Language-Conditional Imitation Learning

Julian Skirzyński, School of Computer Science

McGill University, Montreal

August, 2020

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of

Master of Computer Science

Abstract

The thesis explores interactions between language use and behavior learning. The core idea is that multiple high-level behaviors can be learned by the agent using linguistic cues in the form of sentences and that learning a language model may allow the agent to follow new sets of instructions unseen in training. To implement this idea, we introduce the Language-Conditional Imitation Learning algorithm which auto-encodes input language sequences and uses the resulting context representation to condition input physical states. To do so successfully, the algorithm optimizes two loss functions where one corresponds to behavioral cloning, and the other one is a reconstruction loss for word sequences. Conducted experiments indicate that our method manages to retain quality performance on the behaviors it was trained on, and despite slightly worse numerical results, it manages to imitate the set of training behaviors in rollouts. Moreover, our algorithm seems to be robust to new words in the sentences and to increasing sentences' length. Nonetheless, the results are not straightforward for testing generalization. Although our method does well on an ambiguous task of imitating an unseen behavior which was not included in the training set (quantitatively and qualitatively), so does the simplest baseline of our algorithm, behavioral cloning. We perform an analysis of our algorithm and find it is likely to generalize due to its ability to discover the proper similarity between train and test sentences. We conclude that this research may be stimulating to the field of Human-Computer Interaction. It is proposed, however, that future work clarifies the issue of generalization, by performing a more complex experiment where behavioral cloning does

not do well, and information encoded in the state is not sufficient to clone the unseen behavior.

Abrégé

La thèse explore les interactions entre l'utilisation du langage et l'apprentissage du comportement. L'idée centrale est que l'agent peut apprendre plusieurs comportements de haut niveau en utilisant des indices linguistiques sous forme de phrases et que l'apprentissage d'un modèle de langage peut permettre à l'agent de suivre de nouveaux ensembles d'instructions invisibles pendant la formation. Pour implémenter cette idée, nous introduisons l'algorithme Language-Conditional Imitation Learning qui code automatiquement les séquences de langue d'entrée et utilise la représentation contextuelle résultante pour conditionner les états physiques d'entrée. Pour réussir, l'algorithme optimise deux fonctions de perte où l'une correspond au clonage comportemental et l'autre est une perte de reconstruction pour les séquences de mots. Les expériences menées indiquent que notre méthode parvient à conserver des performances de qualité sur les comportements sur lesquels elle a été entraînée, et malgré des résultats numériques légèrement moins bons, elle parvient même à imiter l'ensemble des comportements d'entraînement dans les déploiements. De plus, notre algorithme semble robuste aux nouveaux mots dans les phrases et à l'augmentation de la longueur des phrases. Néanmoins, les résultats ne sont pas simples pour tester la généralisation. Bien que notre méthode fonctionne bien sur une tâche ambiguë consistant à imiter un comportement invisible qui n'a pas été inclus dans l'ensemble d'apprentissage (quantitativement et qualitativement), il en va de même pour la ligne de base la plus simple de notre algorithme, le clonage comportemental. Nous effectuons une analyse de notre algorithme et constatons qu'il est susceptible de se généraliser en raison de sa capacité à découvrir la similitude appropriée entre les phrases

de train et de test. Nous concluons que cette recherche peut être stimulante dans le domaine de l'interaction homme-machine. Il est proposé, cependant, que les travaux futurs clarifient la question de la généralisation, en effectuant une expérience plus complexe où le clonage comportemental ne fonctionne pas bien et où les informations encodées dans l'état ne sont pas suffisantes pour cloner le comportement invisible.

Acknowledgements

I would like to thank Prof David Meger for his guidance in conducting the research presented in this thesis, numerous critical remarks given during the writing process, and for funding I received through his grants. I would also like to thank my wife, Sara, for putting up with my ever-lasting process of graduation, continuous emotional support and her help in creating some of the figures.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Imitation learning (IL) is one of the major tools used in robotics and fields for which it is easier for humans to convey information by *showing* what to do rather than to express it on a different level of generality. The simplest formulation assumes that we present an agent with a set of state-action pairs and hope it will learn how to approximate the function that generated this data, and thus, master the general set of presented skills. Such an approach, however, is limited because it assumes that all the necessary information is included in the representation of the state. Note that with traditional methods of IL, learning multiple distinct behaviors that are shown by the same user in a single dataset may prove impossible since similar sensory readings could lead to drastically different actions. It has been already shown that if we make the process of imitation learning to depend on additional factors informative to decision-making, i.e. if we condition it, then the agent begins to ingest more than one behavior ([19, 22, 54]). Although the main strength of IL still relies on *showing*, these findings suggest that conditioning on additional information could increase the accuracy of the method and supply it with new capabilities.

The main contribution of this work is extending the framework of Conditional Imitation Learning (CIL) [22] through adding language as the conditioning information. The cited work proposed a supervised learning problem where instead of state-action pairs, we use state-context-action triples and condition states on contexts. In this and in

other works that consider imitating multiple behaviors at once [19, 54], a conditioning vector is based on or derived from a fixed (and usually small) set of possible settings. For self-driving cars, which served as the main focus for all the aforementioned studies, the settings may include driving modes (e.g. following a car), high-level behaviors (e.g. turning right) or even affordances of the perceived scene (e.g. distance to the sidewalk). Language, on the other hand, exhibits Chomskian infinite variety property and enables to create thousands of distinct sentences that describe the same high-level behavior. Moreover, it allows forming new sentences that relate to trajectories the expert did not demonstrate. Since language is compositional (the meaning of an expression is determined by its structure and the meaning of its constituents), understanding sentences for a number of high-level behaviors allows understanding sentences for a behavior that was not presented by the expert, but was constructed similarly. For humans then, compositionality entails generalization to new expressions. For artificial agents, however, the effects of compositionality are unclear. The major motivation for this study is to investigate the problem of language in conditional imitation learning and test whether its inherent generalization quality is transferable. Our central hypothesis is that when added to the CIL framework as context, language will **(a)** enable imitating training behaviors independent of the particular language description (generalization over unseen context) and **(b)** enable executing new behaviors (generalization over unseen behaviors, see zero-shot learning [67]). See Figure 1 for an overview of this research agenda.

In this thesis, we introduce Language-Conditional Imitation Learning algorithm (L-CIL). Our algorithm conditions states on decodable representations of input sentences, thereby grounding action in language and grounding language in action. The idea underlying L-CIL is that sentences can be projected onto a latent space, which preserves their common meaning and gives a sensible nearness relation as sentences and behaviors change. The concept of a latent space has been already used in general studies on conditioning behavior on linguistic input [13, 15, 16, 21, 57, 94]. Nevertheless, the author is unfamiliar with works that use the latent space to learn multiple behaviors at once,

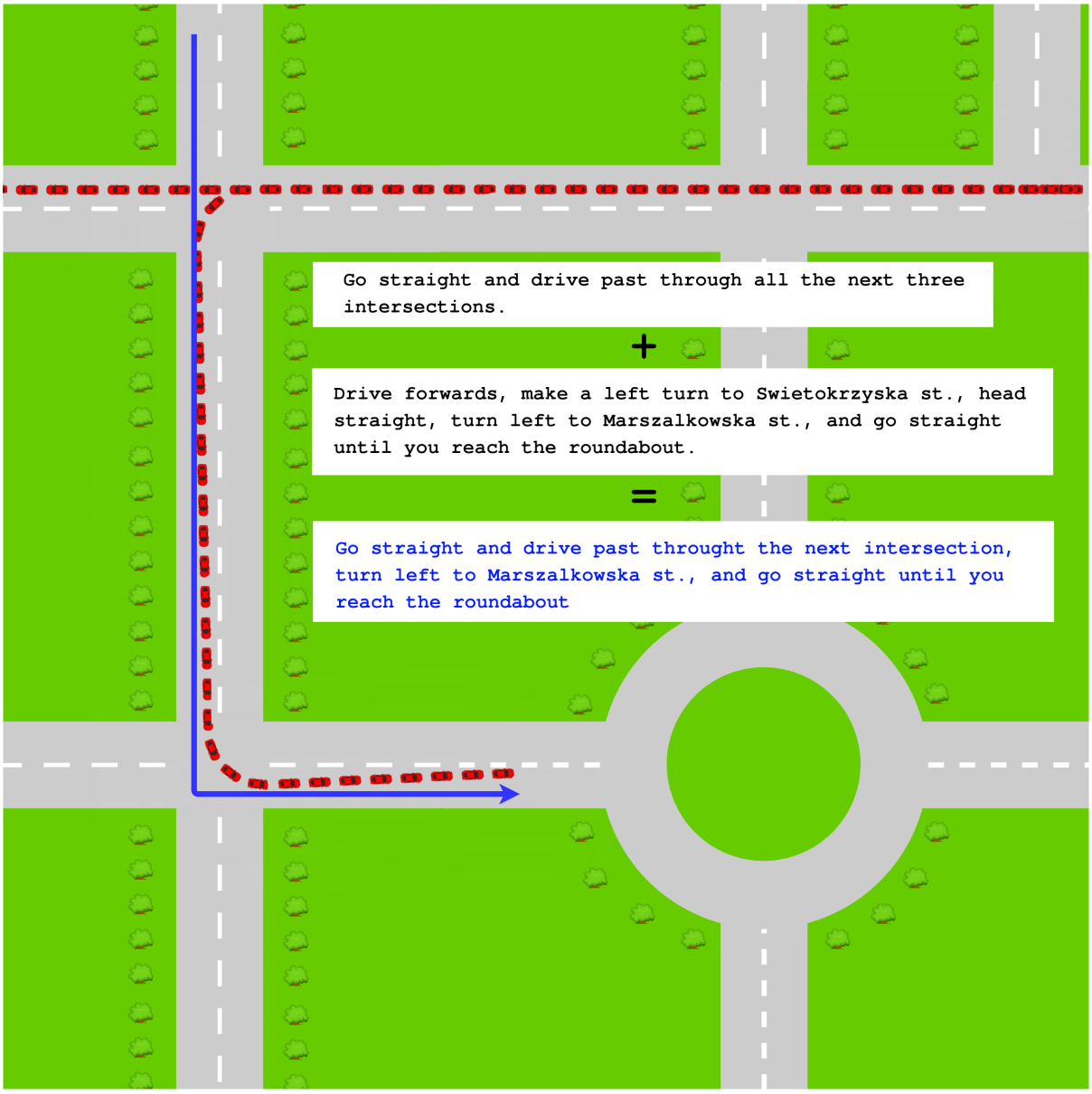


Figure 1: Trajectories generated by the agent (the car) when cued by language (black-colored text commands) and a hypothetical reference trajectory (blue arrow) described by language (blue-colored text command). A successful language-conditional imitation learning algorithm would train on a single dataset containing both behaviors described by the black sentences. During test time it could differentiate between them cued by sentences synonymous to those used in training. Ideally, it could be also queried on sentences that describe a new behavior, if this description used known expressions (see the blue sentence), and perform the described behavior despite not seeing it explicitly.

and at the same time, force it to capture hidden variables defining the meaning of a sentence. L-CIL addresses both of those issues by defining a modified learning task. On top of predicting the action based on a state and a hidden representation of context, it aims to use the hidden representation to predict the context itself. In our experiments, we show that L-CIL overcomes the advantage of Conditional Imitation Learning which uses simple, explicit hidden vectors and report quantitatively and qualitatively comparable performance in standard multiple-behavior learning scenarios. Our tests also indicate that the algorithm is able to follow an unseen behavior, once again quantitatively and qualitatively. The analysis we performed revealed that it infers the similarity of conditioning vectors to the extent that allows it to do so. We believe, however, that further research is needed to corroborate this supposition. The current experimental setup also allowed behavior cloning to imitate the unseen behavior well, although this approach did not have access to the external information.

The thesis is constructed the following way. The first chapter sets up the theoretical framework for the work and covers seemingly all the relevant themes that are connected to the presented research, i.e. basic definition of Markov Decision Processes, background in reinforcement learning, imitation learning, its types, and conditional imitation learning. Chapters 3, 4 and 5 focus on related work in the field of imitation learning (Chapter 3), language in machine learning (Chapter 4) and related work in psychology and neurosciences (Chapter 5). Language-Conditional Imitation Learning algorithm is described in Chapter 6. Chapter 7 details the experimental setup and Chapter 8 describes experimental results including the general performance and generalization capabilities of the algorithm. The last chapter provides comments on the results and tries to sketch a roadmap for the future.

Part I

Technical Foundations and Related Work

Chapter 2

Theoretical Framework

This chapter introduces notions and approaches relevant to the work at hand. The most general definitions are on Markov decision processes (MDPs) and on Imitation Learning (IL) and can be found in the first two sections. Then, given that the introduced algorithm is a version of behavioral cloning (BC), ensuing sections describe it and its alternative in the world of IL, inverse reinforcement learning (IRL). The last section acquaints the reader with the setting used throughout this thesis, and details the problem of conditional imitation learning.

2.1 Markov Decision Process

The notion of a Markov Decision Process underlies all the subsequent definitions. It is a formal tool that represents problems that involve agents that interact with some environment, i.e. observe states and take actions. Let's define a *Markov chain* first.

Definition 1 (Markov Chain). *A stochastic process $X = \{X_n : n \geq 0\}$ on set S is a Markov Chain if, for any $i, j \in S$ and $n \geq 0$,*

$$\mathbb{P}(X_{n+1} = j \mid X_0, ..., X_n) = \mathbb{P}(X_{n+1} = j \mid X_n), \tag{2.1}$$

$$\mathbb{P}(X_{n+1} = j \mid X_n = i) = p_{ij}. \tag{2.2}$$

Condition 2.1 defines the Markov property. It means that the next state X_{n+1} for any given n is conditionally independent of the history of previous transitions $X_0, ..., X_{n-1}$ given the present state X_n.

Definition 2 (Markov Decision Process). *A Markov decision process (MDP) is a process that satisfies the Markov property. It is represented by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$ where S is a set of states; A is a set of actions; $\mathcal{T}(s, a, s') = \mathbb{P}(s_{t+1} = s' \mid s_t = s, a_t = a)$ is a state transition function; $\gamma \in (0, 1)$ is a discount factor; $\mathcal{R} : \mathcal{S} \longrightarrow \mathbb{R}$ is a reward function.*

Note that $\mathcal{R}$ could be also represented as a function dependent on state-action pairs $\mathcal{R} : \mathcal{S} \times \mathcal{A} \longrightarrow \mathbb{R}$. Policy $\pi : \mathcal{S} \longrightarrow \mathcal{A}$ denotes a deterministic function that controls agent's behavior in an MDP and a nondeterministic $\pi : \mathcal{S} \longrightarrow Prob(\mathcal{A})$ defines a probability distribution over the actions.

2.2 Reinforcement Learning

A class of methods that learn policy π through trial and error optimization process is called reinforcement learning (RL).

Definition 3 (Expected Reward). *We denote*

$$\mathcal{J}(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | \pi\right] \tag{2.3}$$

as the expected reward of policy π accumulated over infinite time horizon t. $r_t = \mathcal{R}(s_t, a_t)$ is a measure of quality of the state or the state-action pair and γ is a discount factor that controls the importance of short-term rewards over long-term ones.

The goal of RL is to map the state of the system s_t to action a_t that affects and changes that state to s_{t+1} so as to maximize expected reward $\mathcal{J}(\pi)$:

$$\pi^* = \arg\max_{\pi} \mathcal{J}(\pi). \tag{2.4}$$

Two common algorithms for finding π^* are value iteration which is a solver for known finite MDPs and Q-learning, an RL method that explores an unknown finite MDP.

Definition 4 (Value Function). *The value function maps states $s \in \mathcal{S}$ to expected rewards obtained by starting in state s and navigating the state-space according to policy π:*

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \, \mathcal{R}(s_t, \pi(s_t)) \mid s_0 = s\right]. \tag{2.5}$$

Definition 5 (Action-value Function). *The action-value function maps state-action pairs $s, a \in \mathcal{S} \times \mathcal{A}$ to expected rewards obtained by starting in state s, executing action a and navigating the state-space according to policy π:*

$$Q^{\pi}(s, a) = \mathbb{E}\left[\mathcal{R}(s_0, a_0) + \sum_{t=1}^{\infty} \gamma^t \, \mathcal{R}(s_t, \pi(s_t)) \mid s_0 = s, a_0 = a\right]. \tag{2.6}$$

Definition 6 (Bellman Equations). *The Bellman equation expresses the relationship between the value of a state and the value of a successor state:*

$$V^{\pi}(s) = \mathcal{R}(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi(s), s') V^{\pi}(s'). \tag{2.7}$$

The Bellman optimality equation is the Bellman equation for the optimal policy:

$$V^{\pi^*}(s) = \max_{a \in \mathcal{A}} \left[\mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi^*(s), s') V^{\pi^*}(s')\right]. \tag{2.8}$$

Value iteration iterates over the value function N_ϵ times and $\forall s \in \mathcal{S}$ updates function V using the Bellman optimality equation:

$$V_k(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') \left(\mathcal{R}(s' \mid s, a) + \gamma V_{k-1}(s')\right) \tag{2.9}$$

for $k = 0, ..., N_\epsilon$. N_ϵ is determined so that $\forall s \in \mathcal{S},\ |V_k(s) - V_{k-1}(s)| \leq \epsilon$ and $V_0(\cdot)$ is initialized randomly. This algorithm enables evaluating and updating the policy in one sweep through the state space. It can be shown that sequence $\{V_k\}$ converges to V^{π^*} [91].

Q-learning generates episodes of transitions $(s_t, a_t, s_{t+1}, r_t), t = 0, ..., T_i$ where s_{T_i} is a terminal state for $i = 0, ..., N$. It uses a policy derived from the action-value function Q to update

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right) \tag{2.10}$$

with α as a learning rate. If all state-action pairs continue to be updated with sufficient frequency, Q-learning is guaranteed to converge to the optimal solution [91].

Intuitively, value iteration bootstraps the approximation of V using its own predictions until they become stable, and Q-learning bootstraps the value of a state-action pair based on the Q-value of the next state. For more detailed information about RL refer to [91].

2.3 Imitation Learning

Imitation learning is the problem of finding a policy mimicking transitions provided in a dataset of trajectories $\tau_i = [\phi_0, ..., \phi_{N-1}]$. Trajectories are sequences of features ϕ that encode encountered characteristics of the environment, stand for state-action pairs or even correspond to single pixels of an input image. Their interpretation depends on a particular problem. In our case:

Definition 7 (Trajectory). *A trajectory $\tau = [\phi_0, ..., \phi_{N-1}]$ is a sequence of of length N containing state-action pairs where $\phi_i = (s_i, a_i)$ for $s_i \in \mathcal{S}, a_i \in \mathcal{A}, i = 0, ..., N-1$.*

Sometimes demonstrations are enhanced with an additional context parameter c_τ that describes conditions under which these demonstrations were gathered. Context may be used to modulate proper behavior. We will omit this notion in the general problem statement but it will become a formalism for conditional imitation learning. Similarly, we will

omit reward signals which could also be collected (see BatchRL [48]) and used in imitation (see [81]). Having a dataset $\mathcal{D} = \{\tau_1, ..., \tau_M\}$, an imitation learning agent tries to optimize policy π^* that satisfies:

$$\pi^* = \arg\min \mathcal{L}(q(\boldsymbol{\phi}), p(\boldsymbol{\phi})), \tag{2.11}$$

where q and p are distributions of features under agent's and expert's policies and $\mathcal{L}$ denotes a similarity measure for distributions. In other words, IL searches for a policy that provides the same state visitation as the expert.

Strategies used for optimizing equation (2.11) vary, similarly to the mode in which the set $\mathcal{D}$ is gathered (offline, online, being enhanced with data from a simulator, etc.) Scientists differentiate between two main approaches, however: inverse reinforcement learning (IRL), which tries to establish a reward function that drives the exhibited behavior, and behavioral cloning (BC), which explicitly imitates this behavior. We shall discuss both approaches in the sections below, putting more emphasis on behavioral cloning, which served as the basis for models presented in this work.

2.4 Inverse Reinforcement Learning

In IRL, having only the access to demonstration set $\mathcal{D}$, we assume that training trajectories were generated by following policy π^* that maximized some unknown reward function r_{π^*}. In order to approximate π^*, an IRL agent tries to recover the mentioned reward function, and act according to its valuations in an iterative manner until it finds a stable solution. It should be noted however, that finding the reward function is an ill-posed problem since one policy can be optimal to many reward functions. To show a general algorithm for solving IRL, we introduce three more definitions.

Definition 8 (State Visitation Frequency). *$\psi^\pi(s)$ is a state-visitation frequency if it measures how often a state will be visited during the execution of policy π and is defined as*

$$\psi^\pi(s) = \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi(s), s')\psi^\pi(s'). \tag{2.12}$$

Definition 9 (State-action Visitation Frequency). *$\mu(a, s)$ is a state-action distribution if it measures how often a state-action pair will be generated during the execution of policy π and is defined as*

$$\mu(a, s) = \psi^\pi(s)\pi(a \mid s). \tag{2.13}$$

Definition 10 (Linear Reward). *We say a reward is linear in $\boldsymbol{\omega}$ if there exist features ϕ_i such that*

$$\mathcal{R}_{\boldsymbol{\omega}}(s, a) = \sum_i \omega_i \phi_i(s, a) = \boldsymbol{\omega}^\top \boldsymbol{\phi}. \tag{2.14}$$

The state distribution and state-action distribution regard visitation frequencies that were estimated from a set of data. The most common way of approximating π^* is presented in Algorithm 1 box (taken from [65]). Note step 7 where optimizing for ω is accomplished with the use of reinforcement learning methods. This shows that IRL intermixes methods of standard optimization with RL.

One advantage of IRL is that by learning the reward, we can analyze it to better understand the expert's behavior. Moreover, the reward can be transferred to a different environment and improved through interactions alone, not requiring new demonstrations. However, the biggest drawback of IRL besides its ill-posedness, lies in the computationally costly need to solve an MDP.

Algorithm 1: Abstract version of inverse reinforcement learning

Input: Expert trajectories $\mathcal{D} = \{\boldsymbol{\tau_i}\}_{i=1}^{N}$
Initialize the reward function and policy parameters $\boldsymbol{\omega}$, $\boldsymbol{\theta}$;
repeat
    Evaluate the state-action visitation frequency $\boldsymbol{\mu}$ of the current policy π_θ;
    Evaluate the objective function $\boldsymbol{\mathcal{L}}$ and its derivative $\boldsymbol{\nabla_\omega \mathcal{L}}$ by comparing $\boldsymbol{\mu}$ and the state-action distribution implied by $\mathcal{D}$;
    Update the reward function parameter $\boldsymbol{\omega}$;
    Update the policy parameter $\boldsymbol{\theta}$ with a reinforcement learning method;
until
return optimized policy parameters $\boldsymbol{\theta}$ and reward function parameter $\boldsymbol{\omega}$

2.5 Behavioral Cloning

We use the term *behavioral cloning* to describe all methods which approximate mappings from state or state and context sets to the set of actions (or trajectories) via supervised learning on imitation data. The context set contains any information relevant to action-taking which is not included in the current state observed by the agent – for instance the initial state. In contrast to IRL, in this setting it is possible to perform all the learning steps without resorting to RL. For the sake of the contents of this work let's consider the state-action type of BC. Formally, for finding policy π^* which generated demonstration data $\mathcal{D} = \{(s_i, a_i)\}_{i=0}^{N}$ we set up a supervised regression problem of the form:

$$\underset{\theta}{\text{minimize}} \; \mathcal{L}\left(\sum_t \pi_\theta(s_t;\theta), a_t\right) \tag{2.15}$$

for $s_t, a_t \in \mathcal{D} = \{(s_i, a_i)\}_{i=0}^{N}$, a function approximator π_θ defined with parameters θ, and a loss function $\mathcal{L}$. Note that we simplified the notation used in the previous sections and explicitly wrote $\mathcal{D}$ as a set of state-action pairs instead of separate trajectories to accentuate we are solving a supervised learning problem. Algorithm 2 box presents an abstract formulation of behavioral cloning (taken from [65]).

