hidden size of 256. We return the last hidden state as the language encoding.

Self-Attention For the self-attention based language encoder (Vaswani et al., 2017), we process the input sentence of individual word embeddings $\mathbb{R}^{L \times d_{\text{embedding}}}$ as a whole to allow the attention to encode word-based relationships. We fix the number of heads to 1, the dimensionality for the query $d_q = 32$, key $d_k = 32$ and value $d_v = 32$ vectors, and the output dimension to $d_o = 64$. We flatten the whole sentence representation to get a vector of size $\mathbb{R}^{L \cdot d_o}$

A.2 Language-Conditioned Soft Actor-Critic

Parameter	Values
batch size	256
hidden size of MLPs	256, 256
MLP activation functions	ReLU
learning rate actor, critic and temperature	1e−3
number of workers	16
initial temperature coefficient α	0.2
buffer size	1e+6
discount γ	0.98
update-to-data ratio	10
polyak ρ	0.995
number of critics	2
critic dropout rate	0.0
critic layer normalization	`False`

Table A.1: The hyperparameters for our **LCSAC** baseline first introduced in Röder et al. (2022).

A.3 Language-Conditioned DroQ

Parameter	Values
batch size	256
hidden size of MLPs	256, 256
MLP activation functions	ReLU
learning rate actor, critic and temperature	1e−3
number of workers	16
initial temperature coefficient α	0.2
buffer size	1e+6
discount γ	0.98
update-to-data ratio	10
polyak ρ	0.995
number of critics	2
critic dropout rate	0.01
critic layer normalization	`True`

Table A.2: The hyperparameters for our novel baseline **LCDroQ** derived from our previous **LCSAC** implementation (Röder et al., 2022) and extended by the **DroQ** algorithm (Hiraoka et al., 2022).

A.4 Hindsight Instruction Prediction from State Sequences

Parameter	Values
learning rate	0.0003
batch size	32
sequence limit	20
max gradient norm	1.0
validation accuracy threshold	0.7
validation dataset ratio	0.2
buffer size	5000 × sequence limit

Table A.3: We provide the hyperparameters that both **HIPSS** and **THIPSS** share.

Parameter	Values
encoder hidden size	256
decoder hidden size	256
activation functions	ReLU
teacher forcing ratio	0.5
dropout rate	0.2
word embedding size	128

Table A.4: The hyperparameters for the seq2seq model **HIPSS**.

Parameter	Values
MLP block hidden size	64
MLP block activation functions	ReLU
MLP block dropout rate	0.1
MLP block kernel initialization	xavier uniform
MLP block bias initialization	normal
encoder number of blocks	2
encoder dropout rate	0.1
encoder number heads	4
encoder d_q	32
encoder d_k	32
encoder d_v	32
encoder attention dropout rate	0.1
encoder kernel initialization	xavier uniform
encoder bias initialization	normal
decoder number of blocks	2
decoder dropout rate	0.1
decoder number heads	4
decoder word embedding size	128
decoder d_q	32
decoder d_k	32
decoder d_v	32
decoder attention dropout rate	0.1
decoder kernel initialization	xavier uniform
decoder bias initialization	normal

Table A.5: The hyperparameters for our Transformer-based encoder-decoder **THIPSS**.

Appendix B

Experimental Results

B.1 Language Experiments

In this section of the appendix, we provide the complete results of our language-conditioned baseline **LCDroQ**, complementing Section 3.5. As a continuation from our previous work Röder et al. (2022), we highlight both settings in **LANRO** that involve 2 and 3 objects. The setting with 2 objects (see Subsection B.1.1) provides a suitable initial setup to investigate language grounding with sparse rewards, but it is also limited as the goal identification by chance is already correct 50% of the time. Our setup with 3 objects (see Subsection B.1.2) makes this more difficult, and therefore provides a thorough verification of our implementations.

The plots in this section use 3 random seeds to calculate the average success rate shown on the y-axis and the environment steps shown on the x-axis. The shaded area depicts the 90% confidence interval of the success rate.

B.1.1 Setting with Two Objects

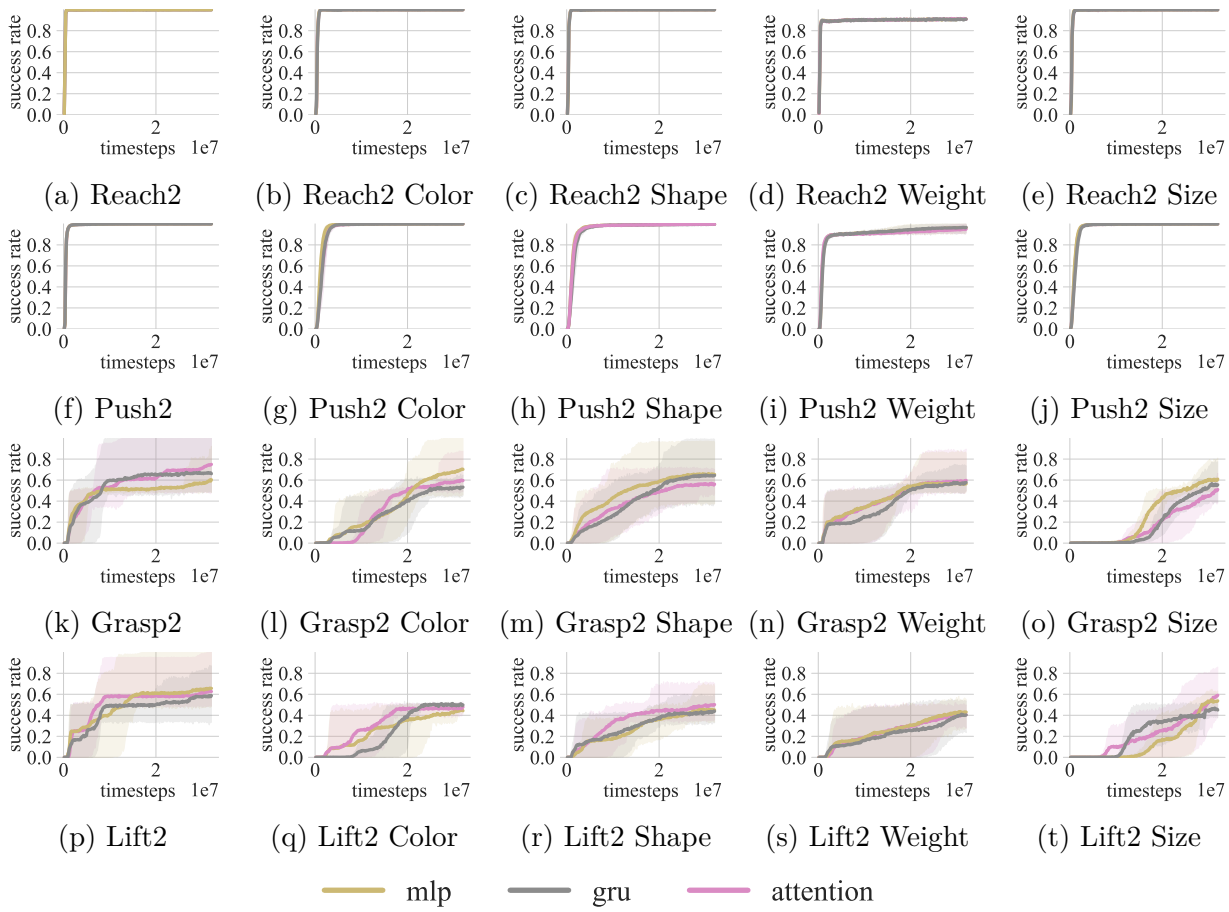
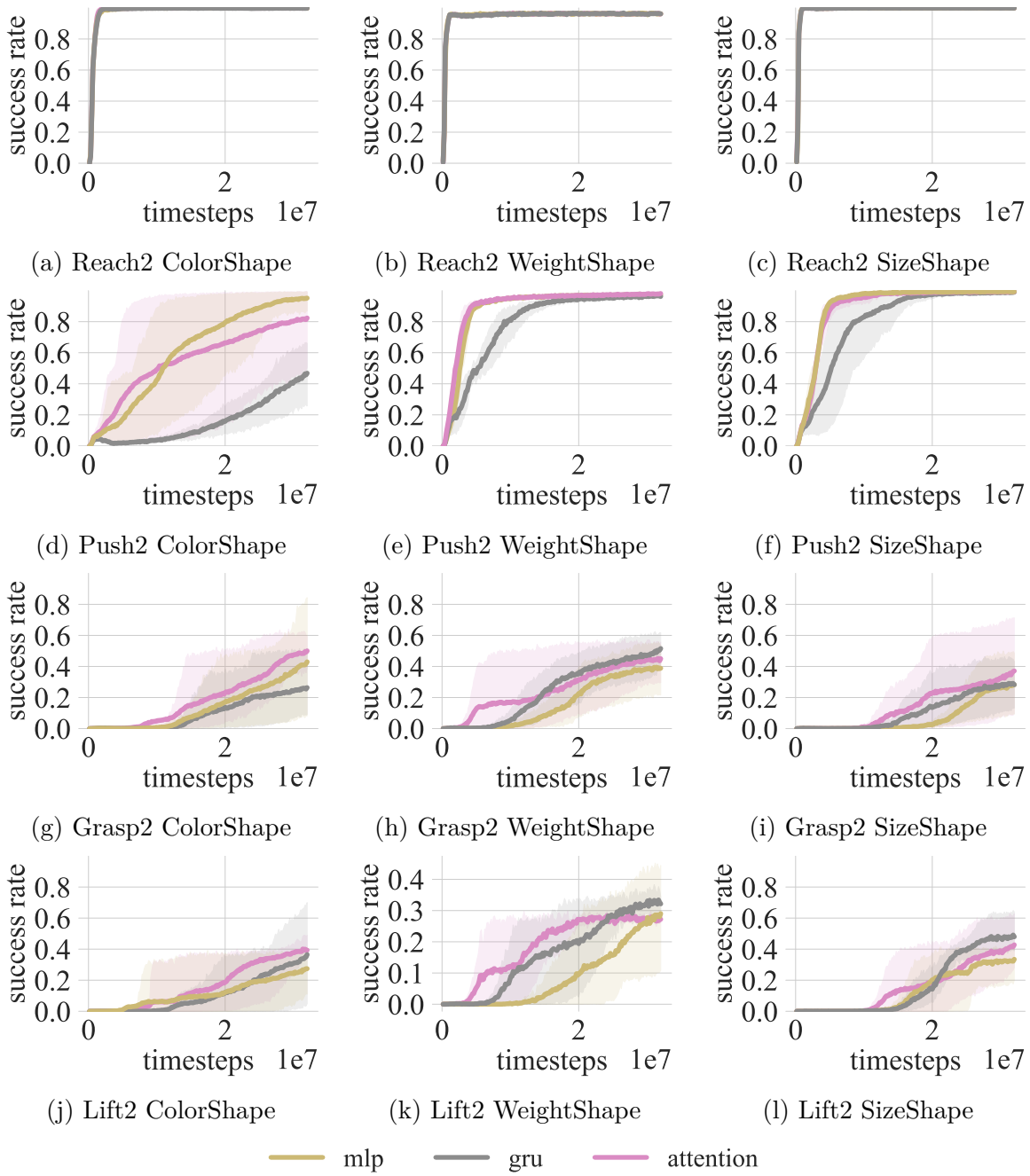


Figure B.1: Illustrated are column-wise from left to right, the *Default*, *Color*, *Shape*, *Weight*, and *Size* options from Table 3.1. We showcase the tasks *reach*, *push*, *grasp*, and *lift* in each row of the figure. Provided are the variations with only 2 objects in the scene.



(a) Reach2 ColorShape (b) Reach2 WeightShape (c) Reach2 SizeShape

(d) Push2 ColorShape (e) Push2 WeightShape (f) Push2 SizeShape

(g) Grasp2 ColorShape (h) Grasp2 WeightShape (i) Grasp2 SizeShape

(j) Lift2 ColorShape (k) Lift2 WeightShape (l) Lift2 SizeShape

Figure B.2: Illustrated are, from left to right, the *ColorShape*, *WeightShape*, and *SizeShape* options from Table 3.1 for the tasks *reach*, *push*, *grasp*, and *lift* in each row of the figure. Scenes with 2 objects are shown. The extended range of property combinations makes these tasks harder to solve.

B.1.2 Setting with Three Objects

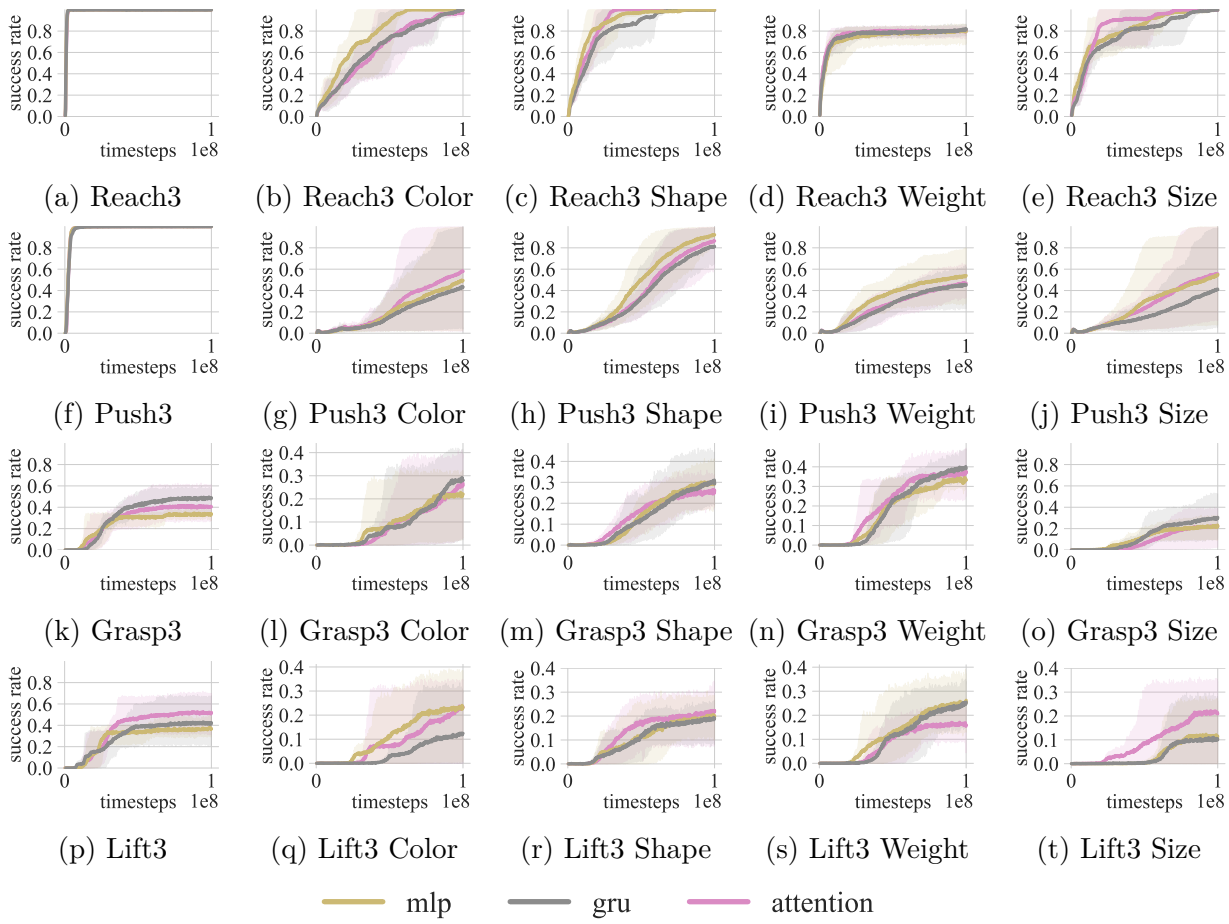
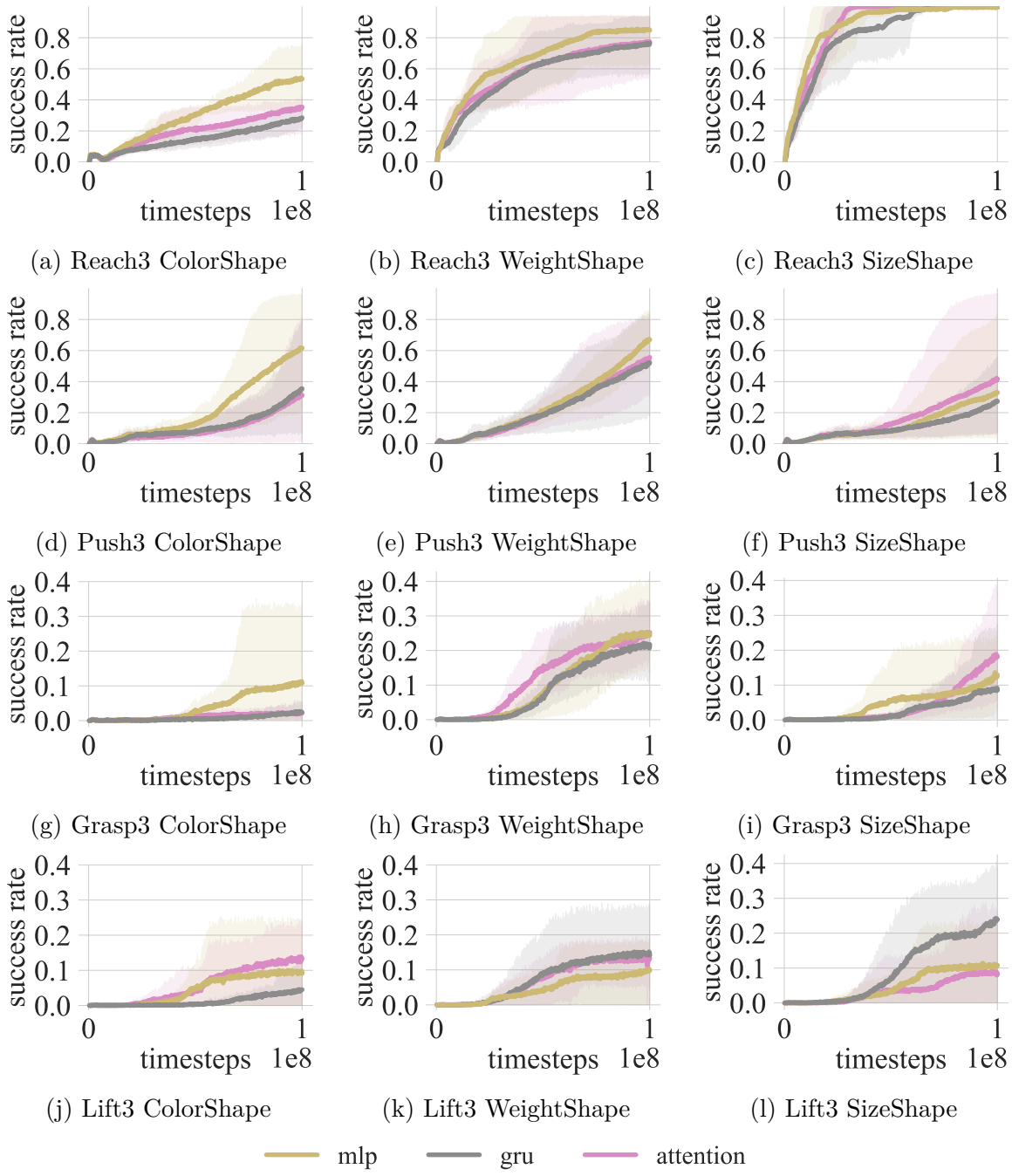


Figure B.3: We illustrate from left to right within each column the *Default*, *Color*, *Shape*, *Weight*, and *Size* options from Table 3.1, while we have the tasks *reach*, *push*, *grasp*, and *lift* in each row of the figure. Provided are the variations with 3 objects in the scene, which are more challenging as there is no 50-50 chance in identifying the correct object, as in the case with 2 objects.



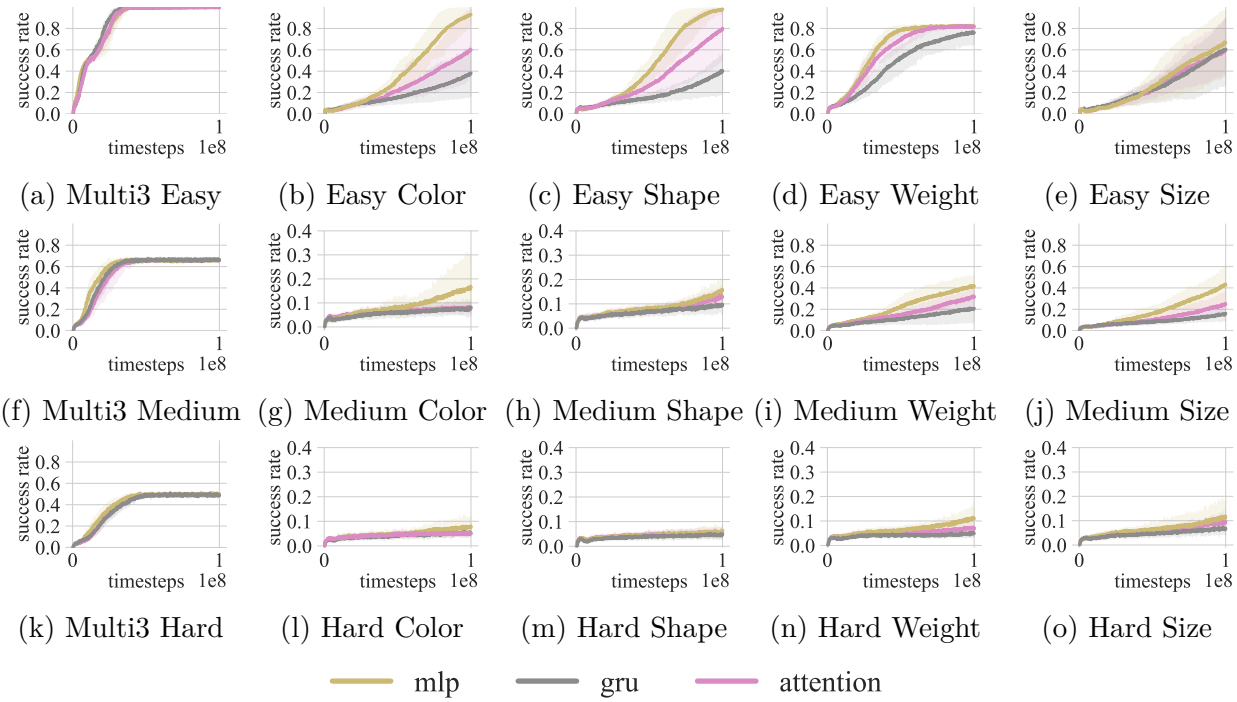
(a) Reach3 ColorShape (b) Reach3 WeightShape (c) Reach3 SizeShape

(d) Push3 ColorShape (e) Push3 WeightShape (f) Push3 SizeShape

(g) Grasp3 ColorShape (h) Grasp3 WeightShape (i) Grasp3 SizeShape

(j) Lift3 ColorShape (k) Lift3 WeightShape (l) Lift3 SizeShape

Figure B.4: Illustrated are, column-wise, the *ColorShape*, *WeightShape*, and *SizeShape* options from Table 3.1 for the tasks *reach*, *push*, *grasp*, and *lift* in each row of the figure. Shown are the settings with 3 objects in the scene. The extended range of property combinations makes these tasks harder to solve.



