Learning to Speak and Hear Through Multi-Agent Communication over a Continuous Acoustic Channel

Kevin Michael Eloff

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Electronic) in the Faculty of Engineering at Stellenbosch University.

Supervisor: Prof. H. Kamper[1]
Co-supervisors: Prof. H. A. Engelbrecht[1], Dr A. Pretorius[2]

[1]Department of Electrical and Electronic Engineering
[2]InstaDeep, Cape Town

March 2023

Acknowledgements

I would like to express my sincerest thanks to the following people for making this work possible:

- My supervisor, Prof. Herman Kamper, for all the fun ideas and enthusiasm of my work. I was consistently motivated by your excitement and interest in the work.
- My co-supervisors, Prof. Herman A. Engelbrecht and Dr Arnu Pretorius, for all the guidance and support throughout this study.
- To Prof. Okko Räsänen, for your interest in this work. I really appreciate your amazing insights and contributions.
- My friends and family, for your continued love and support.
- To Kim, for all your love and assistance.
- To InstaDeep, for an amazing internship, guidance, and financial assistance.

Stellenbosch
UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

Plagiaatverklaring / *Plagiarism Declaration*

1. Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
   Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.

2. Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
   I agree that plagiarism is a punishable offence because it constitutes theft.

3. Ek verstaan ook dat direkte vertalings plagiaat is.
   I also understand that direct translations are plagiarism.

4. Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
   Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism

5. Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
   I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

Studentenommer / *Student number*	Handtekening / *Signature*
Voorletters en van / *Initials and surname*	Datum / *Date*

Abstract

English

Human infants acquire language in large part through continuous signalling with their caregivers. By interacting and communicating with their caregivers, infants can observe the consequences of their communicative attempts (e.g. through parental response) that may guide the process of language acquisition. We find many similarities between human language acquisition and the intuition of intrinsic motivation which serves as a basis of reinforcement learning. In contrast, current trends in natural language processing disregard this, instead focusing on having larger models and more data to learn the statistical relationships between words with none of the original goals of language in mind.

Multi-agent reinforcement learning has proven effective for investigating emergent communication between social agents. Most of these studies, however, focus on communication with discrete symbols. Humans learn language over a continuous channel and language has evolved through gestures and spoken communication, both of which are inherently continuous. This channel is also time-varying: interactions take place in unique settings with different channel acoustics and types of noise. These intricacies are lost when agents communicate directly with purely discrete symbols.

We therefore ask: are we able to observe emergent language between agents with a continuous communication channel? And if so, how does learned continuous communication differ from discrete communication? Our objective is to provide a platform to study emergent continuous signalling in order to see how it relates to human language acquisition and evolution. We propose a messaging environment where a Speaker agent needs to convey a set of attributes to a Listener over a noisy acoustic channel.

This thesis makes two core contributions. Firstly, in contrast to recent studies on language emergence, we train our agents with deep Q-learning rather than REINFORCE. When using DQN, we show significant performance gains and improved compositionality. Secondly, we provide a platform to study spoken emergent language between agents. To showcase this, we compare discrete and acoustic emergent languages. We show that, unlike the discrete case, the acoustic Speaker learns redundancy to improve Listener coherency when longer sequences are allowed. We also find that the acoustic Speaker develops more compositional communication protocols which implicitly compensates for transmission errors over the noisy channel. In addition, we show early experiments with promising results in language grounding (to English) and effective generalisation to real-world communication channels.

Afrikaans

Menslike babas verwerf taal grootliks deur voortdurende seine met hul versorgers. Deur interaksie en kommunikasie met hul versorgers, kan babas die gevolge van hul kommunikatiewe pogings waarneem (bv. deur ouerlike reaksie) wat die proses van taalverwerwing kan lei. Ons vind baie ooreenkomste tussen menslike taalverwerwing en die intuïsie van intrinsieke motivering wat as basis van versterkende leer dien. Hierteenoor ignoreer huidige neigings in natuurlike taalverwerking dit, maar fokus eerder daarop om groter modelle en meer data te hê om die statistiese verwantskappe tussen woorde te leer met geen van die oorspronklike doelwitte van taal in gedagte nie.

Multi-agent versterking leer het bewys effektief vir die ondersoek van ontluikende kommunikasie tussen agente. Die meeste van hierdie studies fokus egter op kommunikasie met diskrete simbole. Mense leer taal oor 'n deurlopende kanaal en taal het ontwikkel deur gebare en gesproke kommunikasie, wat albei inherent aaneenlopend is. Hierdie kanaal is ook tyd-variërend: interaksies vind plaas in unieke omgewings met verskillende kanaal akoestiek en tipes geraas. Hierdie ingewikkeldhede gaan verlore wanneer agente direk met suiwer diskrete simbole kommunikeer.

