University of Amsterdam

MSc Artificial Intelligence
Master Thesis

Addressing Goal Misgeneralization with Natural Language Interfaces

---

by

Giulio Starace

13010840

48 ECTS
September 2023

Supervisor:
Niklas Höpner

Examiner:
Herke van Hoof

External Supervisor:
Juan Rocamonde

Second Reader:
Niklas Höpner

UvA

Informatics Institute (IvI)

Abstract

We present an approach to address the issue of *goal misgeneralization*, an intriguing phenomenon most intuitively linked to sequential decision making (SDM) models. Here, a policy trained to complete a particular goal during training, misgeneralizes in an out-of-distribution test environment and instead capably pursues some other confounding goal. We view goal misgeneralization as a consequence of *causal confusion*, a phenomenon in which machine learning models learn the wrong causal model for some predictive behaviour due to spurious correlations. We posit that the way in which we specify tasks to our SDM agents is a key factor in their proclivity to suffer from causal confusion and goal misgeneralization. Using the framework of multi-task imitation learning, in the context of goal misgeneralization, we study the effects of conditioning on (more expressive) factored task representations derived from natural language, as opposed to simply conditioning on rewards. To this end, we present an implementation for specifying tasks to behavioural cloning agents by conditioning on natural language. Compared to a reward-conditioned baseline, we show that this approach diminishes the extent of goal misgeneralization in a toy environment, but nevertheless still suffers from the phenomenon. We perform some diagnostic experiments for further analysis of our approach, and provide some discussions around current limitations and potential future work.

Acknowledgements

First and foremost, I would like to thank my family for their unwavering support and love. While I may never be able to fully discuss my thesis with you, all that matters to me is the closeness and love you never fail to share. Grazie.

I would like to thank my supervisor, Niklas, for his guidance and for devoting so much attention to this thesis. I am quite sure that if it wasn't for Niklas, I would still be reading papers. Your openness to a self-proposed topic, availability for weekly meetings, your contributions and feedback on my ideas have all proven incredibly valuable. Thank you.

Similarly, I cannot continue without thanking my co-supervisor Juan. Thank you for agreeing to supervise me as part of the SPAR programme, and for remaining available to the same extent well past the end of the program. Our bi-weekly meetings were an excellent source of discussions and idea-bouncing that proved to be very valuable. Thank you.

To Benita, thank you for your patience, your support and your affection. Most of all, thank you for keeping me sane by reminding me to find a balance between work and life, staying healthy and positive. Our calls, the times we spent cooking together, our travels, your letters and your company, ever so fleeting, may not have seemed much, but proved to be essential. Gracias.

To my friends Victor, Karolina, Chase and Luuk: Thank you for keeping me company at the fifth floor flexdesks, for sharing lunch by the third floor microwave or at Cafe Neo[1], for the discussions on grounding and whatever goal misgeneralization is supposed to be and for hyping me up when the results finally came. Thank you for being there. I love you guys.

[1] this message was NOT sponsored by Cafe Neo.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Sequential decision making (SDM) is a key subfield in the area of artificial intelligence (AI) that studies the problems and algorithms associated with autonomously interacting with an environment with the purpose of completing some task through a sequence of actions. With the increased popularity in machine learning (ML), some attention is now being devoted to rendering existing tool-like solutions more autonomous, for instance by allowing large language models (LLMs) to continuously prompt themselves, looping output into input for a number of iterations with the aim of achieving some higher level goal. Several contributions, including AutoGPT (Yang et al., 2023) and others (Bakhtin et al., 2022; Park et al., 2023; Zhou et al., 2023b), have been made in this vein, making SDM more relevant than ever. While already well-established as a field for decades (Bellman, 1957; Bryson, 1996; Klopf, 1972; Minsky, 1954), the recent widespread adoption of deep neural networks (NNs) in supervised and unsupervised learning in vision (He et al., 2016; Krizhevsky et al., 2012; Mildenhall et al., 2021) and language (Brown et al., 2020; Devlin et al., 2019; Mikolov et al., 2013; Vaswani et al., 2017) has led to similar deep learning (DL) approaches to SDM, particularly in the field of reinforcement learning (RL) (Sutton and Barto, 2018), achieving impressive results on a wide range of problems (Fawzi et al., 2022; Mnih et al., 2013; OpenAI et al., 2019; Silver et al., 2018).

These advances are certainly impressive, but their inherently online nature has hindered their integration in contexts where interacting with the environment is either expensive or dangerous, such as consumer robotics (Singh et al., 2022b), healthcare (Liu et al., 2020a) and autonomous vehicles (Kiran et al., 2022). In synchrony with ML's shift to more data-driven approaches, much attention is now being dedicated to *offline* solutions to SDM, such as batch RL (Levine et al., 2020; Prudencio et al., 2022) and imitation learning (Schaal, 1999). Rather than interacting with the environment in real-time, these of-

fline solutions aim to learn control policies using a fixed dataset of previously collected interactions. Making use of large, diverse datasets can lead to improved generalization, and still affords the option to fine-tune the resulting policies online at a later stage, drawing parallels with the "pre-train first, then fine-tune" trend in present-day ML.

1.1 Motivation

ML solutions can be vulnerable to *distributional shift*, where the data distribution changes between training and deployment (Quinonero-Candela et al., 2008). This issue of out-of-distribution (OOD) generalization (Shen et al., 2021) is pervasive to the current paradigm of ML research (Arjovsky, 2020). *Causal confusion* is a phenomenon closely linked to OOD misgeneralization in which the learner wrongly identifies the causal model of the data due to spurious correlations (de Haan et al., 2019). The more classic failure mode of causal confusion is *capability misgeneralization*, where the model generally fails to take correct or useful actions in deployment (Gupta et al., 2022; Tien et al., 2022). Closely tied to the field of AI safety (Hendrycks et al., 2022, 2023; Ngo et al., 2022), Langosco et al. (2022) and later Shah et al. (2022) identify *goal misgeneralization* as an additional, potentially more dangerous failure mode, where the model capably pursues a different goal in deployment. The authors identify goal misgeneralization as being of particular concern due to the retained capability of the model when generalizing, allowing for the possibility of visiting arbitrarily undesirable states.

In this work, we set out to tackle the issue of goal misgeneralization by improving the expressiveness by which we specify the goal or task. We focus on this direction because we hypothesize that goal misgeneralization may at times be caused by the limited nature by which tasks are specified. For instance, specifying a task to an RL agent via sparse rewards awarded only upon task completion may be too coarse of a specification for tasks that require more nuance in the behavior throughout the entirety of the trajectory (Vamplew et al., 2022). We focus on the problem area of of sequential decision making (SDM) and, inspired by the recent applications of language in other fields (Dosovitskiy et al., 2022; Ramesh et al., 2022; Rombach et al., 2022), we choose explore the use of language for task specification. We choose this direction because we view task specification as a *communication* between task requester and task executor. For communicative intents, we view language as the best option for providing a more natural and expressive interface between human and machine to specify goals. After all, this is how humans communicate desired outcomes to each other.

1.2 Contributions

With this work, we make the following contributions:

1. We are the first to employ natural language in an attempt to address the issue of goal misgeneralization.
2. To this end, we are the first to explicitly frame and treat goal misgeneralization in the context of multi-task learning, and present a new definition of the phenomenon.
3. Correspondingly, we outline an original framework for task specification in SDM, and outline why focusing on task specification may help with causal confusion.
4. We describe, implement and train an implementation under this framework and demonstrate how existing and future foundation models may be leveraged for the problem.
5. Finally, we are the first to the best of our knowledge to explicitly link goal misgeneralization to Occam's razor, and provide actionable advice to researchers for future work on the phenomenon.

We make our code publicly available at github.com/thesofakillers/nlgoals, along with a couple of demonstration videos.

1.3 Outline

The remainder of this document will be structured as follows. In Chapter 2, we provide the necessary background, introducing prior work and theory necessary for the following chapters. In Chapter 3, we first outline our task specification framework, discuss the role of specification in CC and define GMG, relating it directly to multi-task learning. We then describe the implementation details for our approach to the problem. In Chapter 4, we describe our experimental setup, the results of our experiments and discuss the implications of our findings. We review related work in Chapter 5, and finally conclude in Chapter 6.

CHAPTER 2

Background

2.1 Sequential Decision Making

Sequential Decision Making (SDM) is the field studying problems and approaches wherein an artificial agent interacts with an environment in the process of pursuing and eventually achieving a specific goal (Frankish and Ramsey, 2014). In this context, we envision the agent as acting according to some *policy* π which maps states S to actions A. States are instantaneous representations of the environment, descriptions of the environment at a given moment. Actions are motions and outputs produced by the agent that may affect the state of the environment. We model the interaction between the agent and the environment as unfolding over discrete time steps. At each time step, the agent observes the state, consults its policy π to select an action, and then executes that action. In the next time step, the environment responds by transitioning to a new state, and the loop continues.

What we are gradually formalizing here is the classical *Markov Decision Process (MDP)* framework (Puterman, 2014) for SDM, shown in Figures 2.1 and 2.2. Aside for states S and actions A, an MDP additionally consists of

- A *transition function* $P : S \times A \times S \to [0, 1]$, describing the probability that action a in state s will lead to state s'.
- A *reward function* $R : S \times A \times S \to \mathbb{R}$, the immediate reward achieved when transitioning from state s to state s' via action a.

The agent interacting with the MDP gives rise to a discrete sequence of states, actions and rewards known as a *trajectory*, indexed by time-steps t and taking the form

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots, S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}.$$

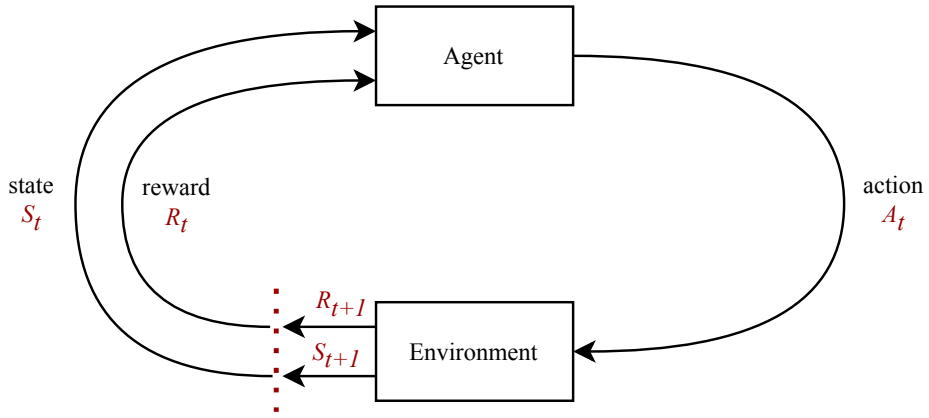

Figure 2.1: The agent-environment loop of a Markov decision process.

We note that the transition function only depends on the current state and action, implying that current state is independent of all but the previous state. This is the *Markov property*.

The reward function is a proxy of how well the agent is doing in pursuing a specific task. *Reinforcement Learning (RL)* can then be defined as learning how to act interactively to maximize expected cumulative reward (Sutton and Barto, 2018).

The transition function of the MDP is often unavailable to the agent and must be modeled either implicitly or explicitly. Similarly, the reward function is often unavailable or underspecified, making RL impractical to use. What is sometimes available instead, are (close to) expert demonstrations of the task at hand. *Imitation Learning (IL)* can be defined as the optimization of a policy π_{IL} such that it behaves similarly to a given expert or set of experts (Schaal, 1999). In our work, we focus on *behavioural cloning (BC)* (Michie et al., 1990), reducing our SDM problem to supervised learning.

In general there is no "one-size-fits-all" solution to SDM. However, MDPs, while limited in some ways, provide sufficient flexibility for a variety of approaches.

2.2 Goal Misgeneralization

Goal Misgeneralization (GMG) is an empirically observed failure mode of machine learning models, particularly closely tied to SDM. Two existing definitions exist for the same phenomenon which we present below for background.

Langosco et al. (2022) are the first to formally identify the issue of GMG, describing it as a form of out-of-distribution (OOD) robustness (Arjovsky, 2020) failure in RL. They specifically distinguish it from "capability generalization failures", where an agent deployed OOD simply fails to take useful

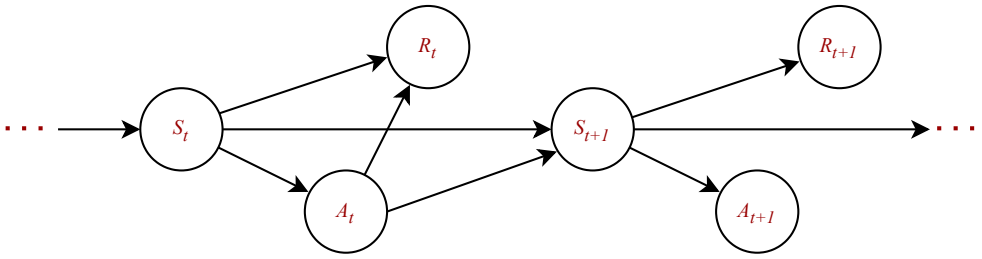

Figure 2.2: A Markov decision process visualized as a causal graph.

actions. Instead, an agent suffering from goal misgeneralization "pursues a goal other than the training reward while *retaining the capabilities*" (emphasis ours). More precisely, their definition consists of the following: Considering a deep RL agent trained to maximize some reward $R : S \times A \times S \mapsto \mathbb{R}$, they assume that, at test time, the agent is deployed in an environment where some aspect has changed, such that the environment can now be considered OOD. Under their definition, GMG occurs if the agent achieves low reward *because* it continues to act capably but instead behaves as if it were in pursuit of some other reward $R' \neq R$.

Shah et al. (2022) also tackle the phenomenon of goal misgeneralization, providing a new definition and additional empirical evidence. Under their definition, goal misgeneralization occurs when, upon deployment in a test setting, a model's capabilities include those necessary for achieving the intended goal, but the model's behaviour is not *consistent* with the intended goal and is instead consistent with some other goal.

While both definitions are certainly helpful in painting a picture of the phenomenon, we find it best grasped through examples, which we present an instance of in Figure 2.3. For this work, we use our own definition for the phenomenon, which we present in Section 3.1.3

2.3 Causal Confusion and Goal Misgeneralization

Inspired by the works of Gupta et al. (2022) and Kirk and Krueger (2022), we hold the view that GMG is a direct consequence of *causal confusion (CC)* (de Haan et al., 2019). This is the phenomenon by which a learner incorrectly identifies the causal model underlying its observations and/or behaviour. This is typically due to spurious correlations between the true cause X for a random event Y and some other variable W that does not causally model Y. We refer to Figure 2.3 for an example of GMG as an instance of CC. Following Occam's razor (Ariew, 1976; Blumer et al., 1987), we posit that CC may lead to GMG when the confounding variable, i.e. the variable spuriously correlated with the causal factor, is easier to learn.

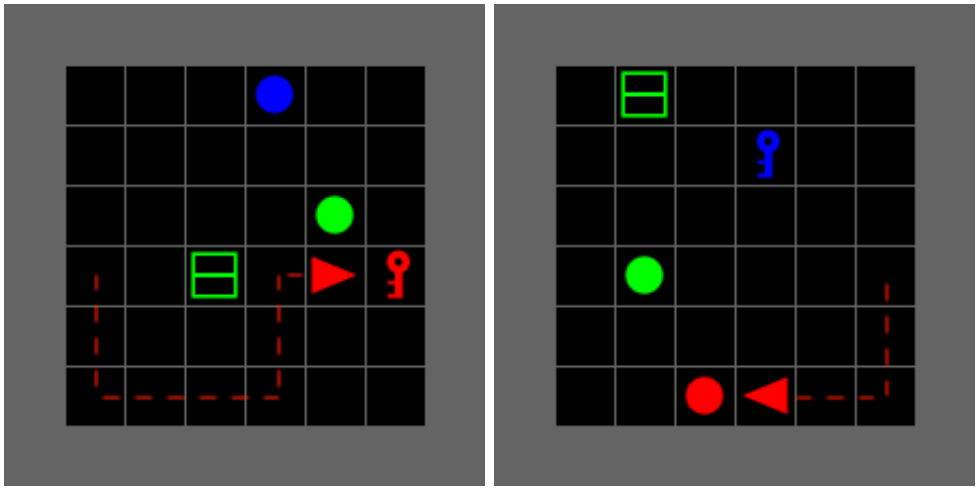

Figure 2.3: An example of goal misgeneralization as a consequence of causal confusion. The intended goal is to move to the key. During training (left), the key is always red. The confounding goal is to move to the red object. During testing (right), the key can be of different colors. In other words, the red color property and the key object type property were spuriously correlated during training. In this case, we expect learning the red color property to be easier than the key object type property, which under Occam's razor would explain why the agent becomes causally confused and misgeneralizes at test time and navigates to the red ball instead.

Accordingly, we note that GMG may therefore be addressed by tackling CC itself. In light of this, we can distinguish three approaches. The first involves performing causal inference with the assistance of interventions on the data so to better discover the underlying causal model. This is the main approach of de Haan et al. (2019). The second approach simply increases the variability of the training data so as to reduce the likelihood of spurious correlations. This is the main approach of Langosco et al. (2022). The final approach focuses on improving the expressiveness of the task specification. We hypothesize that overly coarse specifications may lead to ambiguity in which task is being requested, increasing the chance of causal confusion. We provide more detail in Section 3.1.2.