(a) Multi3 Easy (b) Easy Color (c) Easy Shape (d) Easy Weight (e) Easy Size

(f) Multi3 Medium (g) Medium Color (h) Medium Shape (i) Medium Weight (j) Medium Size

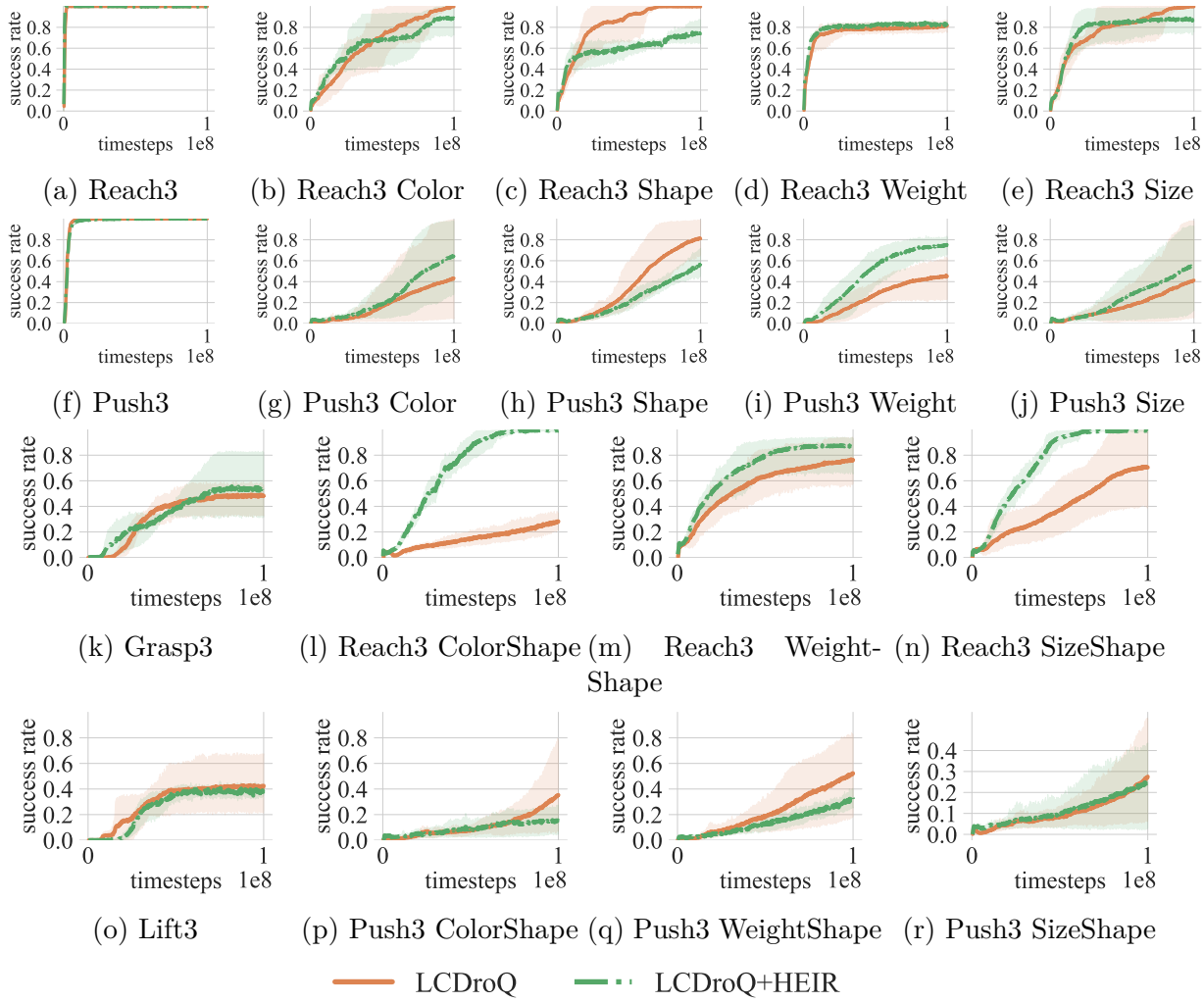
(k) Multi3 Hard (l) Hard Color (m) Hard Shape (n) Hard Weight (o) Hard Size

Figure B.5: We depict column-wise the *Default*, *Color*, *Shape*, *Weight*, and *Size* options from Table 3.1 for the *easy*, *medium*, and *hard multitask* settings in each row of the figure. For all tasks, we include 3 objects. The extended range of property combinations, in combination with the multiple tasks involved, makes these settings the hardest to solve (see Section 3.4.1).

B.2 Hindsight Instruction Experiments

In this section of the appendix, we provide complementary experimental results of our different language encoder types and a broader selection of tasks used in our hindsight experiments using **HEIR**, **HIPSS**, and **THIPSS**.

B.2.1 Expert Feedback without Hindsight Bias



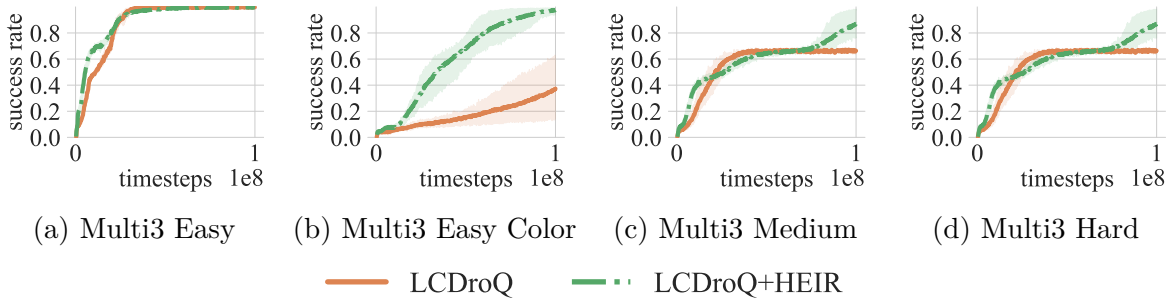
(a) Reach3 (b) Reach3 Color (c) Reach3 Shape (d) Reach3 Weight (e) Reach3 Size

(f) Push3 (g) Push3 Color (h) Push3 Shape (i) Push3 Weight (j) Push3 Size

(k) Grasp3 (l) Reach3 ColorShape (m) Reach3 Weight-Shape (n) Reach3 SizeShape

(o) Lift3 (p) Push3 ColorShape (q) Push3 WeightShape (r) Push3 SizeShape

Figure B.6: Complementary experiments for **LCDroQ** and **HEIR**, utilizing the **GRU** as language encoder.



(a) Multi3 Easy (b) Multi3 Easy Color (c) Multi3 Medium (d) Multi3 Hard

Figure B.7: Complementary *multitask* experiments for **LCDroQ** and **HEIR**, utilizing the **GRU** as language encoder.

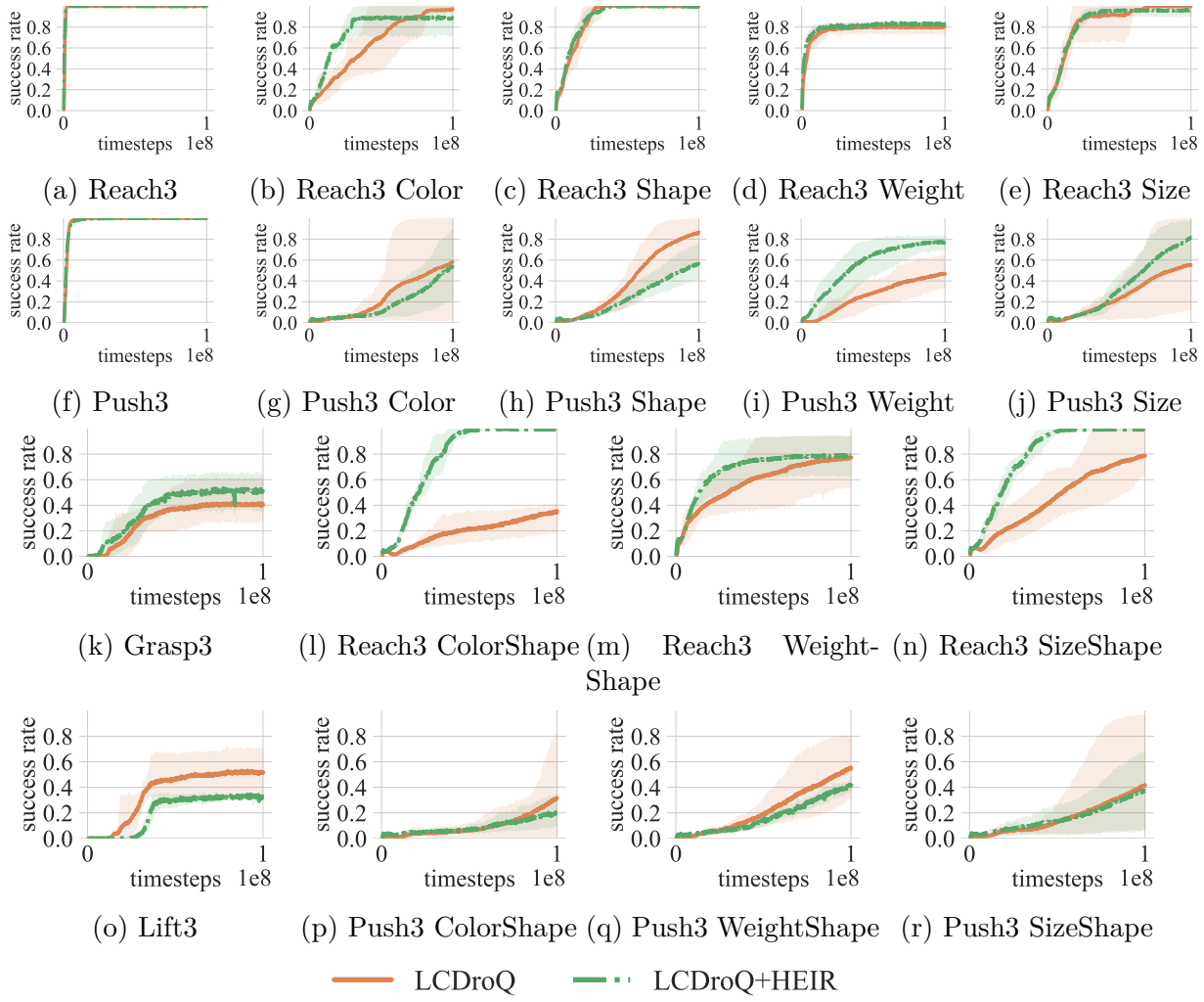


Figure B.8: Complementary experiments for **LCDroQ** and **HEIR** utilizing the **self-attention** as language encoder.

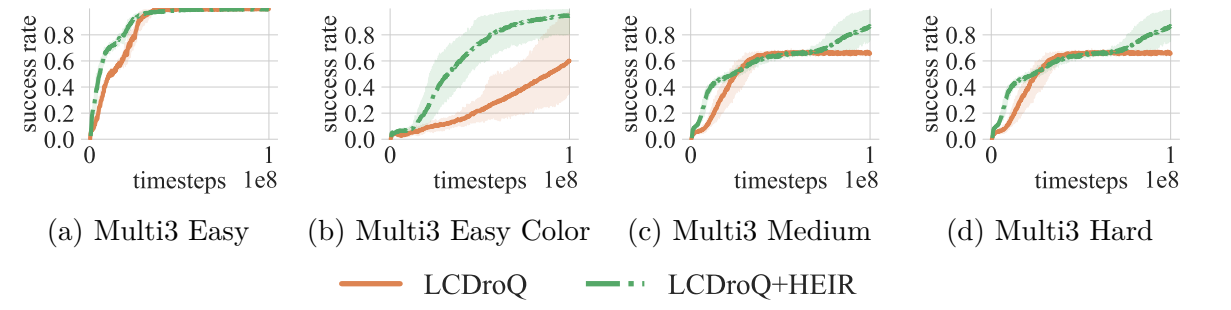


Figure B.9: Complementary multitask experiments for **LCDroQ** and **HEIR** utilizing the **self-attention** as language encoder.

B.2.2 Hindsight Instruction Prediction Experiments

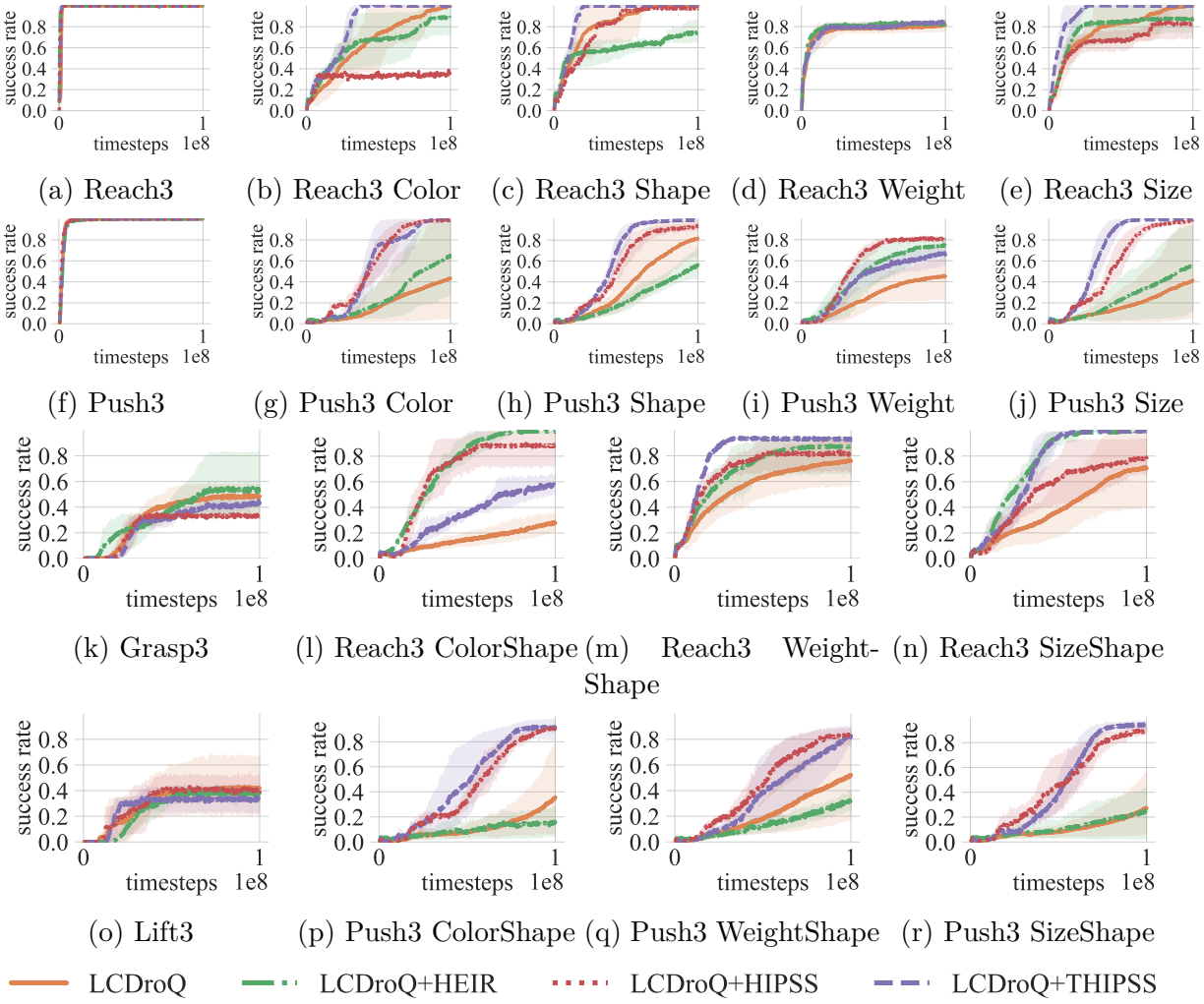


Figure B.10: Complementary experiments for **LCDroQ**, **HEIR**, **HIPSS**, and **THIPSS** utilizing the **GRU** as language encoder.

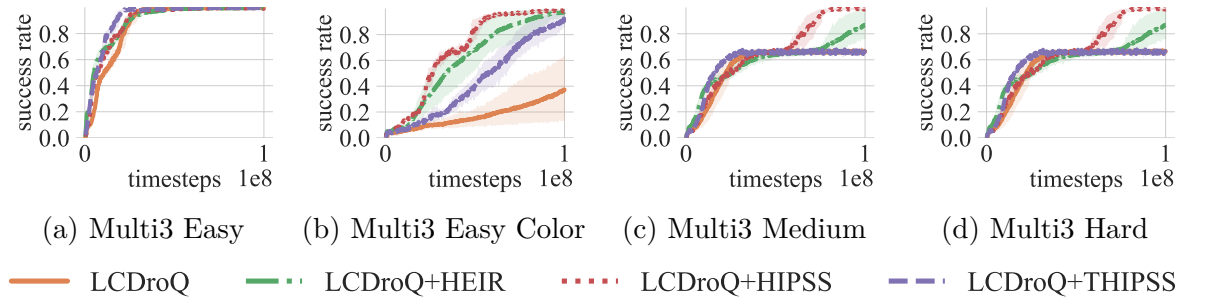


Figure B.11: Complementary multitask experiments for **LCDroQ**, **HEIR**, **HIPSS**, and **THIPSS** utilizing the **GRU** as language encoder.

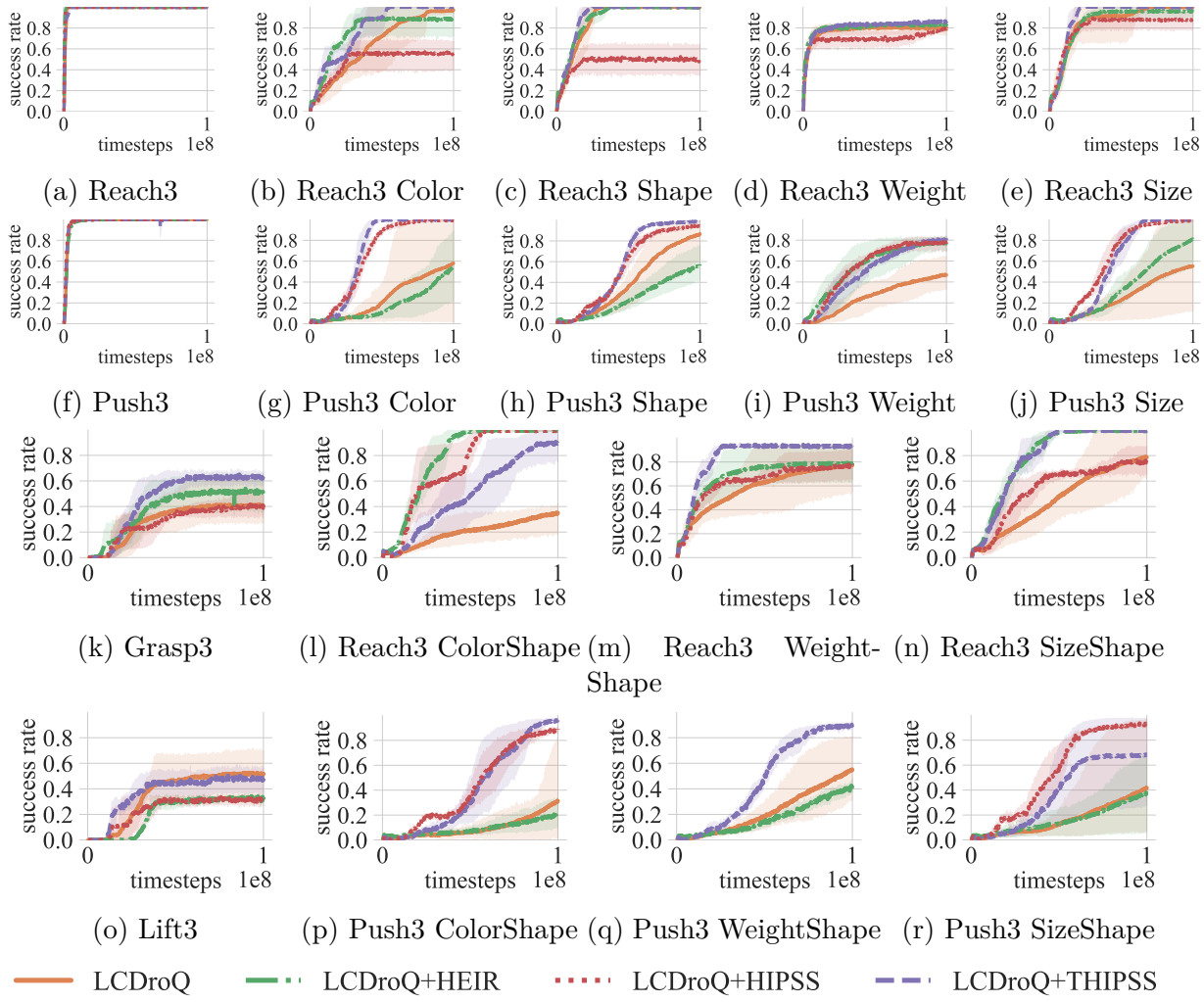


Figure B.12: Complementary experiments for **LCDroQ**, **HEIR**, **HIPSS**, and **THIPSS** utilizing the **self-attention** as language encoder.