Ons vra dus: is ons in staat om opkomende taal tussen agente met 'n deurlopende kommunikasiekanaal waar te neem? En indien wel, hoe verskil aangeleerde deurlopende kommunikasie van diskrete kommunikasie? Ons doelwit is om 'n platform te bied om ontluikende deurlopende seine te bestudeer om te sien hoe dit verband hou met menslike taalverwerwing en -evolusie. Ons stel 'n boodskap-omgewing voor waar 'n spreker-agent 'n stel eienskappe aan 'n luisteraar moet oordra oor 'n lawaaierige akoestiese kanaal.

Hierdie tesis lewer twee kernbydraes. Eerstens, in teenstelling met onlangse studies oor taalopkoms, lei ons ons agente op met diepgaande Q-leer eerder as REINFORCE. Wanneer ons DQN gebruik, toon ons aansienlike prestasiewinste en verbeterde samestelling. Tweedens bied ons 'n platform om gesproke opkomende taal tussen agente te bestudeer. Om dit ten toon te stel, vergelyk ons diskrete en akoestiese opkomende tale. Ons wys dat, anders as die diskrete geval, die akoestiese luidspreker oortolligheid leer om luisteraarsamehang te verbeter wanneer langer reekse toegelaat word. Ons vind ook dat die akoestiese spreker meer komposisionele kommunikasieprotokolle ontwikkel wat implisiet kompenseer vir transmissiefoute oor die raserige kanaal. Daarbenewens toon ons vroeë eksperimente met belowende resultate in taalbegronding (na Engels) en effektiewe veralgemening na werklike kommunikasiekanale.

Contents

List of Figures

List of Tables

Nomenclature

Variables and functions

t	Environment time-step.
S_t	Environment state random variable at time-step t.
A_t	Action taken random variable at time-step t.
R_t	Reward received random variable at time-step t for taking action A_{t-1} in state S_{t-1}.
G_t	Environment return, the sum of all future rewards.
$\pi(s)$	Agent policy, defines actions given the state.
$q_\theta(s, a)$	Deep Q-Network approximating the action-value function parameterised by θ.
θ	Neural network parameters.
$\mathcal{L}(\theta)$	Neural network loss with respect to θ.
$\nabla_\theta \mathcal{L}(\theta)$	Loss gradient with respect to θ.
$\boldsymbol{s}$	A set of attribute values.
$\hat{\boldsymbol{s}}$	Predicted set of attribute values.
$\boldsymbol{c}$	Sequence of discrete units (typically phones) generated by a Speaker.
L	Maximum sequence length of $\boldsymbol{c}$
N	Number of attributes in $\boldsymbol{s}$.
M	Number of distinct values each attribute can be.
ϵ	ϵ-greedy exploration probability.
R	Average reward per attribute or environment reward.
$\mathcal{P}$	Synthesiser phone set.
$\boldsymbol{w}$	Sampled discrete-time waveform.
$f(\boldsymbol{w})$	Channel noise function.
X	Mel-spectrogram sequence.
$\mathcal{N}$	Dataset of Clotho real background noise samples.
$\mathcal{H}$	Dataset of AIR room impulse responses.

Acronyms and abbreviations

ASR	automatic speech recognition
DIAL	differentiable inter-agent learning
DQN	deep Q-network
DTW	dynamic time-warping
GRU	gated recurrent unit
ILM	iterated learning model
LSTM	long short-term memory
MARL	multi-agent reinforcement learning
MLP	multilayer perceptron
NLP	natural language processing
RIAL	reinforced inter-agent learning
RL	reinforcement learning
RMS	root-mean-square
SNR	signal-to-noise ratio

Chapter 1

Introduction

1.1. Motivation

Reinforcement learning (RL) is increasingly being used as a tool to study language emergence [1–6]. By allowing multiple agents to communicate with each other while solving a common task, a communication protocol needs to be established. The resulting protocol can be studied to see if it adheres to properties of human language, such as compositionality, generalisation and redundancy [7–10]. The tasks and environments themselves can also be studied, to see what types of constraints are necessary for human-like language to emerge [11]. Referential games are often used for this purpose [12–14]. While these studies open up the possibility of using computational models to investigate how language emerged and how language is acquired through interaction with an environment and other agents, most RL studies consider communication using *discrete* symbols.