While each of these approaches have merit, we decide to focus on the third. Our motivation is manifold. First, we expect implementations under the first approach to become increasingly more difficult as the field shifts towards offline-learning (Lange et al., 2012; Levine et al., 2020; Prudencio et al., 2022). Secondly, while the simplicity of the second approach coupled with recent advancements in scaling laws (Hoffmann et al., 2022; Kaplan et al., 2020) is promising, we note that increasing the variability of the training data has no guarantee of de-correlating confounding variables, especially when the

spurious correlations are unknown, rendering estimating how much and what kind of variability to work on potentially difficult for more insidious cases of GMG (Kirk and Krueger, 2022). We choose to focus on the approach of improving task specification not only because we view it as an under-explored option, but more importantly because, as we will outline in Chapter 3.1, we view GMG as intrinsically tied to multi-task learning (Caruana, 1997), which itself is intrinsically tied to task specification.

2.4 Grounding

Our approach to improving task specification will focus on the notion of specifying tasks through natural language rather than rewards. We refer to this as *language-informed sequential decision making (LISDM)*. Any treatment of LISDM will necessarily involve grounding, i.e. ensuring that the language used to specify tasks is mapped to representations that can be correctly interpreted by the agent.

More specifically, the *symbol grounding problem* is the problem of ensuring that the meaning of abstract symbols (e.g. words) is anchored to more than the symbols themselves (Harnad, 1990). That is, symbols, on their own, are meaningless. They only acquire meaning when they are grounded to some external referent (Bender and Koller, 2020) or relation (Piantadosi and Hill, 2022). For example the word "cat" is meaningless on its own, but when it is grounded to the concept of a cat, it acquires meaning. The grounding problem is a problem of communication (Clark and Brennan, 1991), and is therefore relevant to task specification.

Addressing the grounding problem, at least in part, is a necessary step for an effective implementation of LISDM. A partial treatment of the problem may be sufficient since we restrict our focus to the context of representation learning: In certain narrowly defined LISDM scenarios, it might be sufficient to ensure that the representations of the same concept in a few modalities are alike (Mooney, 2008).

The grounding problem warrants attention primarily because the effectiveness of using language to make decisions in a specific environment hinges on the accurate alignment of the word meanings with that environment. Failing this, the effectiveness of language representations in guiding decisions could be significantly compromised. Take, for instance, the case where we are endeavoring to use language for decision-making in a restaurant setting. It becomes essential to ensure that the representation for the term "apple" aligns closely with the representation of an actual apple present in the restaurant. This is because we aim to be able to use the word "apple" to refer specifically to the apple in that restaurant, rather than some other apple in a different restaurant or another item in the same restaurant that is not an apple.

CHAPTER 3

Method

3.1 Preliminaries

As mentioned, in this work we tackle the issue of GMG in SDM by focusing on improving task specification. Below, we first outline specifically what we mean by task specification, and later discuss the implications for our own definition of GMG.

3.1.1 Task specification

Task specification is the scenario in which a *requester* $\mathcal{R}$ specifies a *task* $\mathcal{T}$ to be performed by an *actor* $\mathcal{A}$[1]. In SDM, The requester expresses a high-level representation $\mathcal{Z}$ of the ideal trajectory of state-action pairs, corresponding to the task they would like to be performed. We specifically allow high-level representations of trajectories because it can occur that the requester does not know exactly what sequence of state-action pairs they want, and are typically more interested in more abstract, higher level desiderata anyway.

The actor is necessarily a multi-task policy, as otherwise task-specification would be futile. The actor receives $\mathcal{Z}$ and "interprets" it by using it as a conditional variable on its policy. Like Cho et al. (2022), we therefore write the actor's policy as $\pi(a \mid s, \mathcal{Z})$, where $\mathcal{Z}$ represents an encoding of the intended task. We underline that $\mathcal{Z}$ can in principle take any form and originate from any source. Examples include rewards, one-hot encodings, demonstrations, preferences (Christiano et al., 2017), formal language (Bansal, 2022), natural language, *et cetera*.

[1] This generalizes self-proposed tasks, in which the actor is also the requester $\mathcal{A} = \mathcal{R}$.

3.1.2 Specification and causal confusion

Suppose we have some latent *notion* N, an abstraction encapsulating some semantic information, that we wish to communicate. The notion is latent, i.e. not observed directly, and we can instead communicate it through some language L which maps the notion N to some corresponding expression N_L. Note that there can be more than one corresponding expression per notion. In general, the mapping between notion and language expression is many-to-many. Under our task specification framework from above, the task we wish to specify $\mathcal{T}$ is the notion we wish to communicate N, and the high-level representation $\mathcal{Z}$ is the expression N_L we use to communicate it.

In the context of communication, a notion N and its corresponding expressions $N_L^1, N_L^2, \ldots$, can be treated as random variables. This assumption can be made given the wide, almost infinite range of possible notions one may wish to communicate, and similarly to the wide range of ways in which a notion can be expressed. These lead to uncertainty which we can treat probabilistically with random variables.

We can therefore quantify the information content of a given notion or expression using the concept of entropy (Shannon, 1948). Entropy effectively quantifies the average level of uncertainty or "surprise" associated with a random variable. For a discrete random variable X, its entropy $H(X)$ is defined as

$$H(X) = -\sum_{x \in X} p(x) \log p(x) \tag{3.1}$$

where $p(x)$ is the probability mass function of X, and the summation is over all possible outcomes x of X. A higher entropy indicates greater uncertainty and thus greater information content. If an outcome is highly uncertain, it means we have very little prior knowledge about what that outcome will be. Therefore, learning the actual outcome provides us with a substantial amount of new information. Conversely, if an event is certain to occur, then learning that this event has indeed occurred doesn't provide us with any new information because we already knew it would happen. Thus, a higher entropy indicates greater uncertainty and thus greater information content.

The entropy of a given notion N and an expression of it N_L therefore serves as the measure of their respective information content. For a notion, we can write

$$H(N) = -\sum_{n \in N} p(n) \log p(n), \tag{3.2}$$

where $p(n)$ is the probability of notion n being the one intended for communication. For an expression, we can write

$$H(N_L) = -\sum_{n_l \in N_L} p(n_l) \log p(n_l), \tag{3.3}$$

where $p(n_l)$ is the probability of expression n_l being the one used for communication.

N_L will typically be a *compressed* representation of N. In other words, the mapping between notion and expression is not necessarily *lossless* in terms of information

$$\mathrm{H}(N_L) \leq \mathrm{H}(N) \tag{3.4}$$

This compression can be either *intrinsic* or *extrinsic*. The former case corresponds to compression that occurs due to the fundamentally limited expressivity of the language L. For example, a language that lacks the grammar and/or vocabulary for expressing negation, will be fundamentally limited from expressing the notion of absence.

Extrinsic compression is compression that occurs due to reasons external to the language itself. This is typically the communicator choosing to use a coarser expression the notion. For example, choosing to communicate "go to the block" rather than "breathe in, activate your muscles such that your right thigh lifts your right foot off the ground and forward, breathe out, breathe in, ...".

Compression, whether intrinsic, extrinsic or either, can lead to *ambiguity*. These are cases where the same expression F, due to underspecification, maps to multiple semantically different notions $N_1, N_2, \ldots$. We view this as a potential avenue for causal confusion to occur.

For instance, under our definitions, we can frame rewards as a language used to communicate some notion of a desired task to SDM agents. When our rewards are underspecified, they can over-compress our task notion, such that the same reward maps to multiple tasks. The policy may therefore suffer from causal confusion and learn to pursue the wrong task.

We therefore posit that causal confusion and hence GMG can be addressed by focusing on how we specify the task, so to reduce ambiguity in the task specification. We move away from rewards (Vamplew et al., 2022) and instead leverage the potentially much higher expressiveness of natural language, spurred by recent advancements in the field of natural language processing (NLP) (Brown et al., 2020; Devlin et al., 2019; Touvron et al., 2023). For a given notion N, assuming the same amount of engineering effort, we expect the compression faced by the language of rewards LR to be higher than the compression faced by natural language NL, i.e. we expect the following

$$\mathrm{H}(N_{LR}) < \mathrm{H}(N_{NL}) \leq \mathrm{H}(N). \tag{3.5}$$

We reason that the language of rewards faces higher intrinsic compression due to its scalar nature, rendering it more difficult to capture nuance than what would be possible with the multidimensionality and compositionality of natural language, which could not only encode more information directly, but could also allow for factored representations which may more easily be

leveraged for generalization. Similarly, we expect the language of rewards to also face higher extrinsic compression when compared to natural language. We reason that task specification is a communication problem, and to this end natural language is the most natural or "comfortable" interface we have as communicators. Rewards, while succinct, may at times be awkward to specify due to the nature of the tasks. This is for instance the case for sparse rewards awarded only upon task completion, or for the denser proxy rewards awarded in the process of reward shaping (Ng et al., 1999).

3.1.3 Defining GMG in the context of multi-task learning

Goal Misgeneralization is inherently Multi-task. Indeed, all definitions and examples of GMG so far have implicitly defined a multi-task setup, with the presence of some goal task c_g and some other confounding task c_c. After all, the definition of GMG implies the existence of at least one other task beyond the one intended by the designers, as without such a task, it would be impossible for the model to pursue it. We instead choose to explicitly define this multi-task setup, relying on the framework from Wilson et al. (2007).

Specifically, let $\mathcal{C} = \{c_i\}_{i=1}^{N}$ be a set of discrete episodic tasks. This could for example the set of all tasks $\mathcal{T}$ with natural language instructions $\mathcal{T}_{NL}$, following the notion and expression notation from the previous section. Let $p_{\text{train}}(\mathcal{C})$ and $p_{\text{test}}(\mathcal{C})$ be the distributions from which the tasks are sampled during training and testing respectively. Each task c_i then defines a separate MDP $M_i = (S, A, R_i, P_i)$, such that the reward and transition functions differ by task. At training time we try to find a task-conditioned policy

$$\pi : S \times \mathcal{C} \to \Delta(A),$$

with an objective conductive to good performance across the tasks. For multi-task RL, such an objective maximizes the expected reward over the distribution of tasks, i.e.

$$\pi^*_{\text{RL}} = \arg\max_{\pi \in \Pi} \mathbb{E}_{c \sim p_{\text{train}}(C)} \left[\mathbb{E}_{\pi_c} \left[\sum_{t=1}^{T_c} \gamma^t R_{t,c} \right] \right], \tag{3.6}$$

where T is the horizon of time steps t and γ is the discount factor. For multi-task IL, such an objective minimizes the expected loss L between policy and expert behaviour over the distribution of tasks, i.e.

$$\pi^*_{\text{IL}} = \arg\min_{\pi \in \Pi} \mathbb{E}_{c \sim p_{\text{train}}\ (C)} \left[\mathbb{E}_{\pi_\varepsilon} \left[L_c \right] \right]. \tag{3.7}$$

Given the above, we define *Goal misgeneralization (GMG)* as the observed phenomenon in which a system successfully trained to pursue a particular goal c_1 in setting X fails to generalize to a new setting Y and instead capably pursues a different goal c_2. A *goal* in this definition can either be a specific

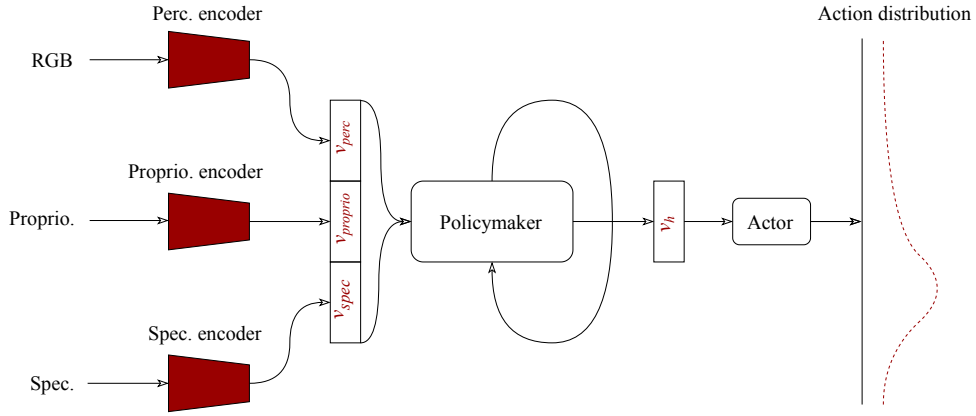

Figure 3.1: A high-level overview the CBC architecture. At every time step in the sequence, a CBC instance receives an RGB representation of the environment state, proprioceptive state and a task specification from the requester. These are encoded by their respective encoders. In the case of GCBC, the specification encoder is CLIPT (see Figure 3.4), while in the case of RCBC, the specification encoder simply multiplies the reward by a one-hot vector encoding the specified task. The output of the encoders is concatenated and fed into the policymaker module, which recursively computes a hidden representation $\boldsymbol{v}_h$ utilizing hidden state from previous steps. Finally, the actor module uses this $\boldsymbol{v}_h$ to compute a distribution over the action space.

state (static) or a behaviour (dynamic). Note that we use the words "task" and "goal" interchangeably, and will do so for the remainder of this work. A system will be in *capable pursuit* of a given goal if a metric M describing the extent of goal achievement (e.g. success rate) is significantly higher than equivalent metric for most other goals in $\mathcal{C}$. Mathematically, we say GMG happens if

$$\exists c_1, c_2 \in \mathcal{C}, \text{ s.t. } p_{\text{test}}(c_1), p_{\text{test}}(c_1) > 0, \tag{3.8}$$

and

$$\mathbb{E}_{\pi_{c_1}}\left[M_{c_2}\right] > \mathbb{E}_{\pi_{c_1}}\left[M_{c_1}\right]. \tag{3.9}$$

We place our definition in between those of Langosco et al. (2022) and Shah et al. (2022), relaxing the former's reliance on RL and Orseau et al. (2018)'s agents and devices framework for simplicity, while focusing on SDM rather than the more general case proposed by the latter, to avoid overly wide characterizations of the phenomenon.

3.2 Conditioned Behavioural Cloning

We are interested in addressing GMG in SDM by improving task specification. We follow our requester-actor task specification framework where our SDM policy is conditioned on our specification. More specifically, our policy takes the current state $s \in S$ and our task specification $z \sim \mathcal{Z}$ as input, and outputs a distribution over possible actions $a \sim A$, $\pi(a|s, z)$. For the sake of simplicity, we take inspiration from Lynch et al. (2020)'s baselines and define a *conditioned behavioural cloning (CBC)* architecture. Represented in Figure 3.1, our CBC policy consists in the following modules:

A **perception encoder**, which at each time step perceives a static view of the environment state as RGB images and outputs a 64-dimensional dense representation $\boldsymbol{v}_{\text{perc}}$. We use the same encoder as Lynch et al. (2020), also adopted in following papers (Lynch and Sermanet, 2021; Mees et al., 2022a,b). This consists in a deep neural network comprising of a series of convolutional layers interleaved with the ReLU activation function. Spatial softmax (Finn et al., 2016) is used to flatten the convolution features into the output dimension. Figure 3.2 outlines the architecture in more detail.

A **proprioceptive encoder**, which at each time step encodes proprioceptive state from the agent (such as e.g. joint positions in the case of a robotic arm) into a sparse representation $\boldsymbol{v}_{\text{proprio}}$. For our experiments (see Chapter 4) the proprioceptive state was simple enough that the encoder was simply the identity function.

A **specification encoder** which encodes the requester's specification into a specification vector $\boldsymbol{v}_{\text{spec}}$. We implement this differently based on which subclass of CBC we are considering (see Sections 3.2.1 and 3.2.2).

A **"policymaker" module** which at each time step takes a concatenation of $\boldsymbol{v}_{\text{perc}}$, $\boldsymbol{v}_{\text{proprio}}$ and $\boldsymbol{v}_{\text{spec}}$ and recursively encodes it into a 2048-dimensional vector $\boldsymbol{v}_h$ using a gated-recurrent-unit (GRU) (Cho et al., 2014). We choose a GRU and this particular dimension following Mees et al. (2022a).

An **actor module** which takes the hidden representation $\boldsymbol{v}_h$ from the policymaker and encodes it into a distribution over our action space. We implement the actor module differently based on whether we are dealing with discrete or continuous action spaces (see Section 3.3).

During training, CBC receives batches of demonstration rollouts, consisting of trajectories of environment state and expert next actions. In parallel to each trajectory, if applicable[2], the batch contains the specification (for example raw text) for the trajectory and the ID of the task completed in the trajectory, the latter of which may or may not be used to assist with handling

[2] As noted in Section 3.2.2, at least one of our training setups allows for trajectories that are not labeled with specification nor task ID.

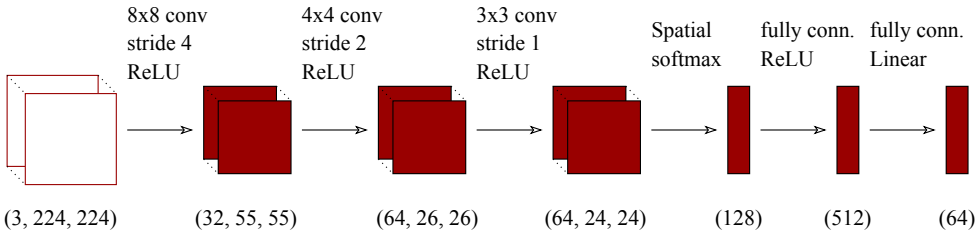

Figure 3.2: The perception encoder of our CBC policy. Not pictured: Layer normalization (Ba et al., 2016) is applied after the final fully connected layer.

multi-task considerations. Only the perception encoder, policymaker and actor are trained, with the other modules either being trained separately or not requiring training at all.

At inference time, CBC receives the current state and specification and samples an action from the computed action distribution. Recurrence is allowed for a specific number of steps (typically equivalent to the length of the trajectories shown during training), after which the hidden state is reset.