Algorithm 2: Abstract version of behavioral cloning

1: Collect a set of trajectories demonstrated by the expert $\mathcal{D}$;
2: Select a policy representation π;
3: Parametrize π with parameters $\boldsymbol{\theta}$;
4: Select an objective function $\boldsymbol{\mathcal{L}}$;
5: Optimize $\boldsymbol{\mathcal{L}}$ w.r.t. the policy parameters $\boldsymbol{\theta}$ using $\mathcal{D}$;
6: **return** optimized policy parameters $\boldsymbol{\theta}$

First we gather the set of trajectories. Then, depending on whether we predict actions or whole trajectories at once we establish an appropriate policy representation, for instance a function $\pi : \mathcal{S} \longrightarrow Prob(\mathcal{A})$. Thereafter, we choose parameters which encode the policy and which we shall optimize. Finally, we start the optimization process by minimizing loss $\mathcal{L}$. The loss acts as a metric which measures the difference between agent- and expert-generated behavior presented in the demonstration set $\mathcal{D}$. Candidate losses may be the following:

Definition 11 (Sample forms of $\boldsymbol{\mathcal{L}}$).

$$\mathcal{L}(x_1, x_2) = (x_1 - x_2)^\top (x_1 - x_2) \tag{2.16}$$

$$\mathcal{L}(x_1, x_2) = (x_1 - x_2)^\top \boldsymbol{\mathcal{W}} (x_1 - x_2) \tag{2.17}$$

$$\mathcal{L}(x_1, x_2) = \sum_i |x_{1,i} - x_{2,i}| \tag{2.18}$$

$$\mathcal{L}(x_1, x_2) = -\sum_i x_{1,i} \ln x_{2,i} \tag{2.19}$$

where $x_1, x_2 \in \mathbb{R}^n$, $x_{i,j}$ is j-th element of vector x_i and $\boldsymbol{\mathcal{W}}$ is a weight matrix.

In this thesis we shall focus on the most common case of BC: supervised behavioral cloning with quadratic loss (2.16). The main task of mastering a number of behaviors will require a slightly enriched version of imitation learning that also accounts for the context. A framework for that type of IL is detailed in the next section.

2.6 Conditional Imitation Learning

The Conditional Imitation Learning (CIL) framework addresses one of main limitations of imitation learning that prevents this method from having widespread use in many practical domains. It tackles the assumption that proper behavior can be inferred from perceptual input only. That is, from the representation of the environment in which the expert is taking actions. As an example in which this assumption is not met the authors of conditional imitation learning, Codevilla *et al.* [22], refer to self-driving:

> Consider a driver approaching an intersection. The driver's subsequent actions are not explained by the observations, but are additionally affected by the driver's internal state, such as the intended destination. The same observations could lead to different actions, depending on this latent state. This could be modeled as stochasticity, but a stochastic formulation misses the underlying causes of the behavior. Moreover, even if a controller trained to imitate demonstrations of urban driving did learn to make turns and avoid collisions, it would still not constitute a useful driving system. It would wander the streets, making arbitrary decisions at intersections.

The solution the authors propose resides in the world of behavioral cloning. Codevilla *et al.* [22] suggest modelling latent information that additionally explains the expert's actions by vector h and expose the learner to this possibly over-complex representation through command $c = c(h)$. The original optimization problem from equation (2.15) changes to

$$\underset{\theta}{\text{minimize}} \; \mathcal{L}\left(\sum_t \pi_\theta(s_t, c_t; \theta), a_t\right). \tag{2.20}$$

The variable c could be an actual command issued in natural language or a form of a signal associated with some behavior (e.g. command "right" or car's blinking light can be both associated with turning). The information carried by c enables the method to

distinguish between two or more different behaviors that occur in the same area of the environment's state-space, and at test time gives the possibility to query it to perform them. This property is crucial for testing zero-shot learning, when we indeed set up c to correspond to spoken language sentences and want to be able to query the algorithm with virtually any expression defined by our vocabulary.

Chapter 3

Related work on Imitation Learning

The main theme for this chapter is to survey a non-exhaustive list of related topics. The primary focus turns from establishing mathematical foundations to showing important works done by other authors that build on these foundations.

In more detail, we provide an overview of existing ideas that are similar to the setting used in this thesis. IRL is mentioned in the first section to show alternative approaches to the problem of imitation. Sample works that justify the use of neural networks in behavioral cloning and point out its weaknesses are discussed in Section 3.2. To grasp the limitations and differences between existing models and the contributions of this thesis, the third section focuses on state-of-the-art works from the area of conditional imitation learning. Finally, learning a language, or more concretely, auto-encoding sentences on top of action prediction can be considered an auxiliary task. Since adding auxiliary tasks to improve performance is the leading idea of multi-task learning (MTL), MTL is mentioned in the last section.

3.1 Inverse Reinforcement Learning

In this section we survey several important and influential works in the area of Inverse Reinforcement Learning. A more detailed discussion can be found in [65].

In a seminal work by Abbeel *et al.* [1] the authors generated a set of M trajectories for helicopter aerobatics denoted as $\{[y_0^0, y_1^0, ..., y_{N_0-1}^0], ..., [y_0^{M-1}, y_1^{M-1}, ..., y_{N_{M-1}-1}^{M-1}]\}$ where $\tau_i = [y_j^i \mid j = 0, ..., N_i - 1], i = 0, ..., M-1$ is a sample trajectory and $y_j^i = (s_j^i, a_j^i)$ is a state-action pair. They assumed there exists a target trajectory $z = [z_j \mid j = 0, ..., T-1]$ and that each τ_j is its imperfect demonstration. Formally, it was assumed that $y_j^k = z_{\tau_j^k} + \omega_j$ where τ_j^k is temporally aligned index that matches observed and desired states, and ω_j is Gaussian noise. The authors also introduced helicopter dynamics $f(\cdot)$, set $z_t = f(z_{t-1}) + \omega_t$ and computed a linear approximation of this model. To solve for the right policy, Abbeel *et al.* [1] used a special case of an MDP – linear quadratic regulator [91], and solved its optimal with iterated application of dynamic programming.

Ng and Russell [62] proposed to write the reward function $\mathcal{R}(s) = \alpha_1\phi_1(s) + ... + \alpha_n\phi_n(s)$ as a combination of bounded basis functions $\phi_i(s) : \mathcal{S} \longrightarrow \mathbb{R}$ and solve a sequence of linear programs (LPs) in search for the weights of this combination. The main constraints $|\alpha_i| \leq 1 \; \forall i$ enforced that subsequent empirical state-valuations derived from $\mathcal{R}$ were greater than state-valuations for the optimal policy after LP-optimization, i.e. $V^{\pi_k}(s_0) \geq V^{\pi_{k-1}}(s_0)$ where $V^{\pi_k}(s_0) = \alpha_1 V_1^{\pi_k}(s_0) + \alpha_d V_d^{\pi_k}(s_0)$ is an α-weighted average over empirical returns $V_i^{\pi_k} = \phi_i(s_0) + \gamma\phi_1(s_0) + ... + \gamma^n\phi_n(s_0)$. The optimal policy itself was found using standard RL value iteration algorithm.

Ziebart *et al.* [98] used maximum entropy principle and implemented Algorithm 1 by optimizing cost parameters θ. They assumed the reward function can be expressed in linear form (c.f. (2.14)) and for trajectory τ defined $c_\theta(\tau) = \sum_{s \in \tau} \theta^\top f_s$. In each iteration, the authors solved for the optimal policy with respect to c_θ via value iteration. Their loss function was $\mathcal{L} = \frac{1}{M} \sum_{\tau_d \in \mathcal{D}} f_{\tau_d} + \sum_s \mu(a, s) f_s$ where $\mu(a, s)$ is state visitation frequency from equation (2.12).

Wulfmeier *et al.* [95] computed the same set of unknowns as Ziebart *et al.* [98] making use of a neural network with maximum entropy objective function $\mathcal{L}(\theta) = \log \mathbb{P}(\mathcal{D} \mid \mathcal{R}) + \log \mathbb{P}(\theta)$. This function maximizes the entropy of state-action visitations while minimizing the distance to expert's state-action visitations.

Finn *et al.* [28] relaxed the problem even more and learned a sample-based approximation of maximum entropy IRL through gradient descent. Concretely, the authors attempted to functionally approximate $c_\theta(\tau)$ in the guided cost learning framework with objective function $\mathcal{L}_{CGL} = \frac{1}{M} \sum_{\tau_d \in \mathcal{D}} c_\theta(\tau_d) + \ln Z$, where M is the number of samples τ and $Z = \int \exp(-c_\theta(\tau)) d\theta$. Their approach was to interchangeably update the cost function based on samples from the demonstrations and the policy being learned with updating the policy with newly computed c_θ.

Ramachandran and Amir [75] decomposed the probability of generating any expert trajectory according to the Bayes rule, i.e. $\mathbb{P}(\mathcal{R} \mid \mathcal{D}) = \frac{\mathbb{P}(\mathcal{D}|\mathcal{R})\mathbb{P}(\mathcal{R})}{\mathbb{P}(\mathcal{D})}$. They approximated each state-action probability $\mathbb{P}_\mathcal{D}((s_i, a_i) \mid \mathcal{R}) = \frac{1}{Z_i} \exp\left[\alpha\, Q(s_i, a_i; \mathcal{R})\right]$ with normalizing factor Z_i. This enabled them to represent $\mathbb{P}(\tau|\mathcal{R})$ for trajectory τ, and in consequence compute the posterior $\mathbb{P}(\mathcal{R}|\tau)$.

Ho and Ermon [40] took yet another approach to find the optimal policy and used the principles of trust-region policy optimization and generative adversarial networks. Other approaches to IRL can be also found in [9, 44].

3.2 Behavioral Cloning

In this section we shall focus on behavioral cloning implemented in neural networks and mainly analyze the structure of resulting models. The goal is to show the robustness neural networks offer and justify their use in the model instantiating L-CIL. Additionally, we will discuss non-standard approaches that try to address the accumulation of error problem and the embodiment problem, because these are imperfections of BC that our method naturally inherited.

3.2.1 Traditional Approaches

Algorithms that implement the sketch from Algorithm 2 are rather homogeneous in their design but for small variations in the choice of the loss function and the decision to exer-

cise the structure of the training environment (model-based vs. model-free distinction). The most common combination is quadratic loss ℓ_2 (2.16) for least squares regression and environment-agnostic scheme that only strives to match the expert behavior in the training states with the highest possible accuracy. This approach is for instance considered in Sammut *et al.* [84] with decision trees used for imitation, and in the historically important ALVINN system for vehicle navigation [74]. Following this early use of neural networks and the resurgence in the field, the most of the current research considers approaches that utilize deep learning.

3.2.2 Deep Learning Approaches

In a work by Duan *et al.* [27] the authors introduced one-shot supervised IL setting with a soft attention component. Their network is constructed of a demonstration network that embeds the trajectory $\{(s_i, a_i)\}_{i=0}^{n}$ into a hidden representation h, a context network that compiles the embedding h and the current state (s_k, a_k) into a joint representation r, and an action network which predicts the action based on r.

Silver *et al.* [85] in their seminal paper on the Go game used imitation learning as a starting point in the creation of AlphaGo. The network the authors utilized had 13 layers which alternated between convolutional layers and rectifier nonlinearity layers with a softmax layer at the end. Its loss function considered merely the likelihood of expert demonstrations $\log \pi_\theta(a_t \mid s_t)$ where θ denoted the parameters of the model, and $(s_t, a_t) \in \mathcal{D}$ stood for the members of the demonstration set.

Zhang *et al.* [97] taught a robot to perform complex manipulation tasks (reaching, grasping, pushing, placing) through recordings made with the use of a virtual reality teleoperator. Their method, similarly to the previously mentioned ones, assumed having a dataset of trajectories of state-action pairs $\mathcal{D} = \{(s_t^{(i)}, a_t^{(i)})\}$ and a neural network that approximated desired policy with $\pi_\theta(a_t \mid s_t)$. The optimized loss consisted of a combination of ℓ_1, ℓ_2, sigmoid cross-entropy, alignment $\frac{a_t^\top \pi_\theta(s_t)}{\|a_t^\top\| \|\pi_\theta(s_t)\|}$ and auxiliary task losses.

Chung *et al.* [20] developed a variational version of a recurrent neural network and used it to imitate handwriting styles. Many more examples are mentioned in an excellent overview paper by Osa *et al.* [65].

3.2.3 Accumulation of Error

In this section we consider a variety of methods that have been proposed to deal with error accumulation. The titular problem occurs when numerous operations are performed consecutively before outputting the result, and each bears a small numerical error. Because these operations depend on one another, the final error is said to *accumulate*.

In the SEARN paper by Daumé III *et al.* [24] the agent learns a mixture of policies iteratively. The algorithm starts with a policy that imitates the expert's behavior, then generates cost-sensitive training examples, sets up a new policy, and finally interpolates it with the previous one. A procedure constructed in this way allows to walk the learner through the search space of policies and tackle the most problematic states.

Ross and Bagnell [80] (SMILe) proposed to iterate over imitating trajectories of a policy learned in the previous iteration, and mix those policies in a similar manner as Daumé et al [24]. After learning policy $\hat{\pi_n}$ on trajectories from policy π_n the algorithm sets $\pi_n = \pi_{n-1} + \alpha(1-\alpha)^{n-1}(\hat{\pi_n} - \pi_0)$, where π_0 is the initially learned expert-imitating policy. As the authors put it themselves "this update is interpreted as adding probability $\alpha(1-\alpha)^{n-1}$ to executing policy $\hat{\pi_n}$ at any step and removing the same probability of executing the queried expert's action" [80]. If the algorithm is stopped at iteration N, it returns a normalized policy $\hat{\pi_N} = \frac{\pi_N - (1-\alpha)^N \pi_0}{1-(1-\alpha)^N}$ which does not rely on the expert anymore.

It seems that one of the most acknowledged papers that sought to ameliorate BC's performance with the discussed concern in mind is Dataset Aggregation (DAgger) [82]. The main assumption for DAgger is to help the agent recover from mistakes it could make. After each iteration the expert gathers information about agent's state visitations, enhances trajectories with examples of successful recoveries, and lets the policy train using the aggregated dataset. Formally, DAgger assumes having a sequence of policies $\pi_{1:N}$, surrogate

loss $\ell(s, \pi)$ being minimized for policy π in state s, distribution of states d_π incurred by π, a true loss of the best policy defined as $\epsilon_N = \min_{\pi \in \Pi} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{s \sim d_{\pi_i}} l(s, \pi)$, and the regret being $\frac{1}{N} \sum_{i=1}^{N} l_i(\pi_i) - \min_{\pi \in \Pi} \frac{1}{N} \sum_{i=1}^{N} l_i(\pi)$ for a general loss of policy $\ell(\pi)$. The major finding of DAgger is that if $N = \tilde{O}(T)$ there exists $\pi_i, i = 1, ..., N$ such that $\mathbb{E}_{s \sim d_{\pi_i}} \ell(s, \pi_i) \leq \epsilon_N + O(1/T)$. In other words, with the ability to obtain targeted information about the expert in desired states, DAgger gives stronger guarantees on regret.

3.2.4 Embodiment

In some cases, especially in robotics, imitation learning encounters an important problem of embodiment. The so called *correspondence problem* [8] arises when the system that is supposed to imitate the behaviors differs from the presenter. A standard method for solving the correspondence is to learn the forward dynamics model and then using this model, plan the trajectory that matches the expert. This is done through straightforward supervised learning setup or with probabilistic techniques like Gaussian mixture models, e.g. [60, 64].

3.3 Conditional Imitation Learning

The purpose of this section is to provide an overview of existing implementations of conditional imitation learning algorithms. We will outline the strengths and weaknesses of each previous approach in order to justify and motivate decisions made in this thesis. We discuss the three most relevant papers.

Codevilla *et al.* [22] attempted learning a finite set of behaviors using a demonstration set that included all of them. They considered a branched version of a feed-forward neural network where context modulated which branch of the network would be utilized to predict the action (steering angle and acceleration). Technically, the authors conditioned input data on one-hot encodings $c(h)$ which corresponded to behavior types h. This en-

abled the network to navigate to goal positions with high accuracy in simulation as well as in real-world driving tests. Note however, that a branched architecture may not easily generalize to settings with a larger number of behaviors. The main reason is that one-hot vectors lose information on similarity between the behaviors they stand for and the network cannot branch indefinitely.

Mehta *et al.* [54] enlarged the command vector with visual affordances – quantitative statistics computed from the visual scene that served as the main input to the learning algorithm. Affordances along with action primitives were firstly used as auxiliary tasks that needed to be computed based on the state alone, and then their predictions $c(h)$ conditioned the action module of the network. The experiments the authors conducted involved self-driving cars once again and reported high accuracy for imitation in simulation. Predicting affordances however, might be understood as projecting the original state-space onto a more meaningful feature space. It should be noted that those features must be hand-crafted before CIL is used, which might not be always feasible.

Chowdhuri *et al.* [19] applied principles of conditioning to teach a fleet of model cars to drive in different behavioral modes. Information about the mode was encoded in a form of a binary tensor $c(h)$ that was concatenated with the current image of the environment before being passed to an intermediate layer of a convolutional neural network. In this case, the results also favored the conditional IL algorithm. The authors restrained from using a branched architecture yet still conditioned on one-hot vectors which do not preserve possible synergies between the behaviors, severely limiting generalization or transfer.

Interestingly, Babes *et al.*, [6] addressed similar concerns about imitation learning and suggested to use IRL to alleviate them. The authors computed different reward functions for unlabeled demonstrations and used them for clustering. This work had its continuation/extensions e.g. in [26, 35, 46, 52].

3.4 Multi-task Learning

There is a close connection between conditional imitation learning and multi task learning (MTL) [12]. In the former case, we provide the learning agent with information crucial to the imitated problem which are otherwise not included in the environmental signal alone. In MTL one uses auxiliary tasks that are connected to the central task and are supposed to increase the effectiveness of learning. Training signals for those tasks provide inductive bias for the learner and this in turn improves its generalization. Multi-task learning may be hence viewed as a form of implicit conditioning in which the conditioning information is encoded in the parameters used for optimization, for instance weights of a neural network. Conversely, conditioning methods sometimes adopt an auxiliary task, e.g. as already mentioned Mehta *et al.* [54], who besides choosing the right action, made the agent to label visual input with proper affordances and predict action primitives. Thus, such a training scheme does not only condition on vectors $c(h)$ but also performs multi-task learning with two tasks: clone the behavior and predict the proper $c(h)$.

There are numerous examples of multi-task setups in imitation learning, e.g [19, 26, 35, 52, 54]. Apart from that, MTL shows promise in areas going beyond IL, e.g. in robotic manipulation [25, 29] or natural language processing, broadly construed [83]. One of the best examples, which might additionally serve as an introduction to the section on language in RL, is the Policy Sketches paper by Andreas *et al.* [4]. Sketches are vectorized descriptions of a sequence of actions that needs to be taken in order to achieve some goal. In the cited paper they are implemented as one-hot vectors that choose between competing low-level policies represented by actor-critic neural networks [58]. On one hand, such a setup enables for generalization and sketches not seen during training are executed with success during testing. On the other hand, the meta-policy which switches between different subpolicies learns which one to choose after receiving a signal external to the environment – a sketch. Finally, sketches might be thought of as an oversimplified language channel with which the teacher communicates her goal to the agent.

Chapter 4

Related Work on Language in Imitation or Reinforcement Learning

The computational study of language data is known as natural language processing (or NLP for short). Nevertheless, this field mostly regards issues connected to understanding, translating or generating phrases in isolation from other representations. Technological and ideological advancements cause growing interests in introducing language to agent learning settings like imitation learning or, maybe more generally, reinforcement learning. Here, scientists may use the impressive toolbox offered by the NLP community to intertwine language with other tasks which may have been previously lacking that component. This section aims to shed some light on works related to the one presented in this thesis and familiarize the reader with other existing techniques that put language somewhere in the learning loop.

The first section is an overview of history of language outside the strict NLP area. The section on grounding helps to relate algorithms which learn denotations of linguistic concepts in the environment to our method of language-conditional imitation learning which grounds language in action. In the section on conditioning, we review a number of papers that conditioned on language, mainly to show the strength of this approach and discuss existing implementations.

4.1 Language use in Historical Perspective

Many historical papers that concerned using language in machine learning focused on robot navigation, either in small-scale environments like a room [89] or in large-scale environments of a whole building [86]. In both cited papers language was parsed into a program of logical form that was later executed by a robot using in-built primitives. Similarly, in MacMahon *et al.*'s paper from 2006 [51] action primitives were called upon after parsing a command into a tree, transforming it into a table, and extracting out a predicate-like program.

Kollar *et al.*'s paper [45] presented a system that follows natural language directions by changing them to a sequence of spatial description clauses. Co-occurrence statistics computed based on a database of tagged images enabled their agent to ground nouns serving as landmarks in the command. To set up a correct route, they also represented the quality of candidate routes by forming a path that is spatially congruent with the command, and modeled actions as the amount of change in the orientation of those paths.

Branavan *et al.* [10] proposed an RL method that transforms high-level descriptions of actions to low-level commands, and used an agent that interacts with the Windows operating system environment. They collected a history of state transitions and iteratively, based on look-ahead features, computed the best-performing policy until convergence.

Artzi and Zettlemoyer [5] employed combinatorial context grammars. They defined a learning algorithm that scores both the resulting lambda-calculus expressions and possible executions of the task created with primitive actions.

4.2 Grounding

Besides typical RL papers, it also is valuable to mention works that come from the field of natural language processing but offer insights on how language can be inserted into an IL scheme. More precisely, relevant approaches show language grounding, that is learning the correspondence between language and features of the environment.

4.2.1 NLP

Yu *et al.* [96] presented a model of autonomous navigation where the agent is shown a visual representation of the world and, given natural language commands, must answer questions. The authors proposed an alternative method of grounding linguistic entities existing in images and treated the image vector representation as a matrix encoding some number of entities. They used word embeddings [56] to detect whether entities from the sentence are in fact encoded in the image, where detection was implemented as a mapping that slides through an image and provides response values. Finally, actions were learned through an action-critic architecture [58].

Sinha *et al.* [88] considered the same kind of problem. The model they proposed fused knowledge encoded in the image and input sentence to then train a modified actor-critic network. Images were turned into latent tensor representations. Sentences encoded as one-hot vectors were transformed by a GRU [17], and then decoupled to some number of vectors to serve as attention filters. Results of navigation in a 2D world implemented by the authors showed good behavior, outperforming other methods which differed only in how the vision and language vectors were aggregated. Moreover, due to properly learned word vectors, decoders trained on one language of instruction were able to translate sentences provided in another language of instruction.

Hermmann *et al.* [39] grounded language in a simulated 3D environment by a combination of several neural network modules and objective functions, drawing from the works on multi-task learning. In their implementation, visual data along with an LSTM-transformed [41] sequence of words was used as an input to language prediction model. It utilized the A3C algorithm [58] and predicated how well the text described the state the agent was currently in. The authors also experimented with additional auxiliary tasks such as reward prediction.

4.2.2 Beyond NLP

It is worth mentioning methods which have been inspired by the above-mentioned ideas on grounding but go beyond mere NLP. Narasimhan *et al.* [61] let their agent play a moderately complicated grid world game by specifying a set of descriptions of the current state of the environment. The agent learned directly through the interaction with this environment using deep RL techniques. A deep Q-network implemented value iteration which relied on information extracted from the scene. This information was represented as a tensor of object- and LSTM-transformed description-embeddings which may in fact be interpreted as some form of conditioning the state space. Narisimhan *et al.*'s work focused more on transfer and it is shown in the paper how the agent generalized over new sets of environments thanks to language.

Similar ideas were researched in Janner *et al.*'s [43] paper where grounding was again understood as conditioning. LSTM module transformed language descriptions into vectors that were multiplied by object-embedding tensor, and outputted a text-conditioned representation. A convolutional neural network took that representation as an input and predicted a value-map that the agent used when making decisions. This model also exhibited transfer capabilities.

4.3 Conditioning

In a recent survey paper of language in reinforcement learning, Luketina *et al.* [49] distinguishes between two categories of language-enhanced algorithms, namely language-conditional and language-assisted RL. For the latter category of algorithms language is not crucial for proper performance and most often comes in an unstructured and descriptive format (instead of an instructive one). We shall omit this topic due to its lesser relevance to the problem considered in this work. Language-conditioned algorithms, on the other hand, connect ideas presented in the previous sections and consider language as one of the key ingredients. To make descriptions of representative examples more structured,

we may divide them after Luketina *et al.* [49] into three categories: those with language in state-action space, with reward shaping or, most importantly, with instruction following.