Spoken language instead operates and emerged over a *continuous* acoustic channel. Human infants acquire their native language by being exposed to speech audio in their environment [15]. By interacting and communicating with their caregivers using continuous signals, infants can observe the consequences of their communicative attempts (e.g. through parental responses) that may guide the process of language acquisition [16]. Continuous signalling is challenging since an agent needs to be able to deal with different acoustic settings and noise introduced through the channel. These intricacies are lost when agents communicate directly with purely discrete symbols.

The primary motivation for our research is to bridge the disconnect between current trends in language processing and our understanding of language acquisition and evolution. As mentioned, spoken language operates and emerged over a *continuous* acoustic channel. Language is also a communication technology [17]: it allows social entities to converse complex meanings through space via speech. In contrast, current trends disregard this, instead focusing on having larger models and more data to learn the statistical relationships between words with none of the original goals of language in mind [18, 19]. A detailed review of language use, philosophy, and goals of language processing is given in Section 3.1. This leads us to look towards an alternative method of language acquisition and processing.

We can draw many similarities between human language acquisition and the intuition

of intrinsic motivation which serves as a basis of modern RL. Language is social, and multi-agent RL is arguably the current state-of-the-art for modelling social interaction. We are also interested in modelling infant language acquisition, where the process of caregiver feedback is loosely akin to an RL reward signal. Thus, we consider RL as an alternative method to model language acquisition in this study.

1.2. Background

Earlier work has considered models of human language acquisition using continuous signalling between a simulated infant and caregiver [20, 21]. However, these models often rely on heuristic approaches and older neural modelling techniques, making them difficult to extend. For example, it is not easy to directly incorporate other environmental rewards or interactions between multiple agents. More recent RL approaches would make this possible, but as noted, have mainly focused on discrete communication [1, 3, 5]. These approaches have proved effective at solving the discrete communication task [4, 6] using REINFORCE [22], an older policy-gradient algorithm with relatively high variance in performance [23]. Our work tries to bridge the disconnect between recent contributions in multi-agent reinforcement learning (MARL) and earlier literature in language acquisition and modelling [24].

One recent exception which does use continuous signalling within a modern RL framework is [25] and the follow-up work [26]. In these studies, a Student agent is exposed to a large collection of unlabelled speech audio, from which it builds up a dictionary of possible spoken words. The Student can then select segmented words from its dictionary to play back to a Teacher, which uses a trained automatic speech recognition (ASR) model to classify the words and execute a movement command in a discrete environment. The Student is then rewarded for moving towards a goal position. We will expand on these studies in Section 3.2.2. We also propose a Student-Teacher setup, but importantly, our agents can generate their own unique audio waveforms rather than just segmenting and repeating words exactly from past observations.

1.3. Research question and objectives

As far as we are aware, this study is the first step in answering the larger research question: are we able to observe emergent language between agents with a continuous acoustic communication channel, trained using RL? We are also interested in investigating how learned continuous communication differs from discrete communication. To summarise, the three main objectives of this study that will assist us in answering these questions are:

1. Propose and develop an environment with the complexities necessary to facilitate

the emergence of spoken communication between cooperative agents. Spoken communication over a lossy channel must be necessary for the agents to solve a common task.

2. Design and train agents that learn to "speak" and "hear" spoken communication in the above-mentioned environment.

3. Analyse the characteristics of the emergent languages developed by these agents. A comparison should be made to previous work in discrete language emergence.

In our investigation of previous work, we noticed that many discrete language emergence studies (e.g. [2, 4, 6]) use REINFORCE as an optimisation technique. Therefore, as an extension to previous work, we would also like to compare REINFORCE to a slightly newer technique, DQN.

1.4. Methodology

In this study, we take a systematic approach to achieve the objectives mentioned in Section 1.3. We first focus on researching and developing a robust environment in which we may perform all the experiments necessary to answer the larger research question. This involves first reviewing recent work in discrete language emergence, spoken language acquisition, and referential games. With these insights, we define an appropriate environment with all the functionality required to observe spoken language acquisition between agents. Next, we develop a realisation of the environment, including all necessary components. This includes the design of each agent, the communication channel, and the training setup.