3.2.1 RCBC: Reward-conditioned Behavioural Cloning

To serve as a baseline for our experiments addressing GMG (see Section 4.2), we formulate a subclass of CBC where the task is specified by conditioning on reward, similarly to (Chen et al., 2021). To fit under our multi-task setup, the specification encoder simply multiplies the reward by a one-hot encoding vector obtained from the task ID similarly to Cho et al. (2022), such that each dimension corresponds to a different task in our multi-task ensemble. Therefore, for e.g. a 6-task ensemble, we have a 6-dimensional one-hot vector for $\boldsymbol{v}_{\text{spec}}$, with the non-zero element corresponding to the reward for task at hand.

3.2.2 GCBC: Goal-conditioned Behavioural Cloning

As stated in previous sections, we posit that task specification can be improved by using natural language instead of rewards, and that this improvement will manifest itself when comparing behaviour in causally confused setups. Following our requester-actor framework, we view natural language specifications as representations of the task that is being specified, or the goal we want our agent to achieve.

Naively we would like to condition our policy on the rich dense representations provided by state-of-the-art language models. However, due to the grounding problem (see Section 2.4), using these representation directly could result in the policy failing to appropriately relate them to their referents in the environment and in the policy's internal representations. Additionally, we

are faced with the reality that language-annotated trajectories are rare and expensive to collect.

To address these issues, we instead train on *visual* representations $\boldsymbol{v}^{\text{vis}}_{\text{spec}}$ of our goals originating directly from the environment (whose state we represent visually, in RGB images), and assume that these visual representations are in a semantic space similar to the equivalent *textual* representations of the same goals $\boldsymbol{v}^{\text{txt}}_{\text{spec}}$, obtained via e.g. some form of contrastive representation learning (Chen et al., 2020; Le-Khac et al., 2020). That is, we assume that when conditioning on the visual goal of e.g. going to object G, this representation will be similar to the representation for the instruction "go to object G". We keep this description relatively high-level at this stage to keep our method general. We follow with more detail about how we implement visual and textual goals in Section 3.4.

Our assumption allows us to leverage the much larger availability of visual data for training. During this phase, the policy is trained on a behavioural cloning objective, i.e. matching the behaviour in the expert demonstrations as closely as possible[3]. Because our (visual) goal representations can be constructed directly from the environment and do not rely on the notion of rewards, they are flexible enough to be applied to any trajectory where the goal can be described with the final image. We therefore can afford to expand our training beyond merely "successful" trajectories as dictated by our task ensemble. Instead, we can encompass a wider range of behaviors, incorporating for instance much cheaper and more available "play data", consisting in exploratory, potentially random behaviour in the environment, covering a wide range of states and actions (Lynch et al., 2020). We posit that the increased variability from this training data should aid with grounding, as the policy learns to interpret incoming representations. At inference time, thanks to our assumption, the visual representations can be swapped for more easily articulable textual representations.

3.3 Actor Modules

We implement our actor modules differently depending on the kind of action space our policy needs to act in. In discrete action spaces, the possible actions the policy can take are finite, for instance "move forward", "turn right", "turn left", et cetera. In continuous action spaces, the possible actions are potentially infinite. These could for instance be a vector describing the (x, y, z, ϕ, θ) coordinates for the position and orientation of a robotic arm. We describe our implementations for both cases below.

[3]The specific implementation will vary depending on which actor module is used. See Section 3.3.

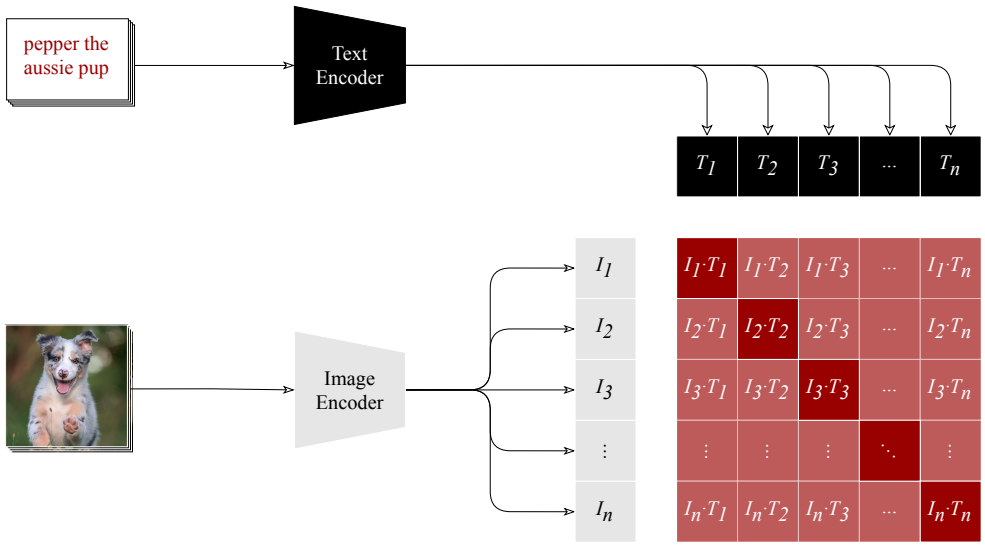

Figure 3.3: A general overview of CLIP. Pairs of images and captions are respectively fed as input to text and image encoders. The encoders are trained contrastively, such that the similarity between the textual and image representations T_i and I_i is maximized for the original pairings (the diagonal) and minimized for the remaining pairings (off the diagonal).

3.3.1 Discrete Action spaces

Our implementation for discrete action spaces is very simple. For an N-dimensional action space, we use a linear layer to project our policymaker's output representation $\boldsymbol{v}_h$ to a $3N$-dimensional hidden representation. We apply ReLU and then use another linear layer to project the hidden representation to a N-dimensional vector. We apply a softmax activation function and train our policy on the cross-entropy loss between the resulting predicted action distribution and the labeled expert action. We sample actions by selecting the action with the highest probability using the argmax operation. We log the action prediction accuracy as our performance metric throughout training.

3.3.2 Continuous Action spaces

For continuous action spaces, we follow Lynch et al. (2020) and use a *discretized logistic mixture likelihood (DLML)*[4,5] (Salimans et al., 2017). This is of particular utility for our use case, where many high-level behaviours may

[4]Also known as "Discretized Mixture of Logistics (DMoL)", "Discretized Logistic Mixture (DLM).", "Mixture of Discretized Logistics (MDL)"

[5]A detailed tutorial, including motivation and implementation details, was produced as part of the thesis: giuliostarace.com/posts/dlml-tutorial

satisfy the same task specification. Utilizing a mixture of distributions can help with modeling this potentially multi-modal problem. For a given continuous output variable y, the details of DLML consist of the following:

1. We assume that there is a latent value v with a continuous distribution.
2. We take y to come from a discretization of this continuous distribution of v. We do this 8-bit representation. What this means is that if e.g. v can be any value between 0 and 255, then y will be any *integer* between those two numbers.
3. We model v using the logistic distribution $v \sim \mathfrak{L}(\mu, d)$, where μ and d are "location" and "scale" parameters respectively, defining the distribution.
4. We then take a further step, choosing to model v as a mixture of K logistic distributions:

$$v \sim \sum_{i}^{K} q_i \mathfrak{L}(\mu_i, d_i), \tag{3.10}$$

   where q_i is some coefficient weighing the likelihood of the ith distribution, which we refer to as "mixture logit".
5. To compute the likelihood of y, we sum its (weighted) probability masses over the K mixtures. We can obtain the probability masses by computing the difference between consecutive cumulative density function (CDF) values of equation (3.10). Note that the CDF of the logistic distribution is a sigmoid function, σ. We therefore write:

$$p(y|\boldsymbol{q}, \boldsymbol{\mu}, \boldsymbol{d}) = \sum_{i=1}^{K} q_i \left[\sigma\left(\frac{y + 0.5 - \mu_i}{d_i} \right) - \sigma\left(\frac{y - 0.5 - \mu_i}{d_i} \right) \right], \tag{3.11}$$

   The 0.5 value comes from the fact that we have discretized v into y through rounding, and therefore successive values of our discrete random variable y are found at this boundary.
6. Finally, we can model edge cases, to avoid assigning probability mass outside the valid range of values. We do this by replacing $y - 0.5$ with $-\infty$ when $y = 0$ and $y + 0.5$ with $+\infty$ when $y = 2^8 = 255$.

We are left with nothing more than a likelihood. We can therefore apply a maximum likelihood estimation (MLE) process to estimate our parameters q, μ and d. Pragmatically, For a given output variable y, using K mixture elements, our model should output K locations, scales and mixture logits.

We therefore implement a separate linear layer for each of our mixture parameters, each projecting the policymaker's output representation $\boldsymbol{v}_h$ into a K-dimensional vector, with each dimension corresponding to one of the mixture elements. For cases in which our output variable y is actually an M-dimensional vector $\boldsymbol{y}$, the linear layers project to a $(K \times M)$-dimensional vector instead. We train the resulting policy architecture on the negative log likelihood loss based on equation (3.11).

To sample actions, we sample a distribution from the mixture based on the predicted mixture logits using the Gumbel-Max trick (Gumbel, 1954). We then use inverse sampling with the sampled distribution location and scale parameters to sample a predicted action. For multivariate actions, we log action cosine similarity and action L1 distance as the performance metrics during training.

3.4 CLIPT: Visual/Textual Goal Representations

In Section 3.2.2, we make the assumption that we have access to nearly aligned visual and textual goal representations. That is, for a given goal, we can represent it visually based on combinations of environment start and end state from demonstrated trajectories, and this visual representation will be semantically similar to an equivalent textual instruction for the goal.

To fulfill our assumption, we propose a number of modifications to CLIP (Radford et al., 2021). CLIP is a method for training two models: a text encoder and vision encoder, such that representations of the same concept are in similar semantic spaces across the two modalities. We refer to Figure 3.3 for an example and overview. CLIP achieves this with two ingredients. First, the authors collect a large dataset of image-caption pairs. Secondly, the authors train their encoders on a contrastive loss objective: maximizing the cosine similarity of the vision and text representations of the B true pairs in a given batch, while minimizing the cosine similarity of the remaining $B^2 - B$ incorrect pairings. This is implemented using the *multi-class N-pair loss* of Sohn (2016), which uses categorical cross entropy. For a given image $\boldsymbol{x}$, its correctly paired caption $\boldsymbol{t}^+$ and the remaining $N-1$ incorrectly paired captions $\{\boldsymbol{t}_i^-\}$, this is defined as

$$\mathcal{L}\left(\boldsymbol{x}, \boldsymbol{t}^+, \{\boldsymbol{t}_i^-\}_{i=1}^{N-1}\right) = \log\left[1 + \sum_{i=1}^{N-1} \exp\left(f(\boldsymbol{x})^\top g(\boldsymbol{t}_i^-) - f(\boldsymbol{x})^\top g(\boldsymbol{t}^+)\right)\right], \tag{3.12}$$

where f is the CLIP vision encoder and g is the CLIP text encoder. We refer readers to the original paper for further details (Radford et al., 2021).

Since the release of the paper, a number of off-the-shelf vision-text encoder pairings have become available (Ilharco et al., 2021; Schuhmann et al., 2022). We aim to leverage the rich representations made available by these models

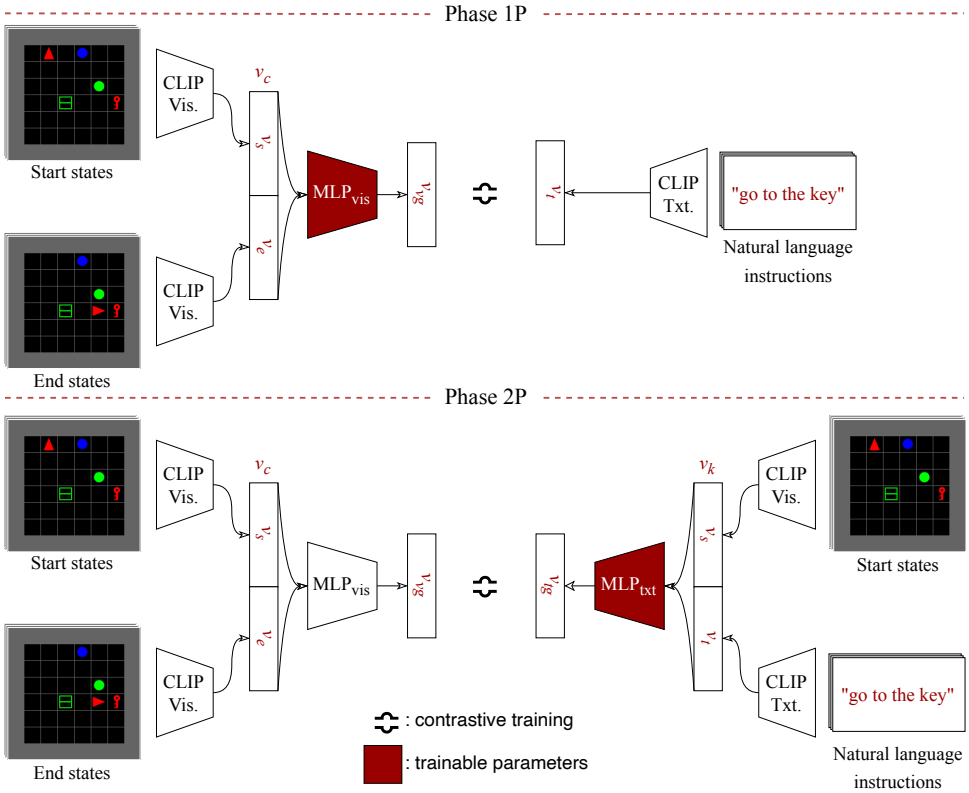

Figure 3.4: A high-level overview of CLIPT. During phase 1P, the start and end state of a trajectory are encoded by the CLIP vision encoder (CLIP Vis) and concatenated into $\boldsymbol{v}_c$. This is projected into $\boldsymbol{v}_{vg}$ by MLP_{vis} which is trained contrastively to match the paired $\boldsymbol{v}_t$ representations of the natural language instructions produced by CLIP's text encoder (CLIP Txt). In phase 2P, the start state of the trajectory is encoded by CLIP Vis and concatenated to $\boldsymbol{v}_t$ to form $\boldsymbol{v}_k$. This is projected into $\boldsymbol{v}_{tg}$ by MLP_{txt}, which is trained contrastively to match the paired $\boldsymbol{v}_{vg}$ representations produced by (frozen) MLP_{vis}.

but envision two limitations which we address with our modifications. Firstly, we expect images of the environment state to be at least slightly out of domain when compared to the dataset CLIP was trained on. The same applies to the instructions we intend to use for textual goal specifications. Secondly, we define a visual goal as being composed of at least two images representing the start and end state of the trajectory. However, CLIPT is trained on single-image inputs, necessitating a modification to this end.

To address these limitations, we perform the following steps:

1. We collect a dataset of language-annotated trajectories.

2. For each sample, we encode the start and end state of the trajectory using CLIP's vision encoder, resulting in u-dimensional representations $\boldsymbol{v}_s$ and $\boldsymbol{v}_e$

3. We concatenate $\boldsymbol{v}_s$ and $\boldsymbol{v}_e$ into a $\boldsymbol{v}_c$ of dimension $2u$.

4. We project $\boldsymbol{v}_c$ back to a u-dimensional vector representing our visual goal, $\boldsymbol{v}_{vg}$, using a multilayer perceptron (MLP_{vis}) with a single u-dimensional hidden layer and ReLU activation function.

5. In parallel, we encode the language annotation using CLIP's textual encoder, resulting in u-dimensional representation $\boldsymbol{v}_t$.

6. Keeping the CLIP encoders frozen, we train the MLP on the same contrastive objective CLIP was trained on, using $\boldsymbol{v}_t$ and $\boldsymbol{v}_{vg}$ instead

The first phase of training (termed as phase 1P) involves training GCBC on the $\boldsymbol{v}_{vg}$ representations from MLP_{vis}. However, we plan to evaluate using textual representations, which could cause a performance gap to emerge due to sub-par representation matching between vision and text modalities, particularly given that the latter does not rely on the context of the start state. We therefore perform a second phase (which we refer to as phase 2P) of training, where we instead work on the textual side. Specifically

1. Using the same dataset from phase 1P, we repeat steps 2-4 from phase 1P.

2. In parallel however, we concatenate $\boldsymbol{v}_s$ with $\boldsymbol{v}_t$ into $\boldsymbol{v}_k$, in a sense providing "context" to our language annotations.

3. Just like in phase 1P, we use a separate but architecturally identical MLP (MLP_{txt}) to project $\boldsymbol{v}_k$ back to a u-dimensional vector $\boldsymbol{v}_{tg}$ representing our textual goal.

4. We train this MLP on the aforementioned contrastive loss with $(\boldsymbol{v}_{tg}, \boldsymbol{v}_{vg})$ pairs, this time additionally freezing MLP_{vis} from phase 1P, so to only train the textual goal representations.

We settle on this two-phase training setup rather than directly training both MLP heads together as we expect that the contrastive objective, mixed with the shared context vector $\boldsymbol{v}_s$, would entice the MLPs to ignore the second halves of the concatenated inputs, $\boldsymbol{v}_e$ and $\boldsymbol{v}_t$ respectively, which would greatly harm our abilities to specify tasks. We refer to the resulting model as CLIPT: Contrastive Language Image Pretraining for Trajectories. Figure 3.4 provides a visual overview of the process.

CHAPTER 4

Experiments and Discussion

Having defined preliminaries and the implementation for our approach to the problem of GMG, we now turn to the experiments to evaluate our methodology and an analysis of the respective results.

4.1 GCBC's Instruction Following Capabilities

We initially focus on the verification of our approach with a specific focus on GCBC's ability to follow instructions. We strive to determine if our model's choices are justifiable, if it can handle somewhat realistic scenarios, and if its capabilities are up to par.