4.3.1 Language in State-Action Space

For the first aforementioned category, the authors of the survey rightfully point out that:

> Environments that use natural language as a first-class citizen for driving the interaction with the agent present a strong challenge for RL algorithms. Using natural language requires common sense, world knowledge, and context to resolve ambiguity and cheaply encode information. [49]

One paper worth mentioning here is by Andreas, Klein and Levine [3] who set up an approach called *learning with latent language*. They firstly created a language interpretation model that "maps from descriptions to predictors", i.e. a model that outputs probability of a text matching some input image. At the same time, they taught the agent concepts when the algorithm searched over natural language strings to minimize the loss incurred by a language interpreter. After that, their model could be evaluated on a new set of inputs. Formally, they introduced a proposal model q that computes a distribution over language strings, and a prediction module f that solves the task at hand. Drawing from q and selecting the hypothesis w that obtains the lowest loss, the algorithm used $f(x; \eta, w)$ for encoded state representation x to make predictions. In this way Andreas with colleagues were teaching the agent to specify the problem in natural language terms by itself and then solve it using this description. Other interesting works that seem to be researching scientific questions in this area are on dialogue or question-answering systems (see for instance [88, 96]).

4.3.2 Reward Shaping

Another way that language can be incorporated into the learning scenarios is closely tied to reinforcement learning. Namely, there is a growing number of studies that use in-

structions given to the agent to specify which reward function it should optimize. This could be done directly, by forming a reward function from linguistic input alone, or in an indirect way.

In the former situation we are dealing with meta-learning setting, for instance as in Co-Reyes *et al.* [21]. In the cited paper the authors restrained from using RL or IL whatsoever and expected the agent to perform given tasks based solely on linguistic input provided in the beginning of the training. They achieved this goal in active learning setting similar to DAgger. After each policy rollout, they specified language corrections and sent them to the agent. Those corrections conveyed information necessary to succeed in the task starting in the state the agent found itself in after the execution of the policy. A growing dataset of traversed trajectories and obtained corrections were used to compute the mean representation of the agent's performance. This vector was concatenated to the instruction representation computed by a convolutional network and the state itself to become an input to an RL algorithm. By doing that, the agent was conditioned on historical transitions, corrections and the current command.

Indirect methods, on the other hand, are often influenced by inverse reinforcement learning literature. A common architecture is a union of two modules, where the reward-learning module grounds language in a segment of a state or a trajectory, and the policy-learning module acts in an environment and receives rewards that are based on this grounding, e.g [37, 50, 94]. To show an example that follows the delineated scheme, Fu *et al.* [30] computed maximum entropy IRL update to recover the reward function. The main change was that the standard input was conditioned by computing a product of itself and a language description embedded with the use of LSTM gates. The authors showed performance of this algorithm in navigation tasks in simulated room environments.

For another example, Goyal, Niekum and Mooney [37] assigned rewards based on the correspondence between language instruction and taken actions. In more detail, their algorithm computed action frequency vectors, encoded the linguistic command, and fused

these two sources of information to predict the probability of a trajectory being related or unrelated to a given description. Under a special formula that takes into account the temporal aspect of action making, these probabilities shaped the original reward function (that always existed in their experiments) and favorably affected the final accuracy.

Bahdanau *et al.* [7] showed a method inspired by an adversarial mode of training instead of IRL, and jointly trained the policy and reward models. Concretely, the policy acted conditioned on the instruction and was optimized under the reward model, whereas the reward model itself was trained as a discriminator. It distinguished between instruction-goal-state pairs generated by the expert and instruction-state pairs retrieved from agent's trajectories. A similar adversarial approach was used by Agrawal *et al.* [2]. More examples revolving around reward shaping can be found in mentioned Luketina *et al.*'s review [49].

4.3.3 Instruction Following

The last category of algorithms which contain the models presented in this thesis regards instruction following. By no means is this category disjoint with others. We have already described a number of methods that set following commands as their main goal, c.f. [2,3,7, 19,21,22,54]. Their common elements are in having even a primitive sort of language from which commands are generated and in conditioning the agent's observations through computing representations of those commands.

Earlier approaches relied on object representations and exploited the structure of an instruction to correspond it to the entities in the world. The result of this procedure, as we noted at the beginning of this section, was a formal language that described which actions to take. Chen and Mooney for instance [15], made use of an established NLP machinery to transform navigation instructions into executable formal plans.

Later methods started to draw from the successes of deep learning and applied conditioning directly by computing language and state representations end-to-end. Misra, Langford and Artzi [57] applied LSTM layers to language input and alongside a history

of previous states and actions, conditioned vanilla policy gradient algorithm [91] to perform the desired high-level actions.

Wang *et al.* [94] encoded language with LSTMs and multiplying it by attention matrices conditioned an action module to take appropriate actions. Their algorithm learned through standard RL mechanisms but besides environment-dependent reward, the agent also received numeric signal from a matching critic that computed alignment between the command and the generated trajectory.

In the paper by Chen *et al.* [16], the authors considered language-conditioned image reconstruction problem to teach a robot navigation in real-life visual urban environment. Instructions transformed into a coherent representation by an LSTM network were concatenated to the output of intermediate layers within an encoder-decoder architecture. This data was used to ultimately predict the distribution over the location of a queried item.

Yet another paper that focused on visual processing and that incorporated instruction following is by Chaplot *et al.* [13]. Therein, to arrive at the final state representation, linguistic information was encoded and mixed with an input image by a fusion model. Specifically, the authors were using Gated-Attention units, and A3C algorithm for policy learning [58].

For even more references, the reader is advised to consult Luketina *et al.*'s survey [49].

4.3.4 Technical Realization of Conditioning

For state-of-the-art methods reviewed so far, the function approximators that predict actions based on states and context (language) are most often, if not always, neural networks. Depending on the whole setup for imitation learning, the practical implementation of conditioning may thus differ. In this work, as well as the majority of papers that revolve around CIL, the conditioning vector is somehow concatenated with the state vector or feature vector to produce predictions about the actions. There are, however, other approaches, seemingly the most general one being so-called Feature-wise linear modu-

lation (FiLM) modulation [71] which advocates the use of FiLM layers instead. Feature-wise linear modulation layers are parts of a neural network that take context (command) information as an input and to condition the computations compute two vectors: α and β. This is accomplished through mapping the output of the i-th layer in the network f_i to a number produced by the affine transformation $\alpha f_i + \beta$. This approach can be easily generalized to the tensor case when each dimension is transformed separately and proves especially effective in visual question answering, as reported by the creators of FiLM themselves.

4.4 Summary

As we mentioned in the introduction to this work, language serves as a great tool for conveying information that cannot be obtained by the agent through the interaction with the environment alone: the context. Like we have seen, the context can be grounded, used to learn the language-based reward or used to condition the computations (which could be thought of as grounding language in action). Contemporary researchers investigate if and how each of these methods could enable the transfer of knowledge from language corpora into learning tasks and the transfer of policies between environments. This thesis focuses on a part of the first open problem. We consider language generalization capabilities in an imitation scenario with multiple presented behaviors, and among other questions ask: does language enable learning behaviors unseen during training through imitation?

Chapter 5

Action-Based Language Acquisition Theory

Psychological literature concerned with language acquisition proposes a plethora of explanations on how humans develop their language competency. One important explanation, to which the model presented in this work conforms to, is the Action-Based Language Acquisition theory (ABL) introduced by Arthur Glenberg and Vittorio Gallese [36]. Generally, the authors adopt a view of ecological psychology which sees people as individuals constantly interacting with the self-contained environment. In their formulation motor control plays a key role in language, and its learning, understanding, and production come from the situated interaction with the world.

In this chapter, we show the correspondence between Language-Conditional Imitation Learning algorithm and Action-Based Language Acquisition theory. The overall congruence that the algorithm presents with respect to this theory proves its substantial representational power. Hence, here presented discussion aims to convince the reader of the value of L-CIL in isolation from its empirical, test performance presented in experimental section of the work.

Because both ABL and L-CIL rely on the notion of conditioning, we begin with reviewing psychological and neuroscientific evidence for conditioning in general. Works in

both of these areas are clearly important to ABL as a psychological theory. For L-CIL they provide support in two of its aspects: theoretical and technical. Psychological findings substantiate the use of conditioning as a statistical method to achieve dependency of one set of data on another. Our discussion will not relate to L-CIL in particular but shall rather show how its conceptualization is rooted in studies on human cognition. The existence and realization of conditioning in neurosciences supports the structure of L-CIL, which as we will discuss in the second part of the thesis, is implemented though artificial neural networks. The presented review will mostly focus on how conditioning information affects human processing stream, a mode that we reflect in our method. We close the chapter by moving to the description of ABL and its comparison with L-CIL.

5.1 Support for Conditioning: Psychology & Neuroscience

For the sake of clarity of this section, let's point out that postulating the dependence on context entails the act of conditioning. Empirical findings rarely inquire how additional information technically affects the processing stream, however, in studies that do concern modelling, context is indeed understood as extra action-dependent information. Examples include Srivastava & Schrater [90] who use Bayesian inference to model choice conditioning on history of options; or Rigoli *et al.* [76,77] who adopt the same formalism and model valuation of choices via a Bayesian network that encodes context variables computing expected reward. Thus, findings suggesting dependency on the context at the same time support the existence of conditioning. This becomes even more evident under neuroscientific point of view. Chawla and Miyapuram [14] state the following:

> Broadly context modulation can be defined by the interaction between 2 different kinds of inputs: first consists of the feed-forward connections from the earlier areas in the preprocessing stream and second consists of the modulatory system that controls the system response to the driving inputs.

Such a description does not only agree with the general understanding of conditioning but is even compatible with the proposed L-CIL's implementation.

5.1.1 Psychology

In general, ample psychological evidence suggests that the context in which the data is acquired affects human reasoning processes. In the visual perception domain, for example, it has been shown that information of surrounding objects modulates how the target object is perceived [14]. Inspecting a handful of illusions, like Ebbinghaus illusion or Titchener circles [79], provides evidence that attributes of observed objects (color, shape, size, etc.) despite staying objectively the same, are regarded as different. Thus, extraneous perceptual data (context) that does not regard the target, heavily impacts what judgement is being made. Similar findings about vision being context-dependent are found in [92] where additional information determines interpretation of perceptual data or in a work by Otten, Seth and Pinto [66] who identify social information as another component modulating behavior.

In the field of decision-making, and that is how we could categorize imitation learning, so-called context effect is ubiquitous. A representative example includes Moore [59] who presented a psychological study in which participants were asked to judge the quality of options they had to chose from. With two special options where one was superficially worse but was in fact normatively superior and a second one was only superficially better, the results differed significantly depending on which one of those was rated first. If that was the superficially attractive option, reported preferences for both options were higher than in the case of a different ordering. Modulating information was in this case the valuation order. Other studies investigated context through a similar approach by focusing on economics [42,73,87] or by exploring the use of cognitive functions [53].

Trueblood *et al.* [93] showed that context effects, besides being strongly present in high-level decision-making research, are present even in simple perceptual tasks. In their experiments, participants were shown rectangular stimuli that differed in height

and width and were asked to select the one with the highest area. Objects were presented alongside decoy objects which invoked one of the three effects. The attraction effect occurred when two choice options $\{X, Y\}$ were enhanced with an additional X- or Y-similar option that was in all aspects inferior to other ones, and increased preference for the option implied by the decoy. In the similarity effect, X- or Y-similar decoy increased the probability for the dissimilar option to be selected. The compromise effect happened when a compromise option was introduced to the set, and eventually was regarded as the best one. In each of those cases, a decision was conditioned on the added element rather than made based on the whole set of options.

5.1.2 Neurosciences

A significant amount of accumulated data shows that conditioning is also a part of human neurological system. We start with the domain of perception, and then move closer to action taking and conditioning in general.

Meng, Cheria and Sinha [55] focused on face processing. They proved that the right fusiform area encodes neural responses to seeing faces and that the encodings depend on the category/context information. Other study by Petro, Paton and Muckli [72] discussed the connections of auditory brain areas to early visual cortex suggesting that sound may additionally modulate perception. Functional cortical connectivity seems to be affected by context also according to Galindo-Leon *et al.* [31].

In decision-making, neuronal correlates were identified independently by Breiter *et al.* [11] and Nieuwenhuis *et al.* [63] when studying simple gambles. The first group found that responses in the amygdala and *nucleus accumbens* seem to depend on context, whereas the second group identified reward-dependent regions (striatum, prefrontal cortex, posterior cingulate) as context-dependent.

Through the paper by Palmer and Kristan Jr. [68] we may bridge the gap between conditional imitation learning algorithms and neuroscience even more. In the cited paper, the authors gathered a plethora of sources for contextual data that affect action making

in animals, and partially brought upon how their localization in the brain. The authors discussed external context in the form of season changes, physical environment characteristics, social conditions, internal context encoding the behavior the agent is performing (running, walking, feeding, etc.), and importantly for language, task-related context. The authors linked task-related conditioning with the correlation of noise between spiking neurons [23] and sign inversion in the connection between secondary somatosensory cortex and prefrontal cortex [18].

A recent, fascinating study by Kumano, Suda and Uka [47] performed an even deeper analysis of task-related context and showed that neurons in the lateral intraparietal area (LIP) "accumulate relevant information preferentially". In the experiment, the scientists investigated brains of monkeys who performed switching between tasks on discriminating direction and depth. The analysis of LIP's activity was conducted based on the provided context, and it turned out that the activity only occurs when the presented stimulus is congruent with the assigned task. In this way, the authors concluded that the LIP area, that was believed to guide monkey's eye movements in the task, is a brain region modulated by contextual data. In this way, Kumano with colleagues extended the findings of Palmer and Kristan Jr. [68] by identifying a whole area neuronally dependent on the perceived context.

In the last paper we can read that:

> A hallmark of human cognition is the flexibility to select an appropriate action in response to identical sensory events depending on the environment or context. [47]

This opinion strongly supports the basis for using conditioning in learning algorithms. The results of this and other cited papers strengthen such methods even more as they indicate that conditioning is a part of our cognition stream in practice. On top of that, the previous section discussed that conditioning on context is also a psychological process. Since cognitive processes may involve higher cognition and since the most obvious

example of high-level context provider is semantic information, the discussed evidence seems to warrant our Language-Conditional Imitation Learning algorithm well. There is however more evidence in favor of the method introduced in this thesis, and we touch upon it in the next section.

5.2 Support for L-CIL: Action-Based Language Acquisition Theory

Action-Based Language Acquisition Theory (ABL) is a language-acquisition theory in the paradigm of ecological psychology. It assumes that interaction within an environment cognitively couples language with motor control and that these two areas make up for one synergetical system. In this section, we analyze ABL with respect to our Language-Conditional Imitation Learning algorithm to show how that it is explicitly grounded in research on cognition.

Formally, action-based language model makes use of MOSAIC and HMOSAIC theories of action control [38]. The fundamental entities in those theories are controllers (sometimes referred to as backward or inverse models) and predictors (forward models). A controller is a mechanism whose goal is to compute context-sensitive motor commands poised to satisfy some goal. Predictors, on the other hand, are used to predict effects of actions. In the nomenclature of imitation learning, controllers can be thought of as algorithms that learn proper behavior through IRL or BC, and predictors are models of forward dynamics. The action-based language acquisition theory assumes that when understanding a sentence, people associate controllers for articulation with controllers for performing a series of actions, and predictions for articulation with predictions for taking a series of actions. A synergy between both of those associations leads to grounding words in actions and vice-versa, actions in words. Consider Figure 2 which illustrates this idea by giving the word "drink" as an example. Glenberg and Gallese explain it the following way:

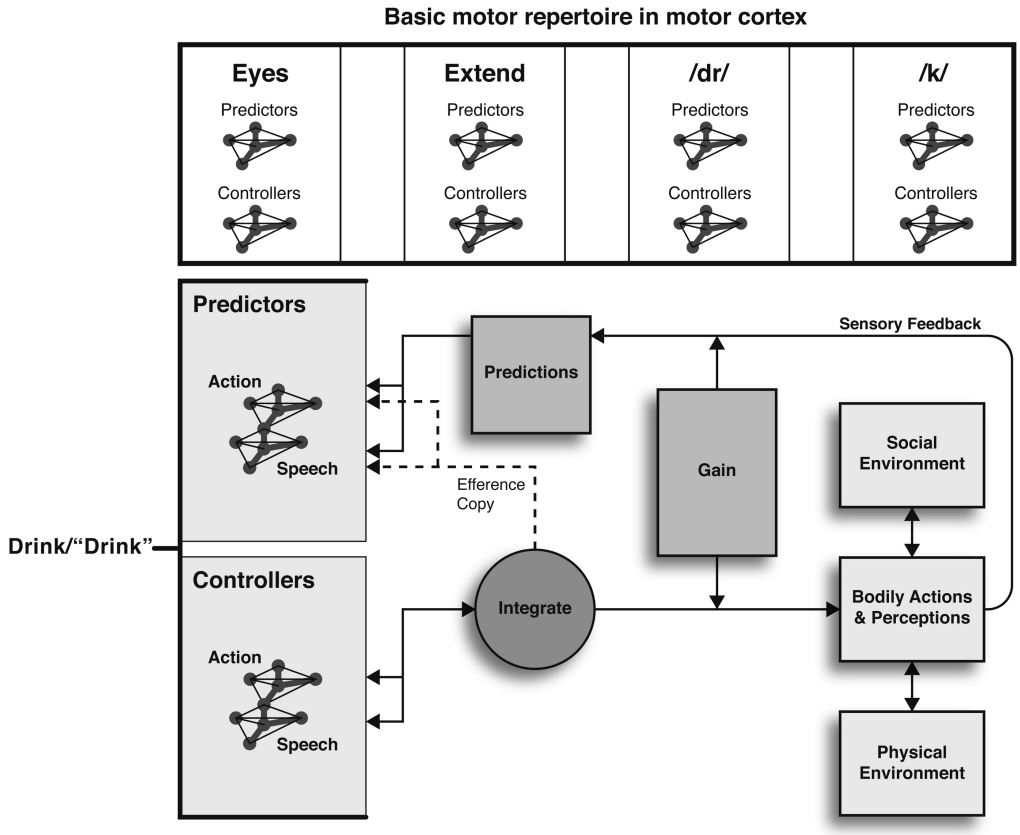


Figure 2: The ABL model for understanding the verb "to drink". Taken from [36].

> The overlap between the speech articulation and action control is meant to imply that the act of articulation primes the associated motor actions and that performing the actions primes the articulation. That is, we tend to do what we say, and we tend to say (or at least covertly verbalize) what we do. Furthermore, when listening to speech, bottom-up processing activates the speech controller, which in turn activates the action controller, thereby grounding the meaning of the speech signal in action. [36]

Note, that Language-Conditional Imitation Learning actually supports this construction under a specific set of assumptions and simplifications. Firstly, linguistic input in form of a sentence that was added to the expert's demonstration set does not have a particular modality. We might thus assume that sequences of word-vectors representing those sequences are in fact encodings of heard speech. In this setting, the controller

for speech articulation might be understood as the module *encoding* the input sequence, and the controller for action taking as the module *imitating* the action. Since encoding of the sentence is the "produced speech", *decoding* the sequence implements the functioning of the predictor. Finally, the second predictor for actions' results is unnecessary as in behavioral cloning the world's dynamics are given to the agent *per se* in the set of demonstrations. Hence, we might say that action predictors are implicitly included in L-CIL. Other elements in the graph from Figure 2 are social environment, bodily action and perceptions, and physical environment – these are all representations of the state the agent is in which exactly corresponds to data used in imitation learning in general. Gain and efference copy pertain to gain control mechanisms that strengthen or inhibit certain processes when the system has to deal with concepts not present in the physical environment. Since imitation through cloning does not assume such scenarios, provided discussion shows that L-CIL implements Glenberg and Gallese's theory and due to that, is strongly grounded in psychology and philosophy of language.

Importantly, ABL is also supported by neurophysiological findings. The authors focus on the existence of mirror neurons that send electrical spikes whenever an object-related action is performed or observed [34]. Later research proved the existence of mirror neurons systems that do not only encode goals of acts, but also retain information on their intention. Moreover, it turned out that such systems can be activated by auditory inputs related to actions or even verbal descriptions [32,33,78]. This evidence is also supportive of the L-CIL algorithm. Like we pointed out above, the sentence accompanying a state may be indeed understood as an input of another modality, and it becomes particularly congruent with ABL if it is assumed to represent speech.

Part II

Technical Contributions

Chapter 6

Language-Conditional Imitation Learning

In this work we are considering the setting described in Codevilla *et al.*'s paper [22], that is imitation learning through behavioral cloning. However, instead of using simple indicators as conditioning information, we will develop a new method that conditions on natural, written language. The title of this chapter corresponds to the algorithm that implements the idea behind this type of conditioning.

In the first section, we describe L-CIL at a high level and provide its formal description. The second section focuses on the implementation of L-CIL as a neural-network. The chapter ends by listing questions worthy of careful analysis, with the major one being: if we use the method to teach the agent behaviors $A_1, ..., A_n$ described with sentences $s_{A_1}, ..., s_{A_n}$, will that at the same time teach it a similar behavior B for which sentence s_B is composed of words or expressions from the s_{A_i}s? Put differently, can the method learn ambiguous behaviors so long as the language descriptions are available?

6.1 Overview and Technical Specification

Recall that any trajectory $\tau = \langle o_0, a_0, o_1, a_1, \ldots, o_t, a_t \rangle$ is a sequence of observation-action pairs over time t. The input to the algorithm consists of a sum of N sets of M expert trajectories, each generated for a different behavior: $\hat{\mathcal{D}} = \{\tau_i^j \mid i = 1, ..., N, j = 1, ..., M\}$ and N sets of sentences describing those behaviors, say $S_i = \{s_i^1, ..., s_i^K\}, \forall i = 1, ..., N$ and some $K \in \mathbb{N}$. Especially, we have that the number of fully formed natural language sentences in any S_i satisfies $K \gg M$. This means that there are multiple ways in which each behavior can be described, which reflects the ambiguous nature of natural language. Then, since we work in the paradigm of conditional imitation learning, appropriate sentences are randomly divided between trajectories so that each τ_i^j is paired with one, distinct $s_i^{\rho(j)}$ where $\rho : [M] \longrightarrow [K]$ is an injection into the set of indices $[K]$. Ultimately, this procedure results in a dataset

$$\mathcal{D} = \{(o_t, s_t, a_t)\}_{t=1}^{T} \tag{6.1}$$

of observation, descriptive sentence and action triples where $T = \sum_{i=1}^{N} \sum_{j=1}^{M} |\tau_j^i|$ is the total length of all the trajectories and $s_t = s_i^{\rho(j)}$ if $(o_t, a_t) \in \tau_i^j$. In this way, we expose the agent to the context embedded in a complex sentence s_t. Note, that the number of datapoints may be in fact much larger than the number of sentences – each trajectory could contain a significant number of observation-action pairs resulting in $T > NK$.

Our algorithm starts by creating a language model to represent words as vectors. It uses word2vec [56] to construct representations of words based on similarities between their neighborhoods. Sentences are then turned into sequences of vectors obtained using this technique. If v_ϕ is a function approximation for word2vec, and $s_i = \langle w_{s_i}^1, ..., w_{s_i}^{l_{s_i}} \rangle$ is a sentence of length l_{s_i} then it is transformed into

$$v_\phi(s_i) := \langle v_\phi(w_{s_i}^1), ..., v_\phi(w_{s_i}^{l_{s_i}}) \rangle, \tag{6.2}$$

and in consequence

$$\mathcal{D} = \{(o_t, v_\phi(s_t), a_t)\}_{t=1}^{T}. \tag{6.3}$$

With this data in the set $\mathcal{D}$ the algorithm begins the optimization process. Let $\mathcal{O}$ be the state space, $\mathcal{S}$ the discrete sentence space and $\mathcal{A}$ the actions space. Additionally, let $\ell_a(x_1, x_2), ell_s(x_1, x2)$ be loss functions that compare actions and sentences representations, respectively and let $\pi_i(\curvearrowleft)$ denote a projection of vector x on its i-th dimension. Finally, let $F(\cdot, \cdot; \theta)$ be a mapping approximating transformation $(o_t, v_\phi(s_t)) \underset{\theta}{\mapsto} (a_t, v_\phi(s_t))$ through parameters θ, where $o_t \in \mathcal{O}, a_t \in \mathcal{A}, s_t \in \mathcal{S}, t \in [T]$. In mathematical terms, instead of using supervised CIL learning objective from equation (2.20), the algorithm uses:

$$\underset{\theta}{\text{minimize}} \sum_t \ell_a \left(\pi_1 \left(F(o_t, v_\phi(s_t); \theta)\right), a_t\right) + \sum_t \ell^2 \left(\pi_2(F(o_t, v_\phi(s_t); \theta)), v_\phi(s_t)\right). \tag{6.4}$$

Using the sentence decoding as an auxiliary task to imitation relies on the low-level idea about the latent space. Namely, a space that enables to do both should focus hidden representations to map points standing for similar mixtures of behaviors to representations that are close to one another. Let's call this the *similarity property*. For example, driving behaviors represented by sentences $s_1 =$ "Go straight and turn right on the next intersection", $s_2 =$ "Drive straight and turn right on the upcoming intersection", and $s_3 =$ "Go straight and turn left on the intersection" compose of high-level behaviors "go straight", "turn right on the next intersection", and "turn left on the next intersection". The latent variables for those sentences should all lie relatively close to each other but $d(v_\phi(s_1), v_\phi(s_2)) < d(v_\phi(s_i), v_\phi(s_3)), i = 1, 2$, where d is a distance metric, should numerically preserve that s_1 and s_2 correspond to the same composite behavior. The *similarity property* property should allow mapping similar state-sentence pairs to similar actions, and in consequence, learn behavior B mentioned in the introduction, knowing only behaviors $A_1, ..., A_n$.