Concretely, we propose the environment illustrated in Figure 1.1, which is an extension of the referential signalling game used in several previous studies [2, 4, 27, 28]. Here $\boldsymbol{s}$ represents a set of attribute values the Speaker must communicate to a Listener agent. Taking these attributes as input, the Speaker produces a waveform as output, which passes over a lossy acoustic channel. The Listener "hears" the utterance from the Speaker. Taking

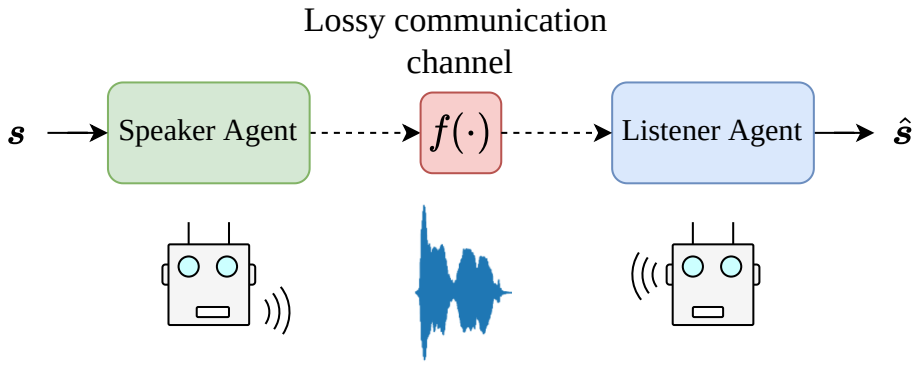


Figure 1.1: Environment setup showing a Speaker communicating to a Listener over a lossy acoustic communication channel f.

the waveform as input, the Listener produces output $\hat{\boldsymbol{s}}$. This output is the Listener's interpretation of the concept (set of attribute values) that the Speaker agent tried to communicate. The agents must develop a common communication protocol such that $\boldsymbol{s} = \hat{\boldsymbol{s}}$. This process encapsulates one of the core goals of human language: conveying meaning through communication [17].

To train the agents, we use a deep Q-network (DQN) [29,30]. This differs from previous studies on language emergence [2,4,6,28] that used REINFORCE [22]. REINFORCE is a policy gradient method known to have relatively high variance [23], while DQN is a value-based method which tends to have higher stability. We then perform a set of experiments that showcase the ability of these agents to learn spoken emergent language in various environmental setups. In these early experiments, we optimise with both REINFORCE and DQN in order to compare the two as per our secondary objective. The rest of the experiments are all related to the investigation of the emergent languages developed by the agents in various situations. These emergent languages are then also compared to discrete emergent communication.

1.5. Contributions

Our bigger goal is to explore the question of whether and how language emerges when using RL to train agents that communicate via continuous acoustic signals. Our proposed environment and training methodology serves as a means to perform such an exploration, and the goal of this study is to showcase some capabilities of the platform. We have two primary contributions in this study. Firstly, we provide a platform to study spoken emergent language between agents. As a concrete example of the types of research questions we can answer with our environment, we consider how discrete and acoustic emergent language differs when agents communicate with sequences of different duration. We show that when longer sequences are allowed, our acoustic Speaker learns a redundant communication protocol, using fewer unique phones[1] per utterance with more repetition of bigrams and trigrams[2]. This redundancy improves the coherence of the Listener agent. In contrast, agents trained to communicate with purely discrete symbols fail to learn such redundancy, resulting in poor communication when these agents are used in very noisy environments. The acoustic agents also tend to develop more compositional communication protocols which implicitly compensate for transmission errors over a noisy channel. In addition to this, we show early experiments with promising results in language grounding (to a known human language – English) and effective generalisation to real-world communication channels.

[1] Phones are the smallest unit of language and are considered the building blocks of speech.

[2] A trigram is a contiguous sequence of three units, a bigram a sequence of two units, and a unigram is a single unit.

As a second contribution, we compared REINFORCE to DQN as an optimisation technique. We show significant performance gains and improved compositionality when using DQN rather than REINFORCE. We highly recommend that future work in language emergence uses DQN; even for discrete language acquisition.

This work is currently under review at the 22nd International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2023).