4.1.1 The CALVIN dataset

Introduced by Mees et al. (2022b), CALVIN (**C**omposing **A**ctions from **L**anguage and **Vi**sio**n**) is a benchmark originally intended for training and evaluating models on long-horizon language-conditioned tasks. That is, a multi-task set of 34 SDM tasks where the expected interaction sequence is considerably long (up to 240 steps, due to task chaining) and the tasks are specified via language. The benchmark consists of, among other things, an environment and a dataset.

The environment simulates a 7-DOF[1] Franka Emika Panda robot arm (Haddadin et al., 2022) interacting with three rectangular objects of different colors and shapes on a desk. The desk additionally features a sliding door and drawer that can both be opened and closed. Finally, a button and switch can be used to activate a green light and light bulb respectively. The observation

[1] *scilicet*: degrees of freedom.

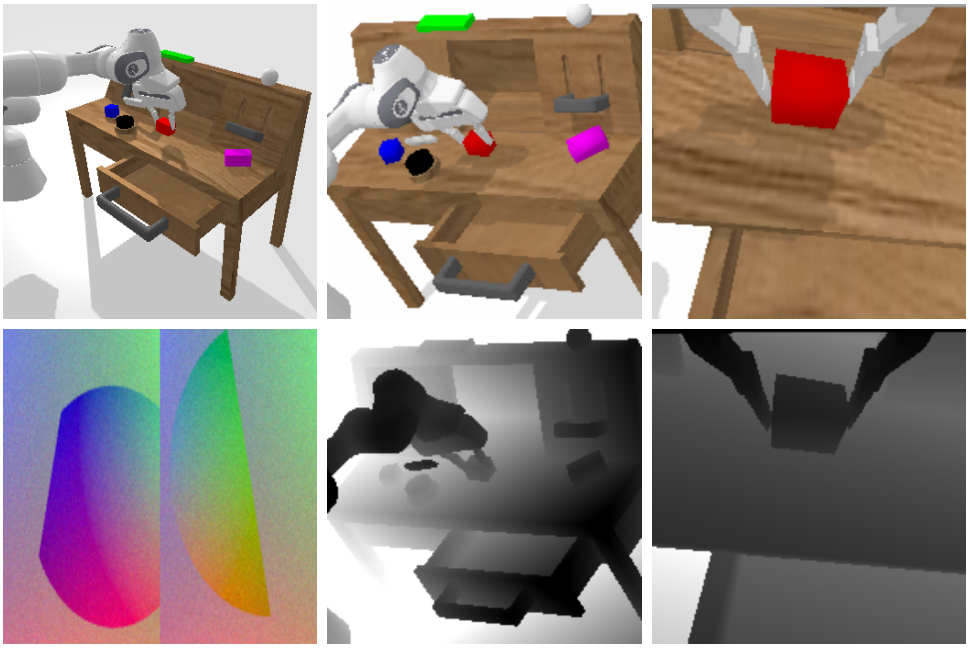

Figure 4.1: An overview of the perceptive state in the CALVIN environment. Top, left to right: scene, static and gripper RGB images. Bottom, left to right: tactile, static depth, gripper depth. Figure courtesy of Mees et al. (2022b)

Table 4.1: The CALVIN 15-D proprioceptive state and 7-D action vectors. **Bold** indicates that these dimensions are present in both proprioceptive state and action space. In case of actions, these can either be "static" (relative to the world frame) or "relative" (displacements relative to the gripper frame). In case of state, this is always static.

Dimension(s)	Description
0-2	**TCP** (x, y, z) **coordinates**
3-5	**TCP** (x, y, z) **euler angle coordinates**
6	Gripper opening width (meters)
7-13	Robotic arm joint state (rad)
14	**Binary gripper action (close** $= -1$**, open** $= 1$**)**

space consists of RGBD[2] images from a static camera as well as from a camera mounted to the robotic arm's gripper, with dimensionality 200×200 and 84×84 respectively. A vision-based sensor additionally provides images of shape $160 \times 160 \times 6$ for tactile information at the gripper. Figure 4.1 presents an overview of the perceived state. Finally the proprioceptive state is also

[2] *scilicet*: red, green, blue and depth channels.

Table 4.2: The 34 tasks of the CALVIN environment and an example natural language annotation. Some tasks are grouped together due to similarity. We note the group sizes with the numbers in the parentheses.

Task	Example annotation
Rotate red/blue/pink block right (3)	"turn the red block right"
Rotate red/blue/pink block left (3)	"turn the blue block left"
Push red/blue/pink block right (3)	"push right the pink block"
Push red/blue/pink block left (3)	"push left the red block"
Move slider left/right (2)	"slide the door to the left"
Open/close drawer (2)	"go open the drawer"
Lift red/blue/pink block table (3)	"pick up the red block"
Lift red/blue/pink block slider (3)	"lift red block from slider"
Lift red/blue/pink block drawer (3)	"lift blue block from drawer'
Place in slider/drawer (2)	"put the grasped object in the slider"
Push into drawer (1)	"push the block into thedrawer"
Stack blocks (1)	"stack blocks on top of each other"
Unstack blocks (1)	"collapse the stacked blocks"
Turn on/off light bulb (2)	"toggle the light switch to turn on the light bulb"
Turn on/off LED (2)	"push the button to turn off the green light"

available in the form of a 15-dimensional vector describing the TCP[3] position and orientation, as well as gripper state and arm joint states. For acting, models operating in the CALVIN environment sample from a continuous 7-dimensional action space describing the position and orientation of the TCP and gripper activation. Table 4.1 provides an overview of what each dimension of the proprioceptive state and action vectors corresponds to. The authors define 34 different tasks that can be performed in the environment and can be labelled with natural language instructions, which we present in Table 4.2. The environment comes with an oracle that can check for the completion of a task in a given sequence of steps, as defined by specific changes of the state between the initial and final steps in the sequence. The environment can take any of four configurations A, B, C, and D, which are structurally identical but present some differences in e.g. table colour.

The CALVIN *dataset* consists of ∼2.2M demonstrations across the four environment configurations. This is largely composed of "unstructured" demonstrations, recordings of exploratory and even random expert interactions with the environment, equivalent to the "play" data described in the previous chapter. The authors then crowd-source over 400 natural language instructions describing the 34 tasks, and procedurally label 1% of the demonstrations with these instructions when possible, i.e. when they find that a given demonstration solves one of the 34 tasks. For our work, we restrict our attention to the D configuration of the environment, equivalent to ∼700K demonstrations. We choose to validate our approach using the CALVIN benchmark due to its more

[3] *scilicet*: tool center tip

realistic and challenging nature. We expect that if our implementations successfully learn an instructable policy for such a benchmark, then they should be straightforward to apply to more easily configurable setups. Additionally, we believe demonstrating applicability to CALVIN could be a good indicator of the soundness and/or generalizability of our approach.

4.1.2 CLIPT

We use CLIPT as the specification encoder of our GCBC solution. As previously mentioned, we train CLIPT separately from GCBC. We do this because we approach the problem from the recently prevalent perspective of utilizing representations from pre-trained foundation models for applications on downstream tasks. We take this approach because we expect future solutions to follow this trend, at least in the short-term. Under this framework, we treat CLIPT as a stand-in for such a foundation model, and GCBC as the beneficiary of the representations.

Training details

We train CLIPT on the subset of CALVIN that is language-annotated, sampling only the start and end state of each trajectory. This amounts to a total of 5124 training samples, and 1011 validation samples, each consisting of two images and the natural language instruction. We use the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 5×10^{-5}. For both training phases, we train for a total of 50 epochs, allowing for early stopping if the validation loss does not improve for 3 epochs, and use the checkpoint with the best validation loss. We use LAION's "CLIP-ViT-L-14-laion2B-s32B-b82K" checkpoint of OpenCLIP (Ilharco et al., 2021) throughout our work. This is a version of CLIP relying on an L-14 variant of the visual transformer (ViT) architecture (Dosovitskiy et al., 2022) for the vision encoder, pretrained on the 2 billion pair subset of the LAION-5B dataset (Schuhmann et al., 2022).

CLIPT sanity checks

We perform a number[4]of sanity checks on the soundness of our CLIPT design. Our main concern is whether CLIPT is making use of the crucial information contained in the second half of its input. We hold this concern due to the fact that the first half of the input to MLP_{vis} and MLP_{txt} is the same, namely the CLIP representation of start state, $\boldsymbol{v}_s$. Given that CLIPT is trained contrastively, the MLP encoders could be pressured into simply relying on the first half of the input while ignoring the second half while constructing

[4]we outline our two main checks here, and present our other CLIPT sanity checks in Appendix A.

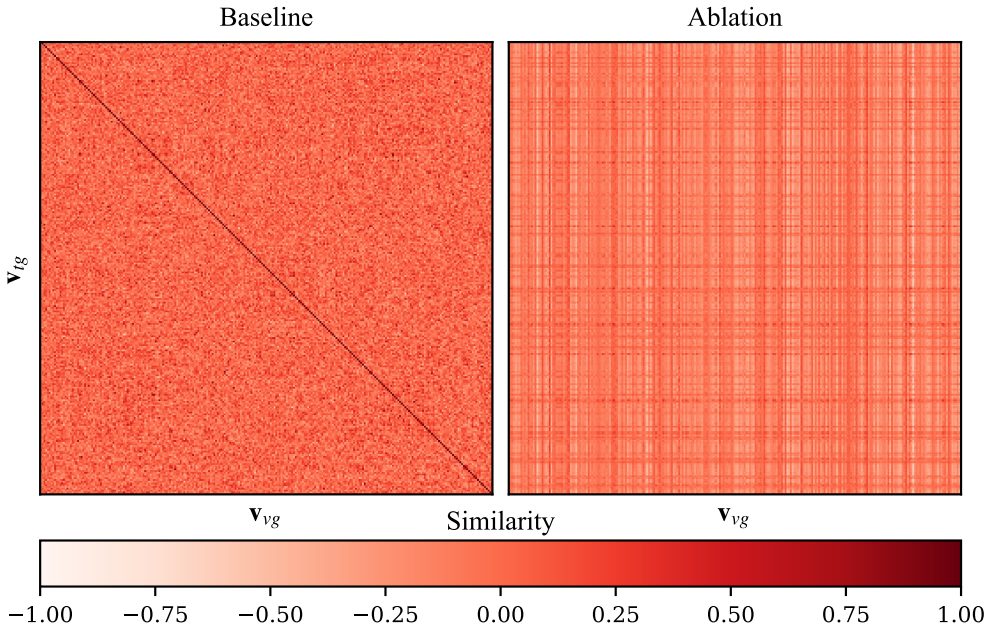

Figure 4.2: Similarity matrices of the baseline (left) and masked (right) representations from CLIPT. The ideal matrix should display a peak in similarity along the diagonal.

representations, as this would contain sufficient information for matching true pairs of goal representations.

The first check we perform consists in setting the second half of the input to a 0-valued vector at inference time, and evaluating whether this affects performance. We expect that if CLIPT is indeed ignoring the second half of its input, then masking it in this way should not affect performance. To perform our check, we operate on the validation split of the CALVIN dataset, sampling 256 trajectories for which we compute pairs of visual and textual trajectory representations using CLIPT's MLP_{vis} and MLP_{txt} respectively. We then compute the cosine similarity between every possible pairing, for a total of 256^2 similarity scores. We then evaluate in two ways. First, we visualize the scores in a similarity matrix. Second, we compute the *top-k accuracy* of CLIPT, that is, the proportion of times that the correctly paired visual representation is in the top-k visual representations ranked by similarity to a given textual representation. We perform this evaluation with normal CLIPT representations as a baseline, and with masked representations as described above as an ablation and compare. Figure 4.2 and the first two rows of Table 4.3 present the results of this first check. We see that the diagonal disappears in the ablation, and that the top-k accuracy metric faces relative drops between -60% and -90%. Based on these results, we had some evidence suggesting that CLIPT was indeed behaving as intended.

Table 4.3: Top-k accuracy of CLIPT. The first two rows show the statistics when using normal and masked representations, on CLIPT trained over two phases. The final two rows shows the statistics of correctly paired (Sing.Ph.) and randomly paired (Rand.P.) representations when training in a single phase.

Top-k accuracy	$k = 1$	$k = 3$	$k = 5$	$k = 10$	$k = 20$	$k = 50$
Baseline Repr.s	0.238	0.457	0.570	0.773	0.906	0.973
Masked Repr.s	0.016	0.031	0.047	0.082	0.176	0.363
Sing.Ph. Repr.s	1.000	1.000	1.000	1.000	1.000	1.000
Rand.P. Repr.s	1.000	1.000	1.000	1.000	1.000	1.000

We perform another check, more closely concerned in verifying the necessity of training in two phases. We posit that training in a single phase, with both MLP_{vis} and MLP_{txt} instantiated from the start, would indeed result in CLIPT learning to ignore the second half of its input due to the shared first half of the input across visual and textual representations. To verify this, we train a version of CLIPT in a single phase as described above and repeat the top-k accuracy evaluation and treat this as a baseline. We then repeat training on a new instance of CLIPT, but pair the representations randomly, treating this as an ablation. More specifically, during training, we use the same start state first halves of the representations used for the baseline, but then sample random instructions for a given end state, rather than the correct pairing. During testing, we repeat the top-k accuracy evaluation using appropriately paired representations. If CLIPT is indeed ignoring the second half, we would expect performance to be unchanged between baseline and ablation. We report the result of this experiment in the final two rows of Table 4.3. We note that in both cases we achieve perfect top-k accuracy, which we explain as only possible if the model learns to ignore the second half and learn to match the (identical) first halves of the input vectors. We therefore conclude that our two-phase training setup is indeed necessary.

4.1.3 GCBC

We turn our attention to our GCBC implementation. Here, we are concerned with determining whether an acceptable performance is achievable and whether a gap forms between the performance when conditioned on visual trajectories and when conditioned on textual trajectories.

Training details

We train GCBC on the play-data trajectories comprising the CALVIN dataset, equivalent to $\sim$740 000 training samples. Each sample consists of a sequence

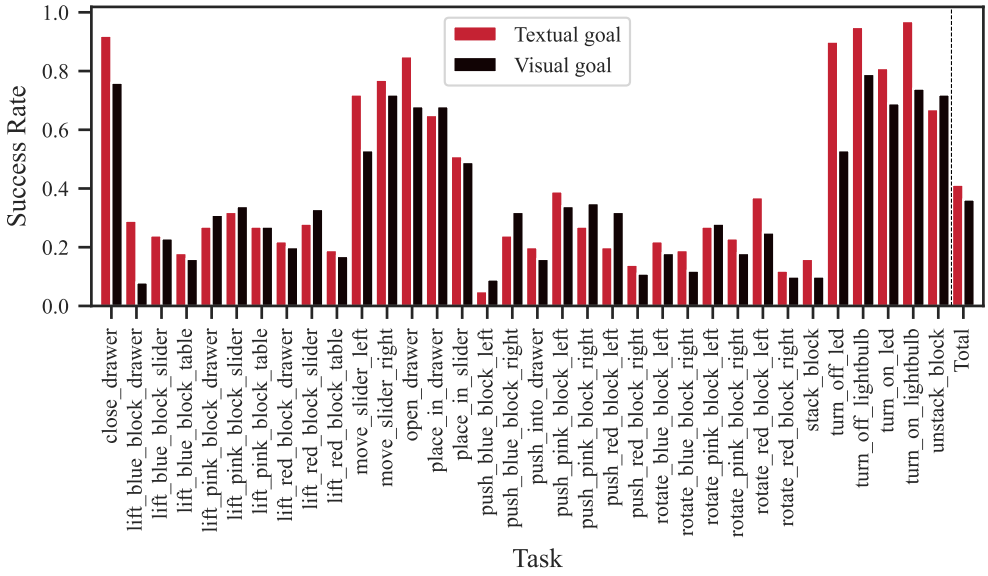

Figure 4.3: GCBC success rate on the CALVIN dataset.

of states and actions, corresponding to the trajectory of a demonstration interacting with the environment and optionally a language annotation. The sequences are of variable length, ranging between 28 and 32 frames. We train for 10 epochs, conditioning on visual trajectory representations from CLIPT. We use the checkpoint from the final epoch for evaluation. We once again use the Adam optimizer (Kingma and Ba, 2015), with a learning rate of $5 \times 10^{-}5$. Because the action space in CALVIN is a multivariate continuous action space, we use the DLML-based continuous actor module described in Section 3.3.2.

Evaluation details and results

We evaluate our GCBC checkpoints by leveraging the oracle included in the CALVIN environment. Specifically, for each of the 34 tasks in our environment we recover 100 valid starting states from our test split. For each of the 100 start states, we condition our policy to complete the task and let it interact with the environment for a maximum of 240 steps, performing a "rollout". We reset the GRU hidden state every 28 steps. At each step, we use the oracle to verify whether the task has been completed. If the task is completed within the 240 steps, we deem this interaction a success. We measure the success rate over the 100 rollouts for each task. To assess the gap between visual trajectory representations (on which the policy is trained) and textual trajectory representations (which we use at inference time), we perform this evaluation twice: once conditioning on textual trajectory representations, and the other conditioning on visual trajectory representations.