Mapping F is a composition of three modules: a representation module R that maps input to context vectors $(o_t, v_\phi(s_t)) \underset{\theta_R}{\mapsto} r_t$, a language decoder module L that decodes the context to input sentence $r_t \underset{\theta_L}{\mapsto} v_\phi(s_t)$ and an action module A that maps the observation conditioned on the context to an action $(o_t, r_t) \underset{\theta_A}{\mapsto} a_t$. All of the modules are function approximators whose parameters $\theta_R, \theta_L, \theta_A$ make up θ.

With that, F can be finally re-written as:

$$F(o_t, v_\phi(s_t); \theta) = \left(A\left(o_t, R\left(o_t, v_\phi(s_t); \theta_R\right); \theta_A\right), L\left(R\left(o_t, v_\phi(s_t)\right); \theta_R\right); \theta_L\right). \tag{6.5}$$

The pseudocode for L-CIL created to retrace the steps of Algorithm 2 can be found in Algorithm 3 box.

Algorithm 3: Pseudocode for Language-Conditional Imitation Learning

1: Collect trajectories and sentences describing them in $\mathcal{D} = \{(o_t, s_t, a_t)\}_{t=1}^{T}$;
2: For $s_t = \langle w_{s_t}^1, ..., w_{s_t}^{l_{s_t}} \rangle$ compute $v_\phi(s_t)$ for all $t \in [T]$;
3: Choose policy representation as $\pi : \mathcal{S} \longrightarrow \mathcal{A}$;
4: Parametrize π with an approximator $F : \mathcal{O} \times \mathcal{S} \underset{\boldsymbol{\theta}}{\longrightarrow} \mathcal{A} \times \mathcal{S}$ dependent on $\boldsymbol{\theta}$;
5: Set $\boldsymbol{\mathcal{L}} = \sum_t \ell_a\left(\pi_1\left(F(o_t, v_\phi(s_t); \boldsymbol{\theta})\right), a_t\right) + \ell^2\left(\pi_2(F(o_t, v_\phi(s_t); \boldsymbol{\theta})), v_\phi(s_t)\right)$;
6: Optimize $\boldsymbol{\mathcal{L}}$ w.r.t. the policy parameters $\boldsymbol{\theta}$ using $\mathcal{D}$;
7: **return** optimized policy parameters $\boldsymbol{\theta}$

6.2 Implementation

We refer to networks consisting of an encoder that projects the input data onto a space of smaller dimensionality, and a decoder that retrieves the original input from that space, as autoencoders. In the simplest linear case, given an input data $\boldsymbol{\mathcal{X}} = [x_1, ..., x_N]^\top$ consisting of N vectors of k dimensions $x_i \in \mathbb{R}^k$ we have the following definition.

Definition 12. *A linear autoencoder is a neural network with parameters* $\theta = \begin{bmatrix} \boldsymbol{\mathcal{W}} \\ \hat{\boldsymbol{\mathcal{W}}} \end{bmatrix}$ *where* $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{h \times k}$ *is a projection matrix onto the latent space* $\boldsymbol{\mathcal{H}} \in \mathbb{R}^{h \times N}$*, and* $\hat{\boldsymbol{\mathcal{W}}} \in \mathbb{R}^{k \times h}$ *is the inverse projection that results in* $\hat{\boldsymbol{\mathcal{X}}} \in \mathbb{R}^{k \times N}$ *where* $||\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}||_2 = ||\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{W}}}\boldsymbol{\mathcal{W}}\boldsymbol{\mathcal{X}}||_2$ *is minimized.*

The 2-norm $|| \cdot ||_2$ is a generalization of ℓ_2 MSE loss to the matrix case, i.e. ℓ_2 norm. We obtain dimensionality reduction by having $h < k$. In general, autoencoder consists of encoder network f_{θ_1} and decoder network g_{θ_2} with the learning objective such that $\ell(x, f(g(x))$ was minimized.

Now, L-CIL was implemented as a composition of a feed-forward neural network and an autoencoder. Let $\oplus$ denote vector concatenation. In our case, the encoder network implemented the representational module R, the decoder network implemented the language module L, and the sum of the modules $R \bigsqcup L$ made for the autoencoder. Feed-forward layers operating on a concatenation of $o \oplus r, o \in \mathcal{O}, r = R(o, v_\phi(s); \theta_R)$ for some $s \in \mathcal{S}$ implemented the action module A. The general structure of L-CIL is depicted in Figure 3. The particular structure of the autoencoder network was based on uni-directional Gated Recurrent Unit [17] and is further detailed in Figure 4.

Parameter-wise, all the feed-forward layers were of size 128, whereas the embedding layer had 32 dimensions. A mini-batch contained 128 elements, the learning rate was set to $3\text{e}{-5}$ and the weights were updated based on the Adam optimizer. The training time was set to 100 or 200 epochs since preliminary hand-run experiments in a simpler, yet similar environment, revealed that after that limit the algorithm begins to overfit. The same runs also indicated that the architectural setup is sufficient to achieve CIL-comparable performance with our algorithm. Due to limited time resources, a complete hyperparameter search on the final model was not performed. For more details on the hyperparameters and their respective values consult Table 6.1.

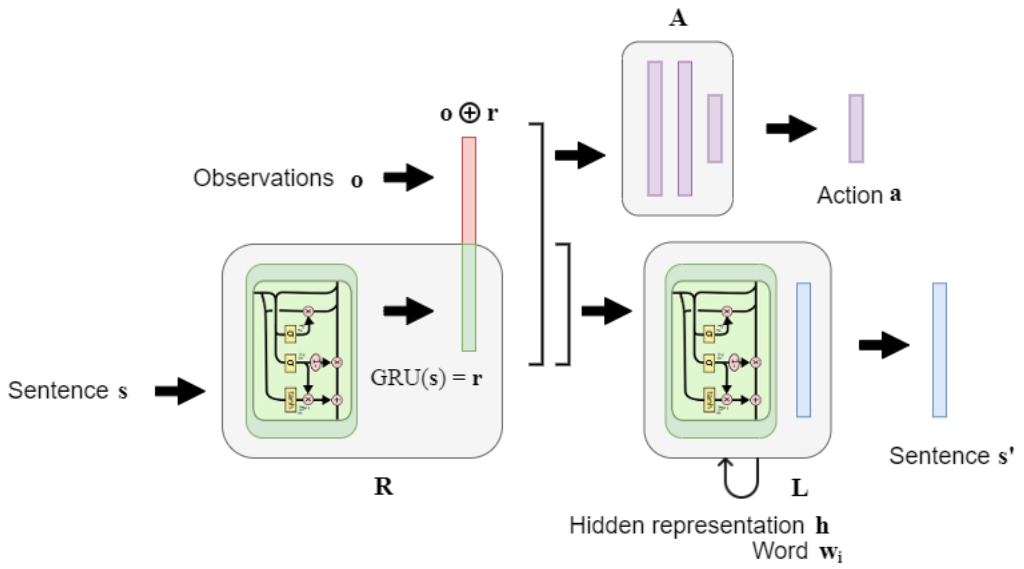


Figure 3: Network architecture for Language-Conditional Imitation Learning. The encoder-GRU implementing the representational module $\boldsymbol{R}$ transforms the input sequence $\boldsymbol{s}$ into an embedding vector $\boldsymbol{r}$ which is passed to the action module $\boldsymbol{A}$. There, it is concatenated with the state $\boldsymbol{o}$ and after being turned to a continuous representation, predicts the action $\boldsymbol{a}$. At the same time, the compressed vector $\boldsymbol{r}$ from the embedding layer serves as the starting point of the decoding procedure where the goal is to output a sequence of vectors identical to the input sequence, i.e. $\boldsymbol{s}$. In this way L-CIL grounds action in language and conversely, language in action, similarly to the assumptions present in action-based theory of language acquisition. The procedure of encoding and decoding $\boldsymbol{s}$ to itself follows the scheme from Cho *et al.* [17].

6.3 Open Questions

The purpose of this section is to list scientific questions that motivated research in language in imitation learning and eventually led to the creation of L-CIL. Note that the first two questions directly correspond to the hypotheses introduced in the Introduction, namely, asking if L-CIL will be able to generalize over unseen context and over unseen behaviors.

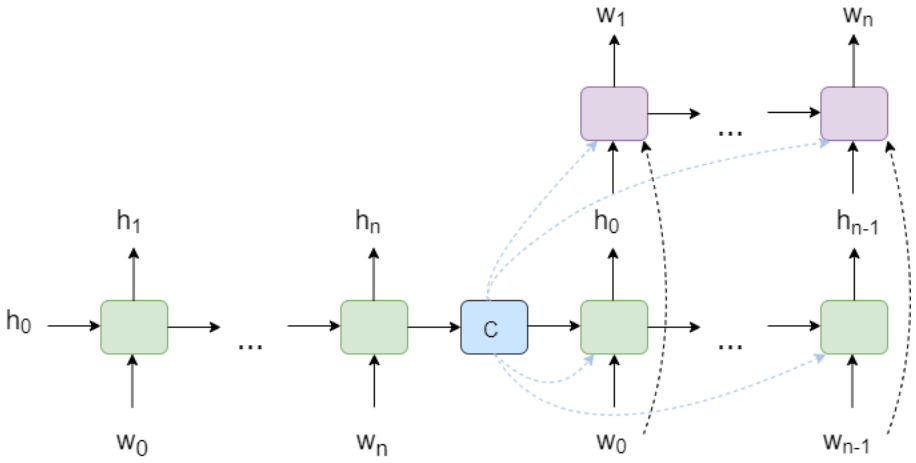


Figure 4: Scheme of data processing for the autoencoder network from Cho *et al.* [17]. The input sequence is transformed by a GRU cell to a context vector. During decoding another GRU cell accepts the computed context, the hidden state from the previous iteration and the previous word of the sentence (or the start-sentence token if none) as input and outputs the most probable next token. Note that one could extract the previous word from the input sequence directly or use the token predicted by GRU in the previous step. For each step of the decoding one of those methods is chosen randomly (i.e. *teacher-forcing* method is used).

Parameter	**Value**
Feed-forward hidden layers size	128
GRU hidden layer size	32
Mini-batch size	128
word2vec vector's size	8
Layers' initialization	$\mathcal{N}(0.1, 0.01)$, bias 0
Activation function	ReLU
Batch normalization	Yes
Learning rate	$3e{-5}$
Optimizer	Adam
Optimizer's parameters	$\beta_1 = 0.9, \beta_2 = 0.999$
Teacher forcing ratio	0.5
Epochs	100/200
Loss function	mixture of MSE and cross-entropy loss (eq. 6.4)
Programming language	Python 2.7
Library	pytorch

Table 6.1: Setup and hyperparameters of the L-CIL network.

Does it work? Firstly, CIL proved to offer strong performance in tasks with multiple high-level behaviors. The previous studies however, used a hand-made conditioning vector representation. It is thus natural to ask if conditioning on real language, in which divisions between the behaviors are more complex to extract, will be similarly successful.

Does it enable generalization? Secondly, language is general. L-CIL tries to address the question, "Does this inherent property increase the imitation algorithm's transfer capabilities?" Specifically, we try to check if learning a set of behaviors will enable the algorithm to follow unseen commands that use similar sentences but correspond to different behaviors. We also check how it performs with sentences containing unseen words.

Why does it do what it does? Thirdly, we elaborate on reasons for presented L-CIL's performance trying to understand its strengths, weaknesses and bottlenecks.

What are the limits of using it on its own? Finally, imitation learning methods are rarely employed for practical tasks autonomously, and are rather used as subcomponents. Our work seeks to discover the limits of Language-Conditional Imitation Learning as a standalone technique and based on that assess if it shows promise for industry-based applications where agents posses some form of a communication channel.

Chapter 7

Experimental Setup

In this chapter, we cover setup for experiments on the use of Language-Conditional Imitation Learning. The first section describes a simple car simulator that was used as the main testbed for measuring efficacy of L-CIL. The next section provides information on the task specification, testing conditions, and describes the data used for learning. The last section presents other methods which were used as a baseline for comparison purposes.

7.1 Environment

Monicar[1] is an autonomous-car simulator [70] written on top of Python's pygame package. It uses a 4-dimensional observation- and a 2-dimensional action space, where observations stand for the car's global position on the map expressed in $\langle x, y, \Theta, v \rangle$ coordinates of horizontal position, vertical position, angle and speed, and action is a $\langle a, \dot{\Theta} \rangle$ vector of linear and angular acceleration. Ranges for the horizontal and vertical positions of the moving car depend on a particular environment, Θ ranges from $-\pi$ to π and v takes values in the $[0, 20]$ interval. Both of the actions are limited to lie between -1 and 1.

Monicar allows for introducing multiple agents on the road with pre-specified behaviors and offers a range of maps appropriate for different styles of driving (roundabout,

[1] Named after its creator Monica Patel.

Figure 5: A sample visual output of the Monicar's environment.

intersection, one-lane, two-lane, merging, etc.) Moreover, there is also a possibility of specifying a custom feature function for state abstraction. A sample image of the simulator is shown in Figure 5. For the purposes of this study the only non-default element used in setting up the Monicar environment was the map constructed based on available examples. Even though a state representation different from the default $\langle x, y, \Theta, v \rangle$ tuple could have yielded more flexible experiment construction, technical difficulties prevented its use. Specifically, preliminary tests with beam state representation that measured distance to the edges of the road with 8 evenly distributed beams (one every $45°$) resulted in a very poor quantitative performance despite hyperparameter search. Due to time limitations this feature function was not eventually used.

7.2 Dataset Definitions

The input to L-CIL consists of two components: observations and sentences. In this section we describe the experiments that were conducted manipulating these inputs and list

behaviors and sentences used in them. We will also outline the approach for creating train and test sets that combined the two.

7.2.1 Behaviors

Let's define two disjoint collections of behaviors. The **multi-behavior** collection contains 6 relatively short driving behaviors which are as follows:

- go straight until a roundabout;
- turn right and go straight until the next intersection;
- turn left and go straight until the next intersection;
- make a u-turn;
- take the first exit on a roundabout and drive straight ahead;
- take the second exit on a roundabout.

The **composite-behavior** collection includes 2 longer driving behaviors:

- go straight until the last intersection turn left, go straight until the next intersection, turn left again and drive forwards until the roundabout;
- go straight until the second intersection, turn right and go straight;

Let's also introduce a third longer behavior with a trajectory fully made out of partial trajectories for composite behaviors. This is the **ambiguous** behavior and may be described the following way:

- go straight until the second intersection, turn left and drive forward until the roundabout.

An experiment, in this context, is formed by defining train and test sets. The precise method for selecting behaviors for each of these sets allows one to study a variety of attributes in each algorithm. We have created three experiments in this way:

1. The **Multi-confusion** experiment where the train set and the test set both contain the same mix of **6** behaviors from the **multi-behavior** collection. The question of interest for this experiment is whether a single method can learn to replicate the full variety of behaviors shown, without confusing between these, only cued by the provided sentences.

2. The **Composite-confusion** experiment where the train set and the test set both contain **2** behaviors from the **composite-behavior** collection. This experiment concerns the same problem as the **Multi-confusion** experiment but uses behaviors with longer trajectories.

3. The **Composite-ambiguous** experiment where the test set contains the **ambiguous** behavior and the train set comprises **2** behaviors from the **composite-behavior** collection. Here, the algorithms can only be expected to succeed if sufficient information can be extracted from the language given as context to enable a never-before-seen behavior to be executed.

To see the map used to perform the experiments with overlaid reference trajectories of the behaviors, refer to Figure 6. The actual trajectories used to establish the train and test sets according to the above specification were generated using a hand-made controller. Each behavior started at a point given by a uniform distribution of pixel values in $\{-20, ..., 20\}$ over the middle of the lane and a random offset in $\{0, ..., 150\}$ added to the vertical starting position ($x = 1$ or $y = 1$, depending on the direction of the movement). Then the controller took a state-dependent fixed set of actions, and approximately followed one of the trajectories shown in Figure 6.

7.2.2 Language

Along with the trajectories there were more than $32\ 000$ sentences that described the behaviors in the **multi-behavior** collection and over $600\ 000$ sentences for the **composite-behavior** collection and the **ambiguous** behavior combined. All descriptions were gener-

Figure 6: Monicar's map for the experiments. Arrows denote sample trajectories generated by the expert and colors separate different behavior collections. Street names are written in the middle of the roads. The trajectory with the yellow glow over it corresponds to the ambiguous behavior.

ated using a context-free grammar (see Grammar 1 and Table 7.2) and a sub-vocabulary of size 43 and 71, respectively. The sub-vocabularies were samples of the main vocabulary V that allowed us to create descriptions of every possible trajectory. It comprised 20 object words, 22 adjectives, 5 parts of proper names, 12 navigational words, 18 place words and 12 prepositions. The length of the sentences varied between 11 and 31. The linguistic input we created in this way followed a particular syntactic structure that did not allow free expression form as normally found in the speech. Although this could be seen as a limitation, our setup is actually on par with the most advanced studies on incorporating language in reinforcement or imitation learning (see Chapter 4). Due to that, we argue it should be rather seen as an advantage of the presented research.

⟨*AND*⟩ ::= 'and'

⟨*COMMA*⟩ ::= ','

⟨*TO*⟩ ::= 'to'

⟨*WITH*⟩ ::= 'with'

⟨*AT*⟩ ::= 'at the next'

⟨*UNTIL_YOU*⟩ ::= 'until you'

⟨*UNTIL_NEXT*⟩ ::= 'until the next'

⟨*TURN*⟩ ::= 'turn' ⟨*DIR*⟩
| 'make a' ⟨*DIR*⟩ 'turn'
| 'go' ⟨*DIR*⟩
| 'complete a' ⟨*DIR*⟩ 'turn'
| 'do a' ⟨*DIR*⟩ 'turn'

⟨*DIR*⟩ ::= 'left' | 'right'

⟨*STRAIGHT*⟩ ::= 'go' ⟨*DIR_STR*⟩
| 'drive' ⟨*DIR_STR*⟩
| 'head' ⟨*DIR_STR*⟩

⟨*DIR_STR*⟩ ::= 'straight' | 'forwards'

⟨*THROUGH*⟩ ::= 'through the' ⟨*N*⟩ | 'past the' ⟨*N*⟩

⟨*N*⟩ ::= 'next' | ϵ

⟨*REACH*⟩ ::= 'reach the' ⟨*N*⟩
| 'get to the' ⟨*N*⟩
| 'arrive at the' ⟨*N*⟩
| 'approach the' ⟨*N*⟩
| 'are at the' ⟨*N*⟩

⟨*INTERSECTION*⟩ ::= 'intersection'
| 'crossing'
| 'junction'
| 'interchange'
| 'crossroads'

⟨*LOCATIONS*⟩ ::= 'Swietokrzyska street'
| 'Marszalkowska street'
| 'Krucza street'
| 'Alley of the Independence'

⟨*ONCE*⟩ ::= 'as soon as you'
| 'once you'
| 'when you'

⟨*ROUNDABOUT*⟩ ::= 'rounadabout'
| 'rotary'
| 'rotonda'
| 'traffic circle'
| 'island'

⟨*FIRST_EXIT_IT*⟩ ::= 'take the first on it'
| 'leave it through the first exit'
| 'exit it at the first exit'
| 'turn right on it'

⟨*SECOND_EXIT_IT*⟩ ::= 'take the second on it'
| 'leave it through the second exit'
| 'exit it at the second exit'
| 'drive through it'
| 'pass it'

⟨*FIRST_EXIT*⟩ ::= 'leave the' ⟨*ROUNDABOUT*⟩ 'through the first exit'
| 'exit the' ⟨*ROUNDABOUT*⟩ 'through the first exit'
| 'take the first exit on the' ⟨*ROUNDABOUT*⟩
| 'turn right at the' ⟨*ROUNDABOUT*⟩
| 'make a right turn at the' ⟨*ROUNDABOUT*⟩
| 'go right at the' ⟨*ROUNDABOUT*⟩

⟨*SECOND_EXIT*⟩ ::= 'leave the' ⟨*ROUNDABOUT*⟩ 'through the second exit'
| 'exit the' ⟨*ROUNDABOUT*⟩ 'through the second exit'
| 'take the second exit on the' ⟨*ROUNDABOUT*⟩
| 'drive through the' ⟨*ROUNDABOUT*⟩
| 'pass the' ⟨*ROUNDABOUT*⟩

Grammar 1: Context-free grammar used to create the sentences.

The introduction of language allowed us to perform an additional experiment defined on top of the three others, namely:

4. The **Linguistic Generalization** experiment. This form of generalization tested if the algorithms (EL-CIL and L-CIL) once learned on some set of sentences, can imitate the same behavior described with synonymous sentences that contain words unseen during training.

For each collection of behaviors, 5 of the object words or adjectives from the vocabulary were held out to enable creating the set of **new-word** sentences. If used, they served only for testing. Otherwise, the algorithms were tested only on sentences that contained unseen combinations of known words. Note that combining **Linguistic Generalization** with the other experiments results in 6 experiments in total. This is shown in Table 7.1.

Test Sentences	Train → Test Behaviors		
	Multi → Multi	Composite → Composite	Composite → Ambiguous
Seen words	**Multi-confusion**	**Composite-confusion**	**Composite-ambiguous**
Unseen words	**Multi-confusion + Ling. Gen.**	**Composite-confusion + Ling. Gen.**	**Composite-ambiguous + Ling. Gen.**

Table 7.1: Classification of the conducted experiments based on the train/test behaviors specification and the type of test sentences.

7.2.3 Training and Testing Demonstrations

Formally, demonstrations took form of state-action-sentence triples. Let's denote the set of training demonstrations as $\mathcal{D}$, the set of testing demonstrations as $\mathcal{T}$ and the set of testing demonstrations accompanied by **new-word** sentences as $\mathcal{N}$. The procedure for establishing these sets iterated over all the behaviors defined in an experiment to obtain behavior-dependent sets $\mathcal{D}_b, \mathcal{T}_b$ and $\mathcal{N}_b$, and then outputted their sum.

In more detail, each behavior was presented in the form of 100 trajectories. Their lengths differed from 200 to 500 steps for behaviors in the **multi-behavior** collection, and from 600 to 1200 steps for those in the **composite-behavior** collection. The **ambiguous** behavior took 700 steps on average. For each behavior b, trajectories were separated in two sets D'_b, T'_b of 80 and 20 elements, and were accompanied by three sets of sentences: 80 training sentences $S(D'_b)$, 20 testing sentences $S(T'_b)$ and 20 **new-word** testing sentences $N(T'_b)$. The pairing of $S(D'_b)$ with D'_b, $S(T'_b)$ with T'_b and $N(T'_b)$ with T'_b followed the procedure described in Section 6.1 and resulted in the above-mentioned sets $\mathcal{D}_b, \mathcal{T}_b$ and $\mathcal{N}_b$, respectively. Since the **ambiguous** behavior received no training, the testing set $\mathcal{T}_{ambiguous}$ was extended to $\mathcal{T}_{ambiguous} = \mathcal{T}_{ambiguous} \cup \mathcal{D}_{ambiguous}$, and then reduced to contain only the state-action pairs from the ambiguous area of the intersection between Swietokrzyska st. and Marszalkowska st. Therein, both of the training behaviors, whose parts made

Sample training sentences
Go straight past the junction with Krucza street until you reach the traffic circle
Go straight ahead past the crossing with Krucza street until you reach the crossroads with Marszalkowska street, and make a right turn
Pull a one eighty when you reach the junction with Krucza street and go straight
Turn right on the roundabout and drive forwards
When you reach the roundabout leave it through the first exit and go forwards
Leave the roundabout through the second exit
Go straight, turn left to Swietokrzyska street, go straight, make a left turn to Marszalkowska street, and go straight ahead till you arrive at the roundabout
Head straight, do a left turn to Marszalkowska street, and drive straight until you approach the rotary
Sample testing sentences
Go straight ahead through the intersection with Krucza street till you arrive at the rotary
Do a right turn to Krucza street and drive forwards until you get to an intersection with Swietokrzyska street
Once you reach the crossing with Krucza street, do a left turn and drive straight
Do a u-ey as soon as you reach the crossing with Krucza street and go forwards
Do a u-turn and go straight
Once you arrive at the roundabout exit it at the first exit, and drive forwards
Turn right at the rotonda and drive straight
Head forwards and when you reach the rotonda take the second exit on it
Head straight, make a left turn to Swietokrzyska street, head forwards, complete a left turn to Marszalkowska street, and go straight ahead until you arrive at the rotary
Go forwards, go left to Marszalkowska street, and drive forwards until you get to the roundabout

Table 7.2: Samples sentences for training and testing L-CIL. Sentences for the ambiguous behavior are in italics.

up the **ambiguous** behavior, took very distinctive actions and language could have potentially provided promising impact on action prediction. By reducing $\mathcal{T}_{ambiguous}$ to only the ambiguous area we also remedied unfair comparisons against other methods. Those methods could reach low average generalization error over the whole trajectory by simply copying the expert's actions taken on the point spatially nearest the queried one.