1.6. Thesis overview

In this thesis, we separate our literature review into two chapters. Firstly, in Chapter 2, we cover the core background concepts relevant to this study without going into detail on recent work. This includes detailed explanations of RL and language processing methods. Secondly, in Chapter 3, we provide a comprehensive literature review of the related work and motivations for considering RL for language acquisition studies. This includes an overview of recent work in referential games and spoken language acquisition. Expanding on this recent work, we define our environment in Chapter 4. Our approach to solving this environment is then discussed, along with details on training and optimisation. In Chapter 5, we briefly overview the experimental setup and introduce the metrics used to analyse the emergent communication. We then systematically work through our experiments in Chapter 6. In this chapter, we first compare REINFORCE to DQN as two approaches for optimising agents in the task of discrete communication (Section 6.1). Next, we investigate whether our agents are capable of learning to speak and hear over a lossy communication channel (Section 6.2). We then perform a detailed analysis of the emergent language characteristics (Section 6.3). Finally, we include two additional experiments related to language grounding and real-world performance (Section 6.4). We end this thesis with a summary and conclusion in Chapter 7; including a detailed overview of environment extensions and future work.

Chapter 2

Background concepts

In this chapter, we outline some of the core concepts from machine learning utilised in this study. Firstly, we provide an introduction to reinforcement learning and how it is used to optimise agent behaviour through rewards, policies, and action-value estimation. We also cover two deep learning approximations of reinforcement learning, namely deep Q-networks and REINFORCE. The latter being used extensively in discrete language emergence literature.

Secondly, we give a brief overview of the various language processing techniques that are used throughout this study. We introduce the reader to the deep learning models used for discrete communication. We then look towards methods of spoken communication in the form of automatic speech recognition and speech synthesis.

2.1. Reinforcement learning

The following explanation of reinforcement learning (RL) is based on Sutton and Barto [31]. It is one of the three main subsets of machine learning, the others being supervised learning and unsupervised learning. In contrast to supervised and unsupervised learning, in RL the data used to update the model is gained through experience, rather than using a predefined dataset. The RL task is generally phrased as some agent acting in an observable environment. This makes RL particularly well-suited to real-world optimisation and control system tasks, such as autonomous vehicles, robotics and game theory. As mentioned previously in Chapter 1, RL has also been extensively used to study language emergence within the discrete communication field [1–6]. This stems from the idea that RL can effectively model social settings, where multiple agents communicate to solve a common task. This is the primary motivation for using RL as a tool in this study. We would like to closely model the human experience of language acquisition, where social interaction and intrinsic reward are vital. Section 3.1 expands on this idea and discusses the motivation behind using RL for language acquisition in more detail.

There are three core concepts to consider in RL: the environment, the agent, and the reward signal. The environment defines the setting in which our agent acts. As an example, consider a game of chess, the environment is the chess board, and the environment state is

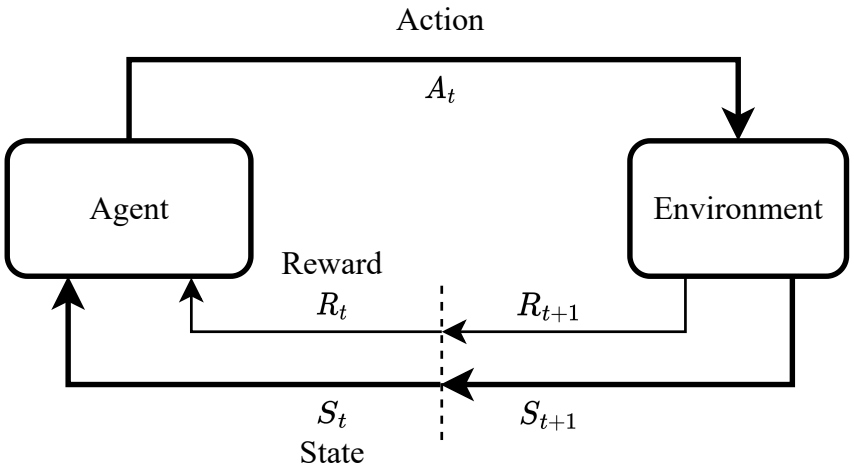


Figure 2.1: Typical agent-environment interaction.

the current position of all the pieces of the chess board. The agent would be the chess player and taking an action would be moving your chess pieces. In a multi-agent RL setting, both players would be agents. The reward signal defines how well the agent is doing given the current environment state and future trajectory, thereby defining a goal. Designing an optimal reward signal is a challenging task. In the same chess example, you could define a positive reward signal for each chess piece the agent takes from the opponent; or perhaps a more detailed approach would be a reward signal proportional to the value of each piece taken. You could also define a reward signal only upon winning or losing the match (referred to as a sparse reward). All of these are valid reward models aimed at winning a game of chess. The agent is tasked with maximising the cumulative reward signal, known as the return. You can see the return as the value of taking a set of moves which take multiple pieces over time – ideally winning the chess game at the end.