Table 4.4: Summary evaluation statistics of the GCBC experiments on CALVIN. The first two rows present the success rate (SR) statistics for our reference checkpoint, trained on CLIPT-2P representations and evaluated with context resets. The following two rows present the SR statistics when we have no context resets (NCR) during evaluation. The final row presents the SR statistics when conditioning on textual trajectory representations from CLIPT-1P, i.e. directly from CLIP's text encoder. Statistics computed across the 34-task set. Multiple seeds were not run due to computational limitations.

	mean	**median**	**min**	**max**	**std err**
Text. SR (ref)	0.41	0.27	0.05	0.97	0.05
Vis. SR (ref)	0.36	0.32	0.08	0.79	0.04
Text. SR (NCR)	0.28	0.15	0.04	0.92	0.04
Vis. SR (NCR)	0.25	0.15	0.01	0.82	0.04
Text. SR (CLIPT-1P)	0.16	0.14	0.00	0.69	0.03

Figure 4.3 and the first two rows of Table 4.4 summarize the results of the evaluation of GCBC on the CALVIN dataset. With textual trajectories, we note a mean success rate of 0.41 and a median success rate of 0.27, going as high as 0.97 for some tasks and as low as 0.05 for others. Prior work achieved a better performance of around 0.647 (Mees et al., 2022b). Direct comparisons are perhaps inappropriate however. We trained for less time due to computational limits and relied on a slightly different architecture with the use of CLIPT, training exclusively on visual goals rather than both vision and language. For context, we perform an additional evaluation to test whether the policy takes the goal conditioning into account or simply relies on the current state. We condition on random vectors rather than the representations from CLIPT, and find that the mean and median success rate drop to 0.09 and 0.04 respectively. There is a gap between the performance on visual and textual trajectories, and surprisingly the policy appears to perform better with textual trajectory representations than on visual trajectory representations, despite being trained on the latter. This gap however, is relatively small. We reserve further discussion and analysis of the results to Section 4.3.

Other experiments and results

We perform additional experiments to better understand the impact of design choices made in GCBC on the performance of the policy. We first inspect the impact of context resets during inference. We do this by repeating evaluation without context resets, and comparing the performance to the baseline where we reset the hidden state every 28 steps. We report our findings in the rows labeled "NCR" in Table 4.4. We can see that omitting context resets during

inference results in a clear drop in performance by around 30%, with the mean success rate dropping from 0.41 to 0.28 when conditioning on textual trajectory representations, and from 0.36 to 0.25 when conditioning on visual trajectory representations. Similar drops are also noted on the median values. The gap between visual and textual performance remains small however. We see that, perhaps unsurprisingly, our GRU-based architecture is sensitive to the sequence lengths longer than what it was trained on, but that this has no effect on whether it handles textual and visual trajectory representations differently.

To better assess the importance of our two-phase training, we repeat evaluation using the CLIPT checkpoint just before the start of the second phase of training. This is when MLP_{vis} has been trained, but MLP_{txt} does not exist yet, such that the textual trajectory representations come directly from CLIP's text encoder. We report our findings in the final row of Table 4.4. Note that the performance when conditioning on visual trajectory representations is unchanged, so we refer to the second row of the table for that. We see that the gap in performance when conditioning on textual trajectories as opposed to visual trajectories is now large, with a mean SR of 0.16 ± 0.03 when conditioning on natural language as opposed to a mean SR of 0.36 ± 0.04 when conditioning on visual goals. This underlines the importance of our two-phase training of CLIPT in allowing us to train on visual trajectory representations and swapping these for textual trajectory representations at inference time.

4.2 Addressing goal misgeneralization

We now turn to the problem of addressing GMG. We rely on a more manageable environment and dataset platform, and construct a toy-scenario for observing the phenomenon.

4.2.1 The BabyAI platform

Introduced by Chevalier-Boisvert et al. (2022), BabyAI is a platform originally designed for the study of grounded language learning via human-in-the-loop experiments. It consists of a collection of 19 grid-world environments[5] of various difficulty, based on the MiniGrid and Gymnasium libraries (Brockman et al., 2016; Chevalier-Boisvert et al., 2023; Towers et al., 2023). Each environment corresponds to a class of tasks, such as moving to an object, placing an object next to another, unlocking doors and more. Distractors, i.e. objects unrelated to the task at hand, may be placed in the environment to complicate the completion of the task. Any instantiation of a BabyAI environment scene will come with an associated language instruction in "Baby Language", a combinatorially rich subset of English unambiguously understood by humans.

[5] Also referred to as "levels".

go to the red ball

open the door on your left

put a ball next to the blue door

open the yellow door and go to the key behind you

put a ball next to a purple door after you put a blue box
next to a grey box and pick up the purple box

Figure 4.4: Example Baby Language instructions. Figure and examples courtesy of Chevalier-Boisvert et al. (2022).

We present example Baby Language instructions in Figure 4.4 and a visual example of the environment state in Figure 2.3. The environment state is represented as a $7 \times 7 \times 3$ egocentric partial observation, encoding the state of the 7×7 square of grid cells in front of the agent. A fully-observable RGB encoding of the environment is also available through MiniGrid wrappers, which is what we use for our work, setting the image resolution to 224×224. Agents in the BabyAI ecosystem additionally perceive a one-dimensional proprioceptive state variable, encoding the direction they are facing. This is encoded using numbers from 0 to 3, corresponding to right, down, left and up. Unlike CALVIN, the action space in in BabyAI environments is discrete and of a single dimension. Specifically, there are 7 actions, encoded as numbers from 0 to 6, corresponding to turn left, turn right, move forward, pick up an object, drop an object, toggle (used e.g. for opening doors) and done.

The platform additionally comes with a bot and verifier, respectively capable of solving any validly specified task and verifying whether a given task has been solved. These are particularly useful, as they can be used to generate demonstrations for imitation learning and offline RL solutions. To this end, we generate two datasets, *BabyAIPlay* and *BabyAISmall*. The former consists of demonstrations of the bot completing tasks from a wide range of possible environments. The latter consists of demonstrations of the bot completing a subset of 6 well-defined tasks from *BabyAIPlay*, each corresponding to a specific configuration of custom-made "GoToObject" and "GoToColor" environments. We refer to Table 4.5 for a more complete overview of the environments in *BabyAIPlay* and *BabyAISmall*. We generate 700 000 training samples and 40 000 validation samples for each dataset, roughly equivalent to 14 GB of compressed data. Like CALVIN, each sample consists of a sequence of states and actions corresponding to a demonstration trajectory of the bot interacting with the environment. The sequences are of varying length, with an average length of 7 frames. Each sample is additionally annotated with the relevant Baby Language instruction and the environment name. Sparse

Table 4.5: An overview of the BabyAI environments, grouped by the datasets they appear in. The first grouping corresponds to the environments used in *BabyAIPlay*, while the second grouping corresponds to the environments used in *BabyAISmall*. The number in parentheses corresponds to the number of tasks associated with each environment.

Environment name (task no.)	**Description**
CustomGoToObj{K/Bx/Bl} (3)	go to the {key/box/ball}, color is irrelevant. 3 distractors.
CustomGoToColor{R/G/B} (3)	go to the {red/green/blue} object, type is irrelevant. 3 distractors.
Custom-GoToObj (1)	go to a {key/box/ball}, color is irrelevant. Random number of distractors
Custom-GoToColor (1)	go to the {red/green/blue} object, type is irrelevant. Random number of distractors.
BabyAI-GoToObj (1)	go to the {color} {object}. No distractors.
BabyAI-GoToLocal (1)	go to a/the {color} {object}. 7 distractors.
BabyAI-PickupDist (1)	pick up a/the {color} {type}. 7 distractors.
BabyAI-PickupLoc (1)	pick up a/the {color} {type} {infront of you/to your right/to your left}. 7 distractors.
BabyAI-PutNextLocal (1)	put the {color} {type} next to the {color} {type}. 7 distractors.
CustomGoToObj{K/Bx/Bl} (3)	go to the {Key/Box/Ball}, color is irrelevant. Three distractors.
CustomGoToColor{R/G/B} (3)	go to the {Red/Green/Blue} object, type is irrelevant. Three distractors.

reward is also available for each sample, equivalent to $1 - 0.9n/n_{max}$, where n is the length of the episode and n_{max} is the maximum episode length as defined internally by each environment.

4.2.2 Goal misgeneralization setup

We choose to focus on BabyAI for our treatment of GMG in part due to the platform's simplicity, but also due to the flexibility and the ease of development offered by the associated ecosystem. Before generating our *BabyAISmall* dataset, we wrap each environment with custom-made wrappers that enforce the spurious correlation between the defining variables of two of our 6 tasks.

Specifically, we generate our data such that keys are always red, and red objects are always keys. In this way, under our hypothesis that GMG is caused by CC, when training on this dataset containing this spurious correlation, we maximize the chances of GMG occurring between the associated tasks. To evaluate for the presence and extent of GMG, at test time we let the policy act in an environment where the property of being a key and the property of being red are not correlated, i.e. keys are not necessarily red and red objects are not necessarily keys. We condition the trained policy on our intended goal, and evaluate the success rate of the policy on both the intended and confounding goal. We specifically define going to the key as our intended goal, and going to the red object as the confounding goal. We choose this setup rather than the complimentary one, where intended and confounding goal are swapped. This is because we expect that, in the presence of confounded variables and the absence of more expressive task specification, Occam's razor will dictate which of the two variables to causally model for the perceived policy success. In other words, we expect the policy to learn the easier variable by default, which in this case we expect to be the property of being red (a single color channel) rather than the property of being a key (a non-arbitrary mixture of shapes and edges).

4.2.3 Training details

Our training setup remains mostly unchanged for CLIPT (see Section 4.1.2), using data from our *BabyAIPlay* dataset instead. We train in two phases on a random subset of 70 000 samples, and validate on 4 000 samples, equivalent to roughly 10% of our total available dataset size. Having pre-trained CLIPT, we then train GCBC on the causally confused *BabyAISmall* dataset. We follow the same training details as when we trained GCBC on CALVIN. However, because the action space in BabyAI is discrete, we use the discrete actor module described in Section 3.3.1. We also train an RCBC policy with the same hyperparameters, to serve as a baseline when evaluating for GMG. As per the implementation outlined in Section 3.2.1, when training RCBC, we multiply the reward by a 6-dimensional one-hot vector for specifying the desired task.

4.2.4 Evaluation details and results

We evaluate for GMG by letting the trained policies interact in the CustomGoToObjK environment, conditioning, as intended by the environment, to go to the key. We specifically wrap the environment such that, now in test time, the key is never red, and that there is always one non-key distractor that is red. In this way, we allow for the possibility of the policy to pursue the confounding goal of going to the red object, which was spuriously correlated with going to the key during training. We leverage the BabyAI bot and verifier to help us in the evaluation. Specifically, we generate 1000 seeds, and let the

Table 4.6: Success rate (gap) on the intended and confounding goals by our GCBC and RCBC policies. We evaluate in a decorrelated environment, where keys are not red and red objects are not keys, and in a causally confused environment, identical to the training set, where the keys are always red and red objects are always keys. For GCBC, we report SR(G) when conditioning on visual trajectory representations (vis) and when conditioning on textual trajectory representations (txt). Standard error calculated over 3 trained seeds.

		SRG	$\mathbf{SR_{IG}}$	$\mathbf{SR_{CG}}$
Decorrelated environment	**RCBC**	-0.74 ± 0.02	0.102 ± 0.005	0.84 ± 0.02
	$\mathbf{GCBC_{vis}}$	-0.48 ± 0.08	0.22 ± 0.05	0.70 ± 0.06
	$\mathbf{GCBC_{txt}}$	-0.59 ± 0.04	0.17 ± 0.03	0.76 ± 0.02
Causally conf. environment	**RCBC**	0.000 ± 0.002	0.996 ± 0.002	0.996 ± 0.002
	$\mathbf{GCBC_{vis}}$	0.000 ± 0.005	0.988 ± 0.003	0.988 ± 0.003
	$\mathbf{GCBC_{txt}}$	0.000 ± 0.004	0.989 ± 0.003	0.989 ± 0.003

bot solve the environment for each seed, so that we can obtain the visual goal (consisting of the agent facing the key in an adjacent tile) to condition our policy with. The environment instance provides the language instruction for textual conditioning. Then, for each seed, we let the policy interact with the environment for a maximum of 100 steps, resetting the hidden state every 7 steps, and use the verifier to determine whether the intended goal or the confounding goal have been completed, keeping track of the success rate of both throughout. We repeat this evaluation for both GCBC and RCBC, and compare the success rates on the intended and confounding goal between the two policies. We additionally repeat the entire evaluation process on an environment identical to the training environment, i.e. where keys are always red and red objects are always keys, to serve as an additional baseline.

We measure the presence and extent of GMG by inspecting the gap between intended goal success rate and confounding goal success rate. More specifically, we define the success rate gap (SRG) metric simply as:

$$\text{SRG} = \text{SR}_{\text{IG}} - \text{SR}_{\text{CG}}, \tag{4.1}$$

where SR_{IG} is the success rate measured on the intended goal and SR_{CG} is the success rate measured on the confounding goal. Based on our GMG definition, a positive SRG indicates that GMG is not occurring, a negative SRG indicates that GMG is occurring, while an SRG of 0 does not lead to any definitive conclusions about whether GMG is occurring or not. While the sign of the SRG indicates the presence of GMG, the magnitude will indicate the extent of GMG, i.e. how frequently our policy generalizes or misgeneralizes. We report our SRG along with the underling SR values in Table 4.6. We firstly note that both RCBC and GCBC achieve a performance close to 100%

on unseen environments from the same distribution of the training set, where keys are always red and red objects are always keys. Because the intended and confounding goal are spuriously correlated here, as expected we observe a SRG of 0. To discuss how the policies do with regard to GMG, we turn our attention to the decorrelated environment, where keys are not red and red objects are not keys. Here, we note that both RCBC and GCBC policies suffer from GMG, with the SRG always below 0. However, we also note that conditioning on goals appears to help, with the extent of GMG being lower with GCBC (magnitudes of 0.48 and 0.59 for visual and textual conditioning respectively) than with RCBC (a much higher magnitude of 0.74).

4.3 Discussion

The results described above are mixed. On the one hand, we have demonstrated that, to some extent, GCBC learns instructable policies on a challenging benchmark. However, we find that this implementation still suffers from GMG, despite the hypothetical improvement in task specification expressiveness, which we expected to nullify the problem. That being said, we do note the silver lining that the extent of GMG is lower with GCBC than with RCBC, suggesting that perhaps the expressiveness of the task specification does indeed help. We now offer some discussion around observations we made throughout the work, the limitations and avenues for future work.

4.3.1 Multi-taskedness and Occam's razor

We note that GMG hinges on the designer's choice of intended goal. Indeed, repeating our evaluation conditioning instead on going to the red object (such that the confounding goal is now going to the key), "eliminated" GMG, inverting the sign of our SRG values. We hypothesize that, in absence of clear specifications, the policy will default to the easier task by Occam's razor, so if our intended goal is easier, then GMG will not occur in a certain sense. We observed similar results in earlier experiments where our multi-task setup was not fully sound: here, we had trained on a mixture of tasks consisting in permutations of the "go to {color} {type}" task class. During training, we artificially enforced a spurious correlation such that red balls always appeared in the bottom right grid cell of the environment. We then defined going to the red ball as our intended goal and going to the bottom right as our confounding goal, drawing inspiration from Langosco et al. (2022)'s CoinRun example. However, going to the bottom right was not explicitly in our multi-task mixture, which instead only contained tasks concerned with navigating to a certain object. We posit that the distribution of tasks in the multi-task mixture was very suggestive of what the causally confused task was, making it easy for our policies to learn to navigate to the red ball rather than to the bottom right when choosing which of the two spuriously correlated variables to causally

model for its perceived performance. Indeed, with this setup, GMG did not occur, with the SRG remaining positive for both RCBC and GCBC. We adjusted our experiments accordingly to ensure that our multi-task mixture was balanced in terms of similarity to intended and confounding goal, and specifically chose our intended goal to be (at least intuitively) more difficult than the confounding goal. We report these observations as they may prove valuable in future investigations on the topic of GMG and CC. One insight that we draw from these observations is that a simple solution to GMG is therefore making the intended goal as "easy" as possible for the policy. We believe there is space for further formalizing task specification under this heuristic, such as envisioning task specification a way of projecting the task space to some other space where the intended goal is "easier", by some metric, than other goals. We however leave this investigation to future work.