7.3 Baselines

Baselines help to measure performance against known standards. Ablations might provide valuable insights about this performance by weighting the importance of method's components. The following chapter, besides showing performance of L-CIL itself, reports results obtained using three other algorithms. Given the structure of the presented model however, all of those methods might be actually considered as ablations. This is because they all miss at least one aspect present in the original formulation of the algorithm.

7.3.1 Behavioral Cloning

The most straightforward approach to our problem is to use BC itself. Of course, by gaining simplicity, we lose all the presumed perks connected to language. Since the only input comes from the observation now, there is no possibility of differentiating between behaviors in areas of the state space where the behaviors diverge. However, such areas might be extremely small. Thus, we hypothesize that in the best scenario BC actually learns to discriminate between the behaviors. In the worst case, it often outputs an action which is the average across seen actions and fails in the imitation task.

The behavioral cloning network that was used in this study consisted of 1 feed-forward hidden layer of 128 neurons and a ReLU activation function. This architecture was found to approximate isolated behaviors with high accuracy in the preliminary runs, and hence was used in the main experiments. Later in the text we will use the name BC only with respect to the network that implements it.

7.3.2 Conditional Imitation Learning

To discriminate between different trajectories residing in the training set one could use the original CIL approach and assign each trajectory a label (e.g. a one-hot vector) denoting its behavior. Theoretically, this should endow the method with categorization capabilities and allow predicting correct actions based on the labels associated with input states. This would fall short with an unseen behavior coming to the picture however. Without knowing the new label *a priori*, the usable information would come from the state only, turning CIL to a generic behavioral cloning algorithm. To test whether this is really the case, we used the vanilla Conditional Imitation Learning as the second ablation.

In technical terms, CIL was represented by a two-layered feed-forward neural network with ReLU activations, batch normalization, and 128 neurons in each layer. Besides the state alone, the input also included a one-hot conditioning vector that acted as a label. It was not used in its raw form but after being transformed through a hidden layer of 128 neurons and $\tanh$ activation function. Note that this structure was not branched as in the original paper, mainly due to the simplicity of the environment.

7.3.3 Encoder Language-Conditional Imitation Learning

Finally, we could indeed use language to obtain compositionality but instead of specifying the auxiliary task of decoding sentences, focus on just on the behavioral cloning part. This approach could posses the same qualities as L-CIL and could offer the same computational power more directly. To test that hypothesis, we set up one last method called EL-CIL (Encoder Language-Conditional Imitation Learning). The flowchart and the hyperparameter specification of the method is nearly identical to this of L-CIL. The only practical distinction is the lack of the decoding step and sentence outputting, i.e. the absence of the language module L.

Chapter 8

Results

In this chapter, we detail quantitative results of the conducted experiments and preview how the agent behaves when it is left to produce rollouts of the learned policy. The quantitative results help us to address the first three questions from Section 6.3: how good is the method, does it generalize and why it does/does not. The qualitative analysis concerns the fourth question: what are the limits of using our method, L-CIL, on its own.

8.1 Quantitative Results

We compare the performance of all the algorithms by analyzing sentence embeddings, decoding accuracy, and learning progress. To ensure reproducible findings, we have run repeated trials and performed statistical analysis on each outcome. The lack of evidence against normal distribution of data ($p > 0.5$), warranted the use of confidence intervals (CI). The shaded area present in all the plots in this section shows the 95% CI computed based on 10 independent runs (5 for BC), unless stated otherwise. Assuming X_i is the set of errors produced in the i-th step of the training iteration, the formula for bounds b_i of the interval was as follows:

$$b_i = \overline{X_i} \pm t_{.975} \sqrt{\frac{Var(X_i)}{10}},$$

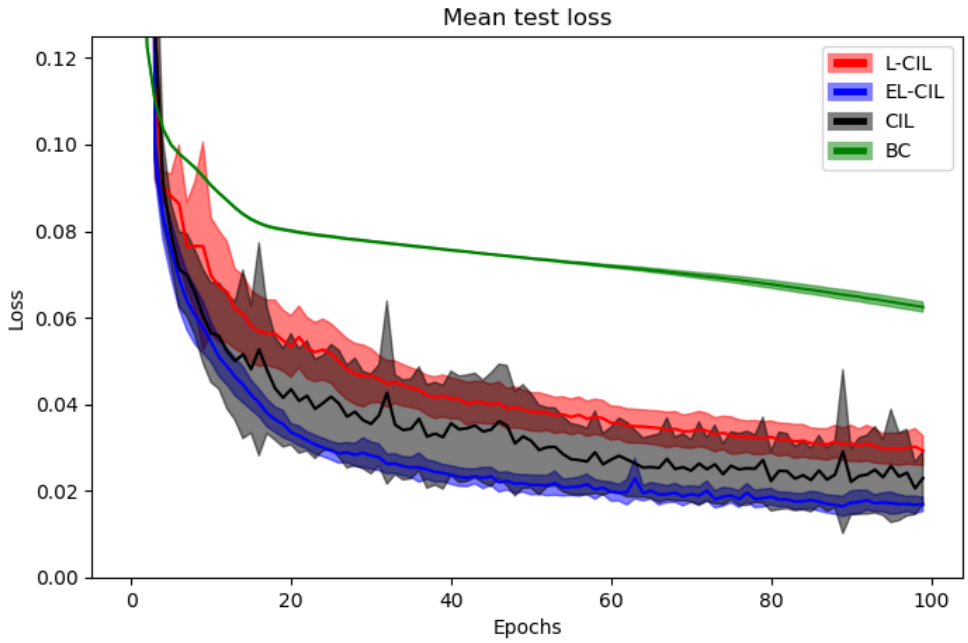
where $t_{.975}$ was the 97.5%-th percentile of the t-distribution with 9 degrees of freedom. We used the t-distribution due to a small sample size.

This section is divided according to the experiments – the **Multi-confusion** experiment, the **Composite-confusion** experiment, and the **Composite-ambiguous** experiment – to then move to the post-experimental analysis. In each experiment we additionally measure **Linguistic Generalization**, for this portion examining only the relevant methods which in fact depend on the form of sentences: L-CIL and EL-CIL. Notably, to show an upper-bound of any algorithm's performance in that scenario, listed figures also present the vanilla CIL algorithm which conditions on one-hot vectors which do not change between training and testing. Therefore, we expect CIL's curve to be identical for both cases.

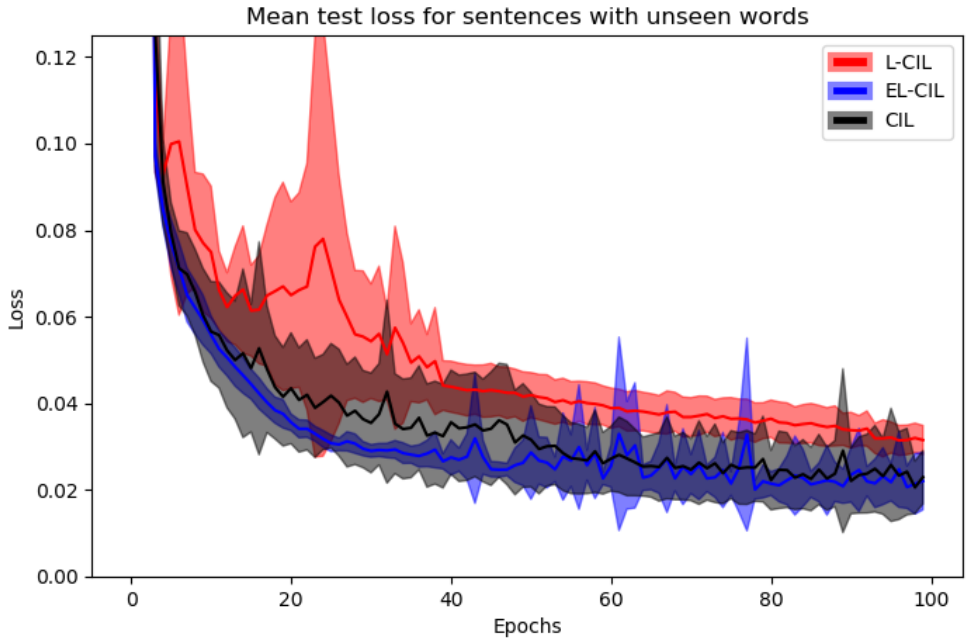
8.1.1 Experiment 1: Multi-confusion

The **Multi-confusion** experiment regarded reproducing a set of 6 training behaviors from the **multi-behavior** collection – driving straight, turning left or right on the intersection, turning around and leaving the roundabout on the first or the second exit. It tested how our algorithm compares to other competing methods and aimed to corroborate that behavioral cloning without any support of additional information fails in imitation tasks extended with conflicting training trajectories.

General Results The plot in Figure 7a reports the mean testing error and confidence bars for all the studied algorithms. BC visibly fails in the experiment due to its inherent inability to differentiate between the training demonstrations. It achieves the highest loss that is beyond the confidence intervals of other methods. For conditional algorithms, we observe that they share a similar performance. Although the results for comparison between L-CIL and EL-CIL seem to favor EL-CIL, the errors of both methods are largely encompassed by the errors of already-established CIL algorithm. Moreover, EL-CIL's low error is a natural consequence of focusing only on the action loss, whereas CIL's low error results from conditioning on simple, discriminative vectors. These two observations indi-



(a) Results for the **Multi-confusion** experiment.



(b) Results for the **Multi-confusion** experiment paired with **Linguistic Generalization**.

Figure 7: MSE loss between predicted and expert actions for all algorithms in the **Multi-confusion** experiment conducted without **Linguistic Generalization** (upper) and with it (lower).

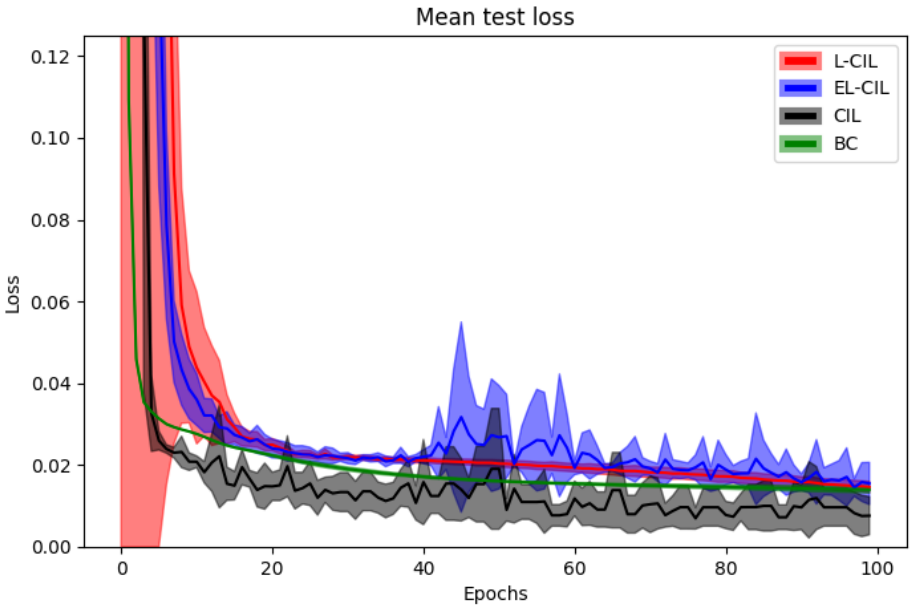
cate that our method does not fall short from either EL-CIL or CIL in this non-generalized task despite being slightly worse in numerical terms.

Linguistic Generalization The results of testing L-CIL and EL-CIL on sentences that contained new, unseen words are plotted in Figure 7b. For **Linguistic Generalization** we see larger error variation than before but very similar learning progress and consequently, the same relative ordering. Identically to the previous case however, even though EL-CIL seems to obtain lower loss than our algorithm, both scores are still strongly coinciding with the confidence intervals for CIL.

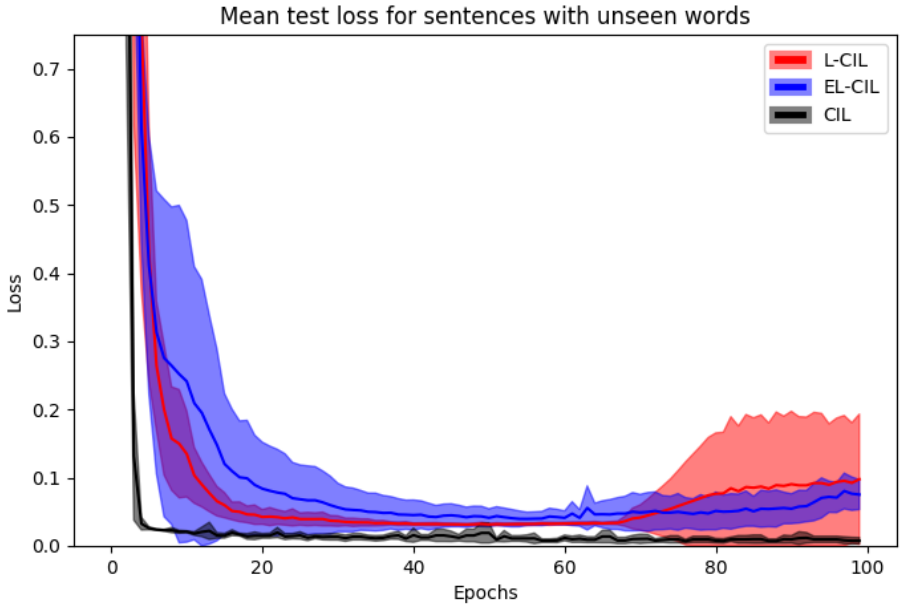
8.1.2 Experiment 2: Composite-confusion

The second experiment regarded reproducing 2 long driving behaviors from the **composite-behavior** collection (see Figure 6). Its purpose was to confirm the findings of the **Multi-confusion** experiment and by using more complex train and test sentences check how well the method we proposed, L-CIL, is adapted to language generalization.

General Results The mean testing errors and confidence intervals presented in Figure 8a indicate that vanilla CIL reaches the lowest error, and the other algorithms render approximately equivalent results. The key observations are that 1) BC exhibits surprisingly good performance, 2) the confidence regions are much narrower than before. Both of these effects are most likely related to the error averaged over long trajectories and the fact that demonstrations showed conflicting actions only in a small area of the map. Given that the relative error difference of our algorithm with respect to CIL is unchanged compared to the previous experiment, all this evidence leads to a conclusion that the here described experiment replicated the findings of the **Multi-confusion** experiment. In this way, we note that L-CIL is not only on par with other methods but is also robust to longer sentences.



(a) Results for the **Composite-confusion** experiment.



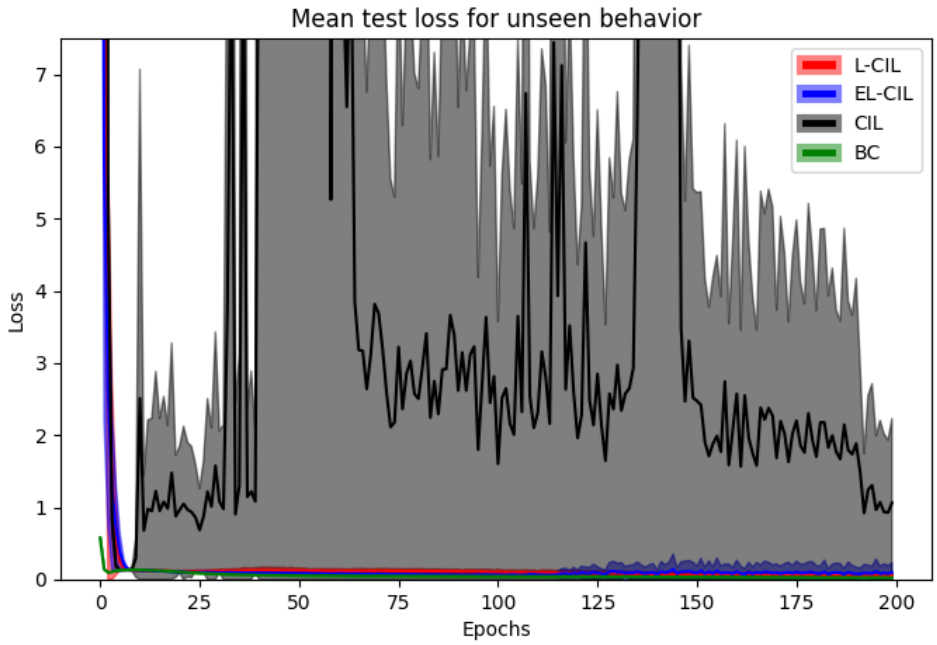
(b) Results for the **Composite-confusion** experiment when paired with **Linguistic Generalization**.

Figure 8: MSE loss between predicted and expert actions for all algorithms in the **Composite-confusion** experiment conducted without **Linguistic Generalization** (left) and with it (right).

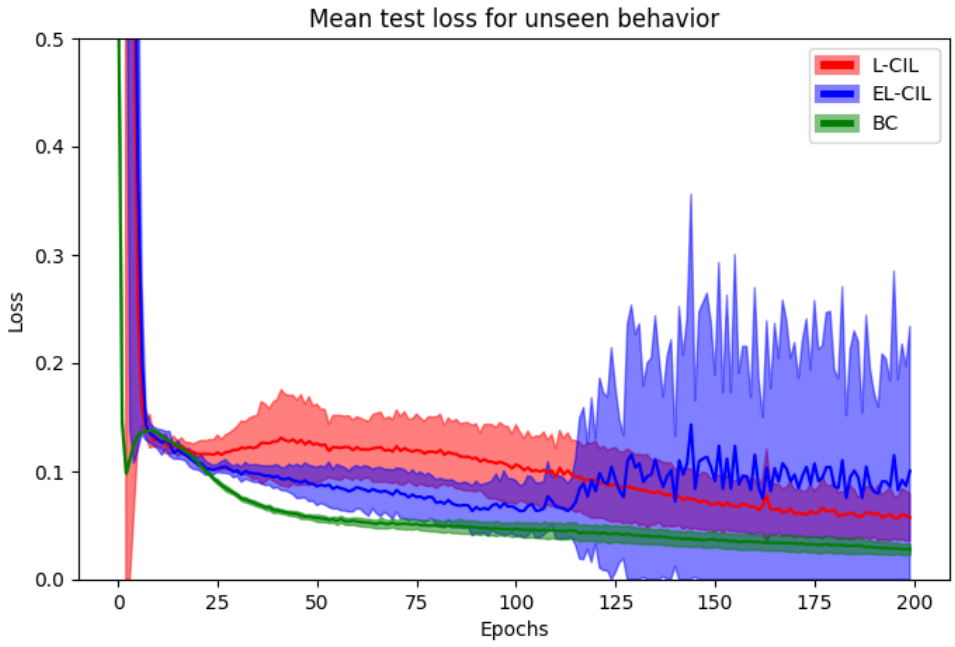
Linguistic Generalization Similarly, as in the previous case, the **Linguistic Generalization** experiment with the **composite-behavior** collection replicated the results of the **Multi-confusion** experiment. EL-CIL and L-CIL showed roughly equivalent performance. Even though the overall errors were higher than when using sentences with seen words, both methods seem to deal with the generalization on the level of new words well.

8.1.3 Experiment 3: Composite-ambiguous

Composite-ambiguous experiment measured generalization capabilities of all the tested algorithms by performing training on 2 long driving behaviors from the **composite-behavior** collection and testing on not fully seen, **ambiguous** behavior. This behavior followed a model driving trajectory that was a mixture of trajectories from the train set (see Figure 6). As it was mentioned in Section 7.2, errors for this experiment were measured only at the intersection between Swietokrzyska and Marszlakowska streets where the two training behaviors diverged. There, the training trajectories showed the most distinctive actions in approximately the same area of the state space. We hypothesized that the only source of information to predict correct movements for these states was encoded in the testing sentences. We assumed that neither CIL nor BC would be able to extract this information since CIL needs to condition on a new one-hot vector, unrelated to the training vectors, and BC can only use the representation of the state. The goal of this experiment was to check if our algorithm, L-CIL, manages to learn the **ambiguous** behavior, and see how it performs in comparison to the ablations, especially EL-CIL. Given the results of our preliminary tests, we decided to double the training time assigned to all the tested methods. Moreover, to simplify the task of learning the language model, we also introduced two variations of language conditional algorithms: frozen L-CIL (L-CIL f) and frozen EL-CIL (EL-CIL f). Both algorithms froze the weights of the encoder network to retain the highest possible quality of the hidden vectors. The former method did so after observing 3 consecutive increases of the sentence loss associated with the ambiguous behavior, whereas the latter method froze the weights after a fixed amount of 20 epochs.



(a) Results for the **Composite-ambiguous** experiment.



(b) Results for the **Composite-ambiguous** experiment – L-CIL, EL-CIL and BC. Note: expanded form of the image above.

Figure 9: MSE loss between predicted and expert actions for algorithms in the **Composite-ambiguous** experiment conducted without **Linguistic Generalization**.