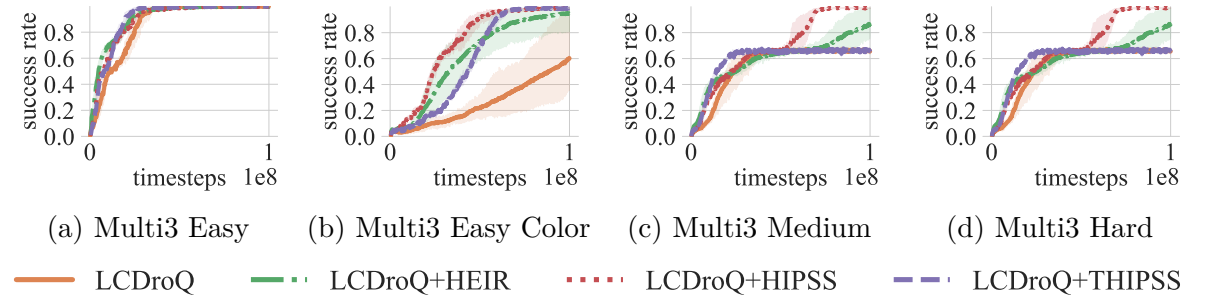


Figure B.13: Complementary multitask experiments for **LCDroQ**, **HEIR**, **HIPSS**, and **THIPSS** utilizing the **self-attention** as language encoder.

B.3 Action Correction Experiments

This section provides the full results of our action correction experiments from Section 5.7.

B.3.1 LCSAC and LCDroQ Experiments

Single Task

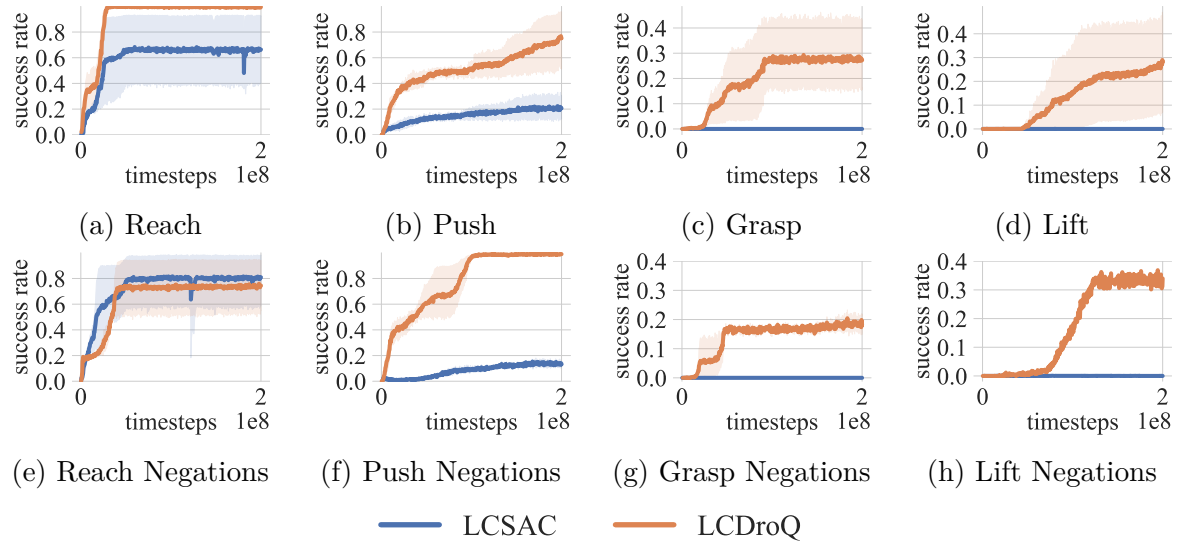


Figure B.14: We provide the additional results for the **GRU** as language encoder, comparing **LCSAC** and **LCDroQ**.

Multitask

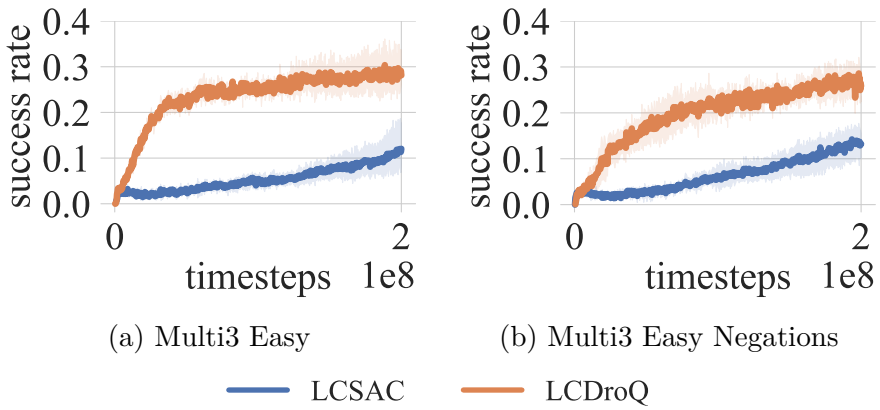
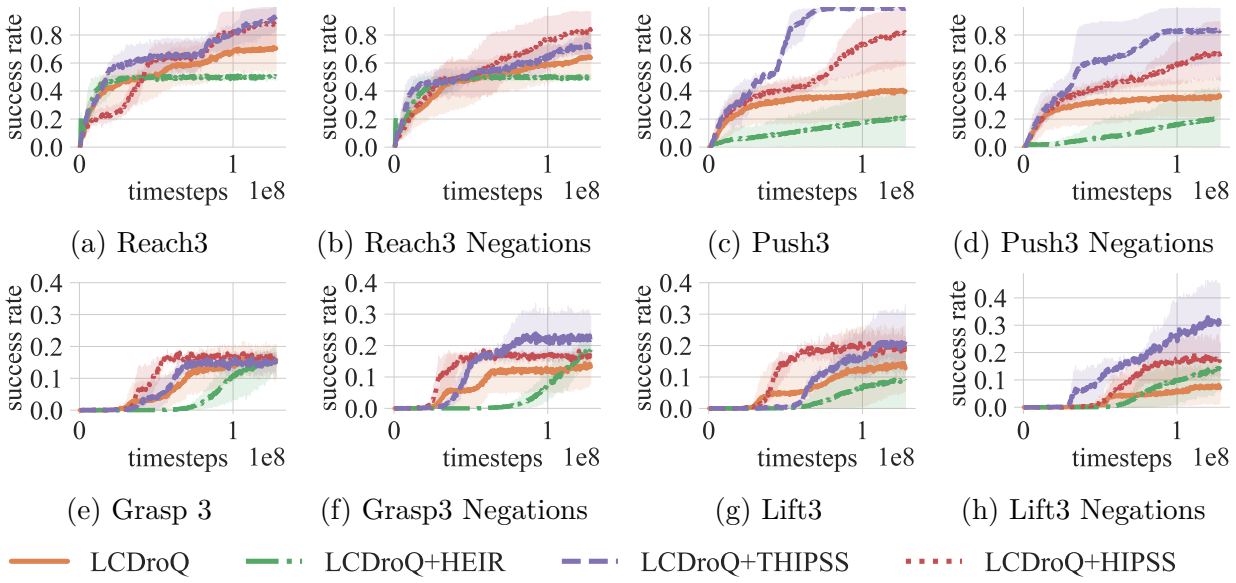


Figure B.15: We provide additional results for the action correction *multitask* setting with and without negations using the **GRU** as language encoder. We compare the performance of **LCSAC** and **LCDroQ**.

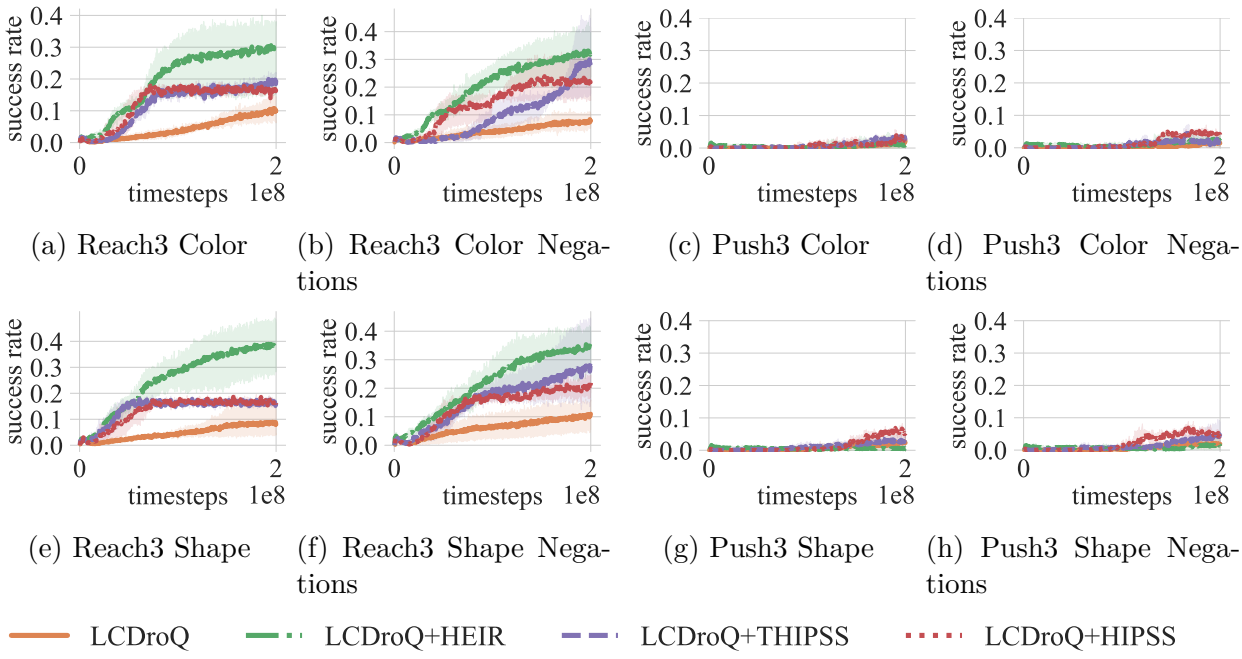
B.3.2 Hindsight Action Correction Prediction



(a) Reach3 (b) Reach3 Negations (c) Push3 (d) Push3 Negations

(e) Grasp 3 (f) Grasp3 Negations (g) Lift3 (h) Lift3 Negations

Figure B.16: We depict additional action correction results for the default tasks, employing the **self-attention** as language encoder in our **LCDroQ**, **HEIR**, **HIPSS** and **THIPSS** approaches.

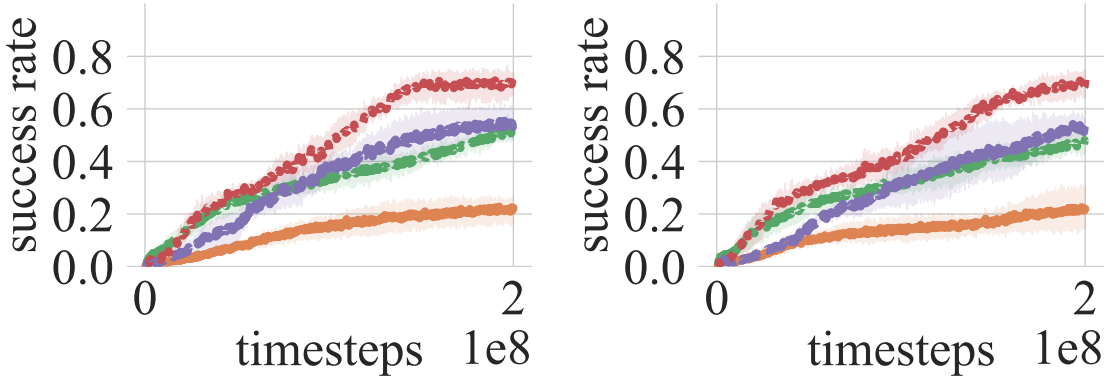


(a) Reach3 Color (b) Reach3 Color Negations (c) Push3 Color (d) Push3 Color Negations

(e) Reach3 Shape (f) Reach3 Shape Negations (g) Push3 Shape (h) Push3 Shape Negations

LCDroQ LCDroQ+HEIR LCDroQ+THIPSS LCDroQ+HIPSS

Figure B.17: We depict additional action correction results for the challenging tasks with the *color* and *shape* options, employing the **self-attention** language encoder in our **LCDroQ**, **HEIR**, **HIPSS** and **THIPSS** approaches.



(a) Multi Easy (b) Multi Easy Negations

LCDroQ LCDroQ+HEIR LCDroQ+THIPSS LCDroQ+HIPSS

Figure B.18: The multitask setting with action corrections, employing the **self-attention** as language encoder in our **LCDroQ**, **HEIR**, **HIPSS** and **THIPSS** approaches.

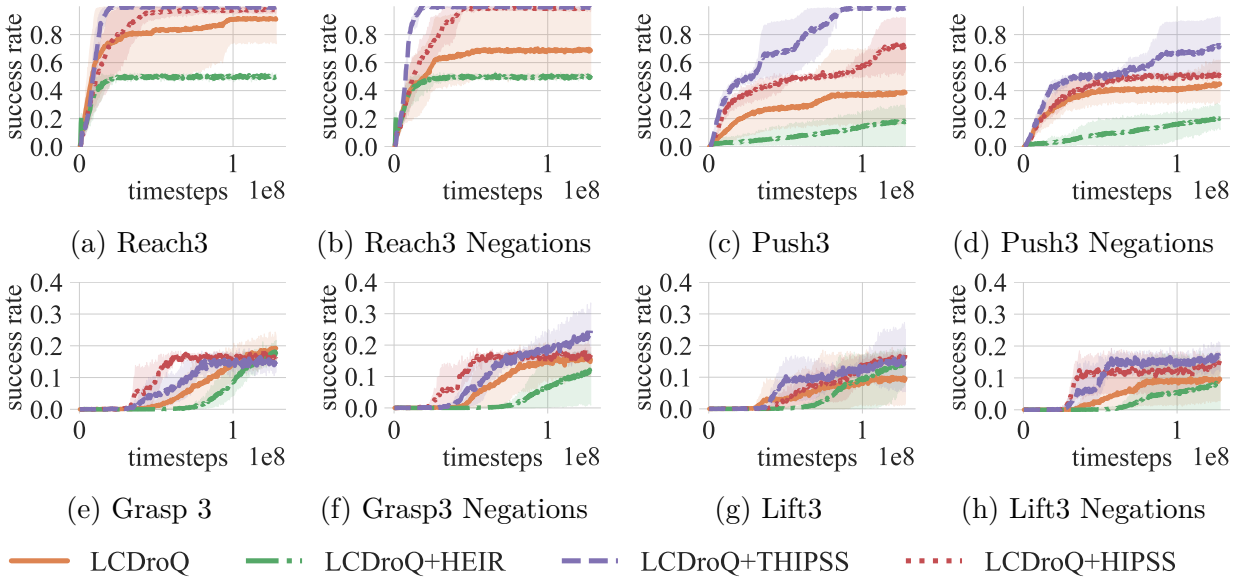


Figure B.19: We depict additional action correction results for the default tasks, employing the **GRU** as language encoder.

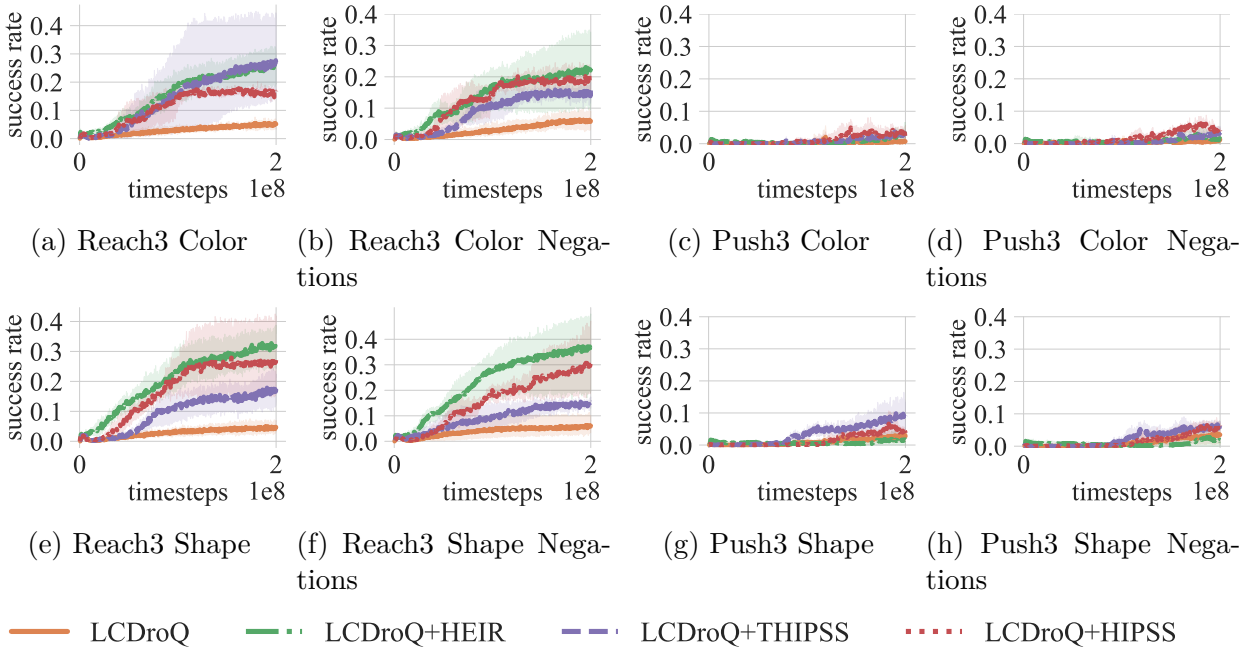
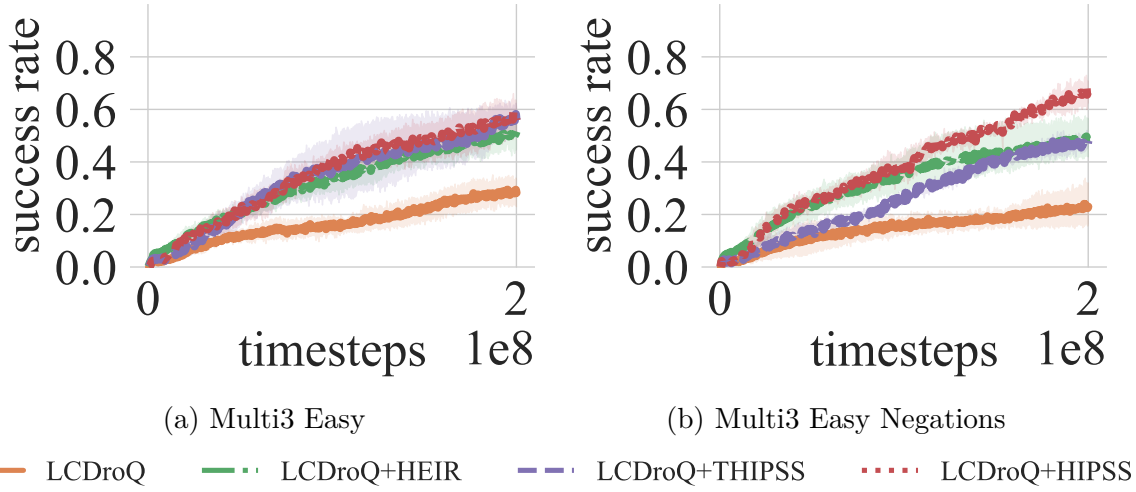


Figure B.20: We depict additional action correction results for the challenging tasks with the *color* and *shape* options, employing the **GRU** language encoder.



(a) Multi3 Easy

(b) Multi3 Easy Negations

Figure B.21: The multitask setting with action corrections, employing the **GRU** as language encoder.