4.3.2 Post-hoc analysis of CLIPT's representations

To diagnose what may still be causing GMG despite our supposed improvement in specification expressiveness, we performed a post-hoc analysis of the representations provided by CLIPT, on which we condition our GCBC policy. Our reasoning is that the ideal representations should have disentangled color and object-type dimensions, such that the policy can leverage this structure to appropriately generalize from the non-confounded goal (e.g. going to the ball) in the mixture to the intended goal (going to the key), despite its spurious correlation with the confounding goal (going to the red object). To this end, we visually inspect the representations from CLIPT for this structure. Specifically, we uniformly sample 1000 textual and visual trajectory CLIPT representations from each of our tasks in the *BabyAISmall* mixture, allowing for instances of non-red keys and non-key red objects. We repeat this for both our two-phase and single-phase training versions of CLIPT (which we refer to as CLIPT-2P and CLIPT-1P), where in the latter case the textual representations come directly from CLIP's text encoder. We fit a UMAP (McInnes et al., 2018) dimensionality reduction model on the representations, mapping to 2 dimensions for our visualization, and report the results in Figure 4.5. We note that clear color and object-type dimensions are not immediately discernible. For CLIPT-1P representations, we notice some variation of color along the vertical axis, while for CLIPT-2P, we see a relatively neatly sorted cluster of colors in the top center portion of the plot. However, this is not clearly disentangled from object-type, with, for instance, keys appearing semantically closer to the colors than to other objects in the projected vector space. This effect is more pronounced in the textual trajectory representations from CLIPT-1P, where the clusters are very well defined, while in for the other configurations there is some overlap between clusters. We suppose that this less-than-ideal structure of our representations may in part explain

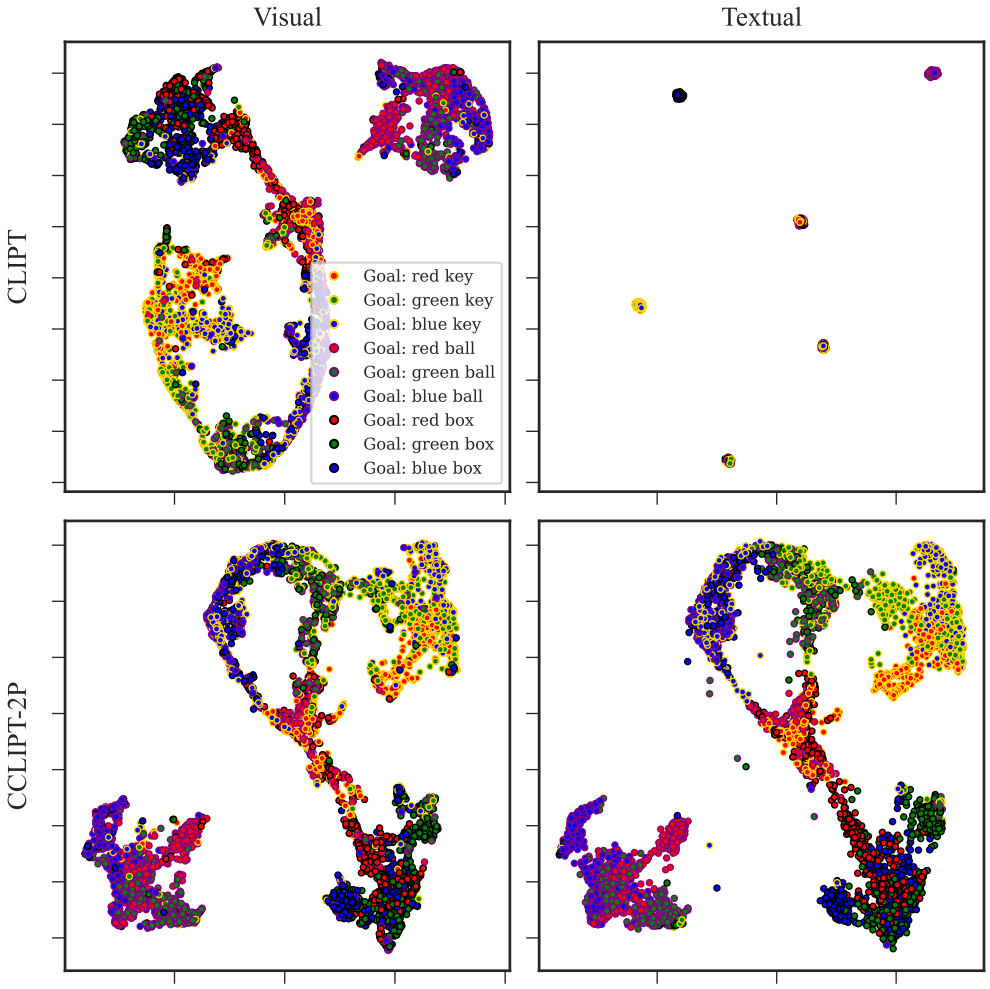

Figure 4.5: Two-dimensional visualization of CLIPT representations obtained using UMAP. We visualize both visual (left column) and textual (right column) trajectory representations. We source the representations from both our single-phase (first row) and two-phase (second row) training variants of CLIPT (CLIPT-1P and CLIPT-2P respectively). We label our representations based on the goal object they are specifying. We use the fill color to indicate the color of the object, and the outline color to identify the object type (yellow for key, black for box, purple for ball).

why GMG still persists even when conditioning on visual and textual goals rather than rewards.

In this theme, we conducted two more experiments. First, to probe the effect of the more well defined clusters of the textual trajectory representations of CLIPT-1P, we trained a "multimodal" variant of GCBC, using CLIPT-1P representations. More specifically, during training, for each batch we randomly

Table 4.7: Success rate (gap) on the intended and confounding goals by our multimodal GCBC policy (GCBC^{MM}). Standard error calculated over 3 trained seeds.

	SRG	$\mathbf{SR_{IG}}$	$\mathbf{SR_{CG}}$
$\mathbf{GCBC^{MM}_{vis}}$	-0.53 ± 0.06	0.20 ± 0.03	0.73 ± 0.05
$\mathbf{GCBC^{MM}_{txt}}$	-0.72 ± 0.02	0.119 ± 0.009	0.83 ± 0.02

choose whether to condition on visual trajectory representations or textual trajectory representations. We then repeated evaluation as described in Section 4.1.3. We report the SRG and SR values in Table 4.7. Comparing to our default implementation of GCBC from Table 4.6, we see that GMG not only is still present, but also more pronounced. This may be explained by the fact that, although more precisely defined, the textual trajectory representations from CLIPT-1P still do not properly disentangle color and object-type dimensions, potentially preventing the model from leveraging any sort of structure to avoid causal confusion and GMG.

We additionally experiment with increasing the variance of our natural language instructions. By default, we relied on the Baby Language instructions, which were limited in vocabulary and structure to the form "go to the (color) object" (and very slight variations depending on the task). Because we do not actually rely on Baby Language, we increase the natural language instruction variance by randomly paraphrasing the instructions using the substitute maps summarized in Table 4.8. More specifically, we parse the words in the Baby Language instructions and randomly substitute them with valid substitutes that convey the same or similar semantic meaning. For example, "go to the red ball" could be replaced with "navigate to the scarlet sphere". We then retrain CLIPT and repeat our representation visualization experiment described at the beginning of this section. We report the results in Figure 4.6, finding that while the variability unsurprisingly widens the clusters in CLIPT-1P textual trajectory representations, the overall results are similar to our default approach: while there is some presence of what one may refer to as a color dimension, this is entangled with our object representations, with, once again, keys being more similar to colors than to other object types.

4.3.3 Other considerations and Future work

Our treatment of the CLIPT representations, while potentially interesting, generally does not rule out other explanations for why GMG is still occurring with our GCBC policy. We note that our policy architecture, while perhaps somewhat complex on first glance, is ultimately rather simple, with most of the parameters defining our single layer GRU policymaker module. Due to computational limitations, we did not perform hyperparameter optimization

Table 4.8: Expression to substitute mapping for our paraphrasing purposes.

Expression	**Available substitutes**
"go to"	"move to", "navigate to", "proceed to", "advance to", "make your way to", "head to"
"pick up"	"pick up", "grab", "take", "get", "remove", "collect"
"blue"	"azure", "sapphire", "cyan"
"green"	"emerald"
"red"	"scarlet", "crimson", "ruby", "carmine", "vermilion"
"purple"	"violet", "lavender", "lilac"
"yellow"	"gold", "amber", "lemon", "mustard", "ochre"
"grey"	"gray", "silver"
"ball"	"sphere", "orb", "globe", "circle", "marble"
"box"	"cube", "cuboid", "chest", "crate", "square", "rectangle"
"key"	-
"object"	"object", "thing", "item"

or devote much effort to exploring better architectures, such as the now very popular Transformer architecture (Vaswani et al., 2017). It is therefore unclear whether our policy simply does not have the capacity to leverage the expressiveness of the CLIPT representations we condition it on. On this note, we are unsure if our modification of CLIPT is detrimental to our needs: we modify CLIP by adding MLP heads which we train in two phases on relatively small datasets – it is entirely possible that in doing so we overfit and lose the expressiveness afforded by the original foundation model. Similarly, it remains unclear whether GMG is primarily alleviated due to the supposedly improved task specificiation or due to the de-correlated pretraining of CLIPT. Nevertheless, we believe that our underlying thesis, that GMG can be addressed by improving task specification, is valid, and we hope future work will devote some attention to formulating better implementations for approaching the problem from this perspective. However, we do not envision LISDM to be the silver bullet for GMG. For instance, we expect future work to be able to show that the use of language may also exacerbate GMG, for example by referring to colors using ambiguous language such as "acquamarine", "turquoise" and "indigo" that instead of "blue". These colors generally fall within the spectra of "blue" and other more commonly used colors such

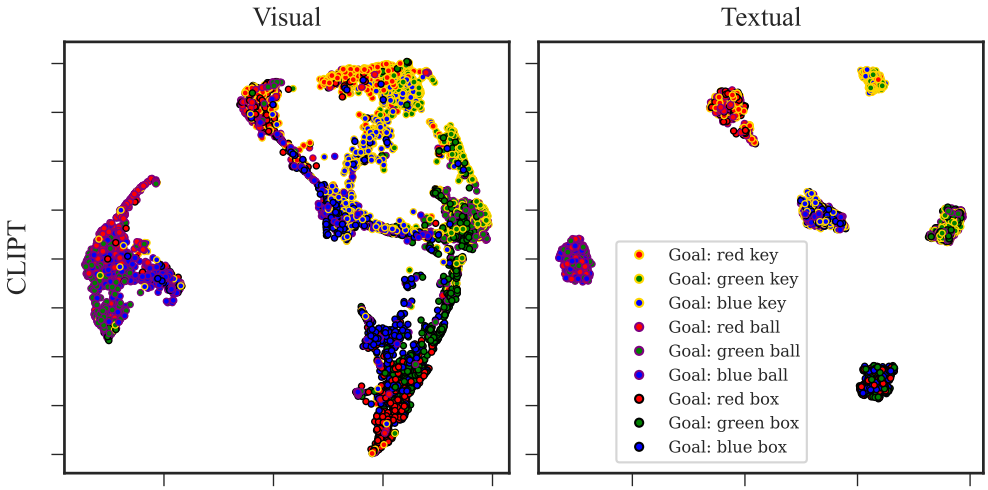

Figure 4.6: Two-dimensional visualization of CLIPT-1P representations obtained using UMAP. This particular CLIPT model was trained on paraphrased natural language instructions, to increase data variance. We visualize both visual (left column) and textual (right column) trajectory representations. We label our representations based on the goal object they are specifying. We use the fill color to indicate the color of the object, and the outline color to identify the object type (yellow for key, black for box, purple for ball).

as "green" or "red", but their exact hue, saturation, and brightness may vary between individuals' interpretation, leading to potential ambiguity.

Future work may also wish to devote more attention to the grounding problem. Indeed, while our approach of using a multimodal foundation models does address the issue, it only does so in part. The use of multimodal models will limit the scope of applications to the modalities handled by the model. In our case, that means we can only handle cases in which the environment state is represented with RGB images. While this is a reasonable assumption for many applications, it is certainly not universal. Additionally, our approach mostly focuses on the grounding between natural language instructions and environment state, with very little attention devoted to action abstractions, also known as *options* (Sutton et al., 1999). We are curious to see work focusing on the grounding between language and action abstractions, similar to Garg et al. (2022) and Jiang et al. (2019), and its applications to GMG. We also believe that future work may generally be necessary in the area of causal representation learning (Schölkopf et al., 2021) and reasoning (Yu et al., 2023; Zhang and Sridharan, 2022), as it is unclear, particularly from our analysis of the CLIPT representations, whether purely associative learning will efficiently achieve the structure in the representations necessary for avoiding causal confusion and goal misgeneralization.

Finally, we believe that future work may benefit from additional revisions to the definition of GMG. Indeed, we view all existing definitions of the phenomenon, including ours, to be limited in some way. We envision that empirical work will benefit tremendously from improved definitions of the term, perhaps devoting more attention to its relation to Occam's razor using elements from statistical learning theory. Aided by this, future work could devote some effort in determining whether GMG occurs in more complex and realistic scenarios, with potentially more dangerous confounding goals, given that existing work so far has mostly been able to demonstrate the phenomenon in innocuous toy environments. We believe that such a demonstration (or lack thereof) would provide a lot of value in terms of calibrating researchers on what problems to focus on.

CHAPTER 5

Related Work

5.1 Natural Language and Sequential Decision Making

Fueled by the Transformer architecture (Vaswani et al., 2017), advancements in natural language processing (Brown et al., 2020; Devlin et al., 2019; Vaswani et al., 2017) are now finding themselves in other domains of ML (Dhariwal et al., 2020; Radford et al., 2021; Rombach et al., 2022) as the field becomes increasingly multi-modal (Baltrušaitis et al., 2019; Xu et al., 2022) and the boundaries between paradigms fade. Moreover, the natural language interface afforded by these models provides a means for leveraging expressive and flexible user input, leading to advancements in "prompt engineering" for in-context learning (ICL) (Dohan et al., 2022; Dosovitskiy et al., 2022; Hertz et al., 2022; Reynolds and McDonell, 2021; Wei et al., 2022). On this note, recent work in SDM has also explored the integration of natural language in existing or new SDM methods (Luketina et al., 2019) and (more famously) viceversa (Ouyang et al., 2022). This work focuses on the former case. Here, Google's SayCan (Ahn et al., 2022) and DeepMind's Gato (Reed et al., 2022) are the current reference works, the first utilizing language models for planning robotics tasks and the latter leveraging a multimodal transformer to perform decision-making aided by language and other modalities. Jiang et al. (2022a)'s VIMA presents a similar approach to Gato, devoting more attention to the design of multimodal prompts. Moving away from purely transformer-based solutions, CLIPort (Shridhar et al., 2021) use CLIP (Radford et al., 2021) and Transporter (Zeng et al., 2021) in a two-stream architecture for language-specified manipulation tasks using imitation learning. More distantly, Zhou and Small (2021) attempt to address generalization issues when using NL goals by proposing a sample-efficient inverse reinforcement learning algorithm based on latent goal relabeling (Nair et al., 2018). Choi et al. (2022) also

tackle generalization through their LMPriors framework which shapes the reward (Ng et al., 1999) based on priors computed by a generative language model. Instead of providing input instructions or feedback, Lampinen et al. (2022) take a different approach by making their agents produce NL explanations as part of their output, which is then used in an additional loss term for learning causal structure and OOD generalization. Yang et al. (2021) focus on developing an operationally safe RL algorithm by proposing a modular *constraint interpreter* capable of mapping NL constraints to spatial and temporal representations of hidden states. DeepMind et al. (2022) and Fan et al. (2022) leverage contrastive vision-language training to address the issue of grounding language representations to the environment, while Watkins et al. (2021) proposes a novel bootstrapping solution for grounding. Sumers et al. (2021) explores reward learning using open-ended linguistic feedback. The same authors contribute formalizations of the type of language a modeled speaker could use for preference expression (Sumers et al., 2022) and most recently by extending hindsight experience replay (HER) (Andrychowicz et al., 2017) to a language-conditioned setting using generative visual-language models (VLM) (Alayrac et al., 2022). A number of language-annotated datasets are available for research in this area (Fan et al., 2022; Jiang et al., 2022b; Liu et al., 2022; Mees et al., 2022b; Shridhar et al., 2020; Zholus et al., 2022). However the field is still clearly in early stages and has yet to settle on a particular direction. To our knowledge, this work is the first to investigate the effects of NL on goal misgeneralization in SDM.

5.2 Causal Confusion and Goal Misgeneralization

The issue of causal confusion was first identified and defined by de Haan et al. (2019) in the context of imitation learning. They address the issue by learning a graph-parametrized policy for each possible causal graph and subsequently performing targeted interventions to select the best policy. Tien et al. (2022) later successfully identify the same phenomenon in the context of preference-based (Christiano et al., 2017) inverse reinforcement learning (IRL) (Ng and Russell, 2000). Concurrently, Gupta et al. (2022) identify causal confusion in the more relevant context of offline RL, and explore active sampling as a means to mitigate the issue. While these works focus mainly on capability failures, Kirk and Krueger (2022) also recognize goal misgeneralization and incentive mismanagement (Farquhar et al., 2022) as two additional failure modes, where we focus on the former of the two. Langosco et al. (2022) are the first to formally define goal misgeneralization based on Orseau et al. (2018)'s definition of agency. The authors demonstrate the phenomenon in a number of RL agents trained on the Procgen (Cobbe et al., 2020) benchmark, proposing increased training data diversity as a means of alleviating the issue. Shah et al. (2022) later generalise the definition, removing the assumed RL framework

necessary in Langosco et al. (2022)'s formalization, and demonstrating the issue in a variety of new settings. Aside from more diverse training data, the authors suggest uncertainty-aware models, better inductive biases and techniques targeting deception as potential routes for mitigation. The issue of goal misgeneralization draws parallels with *reward hacking* (Pan et al., 2022; Skalse et al., 2022) and *reward tampering* (Everitt et al., 2021) where there is a misalignment between the designers intended behaviour and the algorithm's behaviour due to misspecified rewards. This misalignment makes goal misgeneralization of particular interest to research in Artificial Intelligence (AI) alignment (Ngo et al., 2022) and AI safety more broadly (Hendrycks et al., 2022; Houben et al., 2022).

5.3 Representation Learning and Foundation Models

Our approach of working on the representations produced by a pre-trained multimodal model such as CLIP (Radford et al., 2021) is closely related to the field of *representation learning*. This is the study of the processes for learning transformations of raw input data into more abstract representations, typically in the form of dense vectors. The idea is that the more abstract nature of the learned features can lead to better generalization, for usefulness and applications on a wide range of downstream tasks, omitting the need for less transportable hand-engineered features. Now a ubiquitous component of modern-day ML, in 2013 Bengio et al. identify the paradigm-shift from feature engineering to representation learning. The authors define desiderata for "good representations", such as local smoothness, hierarchically-organised explanatory factors shared across tasks and a sparse activation for a specific input. They compare representation learning in the context of probabilistic graphical models (PGM) and neural networks and outline connections to manifold learning and invariance (Bronstein et al., 2021). More recently, Liu et al. (2020b) write a book covering representation learning in NLP, while Xie et al. (2020) provide a statistical perspective, linking representation learning to factor analysis (Rubin and Thayer, 1982) and multidimensional scaling (Kruskal, 1964). Readers of our work may find the ideas of Schölkopf et al. (2021) of particular interest. Here, the authors relate fundamental concepts of causal inference to representation learning, defining *causal representation learning* as the problem of discovering high-level causal variables from low-level observations. Lopez-Paz et al. (2017) work precisely on this issue of causal discovery in the context of objects in images, leveraging proxy variables for identifying causal relationships between entities in images. In the context of SDM, Lan et al. (2022) study the generalization of state representations in RL, relying on the notion of "effective dimension". They provide a bound on the generalization error that arises when using a given k-dimensional representation.