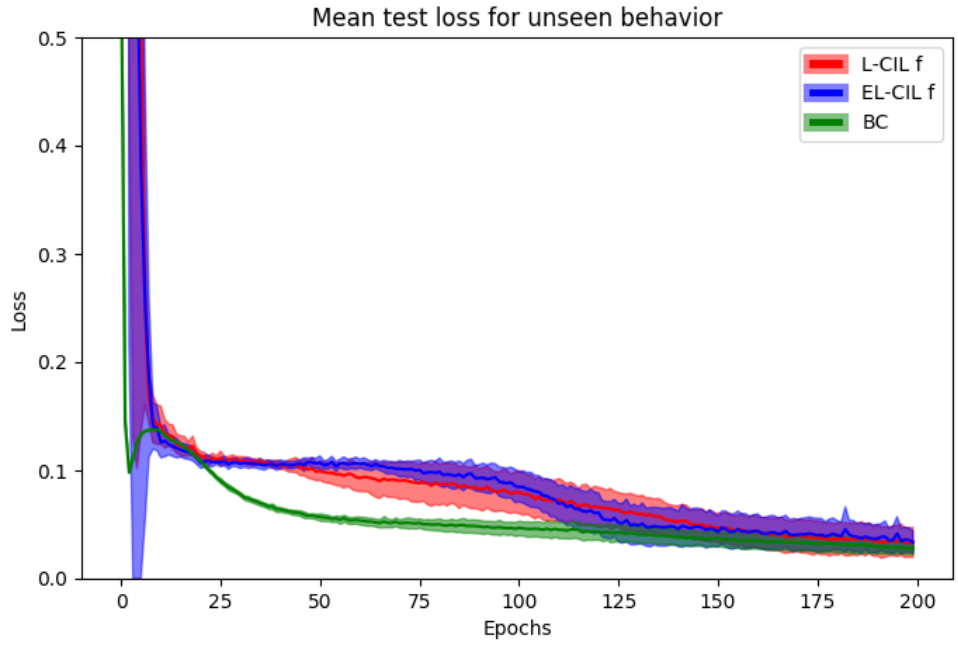
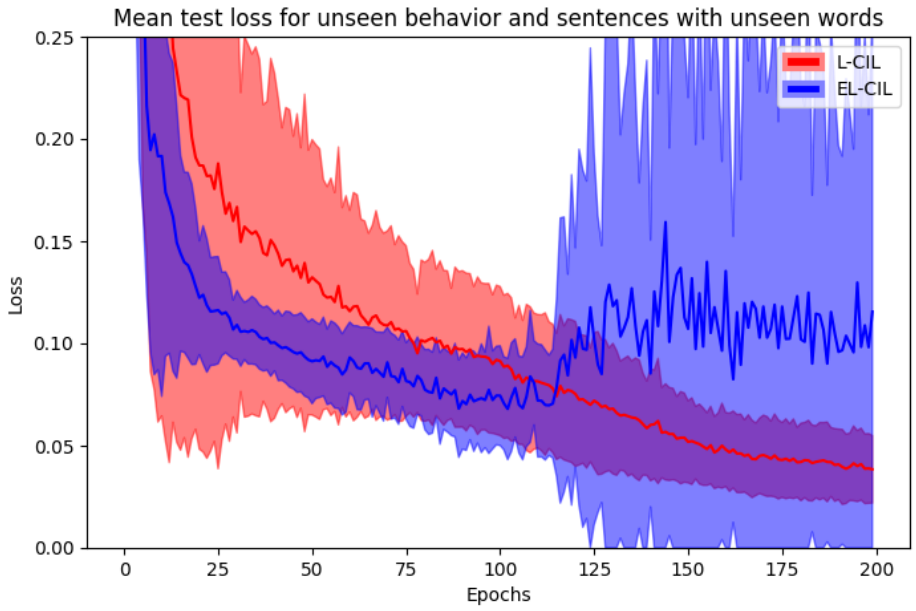
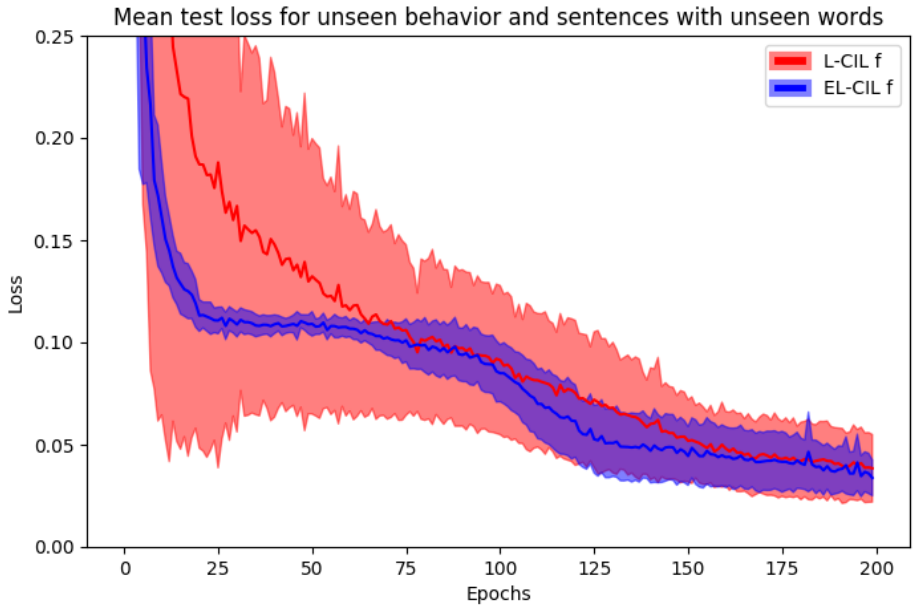


Figure 10: Results for the **Composite-ambiguous** experiment for frozen L-CIL, frozen EL-CIL and BC.

General Results In Figures 9a and 9b one can find two MSE learning curve comparisons. The first one shows the performance of all the algorithms, but extremely large errors of CIL, which diverged during learning, hide important data. The second plot displays errors of only L-CIL, EL-CIL and BC. According to our pessimistic hypothesis, BC seemed to extract enough information from the state in order to provide very good numerical approximation to the optimal actions. Both EL-CIL and L-CIL performed worse than BC. To see if we can remedy this unfavorable result, we tested if increasing the quality of L-CIL's encodings affects the ordering of the methods. We tried to achieve better encodings by freezing the encoder network. The plot visible in Figure 10 shows the results of this manipulation done for both L-CIL and EL-CIL. We can observe that all algorithms reached equivalent performance and were much more stable than before. Moreover, both algorithms succeeded in matching BC's performance despite more challenging optimization that also included language. It is even more impressive for L-CIL that needed to learn the decoding procedure.



(a) Results for the **Composite-ambiguous** experiment paired with **Linguistic Generalization**.



(b) Results for the **Composite-ambiguous** experiment paired with **Linguistic Generalization** for frozen L-CIL and frozen EL-CIL.

Figure 11: MSE loss between predicted and expert actions for algorithms in the **Composite-ambiguous** experiment conducted with **Linguistic Generalization**.

Linguistic Generalization Results for the task in which both the behaviors and the sentences were ambiguous once again depended on the version of the algorithms. As seen in Figure 11a, L-CIL was more stable for that task than EL-CIL. Mind that the error of CIL was as big as in Figure 9, and hence it is not included in the referred plot. However, after freezing the weights of the encoder to secure high-end sentence encodings, both methods achieved equivalent performance again (see Figure 11b). The error they eventually reach, albeit showing more variation during learning, is numerically comparable to the one obtained without **Linguistic Generalization**. Hence, here discussed experiment seems to show that language conditional algorithms are adapted to abstracting language even when the tested behavior is new. Moreover, the results strengthen the hypothesis that our algorithm exhibits the *similarity property* since it was able to follow the **ambiguous** behavior prompted by a sentence with new words. This could have been achieved only if to predict the actions, the method used similarity between unseen and training sentences.

8.1.4 Summary of Action Prediction Results

Table 8.1 summarizes all the mentioned results showing the mean of the error between predicted and expert actions attained at the end of the training. It was at 200th epoch for **Composite-ambiguous** experiments and 100th for all other experiments. The main information included therein is that our method, when properly modified, offers similar performance to other benchmark models, despite more challenging optimization scheme that also accounts for language. There is not enough evidence, however, to state whether it significantly helps following unseen, ambiguous behaviors. Most likely, this is due to an underlying issue with the experimental task that we will discuss in detail momentarily. Still, conditioning on language is superior to conditioning on one-hot vectors when it comes to ambiguity, as CIL failed in **Composite-ambiguous** experiment.

It is possible to notice some trends in the data. Firstly, BC is surprisingly accurate in the **Composite-ambiguous** experiment. Our hypothesis, which we will explore in the

Algorithm	Experiment					
	Multi-confusion		Composite-confusion		Composite-ambiguous	
	–	Ling. Gen.	–	Ling. Gen.	–	Ling. Gen.
BC	0.062*	-	0.014*	-	**0.028**	-
CIL	0.021	-	**0.008**	-	1.064*	-
EL-CIL	**0.017**	**0.022**	0.016*	0.075*	0.101*	0.116*
L-CIL	0.029*	0.032*	0.015*	0.098*	0.057*	0.069*
EL-CIL f	-	-	0.014*	0.056	0.034	**0.034**
L-CIL f	-	-	0.015*	**0.033**	0.033	0.038

Table 8.1: Mean error over actions for different experimental settings for all the methods and behaviors under scrutiny computed after 100 epochs or, for both of the **Composite-ambiguous** settings, after 200 epochs. Columns correspond to particular experiments conducted with ("Ling. Gen.") or without ("–") **Linguistic Generalization**. The value in bold in each column pinpoints the lowest mean error in an experiment. Asterisks indicate whether the difference with the lowest mean error was significant (asterisk) or not.

qualitative analysis of the results, is that too much information was included in the state of the world alone. A likely scenario is that the car was turning one direction earlier than it did the other direction and BC learned to differentiate samples of separate behaviors based on that. This was not possible in the **Multi-confusion** experiment, and hence BC obtained worse results, because there were many more actions which often occurred in the same area of the state space.

Secondly, EL-CIL is the best method in the **Multi-confusion** experiments where our train and test sets contain the same behaviors, and we test for differentiation between them. Its difference to CIL is nevertheless not significant. We speculate that EL-CIL is better than other algorithms because it is allowed to optimize the conditioning vectors to specifically predict the actions. This stands in contrast to L-CIL which optimizes two objectives at once, thus making itself more prone to higher errors, or CIL for which the vectors are fixed.

Thirdly, our L-CIL algorithm is significantly worse than the best method in the **Multi-confusion** and **Composite-confusion** experiments conducted without **Linguistic Generalization**. We have already said that this effect is caused by optimizing two loss functions and that it is most likely inconsequential in practice in some cases. For instance, error plots in the **Multi-confusion** experiment show that the error of both EL-CIL and L-CIL in large chunks coincides with the confidence interval for vanilla CIL. As a consequence, it seems reasonable to assume the practical performance will not differ for either of these algorithms. To elucidate whether the level of accuracy of our method is indeed sufficient, in the next section we study all the algorithms qualitatively.

Fourthly, both language methods exhibit generalization properties. Adding new words does not seem to strongly affect their final results, irrespective of the sentence length (the sentences in the **Multi-confusion** experiment were rather short in comparison to the sentences in the **Composite-confusion** experiment). More importantly, language conditional algorithms succeeded in the **Composite-ambiguous** experiment with **Linguistic Generalization**, and reduced the error to similar values as in the simpler version of this experiment. This shows that they are robust not only to changing words in the sentences or varying their length, but also to modifying their denotation.

Finally, freezing the encoder's weights seems to greatly affect the performance of language conditional methods. Both EL-CIL and L-CIL achieved significantly higher results after freezing the encoder in **Linguistic Generalization** experiments, and the standard **Composite-ambiguous** experiment. This also allowed them to perform with the same quality as BC for that experiment. It is valuable to notice, however, that freezing the weights is implemented robustly only in L-CIL, whereas in EL-CIL it is performed arbitrarily. It is unclear how to implement the same procedure for another set of behaviors, and without freezing the parameters, L-CIL f easily outperforms EL-CIL.

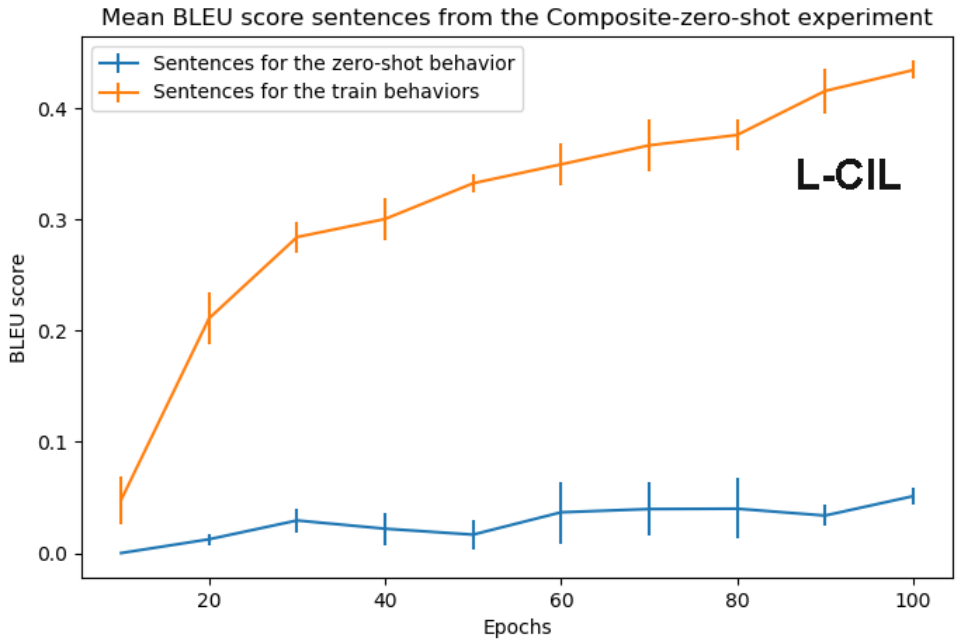
8.1.5 Language Encoding and Decoding Analysis

Results presented in the previous subsection seem to show that conditioning on language is on par with label-conditioning for **Confusion** experiments and does not do harm in **Composite-ambiguous** experiment, this time in contrast to label-conditioning. It is not clear, however, if our algorithm offers any advantages over EL-CIL (or its frozen version over EL-CIL f) and if its ability to do well in the **Composite-ambiguous** experiment comes from neglecting the language and focusing on the details of the state space (as seemingly BC) or not. Further analysis shows that this is not the case and L-CIL, in contrary to other methods, possesses the ability to correctly assess similarity between input sentences.

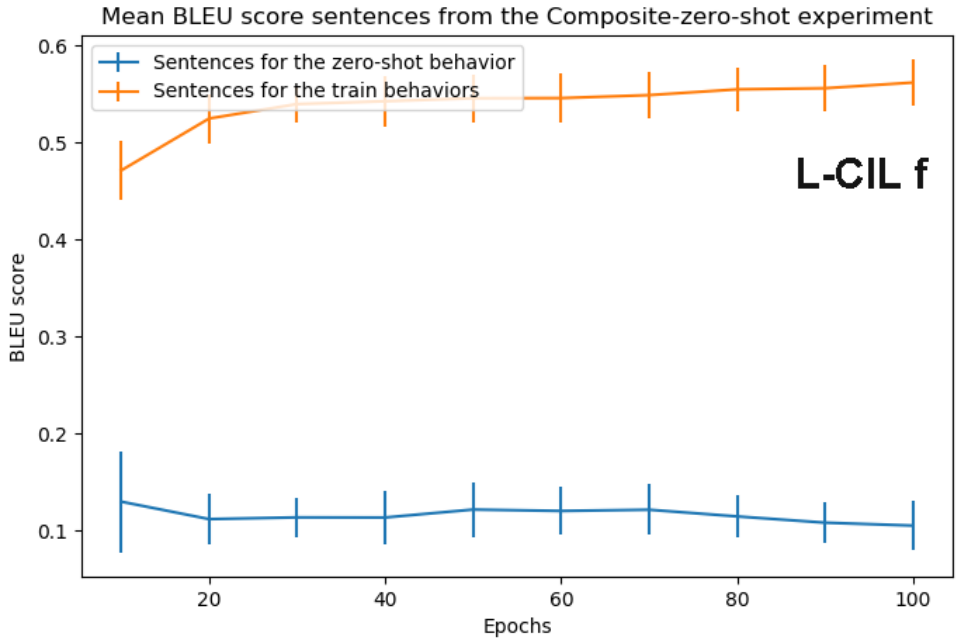
Initially, we looked at the sentence decoding error. We hypothesized that it was learning the language model that enabled L-CIL to outperform CIL in the **Composite-ambiguous** experiment, and that a better language model available for L-CIL f led to better performance. Formally, the output of our algorithm was a L-element vector of vectors v from the latent word2vec space. For n-element input sequence of words s, we defined the decoding to be a sequence of n first elements of v, say $v[:n]$. To analyze the quality of this decoding, we turned it to a sentence s' by mapping each element $v[i]$ to a word whose latent representation had the lowest euclidean distance to it. We measured the decoding error using the BLEU score algorithm [69] which numerically captures the quality of machine translation outputs. Let's denote $M(i)$ as the number of i-grams (i consecutive words of a sentence) that are identical between the input sentence s and the auto-encoded output s' and let $H(i)$ be the number of i-grams of s. Since in our case s and s' have the same length, the BLEU score $BLEU(s', s)$ was defined as

$$BLEU(s, s') = \prod_{i=1}^{4} \left(\frac{M(i)}{H(i)} \right)^{\frac{1}{4}}. \tag{8.1}$$

In this way, BLEU measured the geometric mean over percentages of i-gram matchings up to 4-grams.

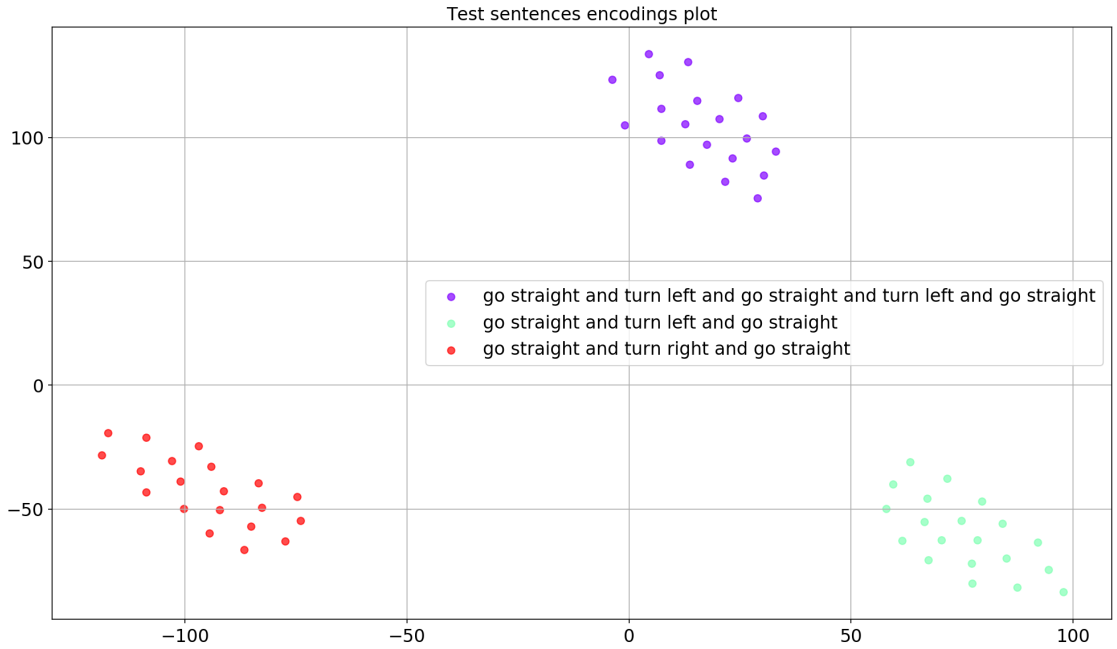


(a) BLEU score for L-CIL.

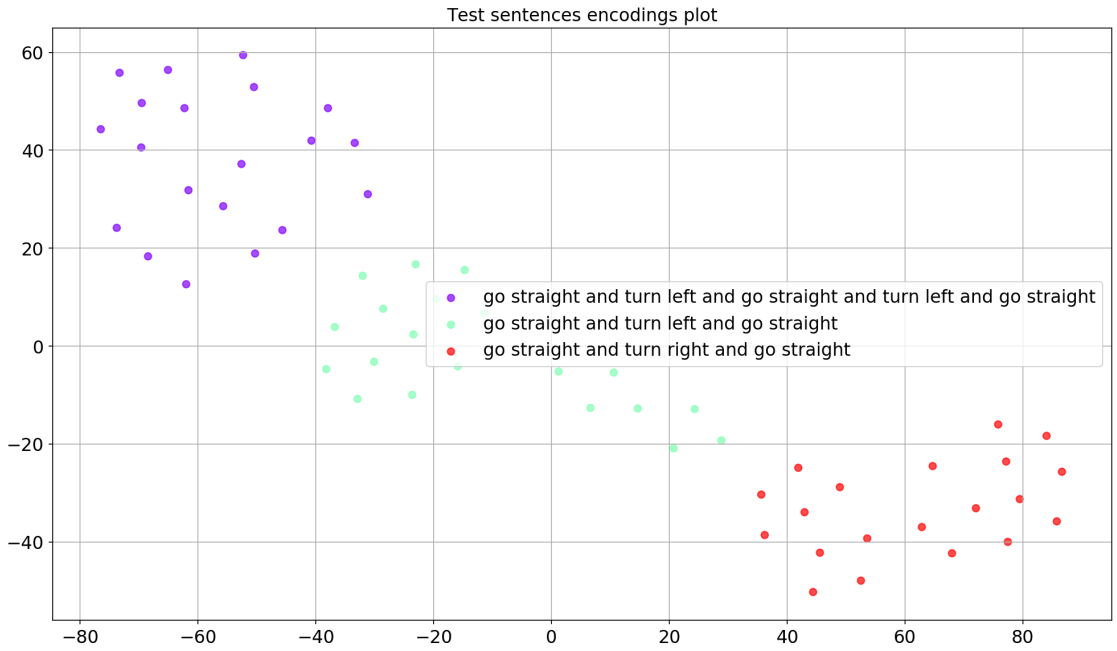


(b) BLEU score for L-CIL f.

Figure 12: BLEU score for the decoded vs target sentences corresponding to the behaviors from the **Composite-ambiguous** experiment computed for the L-CIL and L-CIL f algorithms. Vertical bars indicate 95 % confidence interval computed over 5 runs.

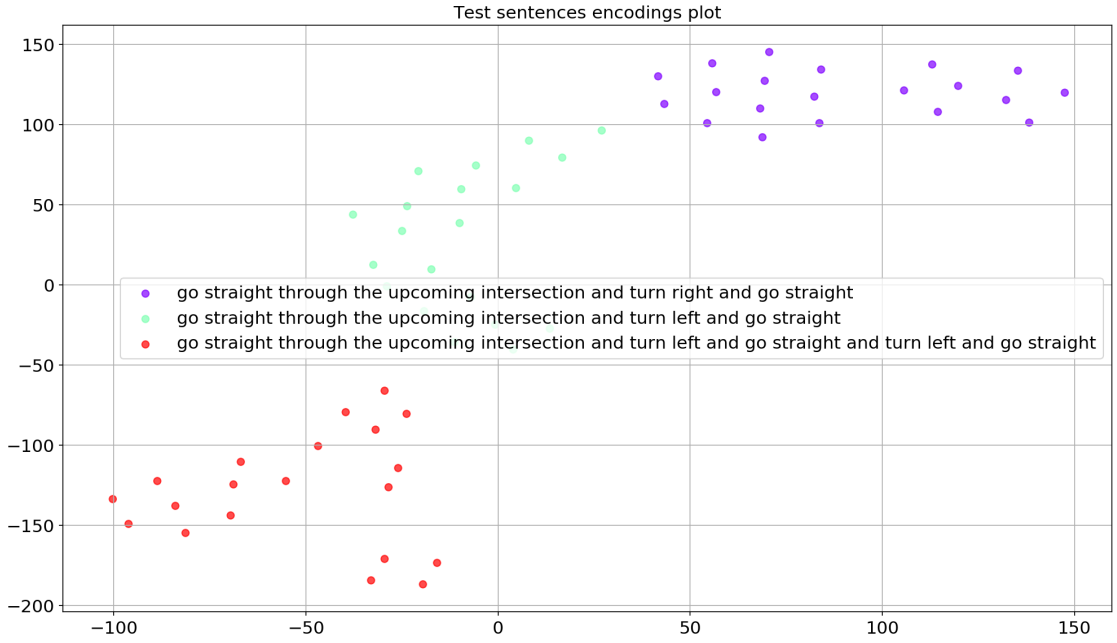


(a) EL-CIL

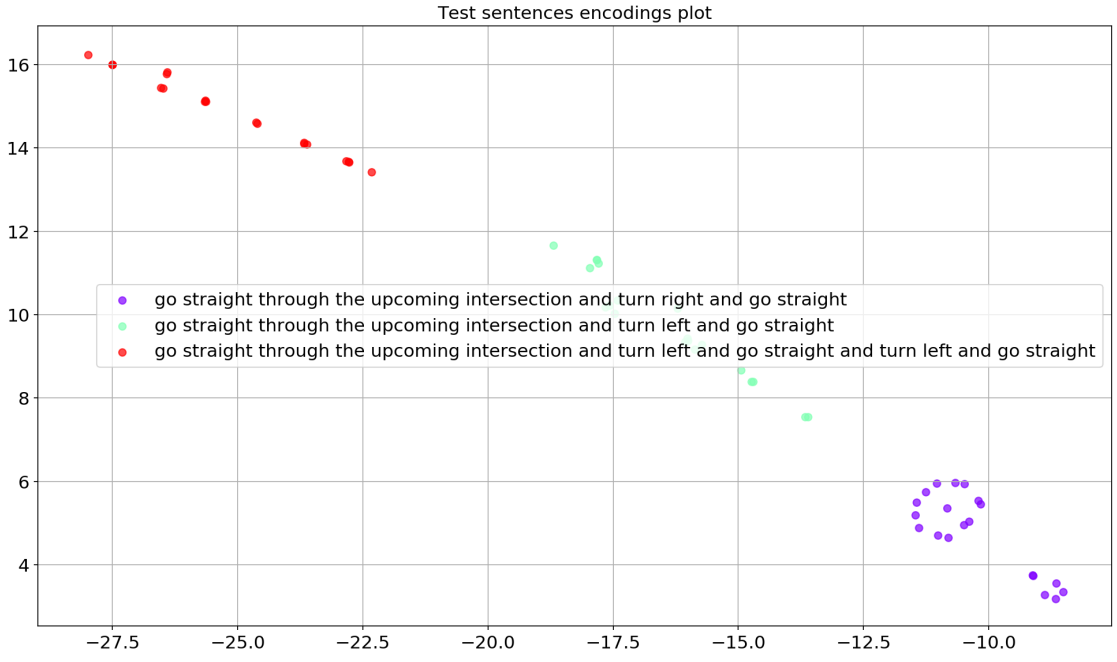


(b) L-CIL

Figure 13: Test sentence embeddings plotted on a 2D plane produced by trained EL-CIL (upper) and L-CIL (lower) algorithms. Different colors denote different types of behaviors. Light green points correspond to the unseen behavior *go straight and turn left and go straight*. Each point denotes a unique sentence.