General Definitions

accuracy Accuracy represents the proportion of correct predictions, $\frac{\#\text{correct predictions}}{\#\text{total predictions}}$. 129, 204

actor A different word for policy in the context of the actor-critic approach. 204

actor-critic A reinforcement learning approach to combine policy gradient methods with value learning. 204

ambiguity The characteristic of word or statement to having multiple interpretations or meanings. 10, 204

critic A different word for the Q-function in the context of the actor-critic approach. 204

episode A single instance of a time-limited MDP, that at most allows T interactions but could terminate earlier if, e.g., a specific condition is reached. 47, 204

goal A goal defines a desired state configuration by the environment and can itself be a state, an image or a language instruction. 52, 204

goal-conditioned policy A function that maps sensory inputs and a goal to a distribution of actions. 53, 204

lapsus linguae Latin phrasing of a slip of the tongue; an error in speech, where the speaker says something unintentional. 28, 152, 204

Markov Decision Process A Markov Decision Process (MDP) defines a sequential decision-making problem with uncertain outcomes. 204

polysemy The property of a symbol or a word to have several meanings. 10, 146, 204

Q-function The action-value function $Q^\pi(s_t, a_t)$ is the expected return of being in state s_t, selecting an action a_t and following the policy π thereafter. 49, 204

slot-filling Slot-filling is the task of identifying and extracting specific pieces of information (slots) from a user's utterance, to fulfill a task or query. 148, 204

success rate The success rate defines the fraction of episodes where the agent is able to successfully achieve the goal state, hence $\frac{\#\text{successful episodes}}{\#\text{total episodes}}$. 55, 204

underspecification The property of an utterance that is incomplete, ambiguous, or lacks the relevant detail to determine its meaning. 146, 204

Value function The value function $V^{\pi}(s_t)$ of a policy π regresses the expected return for a state s_t when following π thereafter. 48, 204

Word2Vec A technique to obtain vector representations from words. 73, 204

List of Symbols

$G(\tau)$ The expected return for a provided trajectory. 204

G_t The expected return from timestep t onwards. 204

$H(X)$ The entropy of a random variable X, defined as $H(X) = -\sum_{x \in X} p(x) \log p(x)$. 204

$I(X;Y)$ The mutual information of two random variables X and Y. 204

Q^π The action-value function for a policy π. 204

$Q_{\bar{\phi}}$ The target critic or Q-function parameterized by $\bar{\phi}$. 204

Q_ϕ The critic or Q-function parameterized by ϕ. 204

T Time horizon of an episode. 204

U Uniform distribution. 87, 204

V^π The state-value function for a policy π. 204

$\mathcal{A}$ The action space as set of all actions. 45, 204

$\mathbb{R}$ Real numbers. 204

$\mathcal{R}$ The reward function returning scalar values. 45, 204

$\mathcal{S}$ The state space as set of all states. 45, 204

$\mathcal{T}$ The transition function to calculate the probability of the next state $p(s' \mid s, a)$ given the current state and action. 45, 204

γ The discount factor $\gamma \in [0, 1)$ specifies the farsightedness of the agent. 46, 204

$\mathcal{N}$ Gaussian distribution. 204

π A function that maps sensory inputs to a distribution of actions, $\pi(\cdot \mid s)$. 44, 204

ρ_0 The initial state distribution. 45, 204

Φ State to goal mapping function. 204

τ_s The state trajectory as sequence of environment states only, $\tau_s = (s_0, s_1, \ldots)$. 106, 118, 204

τ A trajectory is a sequence of environment states and actions, $\tau = (s_0, a_0, s_1, a_1, \ldots)$. 47, 204

a An action output by the policy. 45, 204

g'_ℓ The hindsight language goal. 204

g_ℓ A language goal. 204

g A state-based goal. 204

$p(X = x \mid Y = y)$ Conditional probability of X taking the value x when Y takes on the value y. 204

$p(X = x)$ The probability of a random variable X taking the value x. 204

s' The next state of a transition. 45, 204

s The current state of the agent. 45, 204

<eos> The end-of-sequence symbol commonly used in NLP to indicate the end of a sentence, terminating language generation. 163, 204

<pad> The placeholder symbol commonly used in NLP. 204

<sos> The start-of-sequence symbol commonly used in NLP to indicate the start of a sentence for language generation. 163, 204

List of Acronyms

AI artificial intelligence. 1, 204

ANN artificial neural network. 35, 204

BERT Bidirectional Encoder Representations from Transformers. 74, 104, 121, 204

BNF Backus-Naur Form. 90, 91, 204, 242

CBOW Continuous Bag of Words. 73, 204

DL deep learning. 34, 204

DNN deep neural network. 56, 204

DOF degrees of freedom. 86, 204

DroQ Dropout Q-functions. 161, 183, 204

FIFO first in - first out. 59, 204

GloVe Global Vectors for Word Representation. 74, 204

GPT Generative Pre-trained Transformer. 4, 121, 204

GRU Gated-Recurrent Unit. 79, 119, 204

HEIR Hindsight Expert Instruction Replay. 23, 27, 110, 144, 182, 204

HER Hindsight Experience Replay. 23, 103, 106, 204

HIGhER Hindsight Generation for Experience Replay. 25, 104, 116, 204

HIPSS Hindsight Instruction Prediction from State Sequences. 25, 27, 116, 117, 131, 144, 182, 204, 254

HRI human-robot interaction. 63, 204

LANRO LANguage RObotics. 86, 181, 204

LCDroQ Language-Conditioned DroQ. 21, 27, 29, 92, 137, 162, 167, 181, 183, 204

LCSAC Language-Conditioned Soft Actor-Critic. 19, 21, 27, 82, 92, 181, 204

LLM large language model. 1, 204

LSTM Long-Short Term Memory. 79, 204

MDP Markov decision process. 45, 204

MHA multi-head attention. 123, 127, 204, 245

ML machine learning. 34, 204

NLP natural language processing. 19, 34, 64, 204

OOD out-of-distribution. 102, 141, 204

PCA Principal Component Analysis. 71, 204

RL reinforcement learning. 5, 34, 44, 204

RNN recurrent neural network. 77, 117, 204

RPY roll pitch yaw. 86, 204

SAC Soft Actor-Critic. 19, 58, 181, 204

seq2seq Sequence-to-sequence. 117, 204

SGD stochastic gradient descent. 40, 204

TD temporal difference. 50, 204

THIPSS Transformer-based Hindsight Instruction Prediction from State Sequences. 25, 27, 126, 144, 182, 204

UTD update-to-data. 161, 204

Bibliography

This bibliography contains 218 references.

Achiam, Joshua (2018). *Spinning up in Deep Reinforcement Learning*. (Visited on 06/26/2024).

Achimova, Asya, Gregory Scontras, Christian Stegemann-Philipps, Johannes Lohmann, and Martin V. Butz (2022). "Learning about Others: Modeling Social Inference through Ambiguity Resolution". In: *Cognition* 218, p. 104862. ISSN: 0010-0277. DOI: 10.1016/j.cognition.2021.104862.

Ahn, Michael, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, and Mengyuan Yan (2023). "Do As I Can, Not As I Say: Grounding Language in Robotic Affordances". In: *Conference on Robot Learning*. Proceedings of The 6th Conference on Robot Learning. Vol. 205. Proceedings of Machine Learning Research. Atlanta, Georgia, USA: PMLR, pp. 287–318.

Akakzia, Ahmed, Cédric Colas, Pierre-Yves Oudeyer, Mohamed Chetouani, and Olivier Sigaud (May 3–7, 2021). "Grounding Language to Autonomously-Acquired Skills via Goal Generation". In: *International Conference on Learning Representations*. 9th International Conference on Learning Representations. Vienna, Austria.

Amari, Shunichi (1967). "A Theory of Adaptive Pattern Classifiers". In: *IEEE Transactions on Electronic Computers* EC-16.3, pp. 299–307. DOI: 10.1109/PGEC.1967.264666.

An, Gaon, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song (2021). "Uncertainty-Based Offline Reinforcement Learning with Diversified Q-ensemble". In: *Advances in Neural Information Processing Systems*. Thirty-Fifth Annual Conference on Neural Information Processing Systems. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., pp. 7436–7447.

Anderson, Michael L. (2003). "Embodied Cognition: A Field Guide". In: *Artificial Intelligence* 149.1, pp. 91–130. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(03)00054-7.

Andreas, Jacob, Dan Klein, and Sergey Levine (Aug. 6–11, 2017). "Modular Multitask Reinforcement Learning with Policy Sketches". In: *International Conference on Machine Learning*. Proceedings of the 34th International Conference on Machine Learning. Ed.

by Doina Precup and Yee Whye Teh. Vol. 70. ICML'17. Sydney, NSW, Australia: PMLR, pp. 166–175. DOI: 10.5555/3305381.3305399.

Andrychowicz, Marcin, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba (Dec. 4–9, 2017). "Hindsight Experience Replay". In: *Advances in Neural Information Processing Systems*. Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett. Vol. 30. Long Beach, CA, USA: Curran Associates, Inc., pp. 5048–5058.

Arumugam, Dilip, Siddharth Karamcheti, Nakul Gopalan, Edward C. Williams, Mina Rhee, Lawson L. S. Wong, and Stefanie Tellex (Feb. 2019). "Grounding Natural Language Instructions to Semantic Goal Representations for Abstraction and Generalization". In: *Autonomous Robots* 43.2, pp. 449–468. ISSN: 0929-5593, 1573-7527. DOI: 10.1007/s10514-018-9792-8.

Ashktorab, Zahra, Mohit Jain, Q. Vera Liao, and Justin D. Weisz (May 2, 2019). "Resilient Chatbots: Repair Strategy Preferences for Conversational Breakdowns". In: *Conference on Human Factors in Computing Systems*. Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. CHI '19. Glasgow, Scotland UK: Association for Computing Machinery, pp. 1–12. ISBN: 978-1-4503-5970-2. DOI: 10.1145/3290605.3300484.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton (July 21, 2016). *Layer Normalization*.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *International Conference on Learning Representations*. The 3rd International Conference on Learning Representations. Ed. by Yoshua Bengio and Yann LeCun. San Diego, CA, USA.

Bai, Chenjia, Lingxiao Wang, Yixin Wang, Zhaoran Wang, Rui Zhao, Chenyao Bai, and Peng Liu (Sept. 2021). "Addressing Hindsight Bias in Multigoal Reinforcement Learning". In: *IEEE Transactions on Cybernetics* 53.1, pp. 392–405. ISSN: 2168-2267, 2168-2275. DOI: 10.1109/TCYB.2021.3107202.

Baillargeon, Renée (2002). "Infants' Physical Knowledge: Of Acquired Expectations and Core Principles." In: *Language, Brain, and Cognitive Development: Essays in Honor of Jacques Mehler*. Cambridge, MA, US: The MIT Press, pp. 341–361. ISBN: 0-262-04197-9.

Barto, Andrew G. and Sridhar Mahadevan (2003). "Recent Advances in Hierarchical Reinforcement Learning". In: *Discrete Event Dynamic Systems: Theory and Applications* 13.1-2, pp. 41–77. ISSN: 0924-6703. DOI: 10.1023/A:1022140919877.

Beattie, Charles, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, and Stig Petersen (Dec. 13, 2016). *DeepMind Lab*.

Bellman, Richard (1957). "A Markovian Decision Process". In: *Journal of Mathematics and Mechanics* 6.5, pp. 679–684. ISSN: 00959057, 19435274.

Benad, Jan, Frank Röder, and Manfred Eppe (2025). "Scilab-RL: A Software Framework for Efficient Reinforcement Learning and Cognitive Modeling Research". In: *SoftwareX* 29, p. 102064. ISSN: 2352-7110. DOI: 10.1016/j.softx.2025.102064.

Bender, Emily M., Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell (2021). "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? ". In: *Conference on Fairness, Accountability, and Transparency*. Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency. FAccT '21. Virtual Event, Canada: Association for Computing Machinery, pp. 610–623. ISBN: 978-1-4503-8309-7. DOI: 10.1145/3442188.3445922.

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Janvin (Mar. 2003). "A Neural Probabilistic Language Model". In: *Journal of Machine Learning Research* 3, pp. 1137–1155. ISSN: 1532-4435. DOI: 10.5555/944919.944966.

Berk, Laura E. (Nov. 1994). "Why Children Talk to Themselves". In: *Scientific American* 271.5, pp. 78–83. ISSN: 0036-8733. DOI: 10.1038/scientificamerican1194-78.

Bischoff, R. and V. Graefe (1999). "Integrating Vision, Touch and Natural Language in the Control of a Situation-Oriented Behavior-Based Humanoid Robot". In: *International Conference on Systems, Man, and Cybernetics*. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics. Vol. 2. Tokyo, Japan: IEEE, pp. 999–1004. DOI: 10.1109/ICSMC.1999.825399.

Bisk, Yonatan, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, Nicolas Pinto, and Joseph Turian (Nov. 16–20, 2020). "Experience Grounds Language". In: *Conference on Empirical Methods in Natural Language Processing*. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. Association for Computational Linguistics, pp. 8718–8735. DOI: 10.18653/v1/2020.emnlp-main.703.

Bisk, Yonatan, Deniz Yuret, and Daniel Marcu (June 2016). "Natural Language Communication with Robots". In: *Conference of the North American Chapter of the Association for Computational Linguistics*. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. San Diego, California: Association for Computational Linguistics, pp. 751–761. DOI: 10.18653/v1/N16-1089.

Bohus, Dan and Alexander I. Rudnicky (2008). "Sorry, I Didn't Catch That!" In: *Recent Trends in Discourse and Dialogue*. Ed. by Laila Dybkjær and Wolfgang Minker. Vol. 39. Springer, Dordrecht, pp. 123–154. ISBN: 978-1-4020-6820-1 978-1-4020-6821-8. DOI: 10.1007/978-1-4020-6821-8_6.

— (July 2009). "The RavenClaw Dialog Management Framework: Architecture and Systems". In: *Computer Speech & Language* 23.3, pp. 332–361. ISSN: 0885-2308. DOI: 10.1016/j.csl.2008.10.001.

Bordes, Antoine, Y. Lan Boureau, and Jason Weston (May 2017). "Learning End-to-End Goal-Oriented Dialog". In: *International Conference on Learning Representations*. Toulon, France.

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini

Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei (2020). "Language Models Are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. 34th Conference on Neural Information Processing Systems. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. NIPS '20. Vancouver, BC, Canada: Curran Associates, Inc., pp. 1877–1901. ISBN: 978-1-71382-954-6.

Cangelosi, Angelo (June 2010). "Grounding Language in Action and Perception: From Cognitive Agents to Humanoid Robots". In: *Physics of Life Reviews* 7.2, pp. 139–151. ISSN: 15710645. DOI: 10.1016/j.plrev.2010.02.001.

Cangelosi, Angelo and Matthew Schlesinger (2015). *Developmental Robotics: From Babies to Robots*. The MIT Press. ISBN: 978-0-262-32529-5. DOI: 10.7551/mitpress/9320.001.0001.

Carta, Thomas, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer (Feb. 6, 2023). "Grounding Large Language Models in Interactive Environments with Online Reinforcement Learning". In: *International Conference on Machine Learning*. Proceedings of the 40th International Conference on Machine Learning. Vol. 202. Proceedings of Machine Learning Research. Honolulu, Hawaii, USA: PMLR, pp. 3676–3713.

Caselles-Dupré, Hugo, Olivier Sigaud, and Mohamed Chetouani (Sept. 26, 2022). "Overcoming Referential Ambiguity in Language-Guided Goal-Conditioned Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Thirty-Sixth Conference on Neural Information Processing Systems. New Orleans, Louisiana, USA.

Chai, Joyce Y., Lanbo She, Rui Fang, Spencer Ottarson, Cody Littley, Changsong Liu, and Kenneth Hanson (Mar. 3, 2014). "Collaborative Effort towards Common Ground in Situated Human-Robot Dialogue". In: *International Conference on Human-Robot Interaction*. Proceedings of the 2014 ACM/IEEE International Conference on Human-Robot Interaction. Bielefeld, Germany: Association for Computing Machinery, pp. 33–40. ISBN: 978-1-4503-2658-2. DOI: 10.1145/2559636.2559677.

Chaplot, Devendra Singh, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov (Feb. 2–7, 2018). "Gated-Attention Architectures for Task-Oriented Language Grounding". In: *Conference on Artificial Intelligence*. Proceedings of the Thirty-Second {AAAI} Conference on Artificial Intelligence. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. Vol. 8. AAAI'18/IAAI'18/EAAI'18. New Orleans, Louisiana, USA: AAAI Press, pp. 2819–2826. ISBN: 978-1-57735-800-8. DOI: 10.5555/3504035.3504379.

Chen, Lili, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch (2021a). "Decision Transformer: Reinforcement Learning via Sequence Modeling". In: *Advances in Neural Information Processing Systems*. Roceedings of the 35th International Conference on Neural Information Processing Systems. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. NIPS '21. Curran Associates, Inc., pp. 15084–15097. ISBN: 978-1-71384-539-3. DOI: 10.5555/3540261.3541417.

Chen, Xinyue, Che Wang, Zijian Zhou, and Keith W. Ross (2021b). "Randomized Ensembled Double Q-Learning: Learning Fast Without a Model". In: *International Conference on Learning Representations*. Vienna, Austria.

Chevalier-Boisvert, Maxime, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio (May 6–9, 2019). "BabyAI: A Platform to Study the Sample Efficiency of Grounded Language Learning". In: *International Conference on Learning Representations*. 7th International Conference on Learning Representations. New Orleans, Louisiana, USA.

Cho, Kyunghyun, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (Oct. 25–29, 2014). "Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation". In: *Conference on Empirical Methods in Natural Language Processing*. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. Doha, Qatar: Association for Computational Linguistics, pp. 1724–1734. DOI: 10.3115/v1/D14-1179.

Christiano, Paul F, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei (2017). "Deep Reinforcement Learning from Human Preferences". In: *Advances in Neural Information Processing Systems*. Proceedings of the 31st International Conference on Neural Information Processing Systems. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. NIPS'17. Long Beach, CA, USA: Curran Associates, Inc., pp. 4302–4310. ISBN: 978-1-5108-6096-4. DOI: 10.5555/3294996.3295184.

Cideron, Geoffrey, Mathieu Seurin, Florian Strub, and Olivier Pietquin (Dec. 1–4, 2020). "HIGhER: Improving Instruction Following with Hindsight Generation for Experience Replay". In: *Symposium Series on Computational Intelligence*. 2020 IEEE Symposium Series on Computational Intelligence (SSCI). Canberra, ACT, Australia: IEEE, pp. 225–232. ISBN: 978-1-72812-548-0. DOI: 10.1109/SSCI47803.2020.9308603.

Clark, Herbert H. and Susan E. Brennan (1991). "Grounding in Communication." In: *Perspectives on Socially Shared Cognition*. Ed. by Lauren B. Resnick, John M. Levine, and Stephanie D. Teasley. Washington: American Psychological Association, pp. 127–149. ISBN: 978-1-55798-121-9. DOI: 10.1037/10096-006.

Colas, Cédric, Pierre Fournier, Mohamed Chetouani, Olivier Sigaud, and Pierre-Yves Oudeyer (June 9–15, 2019). "CURIOUS: Intrinsically Motivated Modular Multi-Goal Reinforcement Learning". In: *International Conference on Machine Learning*. Proceedings of the 36th International Conference on Machine Learning. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, CA, USA: PMLR, pp. 1331–1340.

Colas, Cédric, Tristan Karch, Nicolas Lair, Jean-Michel Dussoux, Clément Moulin-Frier, Peter Ford Dominey, and Pierre-Yves Oudeyer (Oct. 21, 2020). "Language as a Cognitive Tool to Imagine Goals in Curiosity-Driven Exploration". In: *Advances in Neural Information Processing Systems*. 34th Conference on Neural Information Processing Systems. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. Vol. 33. NIPS '20. Vancouver, BC, Canada: Curran Associates, Inc., pp. 3761–3774. ISBN: 978-1-71382-954-6.

Colas, Cédric, Tristan Karch, Clément Moulin-Frier, and Pierre-Yves Oudeyer (June 2, 2022). "Vygotskian Autotelic Artificial Intelligence: Language and Culture Internalization for Human-Like AI". In: *Nature Machine Intelligence* 4.12, pp. 1068–1076. DOI: 10.1038/s42256-022-00591-4.

Confalonieri, Roberto, Marco Schorlemmer, Oliver Kutz, Rafael Penaloza, and Manfred Eppe (Apr. 2016). "Conceptual Blending in EL++". In: *International Workshop on Description Logics*. Proceedings of the 29th International Workshop on Description Logics. Ed. by Maurizio Lenzerni and Rafael Peñaloza. Cape Town, South Africa: RWTH Aachen, p. 13.

Coumans, Erwin and Yunfei Bai (2016–2021). *Pybullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning*.

Cruz, F., J. Twiefel, S. Magg, C. Weber, and S. Wermter (2015). "Interactive Reinforcement Learning through Speech Guidance in a Domestic Scenario". In: *International Joint Conference on Neural Networks*. Killarney, Ireland: IEEE, pp. 1–8. DOI: 10.1109/IJCNN.2015.7280477.

Cui, Yuchen, Siddharth Karamcheti, Raj Palleti, Nidhya Shivakumar, Percy Liang, and Dorsa Sadigh (Jan. 6, 2023). ""No, to the Right" – Online Language Corrections for Robotic Manipulation via Shared Autonomy". In: *International Conference on Human-Robot Interaction*. Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction. HRI '23. Stockholm, Sweden, pp. 93–101. ISBN: 978-1-4503-9964-7. DOI: 10.1145/3568162.3578623.

Cybenko, G. (Dec. 1989). "Approximation by Superpositions of a Sigmoidal Function". In: *Mathematics of Control, Signals, and Systems* 2.4, pp. 303–314. ISSN: 0932-4194, 1435-568X. DOI: 10.1007/BF02551274.

De Lazcano, Rodrigo, Kallinteris Andreas, Jun Jet Tai, and Jordan Terry (2023). *Gymnasium Robotics*. Version 1.2.0.

Deisenroth, Marc Peter, A. Aldo Faisal, and Cheng Soon Ong (2020). *Mathematics for Machine Learning*. Cambridge, New York, NY: Cambridge University Press. ISBN: 978-1-108-45514-5.

Der, Ralf and Georg Martius (2011). "The Sensorimotor Loop". In: *The Playful Machine*. Red. by Rüdiger Dillmann, David Vernon, Yoshihiko Nakamura, and Stefan Schaal. Vol. 15. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 23–58. ISBN: 978-3-642-20252-0 978-3-642-20253-7. DOI: 10.1007/978-3-642-20253-7_3.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (May 24, 2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Conference of the North American Chapter of the Association for Computational Linguistics*. Proceedings of the 2019 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies. Vol. 1. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

Dodge, Jesse, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston (2016). "Evaluating Prerequisite Qualities for Learning End-to-End Dialog Systems". In: *International Conference on Learning*

Representations. Th International Conference on Learning Representations. San Juan, Puerto Rico.