They demonstrate the usefulness of the bound in the context of successor representations (Dayan, 1993). Closely related to CLIP, Guo et al. (2019) and Le-Khac et al. (2020) provide surveys covering multimodal and contrastive representation learning respectively. The latter builds on the aforementioned work from Bengio et al. (2013) and identify core principals for learning good representations: distributedness, abstraction, invariance and disentanglement. Much modern work in the field points to foundation models and the use of their representations for downstream tasks (Bommasani et al., 2021; Zhou et al., 2023a). While originating mostly in the field of NLP (Brown et al., 2020; Devlin et al., 2019; Touvron et al., 2023), attention has increasingly included or shifted to other modalities. Here, CLIP (Radford et al., 2021) is the most prominent example. Leveraging a carefully curated dataset of image-caption pairs and a contrastive loss, CLIP learns semantically similar representations across text and vision symbols. Similar to CLIP, ALIGN (Jia et al., 2021) relaxes the need for careful curation of the dataset, instead relying on a noisier but larger dataset of images and alt-text descriptions. They demonstrate that expert knowledge is not necessary for dataset curation in contrastive learning. BEIT-3 (Wang et al., 2022) and M3AE (Geng et al., 2022) take a different approach: instead of training an encoder for each modality and ensuring that the representations from each are similar, they train a single encoder capable of handling both modalities, avoiding the need for paired data and hence enabling larger scale training. FLAVA (Singh et al., 2022a) operates in a similar vein, utilizing a single model for multimodal representations. It however also includes cross-modal contrastive objectives, aiming for performance on image-only, text-only and image-text downstream tasks with the same single model. Florence (Yuan et al., 2021) expands the dimensions covered by their representations, covering for instance both static (images) and dynamic (videos) inputs, or coarse (scene) and fine-grained (object) tasks.

5.4 Grounding

The work by Harnad (1990) typically acts as the starting reference for most work on the symbol grounding problem. In machine learning and linguistics, the focus is typically on grounding in the context of of natural language understanding (NLU) and meaning (Winograd, 1972). Clark and Brennan (1991) and DeVault et al. (2006) are examples of early work in this theme. A commonly held perspective is that referents from other modalities are a necessary ingredient for grounding language. Mooney (2008) make early connections to perception of visual symbols as a means of grounding natural language for NLU. Contemporary to their influential work on distributional semantics (Baroni et al., 2014a,b), Bruni et al. (2014) propose *multimodal* distributional semantics to address the lack of perceptual grounding of distributional models. In particular, they leverage computer vision techniques to

extract "visual words" from images and incorporate these into the training data such that the distributional representations can capture these references. More recently, some focus has been dedicated to comparing how grounded language and meaning is learned in humans and in machines (Lake and Murphy, 2023). Linzen (2020) touch on the subject in their position paper critiquing the current paradigm for NLU evaluation. They argue, among other things, that humans do not learn language from text alone, but additionally through their experience of the world, and that more careful attention should be dedicated to this direction. In parallel, Bisk et al. (2020) make similar claims around the usefulness of world experience for linguistic grounding and NLU. They propose the notion of "World Scopes (WS)" as a framework for auditing progress in NLP. Bender and Koller (2020) outline how current state-of-the-art techniques are limited in their ability to acquire meaning, due to the lack of grounding referents. Piantadosi and Hill (2022) directly respond to this critique, offering an alternative explanation for how grounding and meaning can be achieved through the lens of conceptual role theory (Block, 1998).

CHAPTER 6

Conclusion

In this work, we presented a first attempt at addressing the issue of goal misgeneralization by focusing on the improvement of task specification through the use of natural language. We devoted our attention to sequential decision making, and provided a preliminary formalization of a task specification framework. Here, we identified requester and executor parties, as well as the notion of a latent specification representation, corresponding to some high-level abstraction of the desired trajectory. Concurrently, we provide a new definition of the goal misgeneralization phenomenon, and are the first to explicitly frame it in the context of multi-task learning and to link it to Occam's razor. We developed our own implementation of a possible solution, first demonstrating its applicability on a challenging benchmark, and later tackling a simpler environment platform for toy scenarios of goal misgeneralization. Here, we showed that, under our implementation, natural language appears to decrease the extent of goal misgeneralization, but nevertheless fails to completely eliminate the problem. We performed some diagnostic experiments to understand possible failure modes, and provide some discussions around current limitations and potential future work.

Bibliography

Michael Ahn, Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T. Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. 2022. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. In *6th Annual Conference on Robot Learning*.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. 2022. Flamingo: A Visual Language Model for Few-Shot Learning. In *Advances in Neural Information Processing Systems*.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Roger Ariew. 1976. *Ockham's Razor: A Historical and Philosophical Analysis of Ockham's Principle of Parsimony*. Ph.D. thesis, University of Illinois at Urbana-Champaign, USA.

Martin Arjovsky. 2020. *Out of Distribution Generalization in Machine Learning*. Ph.D. thesis, New York University, USA.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. In *Advances in Neural Information Processing Systems*, volume 30, Barcelona, Spain. Curran Associates, Inc.

Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, Mojtaba Komeili, Karthik Konath, Minae Kwon, Adam Lerer, Mike Lewis, Alexander H. Miller, Sasha Mitts, Adithya Renduchintala, Stephen Roller, Dirk Rowe, Weiyan Shi, Joe Spisak, Alexander Wei, David Wu, Hugh Zhang, and Markus Zijlstra. 2022. Human-level play in the game of Diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2019. Multimodal Machine Learning: A Survey and Taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):423–443.

Suguman Bansal. 2022. Specification-Guided Reinforcement Learning. In *Static Analysis*, Lecture Notes in Computer Science, pages 3–9, Cham. Springer Nature Switzerland.

Marco Baroni, Raffaella Bernardi, and Roberto Zamparelli. 2014a. Frege in Space: A Program for Composition Distributional Semantics. In *Linguistic Issues in Language Technology, Volume 9, 2014 - Perspectives on Semantic Representations for Textual Inference*. CSLI Publications.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014b. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland. Association for Computational Linguistics.

Richard Bellman. 1957. *Dynamic Programming*. Princeton University Press.

Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, Nicolas Pinto, and Joseph Turian. 2020. Experience Grounds Language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8718–8735, Online. Association for Computational Linguistics.

Ned Block. 1998. Conceptual Role Semantics.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. 1987. Occam's Razor. *Information Processing Letters*, 24(6):377–380.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. 2021. On the Opportunities and Risks of Foundation Models. *arXiv:2108.07258 [cs]*.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. 2021. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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. *arXiv:2005.14165 [cs]*.

E. Bruni, N. K. Tran, and M. Baroni. 2014. Multimodal Distributional Semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

A.E. Bryson. 1996. Optimal control-1950 to 1985. *IEEE Control Systems Magazine*, 16(3):26–33.

Rich Caruana. 1997. Multitask Learning. *Machine Learning*, 28(1):41–75.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision Transformer: Reinforcement Learning via Sequence Modeling.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1597–1607. PMLR.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2022. BabyAI: A Platform to Study the Sample Efficiency of Grounded Language Learning. In *International Conference on Learning Representations*.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. 2023. Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Myungsik Cho, Whiyoung Jung, and Youngchul Sung. 2022. Multi-task Reinforcement Learning with Task Representation Method. In *ICLR 2022 Workshop on Generalizable Policy Learning in Physical World*.

Kristy Choi, Chris Cundy, Sanjari Srivastava, and Stefano Ermon. 2022. LMPriors: Pre-Trained Language Models as Task-Specific Priors. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

Paul F Christiano, 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*, volume 30. Curran Associates, Inc.

Herbert H. Clark and Susan E. Brennan. 1991. Grounding in communication. In *Perspectives on Socially Shared Cognition*, pages 127–149. American Psychological Association, Washington, DC, US.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. 2020. Leveraging Procedural Generation to Benchmark Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning*, pages 2048–2056. PMLR.

Peter Dayan. 1993. Improving Generalization for Temporal Difference Learning: The Successor Representation. *Neural Computation*, 5(4):613–624.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. 2019. Causal Confusion in Imitation Learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Interactive Agents Team DeepMind, Josh Abramson, Arun Ahuja, Arthur Brussee, Federico Carnevale, Mary Cassin, Felix Fischer, Petko Georgiev, Alex Goldin, Mansi Gupta, Tim Harley, Felix Hill, Peter C. Humphreys, Alden Hung, Jessica Landon, Timothy Lillicrap, Hamza Merzic, Alistair Muldal, Adam Santoro, Guy Scully, Tamara von Glehn, Greg Wayne, Nathaniel Wong, Chen Yan, and Rui Zhu. 2022. Creating Multimodal Interactive Agents with Imitation and Self-Supervised Learning.

D. DeVault, Iris Oved, and Matthew Stone. 2006. Societal Grounding Is Essential to Meaningful Language Use. In *AAAI Conference on Artificial Intelligence*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. 2020. Jukebox: A Generative Model for Music.

David Dohan, Winnie Xu, Aitor Lewkowycz, Jacob Austin, David Bieber, Raphael Gontijo Lopes, Yuhuai Wu, Henryk Michalewski, Rif A. Saurous, Jascha Sohl-dickstein, Kevin Murphy, and Charles Sutton. 2022. Language Model Cascades.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2022. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

Tom Everitt, Marcus Hutter, Ramana Kumar, and Victoria Krakovna. 2021. Reward tampering problems and solutions in reinforcement learning: A causal influence diagram perspective. *Synthese*, 198(27):6435–6467.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. 2022. MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge. In *Thirty-Sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Sebastian Farquhar, Ryan Carey, and Tom Everitt. 2022. Path-Specific Objectives for Safer Agent Incentives. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(9):9529–9538.

Alhussein Fawzi, 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. 2022. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53.

Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. 2016. Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519, Stockholm, Sweden. IEEE Press.

Keith Frankish and William M. Ramsey. 2014. *The Cambridge Handbook of Artificial Intelligence*. Cambridge University Press.

Divyansh Garg, Skanda Vaidyanath, Kuno Kim, Jiaming Song, and Stefano Ermon. 2022. LISA: Learning Interpretable Skill Abstractions from Language. *Advances in Neural Information Processing Systems*, 35:21711–21724.

Xinyang Geng, Hao Liu, Lisa Lee, Dale Schuurmans, Sergey Levine, and Pieter Abbeel. 2022. Multimodal Masked Autoencoders Learn Transferable Representations. In *First Workshop on Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022*.

Jindong Gu, Zhen Han, Shuo Chen, Ahmad Beirami, Bailan He, Gengyuan Zhang, Ruotong Liao, Yao Qin, Volker Tresp, and Philip Torr. 2023. A Systematic Survey of Prompt Engineering on Vision-Language Foundation Models.

E. J. Gumbel. 1954. Statistical Theory of Extreme Values and Some Practical Applications. A Series of Lectures. Technical Report PB175818, National Bureau of Standards, Washington, D. C. Applied Mathematics Div.

Wenzhong Guo, Jianwen Wang, and Shiping Wang. 2019. Deep Multimodal Representation Learning: A Survey. *IEEE Access*, 7:63373–63394.

Gunshi Gupta, Tim G. J. Rudner, Rowan Thomas McAllister, Adrien Gaidon, and Yarin Gal. 2022. Can Active Sampling Reduce Causal Confusion in Offline Reinforcement Learning? In *NeurIPS ML Safety Workshop*.

Sami Haddadin, Sven Parusel, Lars Johannsmeier, Saskia Golz, Simon Gabl, Florian Walch, Mohamadreza Sabaghian, Christoph Jähne, Lukas Hausperger, and Simon Haddadin. 2022. The Franka Emika Robot: A Reference Platform for Robotics Research and Education. *IEEE Robotics & Automation Magazine*, 29(2):46–64.

Stevan Harnad. 1990. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1):335–346.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Dan Hendrycks, Nicholas Carlini, John Schulman, and Jacob Steinhardt. 2022. Unsolved Problems in ML Safety.

Dan Hendrycks, Mantas Mazeika, and Thomas Woodside. 2023. An Overview of Catastrophic AI Risks.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. 2022. Prompt-to-Prompt Image Editing with Cross Attention Control.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training Compute-Optimal Large Language Models.

Sebastian Houben, Stephanie Abrecht, Maram Akila, Andreas Bär, Felix Brockherde, Patrick Feifel, Tim Fingscheidt, Sujan Sai Gannamaneni, Seyed Eghbal Ghobadi, Ahmed Hammam, Anselm Haselhoff, Felix Hauser, Christian Heinzemann, Marco Hoffmann, Nikhil Kapoor, Falk Kappel, Marvin Klingner, Jan Kronenberger, Fabian Küppers, Jonas Löhdefink, Michael Mlynarski, Michael Mock, Firas Mualla, Svetlana Pavlitskaya, Maximilian Poretschkin, Alexander Pohl, Varun Ravi-Kumar, Julia Rosenzweig, Matthias Rottmann, Stefan Rüping, Timo Sämann, Jan David Schneider, Elena Schulz, Gesina Schwalbe, Joachim Sicking, Toshika Srivastava, Serin Varghese, Michael Weber, Sebastian Wirkert, Tim Wirtz, and Matthias

Woehrle. 2022. Inspect, Understand, Overcome: A Survey of Practical Methods for AI Safety. In Tim Fingscheidt, Hanno Gottschalk, and Sebastian Houben, editors, *Deep Neural Networks and Data for Automated Driving: Robustness, Uncertainty Quantification, and Insights Towards Safety*, pages 3–78. Springer International Publishing, Cham.

Gabriel Ilharco, Mitchell Wortsman, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. 2021. OpenCLIP. Zenodo.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. 2021. Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pages 4904–4916. PMLR.

YiDing Jiang, Shixiang (Shane) Gu, Kevin P Murphy, and Chelsea Finn. 2019. Language as an Abstraction for Hierarchical Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. 2022a. VIMA: General Robot Manipulation with Multimodal Prompts.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li, Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. 2022b. VIMA: General Robot Manipulation with Multimodal Prompts.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez. 2022. Deep Reinforcement Learning for Autonomous Driving: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926.

Robert Kirk and David Krueger. 2022. Causal confusion as an argument against the scaling hypothesis.

A. Harry Klopf. 1972. *Brain Function and Adaptive Systems: A Heterostatic Theory.* Air Force Cambridge Research Laboratories, Air Force Systems Command, United States Air Force.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

J. B. Kruskal. 1964. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27.

Brenden M. Lake and Gregory L. Murphy. 2023. Word meaning in minds and machines. *Psychological Review*, 130(2):401–431.

Andrew K. Lampinen, Nicholas Roy, Ishita Dasgupta, Stephanie Cy Chan, Allison Tam, James Mcclelland, Chen Yan, Adam Santoro, Neil C. Rabinowitz, Jane Wang, and Felix Hill. 2022. Tell me why! Explanations support learning relational and causal structure. In *Proceedings of the 39th International Conference on Machine Learning*, pages 11868–11890. PMLR.

Charline Le Lan, Stephen Tu, Adam Oberman, Rishabh Agarwal, and Marc G. Bellemare. 2022. On the Generalization of Representations in Reinforcement Learning. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, pages 4132–4157. PMLR.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. 2012. Batch Reinforcement Learning. In Marco Wiering and Martijn van Otterlo, editors, *Reinforcement Learning: State-of-the-Art*, Adaptation, Learning, and Optimization, pages 45–73. Springer, Berlin, Heidelberg.

Lauro Langosco Di Langosco, Jack Koch, Lee D. Sharkey, Jacob Pfau, and David Krueger. 2022. Goal Misgeneralization in Deep Reinforcement Learning. In *Proceedings of the 39th International Conference on Machine Learning*, pages 12004–12019. PMLR.

Phuc H. Le-Khac, Graham Healy, and Alan F. Smeaton. 2020. Contrastive Representation Learning: A Framework and Review. *IEEE Access*, 8:193907–193934.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models.

Tal Linzen. 2020. How Can We Accelerate Progress Towards Human-like Linguistic Generalization? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5210–5217, Online. Association for Computational Linguistics.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2022. Reinforcement Learning on Web Interfaces using Workflow-Guided Exploration. In *International Conference on Learning Representations*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Computing Surveys*, 55(9):195:1–195:35.

Siqi Liu, Kay Choong See, Kee Yuan Ngiam, Leo Anthony Celi, Xingzhi Sun, and Mengling Feng. 2020a. Reinforcement Learning for Clinical Decision Support in Critical Care: Comprehensive Review. *Journal of Medical Internet Research*, 22(7):e18477.

Zhiyuan Liu, Yankai Lin, and Maosong Sun. 2020b. *Representation Learning for Natural Language Processing*. Springer Nature.

David Lopez-Paz, Robert Nishihara, Soumith Chintala, Bernhard Scholkopf, and Leon Bottou. 2017. Discovering Causal Signals in Images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6979–6987.

Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. 2019. A Survey of Reinforcement Learning Informed by Natural Language. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 6309–6317, Macao, China. International Joint Conferences on Artificial Intelligence Organization.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. 2020. Learning Latent Plans from Play. In *Proceedings of the Conference on Robot Learning*, pages 1113–1132. PMLR.