(a) EL-CIL f



(b) L-CIL f

Figure 14: Test sentence embeddings plotted on a 2D plane produced by trained EL-CIL f (upper) and L-CIL f (lower) algorithms. Different colors denote different types of behaviors. Light green points correspond to the unseen behavior *go straight and turn left and go straight*. Each point denotes a unique sentence.

Figure 12a shows the BLEU score for the decoded vs target sentences associated with the behaviors used in the **Composite-ambiguous** experiment (2 train and 1 test behavior). What is striking is that the BLEU score for sentences describing the **ambiguous** behavior improves only slightly during training, in contrast to the BLEU score for train sentences which is clearly increasing. Similar findings relate to the frozen L-CIL algorithm for which the BLEU score is higher overall, but likewise static (see Figure 12b). This indicates that learning does occur, but it does not affect the performance of the algorithm evaluated on unseen data. Due to this finding, we decided to look further to explain L-CIL's ability to generalize.

A key enabler of our algorithm's performance is the quality of its encodings. Figure 13 shows T-SNE plots for several sentences' hidden representations produced by two language-conditional algorithms and projected onto a 2-dimensional plane. The embeddings are clustered well in both cases. However, for EL-CIL they are clearly independent, whereas L-CIL saves some information present in the linguistic cue. Thanks to that, its embeddings resemble those of the behavior which turns left at the ambiguous area (same as the **ambiguous** behavior), which in turn invokes similar actions. Note that CIL's one-hot vectors are also unable to generalize since they belong to independent dimensions of the latent space by definition. The importance of the encodings is replicated for frozen L-CIL and EL-CIL (Figure 14). Thanks to freezing, EL-CIL f generates embeddings with similar continuity as L-CIL in Figure 13b. However, they are very distant in the embeddings space, contrary to the embeddings found by L-CIL f (Figure 14b).

In total, our insights indicate that language conditional methods generalize thanks to high-quality language embeddings. The decoding step in L-CIL seems to further improve this quality and that is the advantage it has over EL-CIL. Although this advantage is not reflected in the final error either of the frozen method attains, in the next section we tested if it influences the qualitative performance of the algorithms.

8.2 Qualitative Results

To discover the limitations of L-CIL, its comparison to EL-CIL and its applicability at this stage of maturity we looked at rollouts obtained when running best versions of either of the algorithms in the training environments. Concretely, the qualitative data for **Composite-ambiguous** experiment was gathered using frozen EL-CIL and frozen L-CIL, whereas we used standard EL-CIL and L-CIL for the two other experiments. We also investigated surprisingly good performance on BC in the **Composite-ambiguous** experiment to see whether it averages the error between two conflicting behaviors or recognizes them based on small differences in the state values, as was our initial intuition.

By and large, our studies revealed that sensible movements are attainable for moderately short behaviors (e.g. used in the **Multi-confusion** experiment) and break down when the length of the behavior, and consequently the sentence, is increased. Despite that, L-CIL f shows promising performance when imitating the **ambiguous** behavior, whereas EL-CIL f does not. Hence, smaller differences between the sentence embeddings seem to cue L-CIL f to take correct actions. Tests for the **ambiguous** behavior also corroborated that BC's ability to reduce the test error comes from the fact it differentiates the states in which two training algorithms differ extremely well. Overall, by performing comparative analysis we find out that our algorithm offers high quality of the rollouts when not solicited by other techniques and could offer a valuable extension to existing software.

The best results can be found in Figures 15, 16 and 17. They show visual changes in the environment caused by the agent's actions when it was placed in one of the possible starting positions and given linguistic commands. Notably, all of the data discussed in this section was gathered using test sentences (without new words) and test states. In the case of sentences with new words used in the **Linguistic Generalization** experiments the errors were higher and behaviors were not as accurate, although one could notice similar trends.

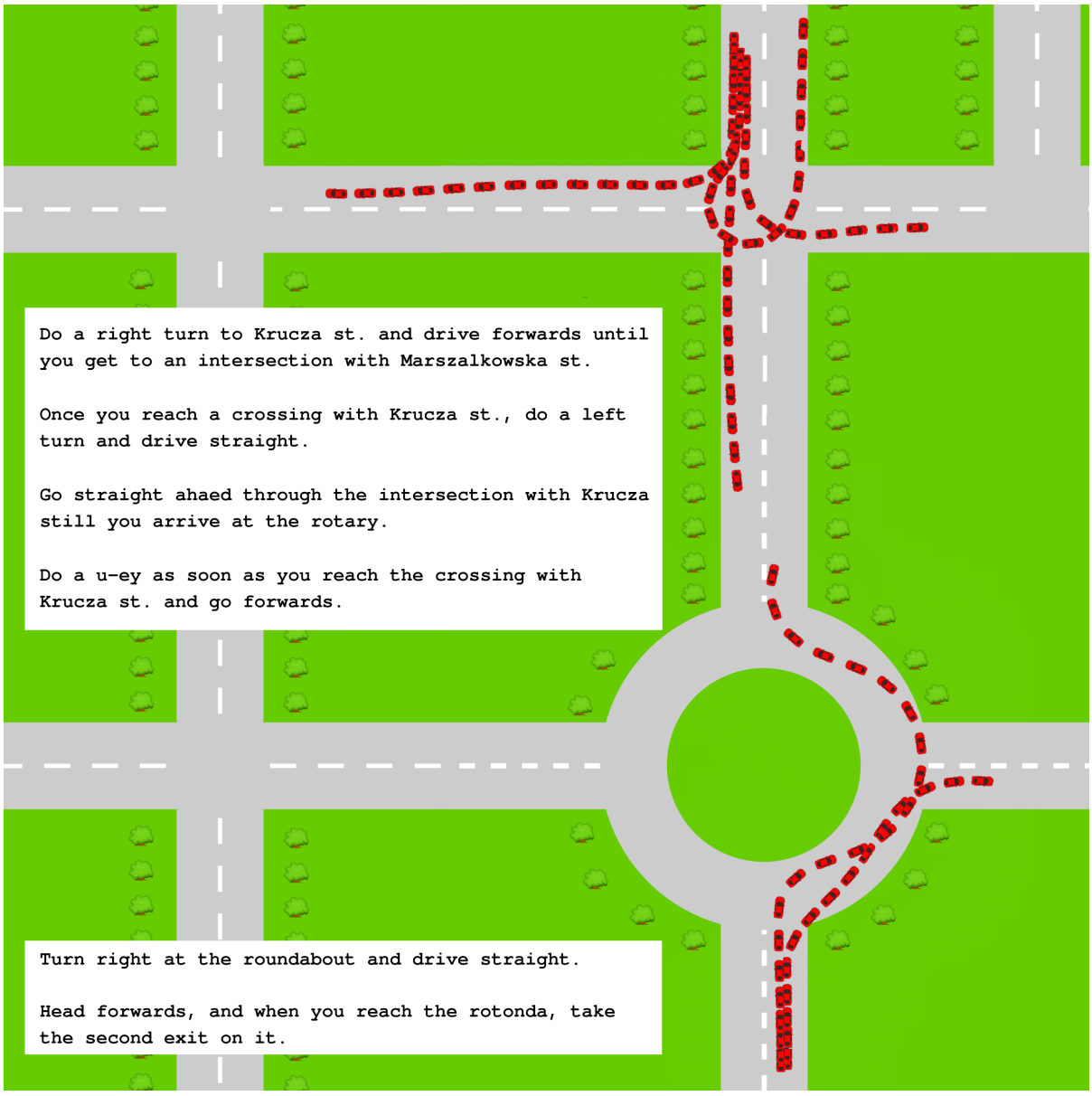


Figure 15: Sample rollouts of the **multi-behavior** collection generated by a trained L-CIL network. The image presents movements across multiple time-frames at once. It also shows sample sentences used to cue the agent, one per every behavior.

Results presented in Figure 15 were obtained by running the L-CIL algorithm as this proved to be the most stable quantitatively. It should be pointed out though, that all of the conditional imitation learning algorithms followed similar trajectories for the **multi-behavior** collection. Despite some variations between their rollouts, overall deviations in comparison to the training trajectories were scarce and Figure 15 could just as well correspond to EL-CIL or CIL. Vanilla behavioral cloning, on the other hand, was unable to perform any sensible behavior, most likely due to too many conflicting datapoints. This supports quantitative results from the previous section and general findings on BC in this kind of setting, e.g. found in the original CIL paper [22].

For the **composite-behavior** collection, trajectories followed by agents irrespective of which algorithm they were trained with, were not that exact. Error in cloning built up rapidly and eventually led to a divergence from the model trajectory. Below, we present a collection of successful runs obtained with our algorithm that despite their overall correctness, are still far from being perfect, as can be seen in Figure 16 or Figure 17.

Note that the behaviors in the **composite-behavior** collection were longer and more complex than other behaviors. Consequently, so were the sentences that described them. Auto-encoding those sentences and predicting the action with the hidden vector was in this light a more demanding task than before. Distortion or failure in cloning the trajectories was presumably caused by the offset of this setup in the form of incremental error. Lengthy sentences affected the learning process directly by creating improper encodings. These, in turn, led to inaccurate action predictions and incremental changes in visited in comparison to known states. This might have eventually led the agent to explore previously unseen areas in which the actions were almost random.

When it comes to the rollouts of the **ambiguous** behavior, it allowed us to obtain a clearer picture of the inner workings of each algorithm. For simplicity, we tested this behavior only in the seemingly ambiguous intersection (see Figure 17). As expected, CIL failed in replicating the correct motion and was completely stuck in the starting point, drove backwards or swirled around the map. Interestingly, EL-CIL f behaved sensibly,

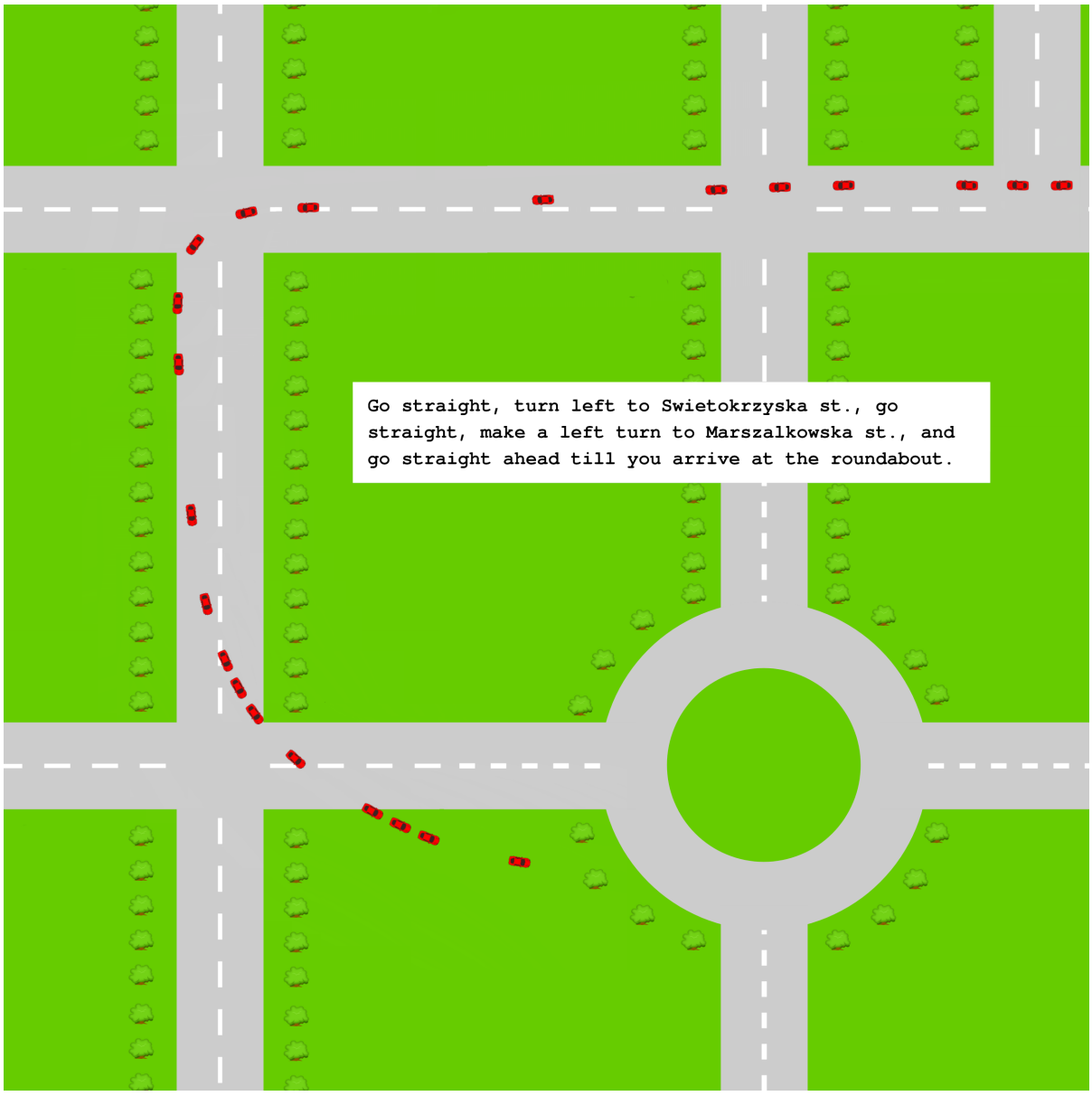
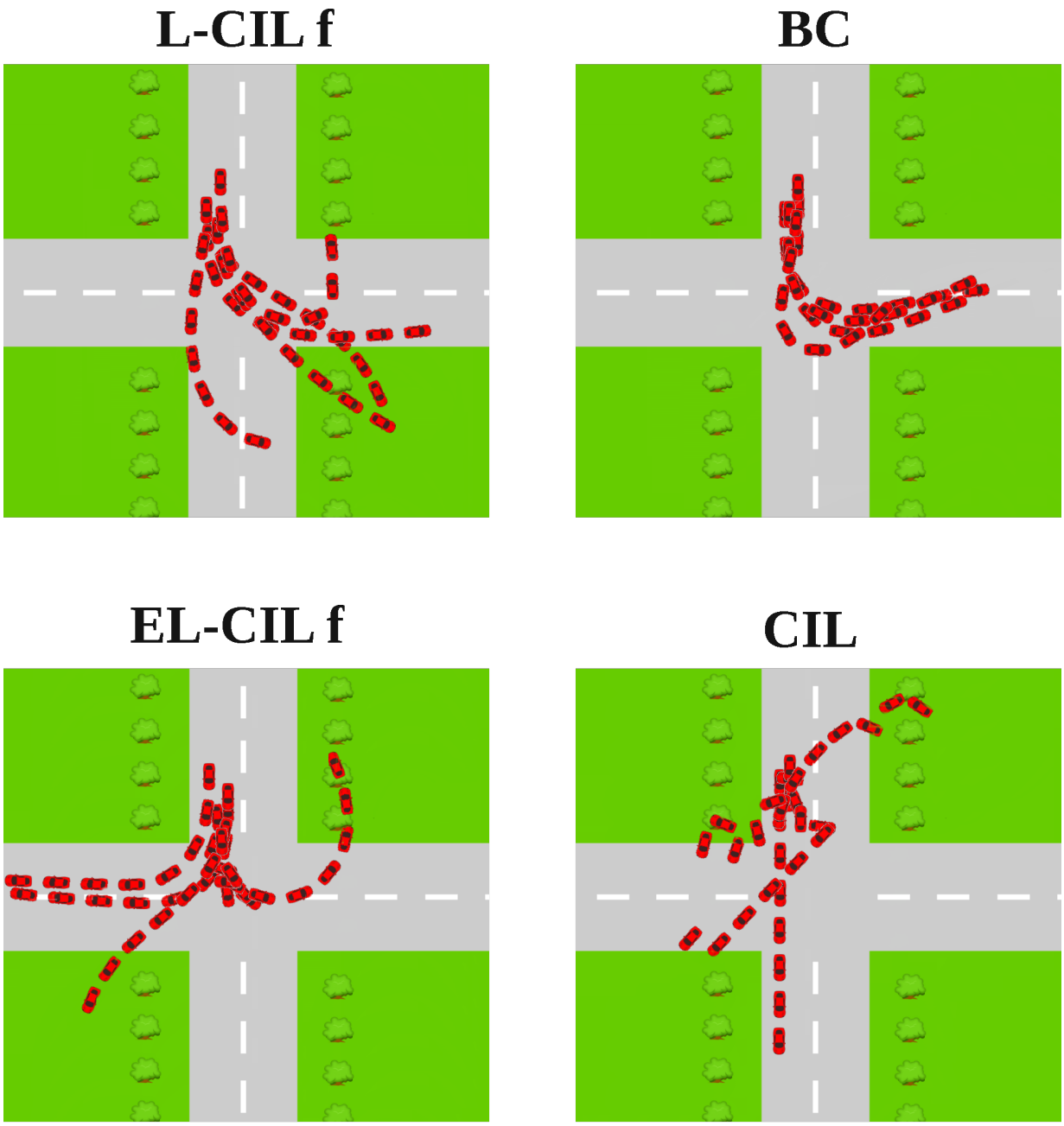


Figure 16: Rollout of the u-shaped trajectory from the **composite-behavior** collection performed by a trained L-CIL network. The image presents movements across multiple timeframes at once. It also shows a sample sentence used to cue the agent.



Go straight, **do a left turn to Marszalkowska street** and drive straight until you approach the rotary

Figure 17: Partial rollouts of the **ambiguous** behavior performed by each tested algorithm. The image presents movements across multiple time-frames at once. It also shows a sample sentence used to cue the agent. The correct movement was to turn left at the intersection which was only completed by BC and frozen L-CIL.

but was either able to perfectly mimic the left turn instead of the right turn or it drove forwards. This points out that huge differences between the embeddings seen in Figure 14a caused the algorithm to wrongly assess the similarity between the unseen and seen sentences. L-CIL f, on the other hand, whose embeddings were closer to one another, was always biased to turn the right direction and performed correct turns a number of times. Finally, BC showcased the most steady qualitative performance, proving that enough cues were present in the trajectories for this experiment to map almost every state to the correct action.

In spite of seemingly unfavorable comparison of our algorithm to BC and its poor performance for longer behaviors, it has to be pointed out that imitation learning methods are known to diverge when not used in a more complex framework. Because the differences of L-CIL with respect to an already-established algorithm CIL are scarce or even favor our method (see the above analysis of the **ambiguous** behavior), we conclude that our algorithm offers competitive practical performance and does have the potential for industrial applications.

Chapter 9

Discussion

L-CIL enables conditional imitation learning that uses full natural and diverse sentences as conditioning information with the analysis indicating its particular utility for generalization/transfer purposes. The former characteristic might be compelling for Human-Computer Interaction (HCI) practitioners since that method of training allows for more understandable communication on the expert side. Moreover, it also shows great promise for industry-based applications where users could specify queries to agents trained with the use of L-CIL algorithm and in return expect sensible behavior. Transfer capabilities are useful in machine learning as a whole and this work seems to prove that their inherent presence in language is compatible with imitation learning. This observation should guide future exploration of that area.

In this chapter, we will try answer questions posed in Section 6.3 and to some extent elucidate the performance of the tested algorithms based on the gathered data. Then we shall focus on future research in L-CIL, its extension and in IL based on language in general.

9.1 Addressing Open Questions

Does it work? **Confusion** experiments on the **multi-behavior** and **composite-behavior** collections indicate that L-CIL offers strong performance when imitating diverse behaviors. Even though it is not the numerically strongest algorithm, qualitative tests showed that the reported differences of at most 0.01 with respect to CIL or EL-CIL have no impact on its practical performance which, as we noted in the text, is indistinguishable from this of the "winning" algorithms. Thus, **yes**, L-CIL does work.

Does it enable generalization? The analysis we performed in the **Composite-ambiguous** experiment and the outcomes of the rollouts that were made, hint that L-CIL assesses similarity between the sentences properly, in contrary to EL-CIL. Additionally, **Linguistic Generalization** applied on top of all three experiments did not pose an increased difficulty for L-CIL. In fact, our algorithm performs roughly as well on new word sentences as those that only contain the words seen in training. Nevertheless, the experimental setup contained loopholes that allowed BC to excel in imitating the **ambiguous** behavior. This altogether indicates that, **yes**, L-CIL does enable generalization, but further tests are needed to measure the limits of this capability.

Why does it do what it does? Subsection 8.1.3 suggested that L-CIL learns well because it **retains similarity between sentences** when encoding them into conditioning vectors. As above, however, this question should be re-addressed in future work with a properly designed generalization study.

What are the limits of using it on its own? We have considered IL in raw acceleration and steering commands here, without DAgger, to emphasize pure learning capability. Qualitative performance of L-CIL in this form points out that **it is comparable to other imitation learning methods and the technical limits are similar for the standard multiple-behavior learning tasks**. It follows the desired short behaviors if they are de-

scribed by shorter sentences but diverges with longer behaviors and longer sentences. After freezing the encoder weights to improve the quality of the sentence embeddings, it also shows decent qualitative performance for the **ambiguous** behavior. Practical use of imitation learning allows to assume that further optimization, modified states, model-predictive control, and many other approaches could boost the final results and add for practical deployment.

9.2 Broader Comments on the Results

In this section, we pinpoint the most pressing issues to research and focus on more far-reaching interpretations of the obtained results. In general, we argue that L-CIL offers competitive performance in conditional tasks but there should be more studies to determine if it truly manages to make use of generalization inherent to language, as hinted by our analysis.

Value of Language-Conditional Algorithms The general idea behind conditioning introduced in the CIL paper was empirically tested to make a difference both in a simulation and in an embodied agent setup. Qualitative and quantitative results of our experiments seem to indicate this is also the case when conditioning information has a more complex structure, i.e. natural sentences. In other words, conditioning on sentence vectors may not fall considerably short from vanilla CIL and the evidence we gathered allows to conjecture that careful optimization may even lead to beating it.

Further tests on generalization capability After freezing the weights of the encoder network, L-CIL, EL-CIL and BC all attained comparable performance for the **Composite-ambiguous** experiment. Further analysis revealed that for language conditional methods this was most likely enabled by the high-quality of sentence embeddings. Then, in qualitative tests we found that embeddings computed by L-CIL and then L-CIL f are better than those computed by EL-CIL or EL-CIL f. The reason for that was because the embed-

dings for behaviorally similar trajectories were closer to one another. In total, this effect – caused by adding the decoding procedure absent in the baselines – allowed L-CIL f to reduce the error and imitate the unseen behavior well. Although this seems as a convincing line of reasoning, similarly correct performance of simple BC calls to further investigate generalization. In particular, all the algorithms should be tested in a more complex set of behaviors where it is impossible to draw enough data from the representation of the state alone. This should prevent BC from succeeding in ambiguity experiments and measure the real extent to which proper language-embedding helps L-CIL. It has to be pointed out though, that introducing such a set is a challenge in itself. The requirement is that the behaviors should be compellingly realistic, diverse, compositional to enable creating new behaviors, and at the same time ambiguous in a non-trivial way for a dynamical simulator.

Similarity Property As it was hinted in the previous chapter, a likely explanation for L-CIL's performance is that it uses similarity between input sentences to produce actions, and retains that similarity in the embedding space. This would be an exact consequence of having the *similarity property* from Section 6.1. To elaborate, an improved version of L-CIL – frozen L-CIL – performed quantitatively and qualitatively well in the **Composite-ambiguous** experiment where we had 2 training behaviors (let's call them behaviors A and B) and 1 test, **ambiguous** behavior. In the problematic intersection area (c.f Figure 6) only behavior A followed the same trajectory as the **ambiguous** behavior, whereas B was the confusing behavior. Now, sentences we generated for the **ambiguous** behavior were structurally alike the sentences for behavior A with a BLEU score of 0.28 vs.0.04 for sentences describing behavior B. It seems that L-CIL f's performance stemmed mainly from this very fact. The *similarity property* is also the cause for L-CIL/L-CIL f's results in the **Linguistic Generalization** experiments. New-word sentences were encoded into less informative vectors on which the conditioning provided less stable outcomes, but they were mapped onto representations within proper clusters irrespective of the length

of a sentence or the type of the behavior it corresponds to (train or test). This is visible in Figures 7b, 8b and Figure 11. Still, considering the success of standard behavioral cloning, the full explanation of whether L-CIL does possess *similarity property* should take place in further work with more demanding set of behaviors.