Dong, Li and Mirella Lapata (Aug. 2016). "Language to Logical Form with Neural Attention". In: *Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Ed. by Katrin Erk and Noah A. Smith. Berlin, Germany: Association for Computational Linguistics, pp. 33–43. DOI: 10.18653/v1/P16-1004.

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby (Oct. 22, 2020). "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *The Eighth International Conference on Learning Representations*. International Conference on Learning Representations. Vienna, Austria.

Eppe, Manfred, Tayfun Alpay, Fares Abawi, and Stefan Wermter (Apr. 2018). "An Analysis of Subtask-Dependency in Robot Command Interpretation with Dilated Cnns". In: *European Symposium on Artificial Neural Networks*. Proceedings of the 26th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2018). Bruges, Belgium, pp. 25–30. ISBN: 978-2-87587-047-6.

Eppe, Manfred, Christian Gumbsch, Matthias Kerzel, Phuong D. H. Nguyen, Martin V. Butz, and Stefan Wermter (Jan. 2022). "Intelligent Problem-Solving as Integrated Hierarchical Reinforcement Learning". In: *Nature Machine Intelligence* 4.1, pp. 11–20. ISSN: 2522-5839. DOI: 10.1038/s42256-021-00433-9.

Eppe, Manfred, Matthias Kerzel, Sascha Griffiths, Hwei Geok Ng, and Stefan Wermter (Nov. 15–17, 2017). "Combining Deep Learning for Visuomotor Coordination with Object Identification to Realize a High-Level Interface for Robot Object-Picking". In: *International Conference on Humanoid Robotics*. 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids). Birmingham, UK: IEEE Press, pp. 612–617. ISBN: 978-1-5386-4678-6. DOI: 10.1109/HUMANOIDS.2017.8246935.

Eppe, Manfred, Phuong D. H. Nguyen, and Stefan Wermter (2019). "From Semantics to Execution: Integrating Action Planning with Reinforcement Learning for Robotic Causal Problem-Solving". In: *Frontiers in Robotics and AI* 6.123. Ed. by Georg Martius. ISSN: 2296-9144. DOI: 10.3389/frobt.2019.00123.

Eppe, Manfred, Sean Trott, and Jerome Feldman (Apr. 2016). "Exploiting Deep Semantics and Compositionality of Natural Language for Human-Robot-Interaction". In: *International Conference on Intelligent Robots and Systems*. 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Daejeon, South Korea: IEEE, pp. 731–738. ISBN: 2153-0866. DOI: 10.1109/IROS.2016.7759133.

Eppe, Manfred, Stefan Wermter, Verena V. Hafner, and Yukie Nagai (Mar. 2021). "Developmental Robotics and Its Role Towards Artificial General Intelligence". In: *KI - Künstliche Intelligenz* 35.1, pp. 5–7. ISSN: 0933-1875, 1610-1987. DOI: 10.1007/s13218-021-00706-w.

Eysenbach, Benjamin, Abhishek Gupta, Julian Ibarz, and Sergey Levine (2019). "Diversity Is All You Need: Learning Skills without a Reward Function". In: *International Confer-*

ence on Learning Representations. The Seventh International Conference on Learning Representations. New Orleans, Louisiana, USA.

Eysenbach, Benjamin and Sergey Levine (Apr. 25, 2022). "Maximum Entropy RL (Provably) Solves Some Robust RL Problems". In: *International Conference on Learning Representations*. Vienna, Austria.

Fang, Meng, Cheng Zhou, Bei Shi, Boqing Gong, Weitao Xi, Tianzhou Wang, Jia Xu, and Tong Zhang (2019). "DHER: Hindsight Experience Replay for Dynamic Goals". In: *International Conference on Learning Representations*. New Orleans, Louisiana, USA.

Faulkner, T. K., Elaine Schaertl Short, and A. Thomaz (2020). "Interactive Reinforcement Learning with Inaccurate Feedback". In: *International Conference on Robotics and Automation*. Paris, France: IEEE, pp. 7498–7504. DOI: 10.1109/ICRA40945.2020.9197219.

Fawzi, Alhussein, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli (Oct. 1, 2022). "Discovering Faster Matrix Multiplication Algorithms with Reinforcement Learning". In: *Nature* 610.7930, pp. 47–53. ISSN: 1476-4687. DOI: 10.1038/s41586-022-05172-4.

Feldman, Jerome (Dec. 2010). "Embodied Language, Best-Fit Analysis, and Formal Compositionality". In: *Physics of Life Reviews* 7.4, pp. 385–410. ISSN: 1571-0645. DOI: 10.1016/j.plrev.2010.06.006.

Feldman, Jerome and Srinivas Narayanan (May 2004). "Embodied Meaning in a Neural Theory of Language". In: *Brain and Language* 89.2, pp. 385–392. ISSN: 0093-934X. DOI: 10.1016/S0093-934X(03)00355-9.

Ferreira, Emmanuel and Fabrice Lefèvre (2015). "Reinforcement-Learning Based Dialogue System for Human–Robot Interactions with Socially-Inspired Rewards". In: *Computer Speech & Language* 34.1, pp. 256–274. ISSN: 0885-2308. DOI: 10.1016/j.csl.2015.03.007.

Fischer, Martin H. and Rolf A. Zwaan (June 2008). "Embodied Language: A Review of the Role of the Motor System in Language Comprehension". In: *Quarterly Journal of Experimental Psychology* 61.6, pp. 825–850. ISSN: 1747-0218, 1747-0226. DOI: 10.1080/17470210701623605.

Fryen, Thilo, Manfred Eppe, Phuong D. H. Nguyen, Timo Gerkmann, and Stefan Wermter (Nov. 11, 2020). *Reinforcement Learning with Time-dependent Goals for Robotic Musicians*. DOI: 10.48550/arXiv.2011.05715.

Fujimoto, Scott, Herke van Hoof, and David Meger (2018). "Addressing Function Approximation Error in Actor-Critic Methods". In: *International Conference on Machine Learning*. Proceedings of the 35th International Conference on Machine Learning. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, pp. 1587–1596.

Gal, Yarin and Zoubin Ghahramani (2016). "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *International Conference on Machine Learning*. Proceedings of The 33rd International Conference on Machine Learn-

ing. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, pp. 1050–1059.

Gallese, Vittorio (Sept. 2008). "Mirror Neurons and the Social Nature of Language: The Neural Exploitation Hypothesis". In: *Social Neuroscience* 3.3-4, pp. 317–333. ISSN: 1747-0919, 1747-0927. DOI: 10.1080/17470910701563608.

Gallouédec, Quentin, Nicolas Cazin, Emmanuel Dellandréa, and Liming Chen (2021). "Panda-Gym: Open-Source Goal-Conditioned Environments for Robotic Learning". In: *Advances in Neural Information Processing Systems*. 4th Robot Learning Workshop: Self-Supervised and Lifelong Learning.

Gardner, Matt, Pradeep Dasigi, Srinivasan Iyer, Alane Suhr, and Luke Zettlemoyer (July 2018). "Neural Semantic Parsing". In: *Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts. Ed. by Yoav Artzi and Jacob Eisenstein. Melbourne, Australia: Association for Computational Linguistics, pp. 17–18. DOI: 10.18653/v1/P18-5006.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. MIT Press. 800 pp. ISBN: 978-0-262-03561-3.

Gurnee, Wes and Max Tegmark (Mar. 4, 2024). "Language Models Represent Space and Time". In: *Conference on Learning Representations*. The Twelfth International Conference on Learning Representations. Vienna, Austria.

Ha, David and Jürgen Schmidhuber (2018). "Recurrent World Models Facilitate Policy Evolution". In: *Advances in Neural Information Processing Systems*. Proceedings of the 32nd International Conference on Neural Information Processing Systems. Ed. by S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, and R Garnett. Vol. 31. NIPS'18. Montréal, Quebec, Canada: Curran Associates, Inc., pp. 2450–2462.

Haarnoja, Tuomas, Haoran Tang, Pieter Abbeel, and Sergey Levine (Aug. 6–11, 2017). "Reinforcement Learning with Deep Energy-Based Policies". In: *International Conference on Machine Learning*. Proceedings of the 34th International Conference on Machine Learning. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. Sydney, NSW, Australia: PMLR, pp. 1352–1361. DOI: 10.5555/3305381.3305521.

Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, and Sergey Levine (July 10–15, 2018). "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *International Conference on Machine Learning*. Proceedings of the 35th International Conference on Machine Learning. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholm, Sweden: PMLR, pp. 1861–1870.

Haarnoja, Tuomas, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine (Jan. 29, 2019). *Soft Actor-Critic Algorithms and Applications*.

Haddadin, Sami, Sven Parusel, Lars Johannsmeier, Saskia Golz, Simon Gabl, Florian Walch, Mohamadreza Sabaghian, Christoph Jahne, Lukas Hausperger, and Simon Haddadin (June 2022). "The Franka Emika Robot: A Reference Platform for Robotics Re-

search and Education". In: *IEEE Robotics & Automation Magazine* 29.2, pp. 46–64. ISSN: 1070-9932, 1558-223X. DOI: 10.1109/MRA.2021.3138382.

Hafner, Danijar, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba (2021). "Mastering Atari with Discrete World Models". In: *International Conference on Learning Representations*. Vienna, Austria.

Hanjie, Austin W., Victor Y Zhong, and Karthik Narasimhan (July 18–24, 2021). "Grounding Language to Entities and Dynamics for Generalization in Reinforcement Learning". In: *International Conference on Machine Learning*. Proceedings of the 38th International Conference on Machine Learning. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 4051–4062.

Harnad, Stevan (June 1990). "The Symbol Grounding Problem". In: *Physica D: Nonlinear Phenomena* 42.1, pp. 335–346. ISSN: 0167-2789. DOI: 10.1016/0167-2789(90)90087-6.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep Residual Learning for Image Recognition". In: *Conference on Computer Vision and Pattern Recognition*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA: IEEE, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

Heinrich, Stefan, Yuan Yao, Tobias Hinz, Zhiyuan Liu, Thomas Hummel, Matthias Kerzel, Cornelius Weber, and Stefan Wermter (Oct. 14, 2020). "Crossmodal Language Grounding in an Embodied Neurocognitive Model". In: *Frontiers in Neurorobotics* 14, p. 52. ISSN: 1662-5218. DOI: 10.3389/fnbot.2020.00052.

Hermann, Karl Moritz, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, Marcus Wainwright, Chris Apps, Demis Hassabis, and Phil Blunsom (June 26, 2017). *Grounded Language Learning in a Simulated 3D World*.

Hill, Felix, Stephen Clark, Karl Moritz Hermann, and Phil Blunsom (Oct. 1, 2019). *Understanding Early Word Learning in Situated Artificial Agents*.

Hill, Felix, Andrew Lampinen, Rosalia Schneider, Stephen Clark, Matthew Botvinick, James L. McClelland, and Adam Santoro (2020). "Environmental Drivers of Systematicity and Generalization in a Situated Agent". In: *International Conference on Learning Representations*.

Hill, Felix, Olivier Tieleman, Tamara von Glehn, Nathaniel Wong, Hamza Merzic, and Stephen Clark (May 3–7, 2021). "Grounded Language Learning Fast and Slow". In: *International Conference on Learning Representations*. 9th International Conference on Learning Representations. Vienna, Austria.

Hiraoka, Takuya, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka (2022). "Dropout Q-functions for Doubly Efficient Reinforcement Learning". In: *International Conference on Learning Representations*.

Hirst, Graeme, Susan McRoy, Peter Heeman, Philip Edmonds, and Diane Horton (Dec. 1994). "Repairing Conversational Misunderstandings and Non-Understandings". In: *Speech Communication* 15.3, pp. 213–229. ISSN: 0167-6393. DOI: 10.1016/0167-6393(94)90073-6.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1997.9.8.1735.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (Jan. 1989). "Multilayer Feedforward Networks Are Universal Approximators". In: *Neural Networks* 2.5, pp. 359–366. ISSN: 0893-6080. DOI: 10.1016/0893-6080(89)90020-8.

Huang, Siyuan, Zhengkai Jiang, Hao Dong, Yu Qiao, Peng Gao, and Hongsheng Li (May 24, 2023a). *Instruct2Act: Mapping Multi-modality Instructions to Robotic Actions with Large Language Model*.

Huang, Wenlong, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter (2023b). "Inner Monologue: Embodied Reasoning through Planning with Language Models". In: *Conference on Robot Learning*. Proceedings of The 6th Conference on Robot Learning. Vol. 205. Proceedings of Machine Learning Research. Auckland, New Zealand: PMLR, pp. 1769–1782.

Hupkes, Dieuwke, Verna Dankers, Mathijs Mul, and Elia Bruni (2020). "Compositionality Decomposed: How Do Neural Networks Generalise?" In: *Conference on Artificial Intelligence*. Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, {IJCAI-20}. Ed. by Christian Bessiere. IJCAI'20. Yokohama, Japan: International Joint Conferences on Artificial Intelligence Organization, pp. 5065–5069. ISBN: 978-0-9992411-6-5. DOI: 10.24963/ijcai.2020/708.

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. ICML'15. Lille, France: PMLR, pp. 448–456. DOI: 10.5555/3045118.3045167.

Jaeger, Bernhard and Andreas Geiger (Dec. 13, 2023). *An Invitation to Deep Reinforcement Learning*.

Jaegle, Andrew, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira (June 22, 2021). "Perceiver: General Perception with Iterative Attention". In: *International Conference on Machine Learning*. Proceedings of the 38th International Conference on Machine Learning. Vol. 139. Vienna, Austria: PMLR, pp. 4651–4664.

Jiang, YiDing, Shixiang (Shane) Gu, Kevin P Murphy, and Chelsea Finn (Dec. 8–14, 2019). "Language as an Abstraction for Hierarchical Deep Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. 33rd Conference on Neural Information Processing Systems. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. Vol. 32. Vancouver, Canada: Curran Associates, Inc., pp. 9419–9431.

Jirak, Doreen, Mareike M. Menz, Giovanni Buccino, Anna M. Borghi, and Ferdinand Binkofski (Sept. 2010). "Grasping Language – A Short Story on Embodiment". In: *Consciousness and Cognition* 19.3, pp. 711–720. ISSN: 1053-8100. DOI: 10.1016/j.concog.2010.06.020.

Kaelbling, Leslie Pack (1993). "Learning to Achieve Goals". In: *International Joint Conference on Artificial Intelligence*. Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France: Morgan Kaufmann, pp. 1094–1098. ISBN: 1-55860-300-X.

Kahneman, Daniel (2011). *Thinking, Fast and Slow*. 1st ed. New York: Farrar, Straus and Giroux. 499 pp. ISBN: 978-0-374-27563-1 978-0-374-53355-7 978-0-606-27564-4.

Kempe, Vera and Patricia J Brooks (2016). "Modern Theories of Language". In: *Encyclopedia of Evolutionary Psychological Science*. Ed. by Viviana Weekes-Shackelford, Todd K. Shackelford, and Viviana A. Weekes-Shackelford. Springer, Cham, pp. 5177–5189. ISBN: 978-3-319-16999-6. DOI: 10.1007/978-3-319-16999-6_3321-1.

Kerzel, Matthias, Erik Strahl, Sven Magg, Nicolás Navarro-Guerrero, Stefan Heinrich, and Stefan Wermter (2017). "NICO — Neuro-inspired Companion: A Developmental Humanoid Robot Platform for Multimodal Interaction". In: *International Symposium on Robot and Human Interactive Communication*. 26th IEEE International Symposium on Robot and Human Communication. Lisbon, Portugal: IEEE, pp. 113–120. ISBN: 978-1-5386-3519-3. DOI: 10.1109/ROMAN.2017.8172289.

Kiefer, Markus and Friedemann Pulvermüller (July 2012). "Conceptual Representations in Mind and Brain: Theoretical Developments, Current Evidence and Future Directions". In: *Cortex* 48.7, pp. 805–825. ISSN: 0010-9452. DOI: 10.1016/j.cortex.2011.04.006.

Kingma, Diederik P. and Jimmy Ba (May 2015). "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations*. Ed. by Yoshua Bengio and Yann LeCun. San Diego, CA, USA.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger. Vol. 25. Lake Tahoe, Nevada, USA: Curran Associates, Inc. ISBN: 978-1-62748-003-1.

LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1989). "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4, pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

LeCun, Yann (2022). *A Path Towards Autonomous Machine Intelligence*.

Levelt, W (July 1983). "Monitoring and Self-Repair in Speech". In: *Cognition* 14.1, pp. 41–104. ISSN: 0010-0277. DOI: 10.1016/0010-0277(83)90026-4.

Li, Alexander, Lerrel Pinto, and Pieter Abbeel (2020a). "Generalized Hindsight for Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Proceedings of the 34th International Conference on Neural Information Processing Systems. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. NIPS '20. Vancouver, BC, Canada: Curran Associates, Inc., pp. 7754–7767. ISBN: 978-1-71382-954-6. DOI: 10.5555/3495724.3496374.

Li, Toby Jia-Jun, Jingya Chen, Haijun Xia, Tom M. Mitchell, and Brad A. Myers (2020b). "Multi-Modal Repairs of Conversational Breakdowns in Task-Oriented Dialogs". In: *ACM Symposium on User Interface Software and Technology*. Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology. UIST '20. Virtual Event, USA: Association for Computing Machinery, pp. 1094–1107. ISBN: 978-1-4503-7514-6. DOI: 10.1145/3379337.3415820.

Liang, Jacky, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng (May 24, 2023). "Code as Policies: Language Model Programs for Embodied Control". In: *International Conference on Robotics and Automation*. 2023

IEEE International Conference on Robotics and Automation (ICRA). London, United Kingdom: IEEE, pp. 9493–9500. DOI: 10.1109/ICRA48891.2023.10160591.

Lillicrap, Timothy P., Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra (2016). "Continuous Control with Deep Reinforcement Learning". In: *International Conference on Learning Representations*. 4th International Conference on Learning Representations. Ed. by Yoshua Bengio and Yann Lecun. San Juan, Puerto Rico.

Lipton, Zachary, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng (Feb. 2018). "BBQ-Networks: Efficient Exploration in Deep Reinforcement Learning for Task-Oriented Dialogue Systems". In: *Association for the Advancement of Artificial Intelligence*. Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence. Vol. 32. New Orleans, Louisiana, USA: AAAI Press, pp. 5237–5244. ISBN: 978-1-57735-800-8. DOI: 10.5555/3504035.3504677.

Lu, Keting, Shiqi Zhang, and Xiaoping Chen (July 17, 2019). "Goal-Oriented Dialogue Policy Learning from Failures". In: *Conference on Artificial Intelligence*. Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. AAAI'19/IAAI'19/EAAI'19, pp. 2596–2603. DOI: 10.1609/aaai.v33i01.33012596.

Luketina, Jelena, Nantas Nardelli, Gregory Farquhar, Jakob N. Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel (July 2019). "A Survey of Reinforcement Learning Informed by Natural Language". In: *International Joint Conference on Artificial Intelligence*. Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI). Macao, China: International Joint Conferences on Artificial Intelligence Organization, pp. 6309–6317. ISBN: 978-0-9992411-4-1. DOI: 10.24963/ijcai.2019/880.

Luo, Jerry, Cosmin Paduraru, Octavian Voicu, Yuri Chervonyi, Scott Munns, Jerry Li, Crystal Qian, Praneet Dutta, Jared Quincy Davis, Ningjia Wu, Xingwei Yang, Chu-Ming Chang, Ted Li, Rob Rose, Mingyan Fan, Hootan Nakhost, Tinglin Liu, Brian Kirkman, Frank Altamura, Lee Cline, Patrick Tonker, Joel Gouker, Dave Uden, Warren Buddy Bryan, Jason Law, Deeni Fatiha, Neil Satra, Juliet Rothenberg, Mandeep Waraich, Molly Carlin, Satish Tallapaka, Sims Witherspoon, David Parish, Peter Dolan, Chenyu Zhao, and Daniel J. Mankowitz (Dec. 14, 2022). *Controlling Commercial Cooling Systems Using Reinforcement Learning*.

Lynch, Corey, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet (Oct. 30–Nov. 1, 2020). "Learning Latent Plans from Play". In: *Conference on Robot Learning*. Proceedings of the Conference on Robot Learning. Ed. by Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura. Vol. 100. Proceedings of Machine Learning Research. Osaka, Japan: PMLR, pp. 1113–1132.

Lynch, Corey and Pierre Sermanet (July 12–16, 2021). "Language Conditioned Imitation Learning Over Unstructured Data". In: *Robotics: Science and Systems*. Proceedings of Robotics: Science and Systems. Ed. by Dylan A. Shell, Marc Toussaint, and M. Ani Hsieh. DOI: 10.15607/RSS.2021.XVII.047.

Ma, Yecheng Jason, William Liang, Vaidehi Som, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman (June 1, 2023). "LIV: Language-Image Representations and Rewards for Robotic Control". In: *International Conference on Machine Learning*. Proceedings of the 40th International Conference on Machine Learning. Vol. 202. Proceedings of Machine Learning Research. Honolulu, Hawaii, USA: PMLR, pp. 23301–23320.

Mandler, Jean M. (Nov. 2004). "Thought before Language". In: *Trends in Cognitive Sciences* 8.11, pp. 508–513. ISSN: 1364-6613. DOI: 10.1016/j.tics.2004.09.004.