Corey Lynch and Pierre Sermanet. 2021. Language conditioned imitation learning over unstructured data. *Robotics: Science and Systems XVII*.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. 2018. UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software*, 3(29):861.

Oier Mees, Lukas Hermann, and Wolfram Burgard. 2022a. What Matters in Language Conditioned Robotic Imitation Learning Over Unstructured Data. *IEEE Robotics and Automation Letters*, 7(4):11205–11212.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. 2022b. CALVIN: A Benchmark for Language-Conditioned Policy Learning for Long-Horizon Robot Manipulation Tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334.

D. Michie, M. Bain, and J. Hayes-Michie. 1990. Cognitive models from subcognitive skills. In *Knowledge-Based Systems for Industrial Control*, pages 71–99. IET Digital Library.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2021. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106.

Marvin Minsky. 1954. *Theory of Neural-Analog Reinforcement Systems and Its Application to the Brain Model Problem.* Ph.D. thesis, Princeton University.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning.

Raymond J. Mooney. 2008. Learning to connect language and perception. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, pages 1598–1601, Chicago, Illinois. AAAI Press.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. 2018. Visual Reinforcement Learning with Imagined Goals. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. 1999. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 278–287, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Andrew Y. Ng and Stuart Russell. 2000. Algorithms for Inverse Reinforcement Learning. In *In Proc. 17th International Conf. on Machine Learning*, pages 663–670. Morgan Kaufmann.

Richard Ngo, Lawrence Chan, and Sören Mindermann. 2022. The alignment problem from a deep learning perspective.

OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. 2019. Dota 2 with Large Scale Deep Reinforcement Learning.

Laurent Orseau, Simon McGregor McGill, and Shane Legg. 2018. Agents and Devices: A Relative Definition of Agency.

Long Ouyang, 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. 2022. Training language models to follow instructions with human feedback.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. 2022. The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models. In *International Conference on Learning Representations*.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior.

Steven Piantadosi and Felix Hill. 2022. Meaning without reference in large language models. In *NeurIPS 2022 Workshop on Neuro Causal and Symbolic AI (nCSI)*.

Rafael Figueiredo Prudencio, Marcos R. O. A. Maximo, and Esther Luna Colombini. 2022. A Survey on Offline Reinforcement Learning: Taxonomy, Review, and Open Problems.

Martin L. Puterman. 2014. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

Joaquin Quinonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. 2008. *Dataset Shift in Machine Learning*. MIT Press.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of*

the 38th International Conference on Machine Learning, pages 8748–8763. PMLR.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical Text-Conditional Image Generation with CLIP Latents.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. 2022. A Generalist Agent.

Laria Reynolds and Kyle McDonell. 2021. Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis With Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695.

Donald B. Rubin and Dorothy T. Thayer. 1982. EM algorithms for ML factor analysis. *Psychometrika*, 47(1):69–76.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. 2017. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications. In *International Conference on Learning Representations*.

S. Schaal. 1999. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233–242.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. 2021. Toward Causal Representation Learning. *Proceedings of the IEEE*, 109(5):612–634.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W. Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R. Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. 2022. LAION-5B: An open large-scale dataset for training next generation image-text models. In *Thirty-Sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Rohin Shah, Vikrant Varma, Ramana Kumar, Mary Phuong, Victoria Krakovna, Jonathan Uesato, and Zac Kenton. 2022. Goal Misgeneralization: Why Correct Specifications Aren't Enough For Correct Goals.

C. E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. 2021. Towards Out-Of-Distribution Generalization: A Survey.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. 2021. CLIPort: What and Where Pathways for Robotic Manipulation. In *5th Annual Conference on Robot Learning*.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10737–10746.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144.

Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. 2022a. FLAVA: A Foundational Language And Vision Alignment Model. *arXiv:2112.04482 [cs]*.

Bharat Singh, Rajesh Kumar, and Vinay Pratap Singh. 2022b. Reinforcement learning in robotic applications: A comprehensive survey. *Artificial Intelligence Review*, 55(2):945–990.

Joar Max Viktor Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. 2022. Defining and Characterizing Reward Gaming. In *Advances in Neural Information Processing Systems*.

Kihyuk Sohn. 2016. Improved Deep Metric Learning with Multi-class N-pair Loss Objective. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Theodore Sumers, Robert D. Hawkins, Mark K. Ho, Thomas L. Griffiths, and Dylan Hadfield-Menell. 2022. How to talk so AI will learn: Instructions, descriptions, and pragmatics. In *Second Workshop on Language and Reinforcement Learning*.

Theodore R. Sumers, Mark K. Ho, Robert D. Hawkins, Karthik Narasimhan, and Thomas L. Griffiths. 2021. Learning Rewards From Linguistic Feedback. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(7):6002–6010.

Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning, Second Edition: An Introduction*. MIT Press.

Richard S. Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211.

Jeremy Tien, Jerry Zhi-Yang He, Zackory Erickson, Anca Dragan, and Daniel S. Brown. 2022. A Study of Causal Confusion in Preference-Based Reward Learning. In *ICML 2022: Workshop on Spurious Correlations, Invariance and Stability*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models.

Mark Towers, Jordan K Terry, Ariel Kwiatkowski, John U. Balis, Gianluca Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Jin Shen Tan, and Omar G. Younis. 2023. Gymnasium. Zenodo.

Peter Vamplew, Benjamin J. Smith, Johan Källström, Gabriel Ramos, Roxana Rădulescu, Diederik M. Roijers, Conor F. Hayes, Fredrik Heintz, Patrick Mannion, Pieter J. K. Libin, Richard Dazeley, and Cameron Foale. 2022. Scalar reward is not enough: A response to Silver, Singh, Precup and Sutton (2021). *Autonomous Agents and Multi-Agent Systems*, 36(2):41.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv:1706.03762 [cs]*.

Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. 2023. Diffusers: State-of-the-art diffusion models.

Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, and Furu Wei. 2022. Image as a Foreign Language: BEiT Pretraining for All Vision and Vision-Language Tasks.

Olivia Watkins, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Jacob Andreas. 2021. Teachable Reinforcement Learning via Advice Distillation. In *Advances in Neural Information Processing Systems*, volume 34, pages 6920–6933. Curran Associates, Inc.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*.

Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. 2007. Multi-task reinforcement learning: A hierarchical Bayesian approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 1015–1022, New York, NY, USA. Association for Computing Machinery.

Terry Winograd. 1972. Understanding natural language. *Cognitive Psychology*, 3(1):1–191.

Jianwen Xie, Ruiqi Gao, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. 2020. Representation Learning: A Statistical Perspective. *Annual Review of Statistics and Its Application*, 7(1):303–335.

Peng Xu, Xiatian Zhu, and David A. Clifton. 2022. Multimodal Learning with Transformers: A Survey.

Hui Yang, Sifu Yue, and Yunzhong He. 2023. Auto-GPT for Online Decision Making: Benchmarks and Additional Opinions.

Tsung-Yen Yang, Michael Y Hu, Yinlam Chow, Peter J Ramadge, and Karthik Narasimhan. 2021. Safe Reinforcement Learning with Natural Language Constraints. In *Advances in Neural Information Processing Systems*, volume 34, pages 13794–13808. Curran Associates, Inc.

Chao Yu, Xuejing Zheng, Hankz Hankui Zhuo, Hai Wan, and Weilin Luo. 2023. Reinforcement Learning with Knowledge Representation and Reasoning: A Brief Survey.

Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, Ce Liu, Mengchen Liu, Zicheng Liu, Yumao Lu, Yu Shi, Lijuan Wang, Jianfeng Wang, Bin Xiao, Zhen Xiao, Jianwei Yang, Michael Zeng, Luowei Zhou, and Pengchuan Zhang. 2021. Florence: A New Foundation Model for Computer Vision.

Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, and Johnny Lee. 2021. Transporter Networks: Rearranging the Visual World for Robotic Manipulation. In *Proceedings of the 2020 Conference on Robot Learning*, pages 726–747. PMLR.

Shiqi Zhang and Mohan Sridharan. 2022. A survey of knowledge-based sequential decision-making under uncertainty. *AI Magazine*, 43(2):249–266.

Artem Zholus, Alexey Skrynnik, Shrestha Mohanty, Zoya Volovikova, Julia Kiseleva, Artur Szlam, Marc-Alexandre Coté, and Aleksandr I. Panov. 2022. IGLU Gridworld: Simple and Fast Environment for Embodied Dialog Agents.

Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, Hao Peng, Jianxin Li, Jia Wu, Ziwei Liu, Pengtao Xie, Caiming Xiong, Jian Pei, Philip S. Yu, and Lichao Sun. 2023a. A Comprehensive Survey on Pretrained Foundation Models: A History from BERT to ChatGPT.

Li Zhou and Kevin Small. 2021. Inverse Reinforcement Learning with Natural Language Goals. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11116–11124.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, Shiding Zhu, Jiyu Chen, Wentao Zhang, Ningyu Zhang, Huajun Chen, Peng Cui, and Mrinmaya Sachan. 2023b. Agents: An Open-source Framework for Autonomous Language Agents.

APPENDIX A

More CLIPT Sanity Checks

We performed a number of additional sanity checks to guide our choices around the design of CLIPT.

A.1 Why not just CLIP?

CLIPT modifies CLIP such that there are two separate MLPs projecting concatenated representations back into the original dimension. This choice is not arbitrary. Indeed, a more comfortable choice would have been to use the representations from CLIP's vision and text encoders directly. As outlined in the previous chapter, we are motivated to make this design decision for two reasons:

1. We define a trajectory to be composed of *at least* start and end state, necessitating the need for two-image representations.

2. We believe images of our state will be slightly out of domain compared to CLIP's training mixture.

We take the first as a definitional aspect, a basic premise, for which we therefore do not envision verification as being necessary. We however do perform a quick experiment to check the validity of our second motivation. We expect the similarity between textual annotations and images of our state to be lower than the similarity between textual annotations and more "realistic" images, which are closer to the distribution of data CLIP was trained on. To test this hypothesis, we collect 8 textual annotations and 8 images of final states of the corresponding trajectories. We compute the cosine similarity between the CLIP representations of all possible pairings, for a total of 64 similarity values. We then repeat this computation, replacing the images

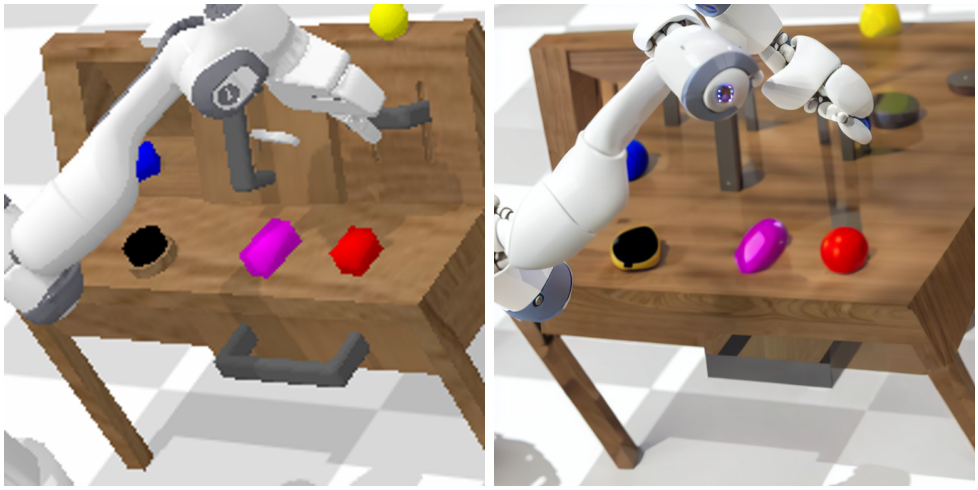

Figure A.1: Comparison of an image from the CALVIN environment and the corresponding "realistic" version obtained via latent diffusion

Table A.1: Similarity statistics between textual annotations and original vs "realistic" state images.

Cosine Similarity	Original state image	"Realistic" state image
mean	0.308	0.350
median	0.311	0.353
maximum	0.384	0.408
minimum	0.234	0.289
std. err.	0.004	0.003

with "realistic" versions obtained via an off-the-shelf text-conditioned image-to-image model. In particular, we use RunwayML's "Stable Diffusion 1.5" checkpoint of Rombach et al. (2022)'s latent diffusion model[1]. Using the HuggingFace diffusers library (von Platen et al., 2023), we set strength to 0.35, guidance scale to 50 and employ the following prompt: "*Photograph of robotic arm interacting with wooden table top. Canon 60D. Realistic, HD, 8k, ultra detailed.*" Figure A.1 presents a sample comparison between an original state image and its "realistic" version. Table A.1 instead summarizes the results of the experiment.

We find that there is a clear difference in the similarity of textual annotations and the original state images and the similarity of textual annotations and the transformed "realistic" state images. In particular, we find the latter case to result in clearly higher similarity. While far from a perfect evaluation, we take this as sufficient evidence that some fine-tuning is necessary to

[1]Courtesy of RunwayML: runwayml.com

adjust the domain-gap between the CLIP pre-training mixture and our environment state. An alternative solution may have been to simply apply the "realism" transform to our environment state. This is however infeasible for three reasons. Firstly, latent diffusion inference is computationally expensive. Secondly, there is a very high variance in the results obtained due to the nature of prompting, resulting in images that rarely preserve the details necessary for the task. See for instance Figure A.1, where the "realistic" version loses the any notion of the sliding door or switch. Lastly, we are still motivated by our first reason above, i.e. the need for two-image representations, so we will need to train a head on top of CLIP regardless.

A.2 Alternatives to CLIPT

We performed a number of experiments similar to the previous section before choosing to embark on our modification of CLIP into CLIPT. Our experiments generally consisted in collecting language annotations and trajectory end states and visualizing similarity matrices in search of well-defined diagonals. The lack of a diagonal would indicate that whatever modification we were testing in that experiment was insufficient to guarantee that we could perform our desired swapping of visual goals language goals. We first tried two alternatives to CLIP, namely FLAVA (Singh et al., 2022a) and ALIGN (Jia et al., 2021), however these did not return promising results. We therefore returned to CLIP and attempted some prompt engineering (Gu et al., 2023; Liu et al., 2023) of the natural language instructions. Specifically we tried converting the instructions to the past tense, paraphrasing them to be more similar in style to image captions, and finally even some BLIP2-based (Li et al., 2023) reformulations. Ultimately however, none of these experiments provided satisfactory results, so we proceeded with our CLIPT modifications.

A.3 Task Classifier

Before proceeding to our GCBC implementation, we wanted to check whether our idea of training on visual representations and evaluation on textual representations from CLIPT was sound on an easier task. Leveraging the fact that CALVIN trajectories annotated with language are additionally also annotated with the task that is being completed, we implemented a simple task classifier network. This consisted in a simple three-linear-layer MLP, trained to classify visual representations of CLIPT across the 34 possible tasks. We used CLIPT-1P, the checkpoint of CLIPT from the first phase of training where the language encoder is simply the CLIP language encoder. At test time, we then evaluated with both visual and textual trajectories and compared the performance across modalities. We report the results in Table A.2.

Table A.2: The test accuracy of our task classifier on visual trajectory representations and textual trajectory representations from CLIPT-1P.

	Accuracy	**Std. Err.**
Visual Trajectory Representations	0.703	0.003
Textual Trajectory Representations	0.857	0.009

To our great surprise, the classifier performed better when classifying textual trajectory representations than when classifying visual trajectory representations, despite being trained on the latter. One possible explanation is that the frames and/or trajectories in the test set are somewhat out of distribution compared to those in the train split, while the textual descriptions do not really change too much between splits. You explanation was somewhat reinforced when noticing that textual accuracy remained almost unchanged between validation and test (~ 0.85) while visual accuracy dropped from 0.89 in validation to 0.70 in test. Given the promising results with our task classifier, we proceeded with our GCBC implementation.

APPENDIX B

More GCBC experiments

B.1 Performance over training

In an attempt to diagnose the relatively underwhelming performance of our policy, we perform our evaluation on the test set at various checkpoints throughout training on the CALVIN dataset. Figure B.1 presents the mean SR over training when conditioning on textual trajectory representations[1]. From the few data-points that we have, we see a very slow increase in performance over training, indicating that perhaps our model is not sufficiently expressive. A more complete analysis is however necessary to make conclusions.

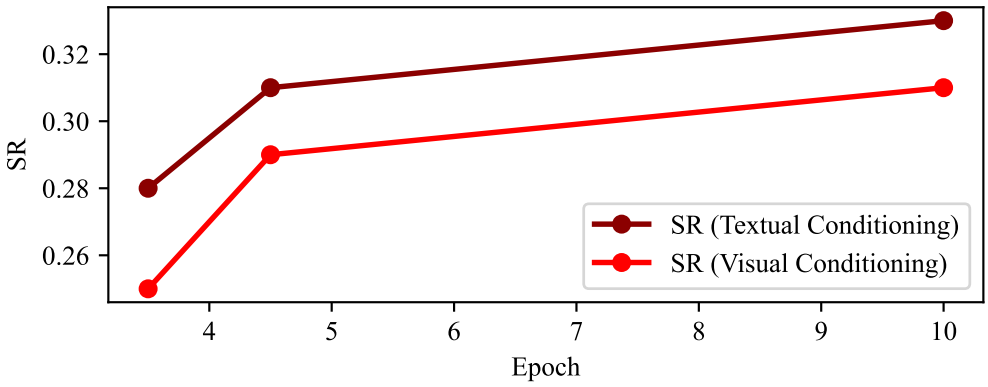

Figure B.1: GCBC success rate over training

[1]We mistakenly performed these evaluations without context resets. However, since we are mainly interested in the trend rather than actual values with this experiment, we deemed this oversight as negligible.