L-CIL: Outlook for Performance with Long Trajectories Failure in obtaining very good qualitative performance in the **Composite-behaviors** collection provides a great direction for future work. The most likely explanation is that our sentences and trajectories were too long, which led to positive feedback cycles in error and worse quality of action-prediction. Nevertheless, the presented qualitative results of L-CIL allow to assume that deepening the network, doing further parameter search or choosing a proper set of sentences might enable the algorithm to do a full rollout in that scenario as well.

9.3 Ideas for the Future

Future work should begin with addressing two important drawbacks of our studies. Firstly, it should introduce a more complex set of behaviors so that the ambiguous behavior was not easily cloned based on the state representations alone. This would present the real impact of L-CIL's language decoding in generalization experiments and could confirm if the insights we gained in this thesis were correct. Secondly, it should focus on hyperparameter optimization which, as already mentioned in the method chapter, was not performed exhaustively in this study and could improve both quantitative and qualitative performance of the algorithm. Additionally, there are numerous directions that should be researched to gain more information on the general quality and characteristics of our Language-Conditional Imitation learning algorithm. They include research on robustness to different forms of sentences, increasing practical usefulness and extended testing. We also mention one possible extension, namely predicting the world's dynamics as an additional auxiliary task.

9.3.1 Improvements

The most salient matter seems to regard the robustness of L-CIL to linguistic data. In the discussed experiments, sentences that described the **ambiguous** behavior resembled sentences for one of the behaviors more than they did sentences for the other one. This eventually affected the final results. As much as this is something expected from the algorithm, it should be pointed out that the similarity of sentences could have been prevented with a different scheme of their generation. Hence, future studies should investigate two other scenarios. Namely, 1) how L-CIL works when neither of the behaviors particularly stands out when it comes to their linguistic description, or 2) when similar sentences describe both the non-congruent behavior(s) and the **ambiguous** behavior. Generally, language aspect of L-CIL should undergo a careful study from both computer science and linguistics perspective.

Another valuable step to undertake is in making the discussed method fully applicable to real life environments. Embodied CIL proved effective when coupled with a self-driving car. Given the dominating topic of this work is also self-driving, it seems like a natural extension to try this with L-CIL. The main challenge would be in improving qualitative performance of L-CIL, establishing a proper state representation that would allow the algorithm to keep its features, and connecting it to a working perception system. With that, training the agent could be done in an offline manner. As mentioned in the introduction to this chapter, a self-driving car (or in fact any other kind of robot) capable of understanding human language would be very stimulating to the field of HCI and could offer manifold potential consumer/industry applications.

Besides self-driving, there are various other settings in which language could positively affect the final performance if not only provide amiable extension that helps humans to communicate with the agent. For this reason, L-CIL should be additionally tested in different simulated environments and more challenging expert datasets. In the presented studies, actions were generated with a hand-made controller. Due to this specification, variance of the actions was rather minimal and the problem could be also posed

using a discrete action space (even though it was not treated as such in the experiments). In more realistic scenarios, the action space could have wider continuous distribution and a stable algorithm should be resistant to these cases as well.

9.3.2 Possible Extensions

Language-Conditional Imitation Learning also entertains room for extensions. Like it was sketched in the first three chapters of the thesis, it follows a multi-task learning approach, specifically an auxiliary learning approach. It is not out of the question that additional tasks may increase the effectiveness of the method. One possible way forward is to better account for the action-based language acquisition theory mentioned in Chapter 5. For now, the algorithm is only concerned with the proper sentence decoding but in the original psychological formulation, humans supposedly predict each action's results too. We said that this element is implicitly contained in the expert's demonstrations, but the overall model of the world's dynamics is actually not known to the agent. Thus, there is a possibility that forcing it to learn this model alongside sentence auto-encoding would use the synergies between these two areas and improve the results even further. This could also address L-CIL's embodiment problem (c.f. Subsection 3.2.4).

Chapter 10

Conclusion

This work presented a new algorithm from the field of imitation learning called Language-Conditional Imitation Learning. To prepare for the problem setup we discussed basic notions in RL and IL as well as related literature that concerned inverse RL, multi-task learning, language use in imitation or reinforcement learning and the general issue of conditioning. To motivate this research philosophically we also went through psychological basis for L-CIL in the form of action-based language acquisition theory and brought upon the topic of conditioning in social and neural sciences. The algorithm, eventually implemented as a neural network, was introduced in Chapter 6. Its main assumption was to use the standard behavioral cloning loss paired with the reconstruction loss for input sentences. Experiments shown in Chapter 8 and elaborated further in the text revealed that L-CIL successfully imitates multiple training behaviors while exhibiting quality performance on the ambiguous one. There was also evidence in favor of the linguistic generalization in which the test sentences were not only different, but also contained unseen words. In our standard generalization experiment we surprisingly observed excellent performance of vanilla behavioral cloning, which was initially expected to fail. Due to that, we concluded that the major objective for future work should be in addressing the drawbacks connected to this study so that it became insurmountable for BC. However, the analysis we performed on L-CIL already elucidated that it succeeds in the generaliza-

tion task due to its architectural setup. The auxiliary loss enabled it to capture the proper similarity between input sentences which, at test time, led it to produce hidden representations (conditioning vectors) that are similar to those successfully used during training. Even though further studies should measure the real extent to which L-CIL generalizes in the face of ambiguity using language, our insights and the qualitative results we obtained by doing rollouts with our algorithm already show great promise for Human-Computer Interaction, NLP and robotics research in general.

Bibliography

[1] ABBEEL, P., COATES, A., AND NG, A. Y. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research 29*, 13 (2010), 1608–1639.

[2] AGRAWAL, A., MALINOWSKI, M., HILL, F., ESLAMI, A., VINYALS, O., AND KULKARNI, T. Generating diverse programs with instruction conditioned reinforced adversarial learning. *arXiv preprint arXiv:1812.00898* (2018).

[3] ANDREAS, J., KLEIN, D., AND LEVINE, S. Learning with latent language. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics* (2017).

[4] ANDREAS, J., KLEIN, D., AND LEVINE, S. Modular multitask reinforcement learning with policy sketches. In *Proceedings of the 34th International Conference on Machine Learning* (2017), JMLR.org, pp. 166–175.

[5] ARTZI, Y., AND ZETTLEMOYER, L. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics 1* (2013), 49–62.

[6] BABES, M., MARIVATE, V., SUBRAMANIAN, K., AND LITTMAN, M. L. Apprenticeship learning about multiple intentions. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (2011), pp. 897–904.

[7] BAHDANAU, D., HILL, F., LEIKE, J., HUGHES, E., HOSSEINI, S. A., KOHLI, P., AND GREFENSTETTE, E. Learning to understand goal specifications by modelling reward. In *International Conference on Learning Representations* (2018).

[8] BILLARD, A., CALINON, S., DILLMANN, R., AND SCHAAL, S. Robot programming by demonstration. *Springer handbook of robotics* (2008), 1371–1394.

[9] BOULARIAS, A., KOBER, J., AND PETERS, J. Relative entropy inverse reinforcement learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (2011), pp. 182–189.

[10] BRANAVAN, S., ZETTLEMOYER, L. S., AND BARZILAY, R. Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of the 48th annual meeting of the association for computational linguistics* (2010), Association for Computational Linguistics, pp. 1268–1277.

[11] BREITER, H. C., AHARON, I., KAHNEMAN, D., DALE, A., AND SHIZGAL, P. Functional imaging of neural responses to expectancy and experience of monetary gains and losses. *Neuron 30*, 2 (2001), 619–639.

[12] CARUANA, R. Multitask learning. *Machine learning 28*, 1 (1997), 41–75.

[13] CHAPLOT, D. S., SATHYENDRA, K. M., PASUMARTHI, R. K., RAJAGOPAL, D., AND SALAKHUTDINOV, R. Gated-attention architectures for task-oriented language grounding. In *Thirty-Second AAAI Conference on Artificial Intelligence* (2018).

[14] CHAWLA, M., AND MIYAPURAM, K. P. Context-sensitive computational mechanisms of decision making. *Journal of experimental neuroscience 12* (2018), 1179069518809057.

[15] CHEN, D. L., AND MOONEY, R. J. Learning to interpret natural language navigation instructions from observations. In *Twenty-Fifth AAAI Conference on Artificial Intelligence* (2011).

[16] CHEN, H., SUHR, A., MISRA, D., SNAVELY, N., AND ARTZI, Y. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 12538–12547.

[17] CHO, K., VAN MERRIËNBOER, B., GULCEHRE, C., BAHDANAU, D., BOUGARES, F., SCHWENK, H., AND BENGIO, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (2014), pp. 1724–1734.

[18] CHOW, S. S., ROMO, R., AND BRODY, C. D. Context-dependent modulation of functional connectivity: secondary somatosensory cortex to prefrontal cortex connections in two-stimulus-interval discrimination tasks. *Journal of Neuroscience 29*, 22 (2009), 7238–7245.

[19] CHOWDHURI, S., PANKAJ, T., AND ZIPSER, K. Multinet: Multi-modal multi-task learning for autonomous driving. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2019), IEEE, pp. 1496–1504.

[20] CHUNG, J., KASTNER, K., DINH, L., GOEL, K., COURVILLE, A. C., AND BENGIO, Y. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems* (2015), pp. 2980–2988.

[21] CO-REYES, J. D., GUPTA, A., SANJEEV, S., ALTIERI, N., ANDREAS, J., DENERO, J., ABBEEL, P., AND LEVINE, S. Guiding policies with language via meta-learning. In *International Conference on Learning Representations* (2018).

[22] CODEVILLA, F., MIILLER, M., LÓPEZ, A., KOLTUN, V., AND DOSOVITSKIY, A. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018), IEEE, pp. 1–9.

[23] COHEN, M. R., AND NEWSOME, W. T. Context-dependent changes in functional circuitry in visual area mt. *Neuron 60*, 1 (2008), 162–173.

[24] DAUMÉ, H., LANGFORD, J., AND MARCU, D. Search-based structured prediction. *Machine learning 75*, 3 (2009), 297–325.

[25] DEISENROTH, M. P., ENGLERT, P., PETERS, J., AND FOX, D. Multi-task policy search for robotics. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (2014), IEEE, pp. 3876–3881.

[26] DIMITRAKAKIS, C., AND ROTHKOPF, C. A. Bayesian multitask inverse reinforcement learning. In *European workshop on reinforcement learning* (2011), Springer, pp. 273–284.

[27] DUAN, Y., ANDRYCHOWICZ, M., STADIE, B., HO, O. J., SCHNEIDER, J., SUTSKEVER, I., ABBEEL, P., AND ZAREMBA, W. One-shot imitation learning. In *Advances in neural information processing systems* (2017), pp. 1087–1098.

[28] FINN, C., LEVINE, S., AND ABBEEL, P. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning* (2016), pp. 49–58.

[29] FOX, R., BERENSTEIN, R., STOICA, I., AND GOLDBERG, K. Multi-task hierarchical imitation learning for home automation. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)* (2019), IEEE, pp. 1–8.

[30] FU, J., KORATTIKARA, A., LEVINE, S., AND GUADARRAMA, S. From language to goals: Inverse reinforcement learning for vision-based instruction following. In *International Conference on Learning Representations* (2019).

[31] GALINDO-LEON, E. E., STITT, I., PIEPER, F., STIEGLITZ, T., ENGLER, G., AND ENGEL, A. K. Context-specific modulation of intrinsic coupling modes shapes multisensory processing. *Science advances 5*, 4 (2019), eaar7633.

[32] GALLESE, V. Before and below 'theory of mind': embodied simulation and the neural correlates of social cognition. *Philosophical Transactions of the Royal Society B: Biological Sciences 362*, 1480 (2007), 659–669.

[33] GALLESE, V. Mirror neurons and the social nature of language: The neural exploitation hypothesis. *Social neuroscience 3*, 3-4 (2008), 317–333.

[34] GALLESE, V., AND GOLDMAN, A. Mirror neurons and the simulation theory of mind-reading. *Trends in cognitive sciences 2*, 12 (1998), 493–501.

[35] GLEAVE, A., AND HABRYKA, O. Multi-task maximum entropy inverse reinforcement learning. *arXiv preprint arXiv:1805.08882* (2018).

[36] GLENBERG, A. M., AND GALLESE, V. Action-based language: A theory of language acquisition, comprehension, and production. *cortex 48*, 7 (2012), 905–922.

[37] GOYAL, P., NIEKUM, S., AND MOONEY, R. J. Using natural language for reward shaping in reinforcement learning. In *International Joint Conferences on Artificial Intelligence* (2019).

[38] HARUNO, M., WOLPERT, D. M., AND KAWATO, M. Hierarchical mosaic for movement generation. In *International congress series* (2003), vol. 1250, Elsevier, pp. 575–590.

[39] HERMANN, K. M., HILL, F., GREEN, S., WANG, F., FAULKNER, R., SOYER, H., SZEPESVARI, D., CZARNECKI, W. M., JADERBERG, M., TEPLYASHIN, D., ET AL. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551* (2017).

[40] HO, J., AND ERMON, S. Generative adversarial imitation learning. In *Advances in neural information processing systems* (2016), pp. 4565–4573.

[41] HOCHREITER, S., AND SCHMIDHUBER, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[42] HUBER, J., PAYNE, J. W., AND PUTO, C. Adding asymmetrically dominated alternatives: Violations of regularity and the similarity hypothesis. *Journal of consumer research 9*, 1 (1982), 90–98.

[43] JANNER, M., NARASIMHAN, K., AND BARZILAY, R. Representation learning for grounded spatial reasoning. *Transactions of the Association for Computational Linguistics 6* (2018), 49–61.

[44] KALAKRISHNAN, M., PASTOR, P., RIGHETTI, L., AND SCHAAL, S. Learning objective functions for manipulation. In *2013 IEEE International Conference on Robotics and Automation* (2013), IEEE, pp. 1331–1336.

[45] KOLLAR, T., TELLEX, S., ROY, D., AND ROY, N. Toward understanding natural language directions. In *Proceedings of the 5th ACM/IEEE international conference on Human-robot interaction* (2010), IEEE Press, pp. 259–266.

[46] KUDERER, M., GULATI, S., AND BURGARD, W. Learning driving styles for autonomous vehicles from demonstration. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (2015), IEEE, pp. 2641–2646.

[47] KUMANO, H., SUDA, Y., AND UKA, T. Context-dependent accumulation of sensory evidence in the parietal cortex underlies flexible task switching. *Journal of Neuroscience 36*, 48 (2016), 12192–12202.

[48] LANGE, S., GABEL, T., AND RIEDMILLER, M. Batch reinforcement learning. In *Reinforcement learning*. Springer, 2012, pp. 45–73.

[49] LUKETINA, J., NARDELLI, N., FARQUHAR, G., FOERSTER, J., ANDREAS, J., GREFENSTETTE, E., WHITESON, S., AND ROCKTÄSCHEL, T. A survey of reinforcement learning informed by natural language. In *International Joint Conferences on Artificial Intelligence* (2019).

[50] MACGLASHAN, J., BABES-VROMAN, M., DESJARDINS, M., LITTMAN, M. L., MURESAN, S., SQUIRE, S., TELLEX, S., ARUMUGAM, D., AND YANG, L. Grounding english commands to reward functions. In *Robotics: Science and Systems* (2015).

[51] MACMAHON, M., STANKIEWICZ, B., AND KUIPERS, B. Walk the talk: Connecting language, knowledge, and action in route instructions. *Def 2*, 6 (2006), 4.

[52] MANGIN, O., AND OUEDEYER, P.-Y. Feature learning for multi-task inverse reinforcement learning.

[53] MAYLOR, E. A., AND ROBERTS, M. A. Similarity and attraction effects in episodic memory judgments. *Cognition 105*, 3 (2007), 715–723.

[54] MEHTA, A., SUBRAMANIAN, A., AND SUBRAMANIAN, A. Learning end-to-end autonomous driving using guided auxiliary supervision. *arXiv preprint arXiv:1808.10393* (2018).

[55] MENG, M., CHERIAN, T., AND SINHA, P. Neural basis of contextual modulation on categorical face perception. *Journal of Vision 9*, 8 (2009), 456–456.

[56] MIKOLOV, T., SUTSKEVER, I., CHEN, K., CORRADO, G. S., AND DEAN, J. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (2013), pp. 3111–3119.

[57] MISRA, D., LANGFORD, J., AND ARTZI, Y. Mapping instructions and visual observations to actions with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (2017), pp. 1004–1015.

[58] MNIH, V., BADIA, A. P., MIRZA, M., GRAVES, A., LILLICRAP, T., HARLEY, T., SILVER, D., AND KAVUKCUOGLU, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (2016), pp. 1928–1937.

[59] MOORE, D. A. Order effects in preference judgments: Evidence for context dependence in the generation of preferences. *Organizational Behavior and Human Decision Processes 78*, 2 (1999), 146–165.

[60] NAIR, A., CHEN, D., AGRAWAL, P., ISOLA, P., ABBEEL, P., MALIK, J., AND LEVINE, S. Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE International Conference on Robotics and Automation (ICRA)* (2017), IEEE, pp. 2146–2153.

[61] NARASIMHAN, K., BARZILAY, R., AND JAAKKOLA, T. Grounding language for transfer in deep reinforcement learning. *Journal of Artificial Intelligence Research 63* (2018), 849–874.

[62] NG, A. Y., RUSSELL, S. J., ET AL. Algorithms for inverse reinforcement learning. In *Icml* (2000), vol. 1, p. 2.

[63] NIEUWENHUIS, S., HESLENFELD, D. J., VON GEUSAU, N. J. A., MARS, R. B., HOLROYD, C. B., AND YEUNG, N. Activity in human reward-sensitive brain areas is strongly context dependent. *Neuroimage 25*, 4 (2005), 1302–1309.

[64] OH, J., GUO, X., LEE, H., LEWIS, R. L., AND SINGH, S. Action-conditional video prediction using deep networks in atari games. In *Advances in neural information processing systems* (2015), pp. 2863–2871.

[65] OSA, T., PAJARINEN, J., NEUMANN, G., BAGNELL, J. A., ABBEEL, P., PETERS, J., ET AL. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics 7*, 1-2 (2018), 1–179.

[66] OTTEN, M., SETH, A. K., AND PINTO, Y. A social bayesian brain: How social knowledge can shape visual perception. *Brain and Cognition 112* (2017), 69–77.

[67] PALATUCCI, M., POMERLEAU, D., HINTON, G. E., AND MITCHELL, T. M. Zero-shot learning with semantic output codes. In *Advances in neural information processing systems* (2009), pp. 1410–1418.

[68] PALMER, C. R., AND KRISTAN JR, W. B. Contextual modulation of behavioral choice. *Current Opinion in Neurobiology 21*, 4 (2011), 520–526.

[69] PAPINENI, K., ROUKOS, S., WARD, T., AND ZHU, W.-J. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics* (2002), Association for Computational Linguistics, pp. 311–318.

[70] PATEL, M. Active preference learning using trajectory segmentation. Master's thesis, McGill University, Retrieved from http://digitool.library.mcgill.ca, 2019.

[71] PEREZ, E., STRUB, F., DE VRIES, H., DUMOULIN, V., AND COURVILLE, A. Film: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence* (2018).

[72] PETRO, L., PATON, A., AND MUCKLI, L. Contextual modulation of primary visual cortex by auditory signals. *Philosophical Transactions of the Royal Society B: Biological Sciences 372*, 1714 (2017), 20160104.

[73] PETTIBONE, J. C., AND WEDELL, D. H. Examining models of nondominated decoy effects across judgment and choice. *Organizational behavior and human decision processes 81*, 2 (2000), 300–328.

[74] POMERLEAU, D. A. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems* (1989), pp. 305–313.

[75] RAMACHANDRAN, D., AND AMIR, E. Bayesian inverse reinforcement learning. In *IJCAI* (2007), vol. 7, pp. 2586–2591.

[76] RIGOLI, F., FRISTON, K. J., MARTINELLI, C., SELAKOVIĆ, M., SHERGILL, S. S., AND DOLAN, R. J. A bayesian model of context-sensitive value attribution. *ELife 5* (2016), e16127.

[77] RIGOLI, F., MATHYS, C., FRISTON, K. J., AND DOLAN, R. J. A unifying bayesian account of contextual effects in value-based choice. *PLoS computational biology 13*, 10 (2017), e1005769.

[78] RIZZOLATTI, G., AND CRAIGHERO, L. The mirror-neuron system. *Annu. Rev. Neurosci. 27* (2004), 169–192.

[79] ROBERTS, B., HARRIS, M. G., AND YATES, T. A. The roles of inducer size and distance in the ebbinghaus illusion (titchener circles). *Perception 34*, 7 (2005), 847–856.

[80] ROSS, S., AND BAGNELL, D. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010), pp. 661–668.

[81] ROSS, S., AND BAGNELL, J. A. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979* (2014).

[82] ROSS, S., GORDON, G., AND BAGNELL, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (2011), pp. 627–635.

[83] RUDER, S. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (2017).

[84] SAMMUT, C., HURST, S., KEDZIER, D., AND MICHIE, D. Learning to fly. In *Machine Learning Proceedings 1992*. Elsevier, 1992, pp. 385–393.

[85] SILVER, D., SCHRITTWIESER, J., SIMONYAN, K., ANTONOGLOU, I., HUANG, A., GUEZ, A., HUBERT, T., BAKER, L., LAI, M., BOLTON, A., ET AL. Mastering the game of go without human knowledge. *Nature 550*, 7676 (2017), 354.

[86] SIMMONS, R. G., GOLDBERG, D., GOODE, A. P., MONTEMERLO, M., ROY, N., SELLNER, B., URMSON, C., SCHULTZ, A. C., ABRAMSON, M., ADAMS, W., ATRASH, A., BUGAJSKA, M. D., COBLENZ, M. J., MACMAHON, M., PERZANOWSKI, D., HORSWILL, I., ZUBEK, R., KORTENKAMP, D., WOLFE, B., MILAM, T., AND MAXWELL, B. A. Grace: An autonomous robot for the aaai robot challenge. *AI Magazine 24* (2002), 51–72.

[87] SIMONSON, I. Choice based on reasons: The case of attraction and compromise effects. *Journal of consumer research 16*, 2 (1989), 158–174.

[88] SINHA, A., AKILESH, B., SARKAR, M., AND KRISHNAMURTHY, B. Attention based natural language grounding by navigating virtual environment. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2019), IEEE, pp. 236–244.

[89] SKUBIC, M., PERZANOWSKI, D., BLISARD, S., SCHULTZ, A., ADAMS, W., BUGAJSKA, M., AND BROCK, D. Spatial language for human-robot dialogs. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 34*, 2 (2004), 154–167.

[90] SRIVASTAVA, N., AND SCHRATER, P. Learning what to want: context-sensitive preference learning. *PloS one 10*, 10 (2015), e0141129.

[91] SUTTON, R. S., AND BARTO, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

[92] TRAPP, S., AND BAR, M. Prediction, context, and competition in visual recognition. *Annals of the New York Academy of Sciences 1339*, 1 (2015), 190–198.

[93] TRUEBLOOD, J. S., BROWN, S. D., HEATHCOTE, A., AND BUSEMEYER, J. R. Not just for consumers: Context effects are fundamental to decision making. *Psychological science 24*, 6 (2013), 901–908.

[94] WANG, X., HUANG, Q., CELIKYILMAZ, A., GAO, J., SHEN, D., WANG, Y.-F., WANG, W. Y., AND ZHANG, L. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 6629–6638.

[95] WULFMEIER, M., ONDRUSKA, P., AND POSNER, I. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888* (2016).

[96] YU, H., ZHANG, H., AND XU, W. Interactive grounded language acquisition and generalization in a 2d world. In *International Conference on Learning Representations* (2018).

[97] ZHANG, T., MCCARTHY, Z., JOW, O., LEE, D., CHEN, X., GOLDBERG, K., AND ABBEEL, P. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018), IEEE, pp. 1–8.

[98] ZIEBART, B. D., MAAS, A., BAGNELL, J. A., AND DEY, A. K. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3* (2008), AAAI'08, AAAI Press, pp. 1433–1438.