Mankowitz, Daniel J., Andrea Michi, Anton Zhernov, Marco Gelmi, Marco Selvi, Cosmin Paduraru, Edouard Leurent, Shariq Iqbal, Jean-Baptiste Lespiau, Alex Ahern, Thomas Köppe, Kevin Millikin, Stephen Gaffney, Sophie Elster, Jackson Broshear, Chris Gamble, Kieran Milan, Robert Tung, Minjae Hwang, Taylan Cemgil, Mohammadamin Barekatain, Yujia Li, Amol Mandhane, Thomas Hubert, Julian Schrittwieser, Demis Hassabis, Pushmeet Kohli, Martin Riedmiller, Oriol Vinyals, and David Silver (June 8, 2023). "Faster Sorting Algorithms Discovered Using Deep Reinforcement Learning". In: *Nature* 618.7964, pp. 257–263. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-023-06004-9.

Matuszek, Cynthia, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox (2012). "A Joint Model of Language and Perception for Grounded Attribute Learning". In: *International Conference on Machine Learning*. Proceedings of the 29th International Coference on International Conference on Machine Learning. ICML'12. Edinburgh, Scotland: Omnipress, pp. 1435–1442. ISBN: 978-1-4503-1285-1.

McCallum, Sabrina, Max Taylor-Davies, Stefano Albrecht, and Alessandro Suglia (2023). "Is Feedback All You Need? Leveraging Natural Language Feedback in Goal-Conditioned RL". In: *Advances in Neural Information Processing Systems*. New Orleans, Louisiana, USA.

Mead, George Herbert, Charles W. Morris, and George Herbert Mead (2000). *Mind, Self, and Society: From the Standpoint of a Social Behaviorist*. Vol. 1. Works of George Herbert Mead. Chicago: Univ. of Chicago Press. 401 pp. ISBN: 978-0-226-51668-4.

Mees, Oier, Lukas Hermann, and Wolfram Burgard (Oct. 2022). "What Matters in Language Conditioned Robotic Imitation Learning Over Unstructured Data". In: *IEEE Robotics and Automation Letters* 7.4, pp. 11205–11212. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2022.3196123.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean (Sept. 6, 2013). "Efficient Estimation of Word Representations in Vector Space". In: *International Conference on Learning Representations*. Scottsdale, Arizona, USA.

Mirchandani, Suvir, Siddharth Karamcheti, and Dorsa Sadigh (Mar. 9, 2021). "ELLA: Exploration through Learned Language Abstraction". In: *Conference on Neural Information Processing Systems*. Proceedings of the 35th International Conference on Neural Information Processing Systems. NIPS '21. Sydney, Australia: Curran Associates Inc., pp. 29529–29540. ISBN: 978-1-71384-539-3. DOI: 10.5555/3540261.3542521.

Mirolli, Marco and Domenico Parisi (2011). "Towards a Vygotskyan Cognitive Robotics: The Role of Language as a Cognitive Tool". In: *New Ideas in Psychology* 29.3, pp. 298–311. ISSN: 0732-118X. DOI: 10.1016/j.newideapsych.2009.07.001.

Mischel, Walter and Ebbe B. Ebbesen (Oct. 1970). "Attention in Delay of Gratification." In: *Journal of Personality and Social Psychology* 16.2, pp. 329–337. ISSN: 1939-1315, 0022-3514. DOI: 10.1037/h0029815.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis (2015). "Human-Level Control through Deep Reinforcement Learning". In: *Nature* 518.7540, pp. 529–533. ISSN: 14764687. DOI: 10.1038/nature14236.

Molnar, Christoph (2022). *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2nd ed.

Moro, Lorenzo, Amarildo Likmeta, Enrico Prati, and Marcello Restelli (2022). "Goal-Directed Planning via Hindsight Experience Replay". In: *International Conference on Learning Representations*. Virtual.

Murphy, Kevin P. (2022). *Probabilistic Machine Learning: An Introduction*. Adaptive Computation and Machine Learning Series. The MIT Press. 864 pp. ISBN: 978-0-262-04682-4.

Nair, Ashvin, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel (2018a). "Overcoming Exploration in Reinforcement Learning with Demonstrations". In: *International Conference on Robotics and Automation*. 2018 IEEE International Conference on Robotics and Automation. Brisbane, QLD, Australia: IEEE, pp. 6292–6299. DOI: 10.1109/ICRA.2018.8463162.

Nair, Ashvin V, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine (Dec. 3–8, 2018b). "Visual Reinforcement Learning with Imagined Goals". In: *Advances in Neural Information Processing Systems*. The Thirty-Second Annual Conference on Neural Information Processing Systems. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. Vol. 31. Montréal, Quebec, Canada: Curran Associates, Inc., pp. 9209–9220.

Narasimhan, Karthik, Regina Barzilay, and Tommi Jaakkola (Dec. 19, 2018). "Grounding Language for Transfer in Deep Reinforcement Learning". In: *Journal of Artificial Intelligence Research* 63.1, pp. 849–874. ISSN: 1076-9757. DOI: 10.1613/jair.1.11263.

Narasimhan, Karthik, Tejas Kulkarni, and Regina Barzilay (2015). "Language Understanding for Text-based Games Using Deep Reinforcement Learning". In: *Conference on Empirical Methods in Natural Language Processing*. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal: Association for Computational Linguistics, pp. 1–11. DOI: 10.18653/v1/D15-1001.

Nguyen, Khanh X, Dipendra Misra, Robert Schapire, Miroslav Dudik, and Patrick Shafto (July 18–24, 2021). "Interactive Learning from Activity Description". In: *International Conference on Machine Learning*. Proceedings of the 38th International Conference on Machine Learning. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 8096–8108.

Oh, Junhyuk, Satinder Singh, Honglak Lee, and Pushmeet Kohli (2017). "Zero-Shot Task Generalization with Multi-Task Deep Reinforcement Learning". In: *International Conference on Machine Learning*. Proceedings of the 34th International Conference on

Machine Learning. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. Sydney, Australia: PMLR, pp. 2661–2670.
Olah, Chris, Alexander Mordvintsev, and Ludwig Schubert (Nov. 7, 2017). "Feature Visualization". In: *Distill*. ISSN: 2476-0757. DOI: 10.23915/distill.00007.
Olah, Christopher (2015). *Understanding LSTM Networks*. (Visited on 02/09/2023).
OpenAI (Mar. 27, 2023). *GPT-4 Technical Report*.
OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba (Jan. 2020). "Learning Dexterous In-Hand Manipulation". In: *The International Journal of Robotics Research* 39.1, pp. 3–20. ISSN: 0278-3649. DOI: 10.1177/0278364919887447.
Oudeyer, Pierre-Yves, Frédéric Kaplan, and Verena V. Hafner (Apr. 2007). "Intrinsic Motivation Systems for Autonomous Mental Development". In: *IEEE Transactions on Evolutionary Computation* 11.2, pp. 265–286. ISSN: 1089-778X. DOI: 10.1109/TEVC.2006.890271.
Ouyang, Long, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe (Mar. 4, 2022). "Training Language Models to Follow Instructions with Human Feedback". In: *Advances in Neural Information Processing Systems*. Proceedings of the 36th International Conference on Neural Information Processing Systems. Vol. 35. NIPS '22. New Orleans, Louisiana, USA: Curran Associates, Inc., pp. 27730–27744. ISBN: 978-1-71387-108-8. DOI: 10.5555/3600270.3602281.
Paulus, Markus (Oct. 2014). "How and Why Do Infants Imitate? An Ideomotor Approach to Social and Imitative Learning in Infancy (and Beyond)". In: *Psychonomic Bulletin & Review* 21.5, pp. 1139–1156. ISSN: 1069-9384, 1531-5320. DOI: 10.3758/s13423-014-0598-1.
Peng, Baolin, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong (2017). "Composite Task-Completion Dialogue Policy Learning via Hierarchical Deep Reinforcement Learning". In: *Conference on Empirical Methods in Natural Language Processing*. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. Copenhagen, Denmark: Association for Computational Linguistics, pp. 2231–2240. DOI: 10.18653/v1/D17-1237.
Pennington, Jeffrey, Richard Socher, and Christopher Manning (Oct. 2014). "GloVe: Global Vectors for Word Representation". In: *Conference on Empirical Methods in Natural Language Processing*. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. Doha, Qatar: Association for Computational Linguistics, pp. 1532–1543. DOI: 10.3115/v1/D14-1162.
Piaget, J. (1926). *The Language and Thought of the Child*. The Language and Thought of the Child. Oxford, England: Harcourt, Brace, pp. xxiii, 246. xxiii, 246.

Piantadosi, Steven T. and Felix Hill (Aug. 12, 2022). "Meaning without Reference in Large Language Models". In: *Advances in Neural Information Processing Systems*. New Orleans, Louisiana, USA.

Plappert, Matthias, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba (Mar. 10, 2018). *Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research*.

Plunkett, Kim, Chris Sinha, Martin F. Møller, and Ole Strandsby (Jan. 1992). "Symbol Grounding or the Emergence of Symbols? Vocabulary Growth in Children and a Connectionist Net". In: *Connection Science* 4.3-4, pp. 293–312. ISSN: 0954-0091, 1360-0494. DOI: 10.1080/09540099208946620.

Pulvermüller, Friedemann (July 2005). "Brain Mechanisms Linking Language and Action". In: *Nature Reviews Neuroscience* 6.7, pp. 576–582. ISSN: 1471-003X, 1471-0048. DOI: 10.1038/nrn1706.

Purver, Matthew, Julian Hough, and Christine Howes (Apr. 2018). "Computational Models of Miscommunication Phenomena". In: *Topics in Cognitive Science* 10.2, pp. 425–451. ISSN: 1756-8757, 1756-8765. DOI: 10.1111/tops.12324.

Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever (2018). *Improving Language Understanding by Generative Pre-Training*.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever (2019). *Language Models Are Unsupervised Multitask Learners*.

Ramesh, Aditya, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever (July 2021). "Zero-Shot Text-to-Image Generation". In: *International Conference on Machine Learning*. Proceedings of the 38th International Conference on Machine Learning. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. Vienna, Austria: PMLR, pp. 8821–8831.

Co-Reyes, John D, Abhishek Gupta, Suvansh Sanjeev, Nick Altieri, John DeNero, Pieter Abbeel, and Sergey Levine (2019). "Guiding Policies with Language via Meta-Learning". In: *International Conference on Learning Representations*. New Orleans, Louisiana, USA.

Rizzolatti, Giacomo and Michael A. Arbib (May 1998). "Language within Our Grasp". In: *Trends in Neurosciences* 21.5, pp. 188–194. ISSN: 0166-2236. DOI: 10.1016/S0166-2236(98)01260-0.

Röder, Frank and Manfred Eppe (2022). "Language-Conditioned Reinforcement Learning to Solve Misunderstandings with Action Corrections". In: *Advances in Neural Information Processing Systems*. New Orleans, Louisiana, USA.

Röder, Frank, Manfred Eppe, Phuong D. H. Nguyen, and Stefan Wermter (Oct. 14, 2020). "Curious Hierarchical Actor-Critic Reinforcement Learning". In: *International Conference on Artificial Neural Networks*. International Conference on Artificial Neural Networks. Vol. 12397. Lecture Notes in Computer Science. Bratislava, Slovakia: Springer, Cham, pp. 408–419. ISBN: 978-3-030-61616-8. DOI: 10.1007/978-3-030-61616-8_33.

Röder, Frank, Manfred Eppe, and Stefan Wermter (Sept. 12–15, 2022). "Grounding Hindsight Instructions in Multi-Goal Reinforcement Learning for Robotics". In: *International Conference on Development and Learning*. 2022 IEEE International Conference on De-

velopment and Learning (ICDL). London, United Kingdom: IEEE, pp. 170–177. ISBN: 978-1-66541-310-7. DOI: 10.1109/ICDL53763.2022.9962207.

Röder, Frank, Ozan Özdemir, Phuong D. H. Nguyen, Stefan Wermter, and Manfred Eppe (Aug. 16, 2021). "The Embodied Crossmodal Self Forms Language and Interaction: A Computational Cognitive Review". In: *Frontiers in Psychology* 12, p. 3374. ISSN: 1664-1078. DOI: 10.3389/fpsyg.2021.716671.

Roh, Junha, Chris Paxton, Andrzej Pronobis, Ali Farhadi, and Dieter Fox (Oct. 30–Nov. 1, 2020). "Conditional Driving from Natural Language Instructions". In: *Conference on Robot Learning*. Proceedings of the Conference on Robot Learning. Ed. by Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura. Vol. 100. Proceedings of Machine Learning Research. PMLR, pp. 540–551.

Rosch, Eleanor (1978). "Principles of Categorization". In: *Cognition and Categorization*. London: Erlbaum, pp. 27–48. ISBN: 978-1-03-263327-5.

Rosenblatt, F. (1958). "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain." In: *Psychological Review* 65.6, pp. 386–408. ISSN: 1939-1471, 0033-295X. DOI: 10.1037/h0042519.

Roy, Deb (Aug. 2005). "Grounding Words in Perception and Action: Computational Insights". In: *Trends in Cognitive Sciences* 9.8, pp. 389–396. ISSN: 1364-6613. DOI: 10.1016/j.tics.2005.06.013.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (Oct. 1986). "Learning Representations by Back-Propagating Errors". In: *Nature* 323.6088, pp. 533–536. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/323533a0.

Schaul, Tom, Daniel Horgan, Karol Gregor, and David Silver (July 6–11, 2015). "Universal Value Function Approximators". In: *International Conference on Machine Learning*. Proceedings of the 32nd International Conference on Machine Learning. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, pp. 1312–1320. ISBN: 978-1-5386-6026-3. DOI: 10.1109/ICRA.2019.8794206.

Schmidhuber, Jürgen (Sept. 2010). "Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990–2010)". In: *IEEE Transactions on Autonomous Mental Development* 2.3, pp. 230–247. ISSN: 1943-0604, 1943-0612. DOI: 10.1109/TAMD.2010.2056368.

Schrittwieser, Julian, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver (2020). "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model". In: *Nature* 588.7839, pp. 604–609. ISSN: 1476-4687. DOI: 10.1038/s41586-020-03051-4.

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov (2017). *Proximal Policy Optimization Algorithms*.

Searle, John R. (Sept. 1980). "Minds, Brains, and Programs". In: *Behavioral and Brain Sciences* 3.3, pp. 417–424. ISSN: 0140-525X, 1469-1825. DOI: 10.1017/S0140525X00005756.

Sejnova, Gabriela, Michal Vavrecka, and Karla Stepanova (Apr. 2, 2024). "Bridging Language, Vision and Action: Multimodal VAEs in Robotic Manipulation Tasks". In: *International Conference on Intelligent Robots and Systems*. Abu Dhabi, UAE: IEEE.

Shao, Lin, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg (2020). "Concept2Robot: Learning Manipulation Concepts from Instructions and Human Demon-

strations". In: *Proceedings of Robotics: Science and Systems* 40.12-14, pp. 1419–1434. ISSN: 0278-3649. DOI: 10.1177/02783649211046285.

Sharma, Pratyusha, Balakumar Sundaralingam, Valts Blukis, Chris Paxton, Tucker Hermans, Antonio Torralba, Jacob Andreas, and Dieter Fox (Apr. 11, 2022). "Correcting Robot Plans with Natural Language Feedback". In: *Robotics: Science and Systems*. Proceedings of Robotics: Science and Systems. New York, NY, USA. DOI: 10.15607/RSS.2022.XVIII.065.

Shi, Lucy Xiaoyang, Zheyuan Hu, Tony Z. Zhao, Archit Sharma, Karl Pertsch, Jianlan Luo, Sergey Levine, and Chelsea Finn (Mar. 19, 2024). "Yell At Your Robot: Improving On-the-Fly from Language Corrections". In: *Robotics: Science and Systems*. Proceedings of Robotics: Science and Systems. arXiv.

Shi, Weiyan and Zhou Yu (July 2018). "Sentiment Adaptive End-to-End Dialog Systems". In: *Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vol. 1. Melbourne, Australia: Association for Computational Linguistics, pp. 1509–1519. DOI: 10.18653/v1/P18-1140.

Shridhar, Mohit, Lucas Manuelli, and Dieter Fox (2022). "CLIPort: What and Where Pathways for Robotic Manipulation". In: *Conference on Robot Learning*. Proceedings of the 5th Conference on Robot Learning. Ed. by Aleksandra Faust, David Hsu, and Gerhard Neumann. Vol. 164. Proceedings of Machine Learning Research. London, UK: PMLR, pp. 894–906.

Silva, Andrew, Nina Moorman, William Silva, Zulfiqar Zaidi, Nakul Gopalan, and Matthew Gombolay (2022). "LanCon-Learn: Learning with Language to Enable Generalization in Multi-Task Manipulation". In: *IEEE Robotics and Automation Letters* 7.2, pp. 1635–1642. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2021.3139667.

Sodhani, Shagun, Amy Zhang, and Joelle Pineau (July 18–24, 2021). "Multi-Task Reinforcement Learning with Context-Based Representations". In: *International Conference on Machine Learning*. Proceedings of the 38th International Conference on Machine Learning. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 9767–9779.

Spilsbury, Sam and Alexander Ilin (July 2022). "Compositional Generalization in Grounded Language Learning via Induced Model Sparsity". In: *Conference of the North American Chapter of the Association for Computational Linguistics*. Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Student Research Workshop. Seattle, Washington: Association for Computational Linguistics, pp. 143–155. DOI: 10.18653/v1/2022.naacl-srw.19.

Spranger, Michael, Jakob Suchan, Mehul Bhatt, and Manfred Eppe (2014). "Grounding Dynamic Spatial Relations for Embodied (Robot) Interaction". In: *Trends in Artificial Intelligence*. Ed. by Duc-Nghia Pham and Seong-Bae Park. Vol. 8862. Springer, Cham: Springer International Publishing, pp. 958–971. ISBN: 978-3-319-13559-5 978-3-319-13560-1. DOI: 10.1007/978-3-319-13560-1_83.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting".

In: *Journal of Machine Learning Research* 15.1, pp. 1929–1958. ISSN: 1532-4435. DOI: 10.5555/2627435.2670313.

Steels, Luc (2008). "The Symbol Grounding Problem Has Been Solved. So What's Next?" In: *Symbols and Embodiment: Debates on Meaning and Cognition*. 1st ed. New York, NY, US: Oxford University Press, pp. 223–244. ISBN: 0-19-921727-0 978-0-19-921727-4.

Steels, Luc and Martin Loetzsch (2012). "The Grounded Naming Game". In: *Experiments in Cultural Language Evolution*. Ed. by Luc Steels. Vol. 3. Amsterdam, Netherlands: John Benjamins Publishing Company, pp. 41–59. ISBN: 978-90-272-0456-1 978-90-272-7495-3. DOI: 10.1075/ais.3.04ste.

Stiennon, Nisan, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano (2020). "Learning to Summarize from Human Feedback". In: *Advances in Neural Information Processing Systems*. Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20. Vancouver, BC, Canada: Curran Associates Inc., pp. 3008–3021. ISBN: 978-1-71382-954-6.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). "Sequence to Sequence Learning with Neural Networks". In: *Conference on Neural Information Processing Systems*. Proceedings of the 27th International Conference on Neural Information Processing Systems. Vol. 2. NIPS'14. Montréal, Quebec, Canada: MIT Press, pp. 3104–3112. DOI: 10.5555/2969033.2969173.

Sutton, Richard S and Andrew G Barto (Nov. 2018). *Reinforcement Learning: An Introduction*. 2nd ed. A Bradford Book. MIT press. 552 pp. ISBN: 978-0-262-03924-6.

Sutton, Richard S. (Aug. 1988). "Learning to Predict by the Methods of Temporal Differences". In: *Machine Learning* 3.1, pp. 9–44. DOI: 10.1007/BF00115009.

Tellex, Stefanie, Nakul Gopalan, Hadas Kress-Gazit, and Cynthia Matuszek (Jan. 31, 2020). "Robots That Use Language". In: *Annual Review of Control, Robotics, and Autonomous Systems* 3.1, pp. 25–55. DOI: 10.1146/annurev-control-101119-071628.

Tellex, Stefanie, Thomas Kollar, Steven Dickerson, Matthew R. Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy (2011). "Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation". In: *Conference on Artificial Intelligence*. Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence. AAAI'11. San Francisco, California: AAAI Press, pp. 1507–1514. DOI: 10.5555/2900423.2900661.

Thierauf, Christopher, Ravenna Thielstrom, Bradley Oosterveld, Will Becker, and Matthias Scheutz (Sept. 2023). ""Do This Instead" – Robots That Adequately Respond to Corrected Instructions". In: *J. Hum.-Robot Interact.* 13.3. DOI: 10.1145/3623385.

Thomason, Jesse, Shiqi Zhang, Raymond Mooney, and Peter Stone (2015). "Learning to Interpret Natural Language Commands through Human-Robot Dialog". In: *International Conference on Artificial Intelligence*. Proceedings of the 24th International Conference on Artificial Intelligence. IJCAI'15. Buenos Aires, Argentina: AAAI Press, pp. 1923–1929. ISBN: 978-1-57735-738-4. DOI: 10.5555/2832415.2832516.

Trott, Sean, Manfred Eppe, and Jerome Feldman (2016). "Recognizing Intention from Natural Language: Clarification Dialog and Construction Grammar". In: *Workshop on*

Communicating Intentions in Human-Robot Interaction. International Symposium on Human and Robot Interactive Communication. New York, NY, USA: IEEE, p. 8.

Tsironi, Eleni, Pablo Barros, Cornelius Weber, and Stefan Wermter (Dec. 2017). "An Analysis of Convolutional Long Short-Term Memory Recurrent Neural Networks for Gesture Recognition". In: *Neurocomputing* 268, pp. 76–86. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2016.12.088.

Turing, Alan (1948). *Intelligent Machinery*.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (June 2017). "Attention Is All You Need". In: *Advances in Neural Information Processing Systems*. Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17. Long Beach, CA, USA: Curran Associates, Inc., pp. 6000–6010. ISBN: 978-1-5108-6096-4. DOI: 10.5555/3295222.3295349.

Vinyals, Oriol, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P Agapiou, Max Jaderberg, Alexander S Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina Mckinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, and Chris Apps (2019). "Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning". In: *Nature* 575, pp. 350–354. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1724-z.

Vygotskij, Lev Semenovič (1985). *Thought and Language*. 17. print. Cambridge, Mass: MIT Press. 168 pp. ISBN: 978-0-262-72001-4.

Vygotsky, L. S. (Apr. 1967). "Play and Its Role in the Mental Development of the Child". In: *Soviet Psychology* 5.3, pp. 6–18. ISSN: 0038-5751. DOI: 10.2753/RPO1061-040505036.

Watkins, Christopher J. C. H. and Peter Dayan (1992). "Q-Learning". In: *Machine Learning* 8, pp. 279–292. ISSN: 0885-6125. DOI: 10.1007/bf00992698.

Waxman, Sandra R. and Dana B. Markow (Dec. 1995). "Words as Invitations to Form Categories: Evidence from 12- to 13-Month-Old Infants". In: *Cognitive Psychology* 29.3, pp. 257–302. ISSN: 00100285. DOI: 10.1006/cogp.1995.1016.

Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou (2024). "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". In: *Conference on Neural Information Processing Systems*. Proceedings of the 36th International Conference on Neural Information Processing Systems. NIPS '22. New Orleans, LA, USA: Curran Associates Inc., pp. 24824–24837. ISBN: 978-1-71387-108-8. DOI: 10.5555/3600270.3602070.

Wermter, Stefan, Cornelius Weber, Mark Elshaw, Vittorio Gallese, and Friedemann Pulvermüller (2005). "Grounding Neural Robot Language in Action". In: *Biomimetic Neural Learning for Intelligent Robots*. Ed. by Stefan Wermter, Günther Palm, and Mark Elshaw. Red. by David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y.

Vardi, and Gerhard Weikum. Vol. 3575. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 162–181. ISBN: 978-3-540-31896-5. DOI: 10.1007/11521082_10.

Wermter, Stefan and Volker Weber (1997). "SCREEN: Learning a Flat Syntactic and Semantic Spoken Language Analysis Using Artificial Neural Networks". In: *Journal of Artificial Intelligence Research* 6, pp. 35–85.

Weston, Jason E (2016). "Dialog-Based Language Learning". In: *Advances in Neural Information Processing Systems*. Proceedings of the 30th International Conference on Neural Information Processing System. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. NIPS'16. Barcelona, Spain: Curran Associates, Inc., pp. 829–837. ISBN: 978-1-5108-3881-9. DOI: 10.5555/3157096.3157189.

Xu, Shusheng, Huaijie Wang, and YI WU (2022). "Grounded Reinforcement Learning: Learning to Win the Game under Human Commands". In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. New Orleans, Louisiana, USA: Curran Associates, Inc., pp. 7504–7519.

Zha, Lihan, Yuchen Cui, Li-Heng Lin, Minae Kwon, Montserrat Gonzalez Arenas, Andy Zeng, Fei Xia, and Dorsa Sadigh (2023). "Distilling and Retrieving Generalizable Knowledge for Robot Manipulation via Language Corrections". In: *2024 IEEE International Conference on Robotics and Automation (ICRA) 2nd Workshop on Language and Robot Learning: Language as Grounding*. Conference on Robotics and Automation. Yokohama, Japan: IEEE, pp. 15172–15179. DOI: 10.1109/ICRA57147.2024.10610455.

Ziebart, Brian D., Andrew Maas, J. Andrew Bagnell, and Anind K. Dey (2008). "Maximum Entropy Inverse Reinforcement Learning". In: *Conference On Artificial Intelligence*. No. 1: Twenty-Third AAAI Conference On Artificial Intelligence Volume. Vol. 3. AAAI'08. Chicago, Illinois: AAAI Press, pp. 1433–1438. ISBN: 978-1-57735-368-3.

List of Figures

1.1 An illustration depicting the sentence, *"she sneezed the napkin off the table"*. **Note**: The prompt to generate this image required considerable effort in tweaking the text to make the system DALL-E (Ramesh et al., 2021) – a model designed to generate detailed visual representations of textual descriptions – successfully produce the desired scene. This emphasizes the aforementioned limitations of current systems in understanding language at the level of actual world dynamics. 2

1.2 We illustrate the relevance of other modalities when processing the sentence *"The big brown grizzly roars loudly"*. To the left, a text-only model has no access to the conceptual appearances related to visual and auditory descriptions. It can only make sense of these on a textual basis. To the right, we illustrate mental concepts that get triggered by hearing key words, such as *"big brown"* and *"roars loudly"*. This figure is taken from our previous work on embodied crossmodal language learning (Röder et al., 2021). 7

1.3 Embodied cognition and its connections to other research fields. The grouping of *Decision Making*, *Artificial Intelligence*, and *Action* could be related to the field of robotics and reinforcement learning (Sutton and Barto, 2018), whereas *Perception*, *Cognitive Psychology*, and *Cognitive Science* in this thesis could be assigned to the field of developmental psychology and developmental robotics (Cangelosi and Schlesinger, 2015; Eppe et al., 2021). *Language* and *Social Cognition* (Mead et al., 2000) have special roles in this thesis, as they define the means of goal communication and what the reward actually corresponds to. 9

1.4 The figure illustrates learning approaches, contrasting the phases of skill acquisition (left) and language grounding (right). In the skill learning phase, image a) depicts learning driven by intrinsic motivation, characterized by play and self-generated goal-setting. Image b) showcases learning influenced by external factors, such as caregiver imitation (Paulus, 2014) and supervision. In the language grounding phase, image c) represents the agent learning from external feedback resulting in outcomes based on intrinsic factors. Image d) exemplifies instruction following, wherein the agent aims to follow an instructor's commands to obtain a reward. – Figure adapted from Röder et al. (2021). 15

1.5 The figure illustrates the idea of mental simulation. Harnessing its world model, the agent mentally pictures itself executing the instruction uttered by the caretaker. However, predictions are limited in terms of their accuracy with respect to the number of steps, visualized by the increasing transparency of the blue cube and hand. – Figure adapted from Röder et al. (2021). 16

1.6 A visualization of our **LANRO** environment with the robot facing 3 different colored cubes. 20

1.7 Main outcome of Chapter 3 with 4 hand-selected experiments. The figure compares the success rate (y-axis) to the number of environment steps (x-axis) for different types of language encoders, including the multilayer perceptron (mlp), gated-recurrent unit (gru) (Cho et al., 2014), and the scaled dot-product attention (Vaswani et al., 2017) (see Subsection 3.2.3 for more details). Enlisted are the plots for the *reach*, *push*, *grasp*, and *lift* tasks. Each of them involves three objects and varies in difficulty regarding their object attributes, such as the color, shape, and weight, and the requested behavior itself. We report the mean success rate along with the 90% confidence interval (shaded area), calculated across 3 random seeds. . 22

1.8 We illustrate the procedure of receiving the expert feedback, upon triggering an outcome to which a hindsight language goal such as *"Hand the red apple"* matches. 23

1.9 We depict a selection of experiments showing the advantage of incorporating the **HEIR** approach into the base algorithm **LCDroQ**. As shown in Figure 1.9d, **LCDroQ+HEIR** outperforms **LCDroQ** by a factor of 3. The plots show the mean success rate along with the 90% confidence interval as a shaded region on the y-axis, as well as the episode timesteps along the x-axis. Each line represents the results averaged over 5 random seeds. . . . 24

1.10 This figure illustrates the concept of a sequence-to-sequence model (Sutskever et al., 2014), applied in our setting as a replay mechanism that translates a trajectory into a corresponding language goal, thereby implementing egocentric speech. 25

1.11 Illustrated is a comparison of the former **HEIR** replay mechanism with the novel egocentric speech approaches **HIPSS** and **THIPSS**. For a selection of tasks, along with variations in color and shape, we depict the average success rate for 5 random seeds, with the 90% confidence interval shown as shaded region on the y-axis and the number of environment steps on the x-axis. Comparing Figures 1.11a and 1.11d reveals significant differences with respect to the performance of the expert approach **HEIR**, possibly due to the different *reaching* and *pushing* behaviors involved. 26

1.12 We depict a scene with the ambiguous instruction *"Hand me the apple, please!"*. Upon observing the robotic agent handing the red apple (bottom left), the instructor utters the action correction, *"No, the green one!"*. The agent then resolves the misunderstanding by taking back the red apple and handing the green one. 29

1.13 The left diagram shows the architecture of a single dropout Q-function with alternating dense, dropout, and normalization layers, paired with ReLU activations. The right illustration depicts the clipped double-Q trick used to calculate the target value for temporal difference learning. Redrawn figure based on Hiraoka et al. (2022). 30

1.14 The performance of **LCSAC** and the dropout Q-function extension **LCDroQ** is compared in a small selection of *multitask* and *push* action correction scenarios. **LCDroQ** consistently outperforms **LCSAC**, as shown by its higher success rate, which accounts for both the standard instruction-following and the resolution of misunderstandings. The y-axis represents the average success rate with a 90% confidence interval (shaded region), plotted against the environment steps on the x-axis. The results are averaged across 5 random seeds. **LCDroQ** shows performance improvements of up to a factor of 8 (Figure 1.14c). 30

1.15 We illustrate the branching mechanism of our egocentric speech model, which predicts either the standard instructions from Chapter 4 (green box) or the extended instructions incorporating action corrections from Chapter 5 (purple box). 31

1.16 We illustrate the sequence-to-sequence approach, which now *teacher forces* the initial instruction g_ℓ = *"push the red cube"* upon the generation of the action correction *"no, the blue cube"*, given the state trajectory τ_s. 32

1.17 We present results for the basis **LCDroQ**, the expert feedback method **HEIR**, and the egocentric speech approaches **HIPSS** and **THIPSS** across various action correction scenarios. These scenarios include *reach*, *push*, and *multitask* tasks, as well as tasks involving negations. The average success rate, computed over 5 random seeds, is displayed on the y-axis, with environment steps plotted on the x-axis. At least one of the egocentric speech approaches consistently leads in performance, demonstrating the benefits of learning to predict action corrections in hindsight. By contrast, the hindsight expert feedback approach **HEIR** focuses exclusively on default instructions and neglects corrective feedback replay, as evident in Figures 1.17a and 1.17d, where it fails to exceed a success rate of 0.5. This result aligns with our configured ratio of episodes containing misunderstandings versus those without (cf. Section 5.4). 33

2.1 Image taken from Molnar (2022, Section 10.1) based on Olah et al. (2017), showing the increasing levels of abstraction (left to right) learned by the *Inception* architecture. 36

2.2 This figure illustrates, on the left, a single neuron with two inputs and one output and, on the right, a neuron with two inputs and two outputs. Both examples highlight the computations being applied to the data x with the help of the weights w_{ij} according to Equation 2.3. 37

2.3 Illustration of various activation functions and their gradients: The sigmoid function σ (left) produces vanishing gradients in regions of very low and high input values, limiting effective training of neural networks. The hyperbolic tangent (center) exhibits similar saturation issues but performs better than sigmoid when input values are consistently positive or negative. The rectified linear unit (right), one of the most widely used activation functions in deep learning, behaves linearly for positive inputs while outputting zero for negative values. 38

2.4 An illustration of a multilayer perceptron with two layers, a hidden layer with two hidden nodes and an output layer with one node. The input layer is not considered a layer in the literature. 39

2.5 Visualization of dropout in a multilayer perceptron with 4 input nodes, 4 hidden nodes, and 3 output nodes. The left side of the figure presents the full network, while the right side demonstrates the effect of dropout, with deactivated connections depicted by the thin gray lines. 42

2.6 Illustration of the reinforcement learning interaction-loop between the agent and the environment. The agent receives the current state s_t as input and, based on this, generates an action a_t. This action transitions the agent into the following state s_{t+1}, accompanied by the corresponding reward r_t in return. 45

2.7 Graphical model of the MDP, showing the sequential dependencies between states s_t, actions a_t, and rewards r_t for timesteps $t = 0, 1, 2, 3$. Each state s_t depends on the previous state s_{t-1} and action a_{t-1}, following the transition dynamics, whereas the rewards r_t are determined by the reward function and actions a_t are selected by the policy. An initial state s_0 and reward r_0 are provided by the environment. 46

2.8 Redrawn figure from Sutton and Barto (2018) of a backup diagram for the value function. 49

2.9 Redrawn figure from Sutton and Barto (2018) of a backup diagram for the state-action function. 50

2.10 We illustrate the language-to-state mapping for the instructions *"pick up the red object"*, mapping to the set $\mathcal{S}_{\bar{g}_\ell}$, and *"pick up the red ball"* mapping to $\mathcal{S}_{\bar{g}_\ell}$. 54

2.11 Graphical model of the extended MDP, visualizing the sequential dependencies of s_t, a_t, and r_t with $t = 0, 1, 2, 3$. We highlight the goal dependence that influences the action selection by the policy and the resulting rewards. The goal g remains constant for the whole episode. The initial state s_0 and the reward r_0 are provided by the environment, whereas the goal comes from the initial goal distribution, $g \sim \rho_g$. 56

3.1 We visualize the stages of processing a language instruction: At the bottom (first stage), we have the instruction *"Hello, how are you?"*. In the second stage, the sentence gets split into individual words. Following, words are then mapped to vector representations that serve the neural network as input in the third stage. 69

3.2 We illustrate an example of the mapping of words to their indices corresponding to a particular embedding. The word *"how"* is mapped to index 73, which retrieves its embedding representation highlighted in red. 71

3.3 We depict the first two principal components as the result of passing the one-hot encodings for the 5 words *"red"*, *"blue"*, *"chair"*, *"table"*, and *"cube"* through the dimensionality reduction. The figure illustrates that the usual word similarities of colors and furniture are not captured. The placement of the words is distorted by the PCA; as it is difficult to visualize a 5-dimensional space in 2D. 72

3.4 This figure depicts the first two principal components of the dense word representations, using the pre-trained *GloVe* embeddings. The 5 words *"red"*, *"blue"*, *"chair"*, *"table"*, and *"cube"* are shown after the dimensionality reduction using PCA. A cluster of colors can be located on the right side, and the furniture-related words are close in space. For this plot, we are using word vectors of size 100 with a vocabulary of 400 000 words. 73

3.5 This figure illustrates the first two principal components of the word representations, using the base BERT model with 110 million parameters and word embeddings of size 768. The 4 sentences *"Push the cube"*, *"Push the red cube"*, *"Push the blue cube"*, and *"Push the chair"* are visualized after the dimensionality reduction using PCA. As expected, BERT is not able to encode the color differences appropriately, which could make it challenging for an RL agent to differentiate these small nuances, which might be crucial for obtaining the sparse reward. 75

3.6 Illustration of an MLP processing the flattened sentence as a large vector of the individual word embeddings of size 2. The sentence length is $L = 4$, hence the MLP input requires 8 input nodes. The words exchange information at the level of hidden activations in the subsequent layers, but otherwise lose any type of sequential information. 77

3.7 To the left, we illustrate the idea of an RNN as a single neuron with a loop connection, transmitting the previous hidden state to the next processing step as additional input. On the right, we show an unrolled version of that neuron, propagating the hidden state to subsequent processing steps. . . . 77

3.8 This figure demonstrates the information flow of a recurrent neural network, that processes the instruction *"Push the green cube"* word by word. The intermediate connections depict the unrolled network in time, propagating the previous hidden state until the final state h_4 is returned. 78

3.9 Diagram of a Gated Recurrent Unit (Cho et al., 2014), showing the update gate u_t, reset gate $\dot{r}_t$, and current memory content interaction, going from h_{t-1} to h_t. The activation functions are sigmoid functions visualized by σ and the hyperbolic tangent, shown as tanh (see Figure 2.3). The illustration is based on Olah (2015). 79

3.10 We illustrate the calculation flow of the scaled dot-product attention layer. This example considers only the first word, with $i = 0$. The instruction *"Push the green cube"* is provided, along with word embeddings of size 2. The illustrated calculations of e_{ij}, a_{ij}, and y_i follow those outlined in Equations (3.15) to (3.17), respectively. 82

3.11 Illustration of our language-conditioned soft actor-critic architecture, receiving as input a sequence of words $g_\ell = (w_0, w_1, w_2, w_3)$, the state of the world s, and, in the case of the critic, also the action a. All of these inputs are encoded by a language encoder ℓ_{enc}, a state encoder s_{enc}, and an action encoder a_{enc}. At the top, the actor is illustrated concatenating the hidden representations of both modalities $[h_\ell, h_s]$, based on which it parameterizes a Gaussian distribution $N(\mu, \sigma)$, from which an action a is sampled. The critic architecture additionally encodes the current action a to concatenate the three hidden states $[h_\ell, h_s, h_a]$, based on which it regresses the state-action value q. An ensemble of Q-functions Q_i is employed, of which the minimum is taken as actual output. 85

3.12 To the left, we have a picture of the real Franka Emika Panda that we use in our virtual **LANRO** benchmark, shown to the right. The real robot image is taken from the official homepage at `https://franka.de` (Visited November 20, 2024). 86

3.13 We illustrate the reach behavior (from left to right). The agent is tasked with reaching the *green* object. 87

3.14 The figure demonstrates the push behavior, showing the agent randomly selecting a direction to move the *green* goal object (from left to right). . . 88

3.15 The grasp behavior visualized from left to right. The agent is reaching for the *orange* object and grasping it. 88

3.16 Demonstration of the agent successfully lifting the *green cuboid* while avoiding contact with other objects. 88

3.17 Backus-Naur Form for our instruction set inspired by Chevalier-Boisvert et al. (2019) – adopted from our prior work Röder and Eppe (2022). 91

3.18 Examples from our POMDP setting in **LANRO**, with the egocentric view to the left (a) and static perspective to the right (b). 91

3.19 We present the results of the *reach* task using the language-conditioned Soft Actor-Critic algorithm (**LCSAC**) from our prior publication (Röder et al., 2022). Each graph shows the number of environment steps (x-axis) and the average success rate (y-axis) for the options *Default*, *Color*, *Shape*, and *ColorShape* (see Table 3.1). The shaded region depicts the 90% confidence interval of 5 random seeds. 93

3.20 The figure compares the different types of language encoders that we use in our language-conditioned RL baseline **LCDroQ**, namely MLP, GRU and self-attention (cf. Subsection 3.2.4). Each graph shows the number of environment steps (x-axis) and the average success rate (y-axis) for 3 trials, with the shaded region being the 90% confidence interval. Illustrated are column-wise, from left to right, the *Default*, *Color*, *Shape*, *Weight*, and *Size* options from Table 3.1, where we have the tasks *reach*, *push*, *grasp*, and *lift* in each row of the figure. 96

3.21 Comparison of our different language encoders across various multitask environments (*easy*, *medium*, and *hard*) with different options (*Default*, *Color*, *Shape*, *Weight*, and *Size*) showing the average success rates (y-axis) over environment steps (x-axis) from 3 random seeds with the shaded area depicting the 90% confidence interval. 98

4.1 We illustrate the example of both the externally and internally prescribed language goals for the undesired outcome. In step **1**, the caretaker utters the instruction *"Hand me the green apple, please!"*. However, in step **2** the agent fails to follow this goal and hands over the *red apple* instead. Wondering why there is no reward, the agent is left with two options to learn from this failure. For step **3a**, the caretaker recognizes this mistake and provides a positive learning experience for the agent by retrospectively changing the goal instruction to one that fits the behavior, in this case, treating it as if it were the intended outcome. Without any external caretaker guidance, in step **3b**, the agent generates an appropriate goal instruction by itself that fits the executed behavior, thus providing a positive learning experience via self-speech. 100

4.2 We depict *"a green elephant with a birthday hat"*, generated with DALL-E 3 (Ramesh et al., 2021). The model shows the individual parts related to the prompt, hinting at compositional capabilities. 102

4.3 Example taken from the paper by Cideron et al. (2020), showing a scene in the BabyAI environment (Chevalier-Boisvert et al., 2019). The agent is instructed to *"Pick the red ball"*. However, it ends up with the wrong object, hence receiving the feedback *"purple ball"*. 105

4.4 Figure taken from the OpenAI blog post *Ingredients of robotics* (Andrychowicz et al., 2017), showing the setting of the Fetch robot pushing the puck that ends up in the wrong location (green circle), instead of the proposed goal location (red circle). For hindsight learning, the actual outcome serves as a virtual goal achieved. 107

4.5 We depict an unsuccessful episode of an already useful policy interacting with the world (from left to right). The agent is instructed to push the *"blue cube"*. However, it ends up pushing the *"red cube"* instead, due to a lack of grounding the color property. This scene provides an optimal episode for hindsight language learning, as the experienced episode is a successful one given the instruction *"push the red cube"*, a potential hindsight instruction. 109

4.6 Upon handing the red apple to the caretaker (left box), the agent receives the hindsight goal instruction, *e.g.*, *"Hand the red apple"* as feedback (right box). Similar to the idea of a social partner proposed in Colas et al. (2020) and Akakzia et al. (2021). 111

4.7 This figure illustrates the **future** sampling strategy. We depict the red goal object and the green object that represents the virtual goal. The strategy considers the state s_5 and following future states of the episode $s_{5...9}$, resulting in the region i, from which we draw candidates for hindsight learning. . 112

4.8 The **episode** strategy considers all transitions i that happened before the hindsight feedback was returned, $i \leq t$. The notion is that all the states ($s_{\leq 5}$) contributed in achieving the hindsight goal configuration, hence pushing the green object. 113

4.9 With the **final** strategy, we sample transitions that, at first, only correspond to the states where the hindsight mechanism was triggered (s_5) and the predecessor transitions leading to that state, while at most go k steps back in time. 114

4.10 We depict a general example of translation, employing a seq2seq model according to the encoder-decoder architecture. As input, the German sentence *"Wie geht es dir?"* is processed by the encoder to output the context vector, which serves as an abstract representation of the input. Following this, the decoder processes the context vector to output the English translation *"How are you?"* word by word. 117

4.11 In the case of **HIPSS**, the encoder RNN takes as input the past hidden state h_{j-1} and the current Markov state s_t embedded by an MLP, to generate a new hidden state h_j. When all vectors of the sequence are processed, the last hidden state h_T, where $T = |\tau_s|$ is the length of the state trajectory, is now our context vector z, an abstract representation of the whole input sequence. The decoder RNN takes as input a starting symbol `<pad>` and autoregressively generates the sentence *"push the red cube"* word by word, by considering the previous decoder hidden state h_{i-1} and the last word prediction $\hat{w}_{i-1}$. Word predictions, made by the MLPs mapping the hidden state to vectors according to the vocabulary size, refer to logits of categorical probability distributions, that we normalize using the softmax function. We sample from these distributions when generating a sentence. For the first word *"push"*, we illustrate this as a vertical histogram for a small sample vocabulary for illustrative purposes. 120

4.12 With **HIPSS** using a Transformer (Vaswani et al., 2017) instead of RNNs, the encoder, consisting of encoder blocks, processes the whole trajectory of states $s_0, \ldots, s_T$, to generate a sequence of hidden encoder states $h_0, \ldots, h_T$. The decoder, made up of decoder blocks, takes as input the full sequence of encoder states and initial target `<pad>` to generate the first logit vector that gets mapped to a probability distribution employing the softmax function (as illustrated by the small vertical histogram), from which the prediction *"push"* is derived. Actual word predictions are made by the MLPs that map the hidden representations to vectors according to the vocabulary size $|V|$. While decoding, a cross-attention operation is employed to consider the encoder states and the decoder predictions generated so far, indicated by the dashed lines. 122

4.13 Illustration of a residual connection with two dense layers, ReLU activations functions, and the skip-connection (identity). 123

4.14 The illustration shows the multi-head attention for H heads running in parallel. The dense layers (feed-forward connections) process the query Q, key K, and value V matrices. Those are split among the multiple heads and afterwards combined via concatenation. The last dense layer defines the output with weights W^O. Redrawn figure based on Vaswani et al. (2017). 125

4.15 We illustrate the causal masking used within the Transformer decoder during training. In the decoder, each word (the query) attends to other words (the keys) based on the computed attention scores according to Equation 3.16. By setting the attention scores of future words to $-\infty$ (depicted by the gray cells), we apply causal attention masking that prevents each position from attending to subsequent words. This is necessary because the attention mechanism processes the entire input sequence in parallel without inherent sequential ordering, and we need to prevent information leakage from future words. For instance, when processing the word *"red"*, the model is allowed to attend only to previous words like *"push"*, *"the"*, and *"red"* itself, but not to future words like *"cube"*. 126

4.16 Illustration of the Transformer architecture, with N_{enc} encoder blocks on the left and N_{dec} decoder blocks to the right. At the bottom, we have the input and the output embeddings to which we apply the positional encoding. Afterwards, the query, key, and value matrices are provided as input to the multi-head attention (MHA) layers (cf. Figure 4.14). In case of the decoder (right side), the aforementioned causal masking is applied (not shown) to prevent looking into the future. Arrows skipping the masked MHA are the residual connections from Figure 4.13, that we jointly illustrate with the layer normalization from Section 2.1.3. Following, we have MLPs processing the output of the MHA. The decoder employs another MHA layer that implements the cross-attention, by receiving the query and key inputs from the encoder's output. Following, the decoder outputs a logit vector (dense layer), that gets normalized by the softmax function to obtain the output probabilities. Redrawn figure based on Vaswani et al. (2017). 127

4.17 To the left, we illustrate the agent interacting with the environment. In the top left, the agent is rewarded by touching the green object as instructed by *"Touch the green object"*. On the bottom left, we present a failure, where the agent touches the wrong object, in this case a blue cuboid, for which a penalty is granted. On the right side of the figure, we illustrate the hindsight procedures that we trigger in each case. In the **successful case**, the outcome serves as an example for training **HIPSS**. In the **case of a failure**, at the bottom right, either **HEIR** or **HIPSS** generate an instruction in hindsight like *"Reach the blue object"* for learning – Figure taken from our work Röder et al., 2022. 130

4.18 The figures provide the results for our 3 proposed replay strategies **episode**, **future**, and **final**, that we tested across the different *reach* tasks of varying difficulty (*Default*, *Color*, *Shape*, and *ColorShape*). All the experiments were conducted with 2 objects on the table (denoted by *Reach2*), using the former **LCSAC+HEIR** combination (Röder et al., 2022). The figures show the environment steps (x-axis) and the success rate (y-axis). Experiments were run using 3 random seeds to plot the mean success rate and their 90% confidence interval (shaded area). While the *future* strategy yields the best results in almost all experiments, it significantly suffers from the effect of hindsight bias in Figure 4.18d, where the performance drops after 5 million environment steps. The experiments made use of the dense embeddings and the GRU as language encoder. 133

4.19 Experimental results from our work on grounded hindsight language learning (Röder et al., 2022). We show the performance of the baseline **LCSAC** and the extensions **HEIR** and **HIPSS** in the *reach* task with 2 objects on the table with 4 levels of difficulty (*Default*, *Color*, *Shape*, and *ColorShape*). The performance gain of **HIPSS** increases with the task complexity; hence a higher number of words and object appearances that could potentially be mixed up by the learning policy. We depict the environment steps on the x-axis, with the average success rate and the 90% confidence interval on the y-axis, employing 5 random seeds. 134

4.20 Replay strategy results for the latest version of **HEIR** paired with **LCDroQ**, showing the mean success rate (y-axis) and the environment steps (x-axis). We are comparing the **episode**, **final**, and **future** strategies within the *reach* task with the *ColorShape* option (see Figure 4.20a), the *push* task with the *SizeShape* option (see Figure 4.20c), and the *multitask easy* setting with and without the *Color* option (see Figures 4.20b and 4.20d). Shown are the results for 5 random seeds with 3 objects on the table, utilizing the self-attention as language encoder. The shaded regions are calculated by the 90% confidence interval. 136

4.21 We highlight a selection of tasks, showing **LCDroQ+HEIR** outperforming the baseline **LCDroQ** in many cases. Illustrated are the experiments using self-attention as the language encoder for 5 random seeds. Depicted are the environment steps (x-axis) in relation to the average success rate, with the shaded region representing the 90% confidence interval (y-axis). We provide the full set of experiments as a comparison in Subsection B.2.1. Here, **LCDroQ+HEIR** denotes the combination of our baseline **LCDroQ** (cf. Subsection 5.5.2) and the expert feedback replay method **HEIR** (cf. Section 4.4). As expected, the perfect hindsight instructions by the expert show significant performance improvements (*e.g.*, Figures 4.21d and 4.21l). 138

4.22 This figure depicts the results for our algorithms **LCDroQ**, **LCDroQ+HEIR**, **LCDroQ+HIPSS**, and **LCDroQ+THIPSS**. We compare them with respect to the amount of environment timesteps (x-axis) it takes to reach a certain success rate, for which we visualize the mean and the 90% confidence interval as shaded area on the y-axis. All experiments employ the GRU as language encoder and contain 5 random seeds. Further results are provided in Subsection B.2.2. 140

4.23 We depict an experiment comparing **HIPSS** and **THIPSS** with respect to their systematic generalization (Hupkes et al., 2020). While the training in both cases achieves similar accuracy values for the most part (left figure), **THIPSS** exceeds the performance of **HIPSS** by up to 0.1 accuracy points after the sudden jump at 330 epochs in the test plot to the right. The table at the bottom depicts, as green cells, the word combinations seen during training and, as red cells, the word combinations that were excluded. The models are trained with all instructions containing the verbs *reach*, *touch*, and *contact*, paired with the colors *red* and *green*. In addition, the verbs *push*, *move*, and *shift* are paired with the color *blue*. For testing, the opposite word combinations, shown as red cells, are used. Hence the models have never seen instruction word combinations containing the *reach*-related verbs paired with the color blue or the *push*-related verbs paired with the colors *red* and *green*. 142

4.24 We examine a case with a more evenly distributed setting of out-of-distribution samples. The training accuracy in the left figure shows very similar results for both **HIPSS** and **THIPSS**, while the latter appears to have a more stable learning curve. On the right, we depict the test accuracy, with **HIPSS** exceeding the performance of **THIPSS** after 100 epochs of training. In the table at the bottom, in-distribution samples used for training are illustrated by the green cells, while the samples that we exclude are depicted by the red cells. More precisely, training the models includes the instructions containing the word combinations *reach* and *red*, *touch* and *green*, *contact* and *blue*, *push* and *green*, *move* and *red*, and *shift* and *blue*. For evaluation, we consider all the other combinations, denoted by the red cells in the table. . 143

5.1 An action correction scenario, where a human issues the ambiguous command *"Hand me the apple, please!"* to the agent. However, the operator intends for the robot to pick up the green apple. The robot chooses to reach for the red apple, either by chance or because it is the closer object. When the operator realizes that the actions deviate from the intended outcome, the action correction *"No, the green one!"* is uttered. Then, the robot reevaluates the original instruction given the correction, resolving the underspecification, to successfully hand over the correct apple. 147

5.2 A scene from **LANRO**, with a green cube, a green cuboid, and a red cube. Figure taken from Röder and Eppe, 2022. 151

5.3 The figure describes the general case of an action correction scenario that fits any of the proposed cases of misunderstanding from Section 5.2. The instructor utters a lingual goal g_ℓ that the agent attempts to follow, interacting with the environment depicted by the RL loop (see Figure 2.6). Implemented by the condition function $\mathcal{C}$, the instructor observes the world state s_t for configurations that constitute a misunderstanding. In case of detecting a misunderstanding, an action correction g_{ac} is returned. Figure inspired by our previous work Röder and Eppe (2022). 154

5.4 To the left in Figure 5.4a, we illustrate the process of correcting a misunderstanding that is closely related to the corrected instruction; while to the right (Figure 5.4b), we depict the case where correcting the misunderstanding changes the whole meaning. Visualized are the rewarding states that, on the left, overlap and, on the right, are completely disjoint. An example of the first case would be the instruction g_ℓ = *"pick up the red object"* corresponding to the set S_{g_ℓ}, that gets corrected by g_{ac} = *"actually, the cube"*, where $g'_\ell = g_\ell \circ g_{\text{ac}}$ now maps to a smaller set of rewarding states $S_{g'_\ell}$. The same idea could be applied to Figure 5.4b, where the instruction g_ℓ= *"pick up the red object"* gets corrected by g_{ac} = *"sorry, the blue object"*, resulting in a rewarding state mapping $S_{g'_\ell}$, disjoint from the original set of states S_{g_ℓ}. 155

5.5 This figure illustrates the possible action correction scenarios implemented by **LANRO**. We depict a setting with a red and green cube, and a green cuboid. To the left, we have an initial valid instruction that follows misunderstandings caused by a lack of common ground or an instructor mistake. To the right, the ambiguity is caused by an underspecified statement. . . . 157

5.6 We illustrate the lack of common ground where the agent misunderstands the instruction. Instead of reaching the *"green cube"*, it reaches the *"green cuboid"* instead. As we do not know the cause of the misunderstanding, we provide the action correction *"No, the green cube"* to contrast the agent's current state (step 2) and object visited with the actual instruction. 159

5.7 The extension of our Backus-Naur form grammar from Figure 3.17 for our action corrections, including the negations. The provided definitions are applicable for all language tasks (see Subsection 3.4.1). 159

5.8 The Backus-Naur form of our task corrections used in the multitask settings of **LANRO**. 160

5.9 Example instructions with action corrections used in **LANRO**. 160

5.10 To the left, we depict the structure of an individual dropout target Q-function $Q_{\bar{\phi}_i}$ according to Hiraoka et al. (2022). The network consists of dense layers, dropout layers, layer normalization and ReLU functions, and outputs a Q-value q_i. On the right, we depict the ensemble of size N with the min operation to select the actual Q-value target prediction q used for training (see Section 2.2.7). This figure is based on Hiraoka et al. (2022). . 163

5.11 The figure illustrates the various ways **HIPSS** can complete the instruction *"<sos> push the green cube"*, considering multiple approaches to hindsight action correction. The initial instruction could be directly terminated with the <eos> symbol, followed by a negation starting with *"not"*, or simply edited with phrases like *"sorry"* or *"actually"*, each potentially followed by a different object description or task behavior. 164

5.12 The hindsight action correction instruction is generated by completing the provided initial instruction, *"push the red cube"* (red box), with the action correction prediction, *"no, the blue cube"* (blue box). 166

5.13 This figure compares the performance of our previous **LCSAC** approach with **LCDroQ**. We depict the environment steps taken (x-axis) and success rate (y-axis) of 5 trials, for which we plot the mean with the 90% confidence interval. We show results for self-attention as the language encoder, while further results are provided in Section B.3.1. The upper plots show the success rates for the normal action corrections, while the plots below depict the more challenging setting with negations. The results indicate that negations are generally more difficult to learn. However, **LCDroQ** outperforms the baseline **LCSAC** in all the presented cases. 168

5.14 We provide a comparative analysis of **LCSAC** and **LCDroQ** in the multi-task environment. Shown are the environment steps (x-axis) and the mean success rate for 5 trials including the 90% confidence interval (y-axis). To the left, we illustrate the default action correction, while to the right, we show the correction setup incorporating negations. The presented results only employ self-attention as the language encoder. 169

5.15 We illustrate the critic ensemble uncertainty as an average of 80 action correction episodes across 5 tasks. The episode timesteps are shown on the x-axis, and the mean return estimate with the standard deviation as a shaded region is shown on the y-axis. One can see that around $t = 10$ timesteps, the uncertainty drops drastically when the action correction is returned by the caretaker upon interacting with the wrong object. 171

5.16 We illustrate the position of the agent's gripper conditioned on the initial instruction (blue dots ●) and after the action correction appears (orange dots ●). The red, green, and blue objects are depicted by the × with their corresponding colors. The agent starts at the circled position and approaches the green object, which it manipulates, hence the cloud of green × markers. After receiving the action correction, it changes its trajectory to approach and manipulate the blue object, denoted by the cloud of blue × markers at the end of the episode. 172

5.17 An exemplary illustration of the *reach* task, focusing on the frequency of generating hindsight instructions, with respect to the overall success rate (Figure 5.17a) and the action correction success rate (AC success rate on the y-axis) only (Figure 5.17b). In Figure 5.17c, we depict the hindsight instruction rate (HI rate) caused by wrong object interactions. These interactions are, as expected, high in the beginning and decrease as the agent progresses. The x-axis shows the environment steps, while the y-axis depicts the respective metrics with their 90% confidence interval. 173

5.18 We depict our baseline **LCDroQ**, next to **LCDroQ+HEIR** as the baseline extended by the expert feedback, and **LCDroQ+HIPSS** and **LCDroQ+THIPSS**, where **HIPSS** and **THIPSS** incorporate the egocentric speech approaches from Section 4.7. Action correction challenges for the *reach*, *push*, *grasp*, and *lift* tasks without any form of color or shape extension are shown. The default action correction tasks are shown in the first row, while those with negations are depicted in the second row. We show the average success rate and the 90% confidence interval for 5 random seeds (y-axis) with respect to the number of environment steps taken (x-axis). We employ the GRU as a language encoder, while we provide further results in Figure B.16. 175

5.19 We illustrate experimental results for our approaches **LCDroQ**, **LCDroQ+HEIR**, **LCDroQ+HIPSS**, and **LCDroQ+THIPSS** in the *reach* and *push* action correction tasks involving the *color* and *shape* extensions. Depicted are the average success rates, with the shaded regions indicating the 90% confidence interval (y-axis), and the environment steps taken (x-axis). All trials employ the GRU language encoder and contain 5 random seeds. 176

5.20 We depict the experimental results for our easy *multitask* settings involving misunderstandings without and with negations. The graphs show the average success rate and the 90% confidence interval on the y-axis with the taken environment steps on the x-axis. Each average is the results of 5 random seeds all using the GRU as language encoder, with more results shown in Section B.3. 177

6.1 This figure contrasts hindsight instructions and action corrections, both of which address undesirable outcomes stemming from various underlying reasons (center, see Section 5.2). In the former case, undesirable outcomes guide retrospective learning, whereas in the latter case, an action correction aims to revise the behavior that led to the unwanted outcome resulting from a misunderstanding. 180

B.1 Illustrated are column-wise from left to right, the *Default*, *Color*, *Shape*, *Weight*, and *Size* options from Table 3.1. We showcase the tasks *reach*, *push*, *grasp*, and *lift* in each row of the figure. Provided are the variations with only 2 objects in the scene. 193

B.2 Illustrated are, from left to right, the *ColorShape*, *WeightShape*, and *SizeShape* options from Table 3.1 for the tasks *reach*, *push*, *grasp*, and *lift* in each row of the figure. Scenes with 2 objects are shown. The extended range of property combinations makes these tasks harder to solve. 194
B.3 We illustrate from left to right within each column the *Default*, *Color*, *Shape*, *Weight*, and *Size* options from Table 3.1, while we have the tasks *reach*, *push*, *grasp*, and *lift* in each row of the figure. Provided are the variations with 3 objects in the scene, which are more challenging as there is no 50-50 chance in identifying the correct object, as in the case with 2 objects. 195
B.4 Illustrated are, column-wise, the *ColorShape*, *WeightShape*, and *SizeShape* options from Table 3.1 for the tasks *reach*, *push*, *grasp*, and *lift* in each row of the figure. Shown are the settings with 3 objects in the scene. The extended range of property combinations makes these tasks harder to solve. 196
B.5 We depict column-wise the *Default*, *Color*, *Shape*, *Weight*, and *Size* options from Table 3.1 for the *easy*, *medium*, and *hard multitask* settings in each row of the figure. For all tasks, we include 3 objects. The extended range of property combinations, in combination with the multiple tasks involved, makes these settings the hardest to solve (see Section 3.4.1). 197
B.6 Complementary experiments for **LCDroQ** and **HEIR**, utilizing the **GRU** as language encoder. 198
B.7 Complementary *multitask* experiments for **LCDroQ** and **HEIR**, utilizing the **GRU** as language encoder. 199
B.8 Complementary experiments for **LCDroQ** and **HEIR** utilizing the **self-attention** as language encoder. 200
B.9 Complementary multitask experiments for **LCDroQ** and **HEIR** utilizing the **self-attention** as language encoder. 200
B.10 Complementary experiments for **LCDroQ**, **HEIR**, **HIPSS**, and **THIPSS** utilizing the **GRU** as language encoder. 201
B.11 Complementary multitask experiments for **LCDroQ**, **HEIR**, **HIPSS**, and **THIPSS** utilizing the **GRU** as language encoder. 201
B.12 Complementary experiments for **LCDroQ**, **HEIR**, **HIPSS**, and **THIPSS** utilizing the **self-attention** as language encoder. 202
B.13 Complementary multitask experiments for **LCDroQ**, **HEIR**, **HIPSS**, and **THIPSS** utilizing the **self-attention** as language encoder. 202
B.14 We provide the additional results for the **GRU** as language encoder, comparing **LCSAC** and **LCDroQ**. 203
B.15 We provide additional results for the action correction *multitask* setting with and without negations using the **GRU** as language encoder. We compare the performance of **LCSAC** and **LCDroQ**. 203
B.16 We depict additional action correction results for the default tasks, employing the **self-attention** as language encoder in our **LCDroQ**, **HEIR**, **HIPSS** and **THIPSS** approaches. 204

B.17 We depict additional action correction results for the challenging tasks with the *color* and *shape* options, employing the **self-attention** language encoder in our **LCDroQ**, **HEIR**, **HIPSS** and **THIPSS** approaches. 205
B.18 The multitask setting with action corrections, employing the **self-attention** as language encoder in our **LCDroQ**, **HEIR**, **HIPSS** and **THIPSS** approaches. 205
B.19 We depict additional action correction results for the default tasks, employing the **GRU** as language encoder. 206
B.20 We depict additional action correction results for the challenging tasks with the *color* and *shape* options, employing the **GRU** language encoder. 206
B.21 The multitask setting with action corrections, employing the **GRU** as language encoder. 207

List of Tables

1.1 Overview of the proposed methods and their contributions. 27

3.1 Overview: Single Task Options 89
3.2 Primary and secondary properties for the task combinations. We provide a fixed order to retain natural-sounding instructions. Enlisted are the default sets (in brackets) and the placeholders for the other property sets. 90

5.1 Hyperparameter settings for the critic ensemble in **LCSAC** and **LCDroQ**, as recommended by Hiraoka et al. (2022). 167

A.1 The hyperparameters for our **LCSAC** baseline first introduced in Röder et al. (2022). 189
A.2 The hyperparameters for our novel baseline **LCDroQ** derived from our previous **LCSAC** implementation (Röder et al., 2022) and extended by the **DroQ** algorithm (Hiraoka et al., 2022). 189
A.3 We provide the hyperparameters that both **HIPSS** and **THIPSS** share. . 190
A.4 The hyperparameters for the seq2seq model **HIPSS**. 190
A.5 The hyperparameters for our Transformer-based encoder-decoder **THIPSS**. 191

List of Algorithms

1 Soft Actor-Critic Algorithm (Haarnoja et al., 2018) 61
2 Language-Conditioned Soft Actor-Critic (Röder et al., 2022). Equations from SAC (Subsection 2.2.7) are reused since the goal can be considered part of the state. 84
3 Hindsight Experience Replay (**future** replay strategy) 108
4 Hindsight Instruction Prediction from State Sequences 131
5 **HIPSS** Action Correction Generation 165