A typical agent-environment interaction is provided in Figure 2.1. Formally, at each time-step t, the agent makes an observation of the environment S_t and outputs an action A_t. The agent executes this action in the environment, updating the environment state. The agent then receives the updated environment state S_{t+1} along with a reward signal R_{t+1}. This is a fully observable environment as the agent state and environment state are the same.

2.1.1. Action-value estimation

The intuition behind RL is relatively straightforward, but how do we define how an agent chooses actions given the environment state? One solution is to estimate the value of a given state and/or actions. If the value can be estimated, we simply need to choose the state or action that would yield the highest value. We know the reward signal R_t indicates how well the agent is doing at time-step t – defining the goal. We can also define

the cumulative reward or return $G_t = R_{t+1} + R_{t+2} + R_{t+3} + \dots$ as the sum of all future rewards. The values and returns may also be defined recursively: $G_t = R_{t+1} + G_{t+1}$. The expected cumulative reward from state s is called the state-value $v(s)$:

$$\begin{aligned} v(s) &= \mathbb{E}[G_t \,|\, S_t = s] \\ &= \mathbb{E}[R_{t+1} + R_{t+2} + R_{t+3} + \dots \,|\, S_t = s] \end{aligned} \tag{2.1}$$

These state-values help define the desirability of certain states or actions. For example, we once again consider a game of chess, where the position of each chess piece is the environment state $S_t = s$. The state-value $v(s)$ estimates how well the player is currently doing and might give an indication of whether they are currently winning. Knowing the value of a state is extremely useful in RL, as it could help an agent choose actions such that they maximise the future return.

We can also condition the state-value function on a particular action, known as state-action-value $q(s, a)$, action-value, or Q-value. We can think of the Q-value as estimating the future reward an agent would receive given it takes action $A_t = a$ while in state $S_t = s$:

$$\begin{aligned} q(s, a) &= \mathbb{E}[G_t | S_t = s, A_t = a] \\ &= \mathbb{E}[R_{t+1} + R_{t+2} + R_{t+3} + \dots \,|\, S_t = s, A_t = a] \end{aligned} \tag{2.2}$$

Using the same chess example, the action-value estimates the value of taking a certain action given the agent is in some state. For instance, taking the opponent's queen would likely yield a high action-value, whereas putting your queen in a position where it could be taken may yield a low action-value.

A policy defines an agent's behaviour as a probability distribution over actions, usually denoted as $\pi(s) = P(a|s)$. This helps us when defining our state-values and action-values, as it gives an indication of trajectory: how an agent will act in future states. The trajectory is important when estimating future rewards, as without it we do not necessarily know how an agent would act. We can update our state-value and state-action-value equations:

$$\begin{aligned} v_\pi(s) &= \mathbb{E}[G_t \,|\, S_t = s, A_t \sim \pi(a|s)] \\ &= \mathbb{E}[R_{t+1} + v_\pi(S_{t+1}) \,|\, S_t = s, A_t \sim \pi(a|s)] \end{aligned} \tag{2.3}$$

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}[G_t \,|\, S_t = s, A_t = a, \pi] \\ &= \mathbb{E}[R_{t+1} + v_\pi(S_{t+1}) \,|\, S_t = s, A_t = a] \\ &= \mathbb{E}[R_{t+1} + q_\pi(S_{t+1}, A_{t+1}) \,|\, S_t = s, A_t = a] \end{aligned} \tag{2.4}$$

Here, we also show the recursive relationship, known as the Bellman Equation [32], where the value of the current state is recursively defined by the value of the next state. This property will come in handy when discussing bootstrapping in Section 2.1.2. In standard RL, we apply a discount factor $\gamma \in [0, 1]$ to the estimated value of the resultant state, defining the desirability for current vs future rewards. In this study, we use a constant $\gamma = 1$. Therefore, we leave out the discount factor for brevity. [31] discusses how this discount factor may be used to tune agent behaviour.

The goal in RL is to find the optimal state-value and state-action-value functions over all policies π: $v^*(s) = \max_\pi v_\pi(s)$ and $q^*(s,a) = \max_\pi q_\pi(s,a)$. These above equations can be combined with dynamic programming or other optimisation techniques to iteratively update our policy π such that we converge on the optimal policy π^*. The policy, state-value, and action-value are all functions. In RL we want to learn these functions from experience. Traditional methods involve tabular functions for v, q and π. Iterative solutions such as dynamic programming, Monte-Carlo evaluation or temporal-difference learning are commonly used to learn the values of these functions [31].

To assist the reader in understanding the equations above, we provide a brief example

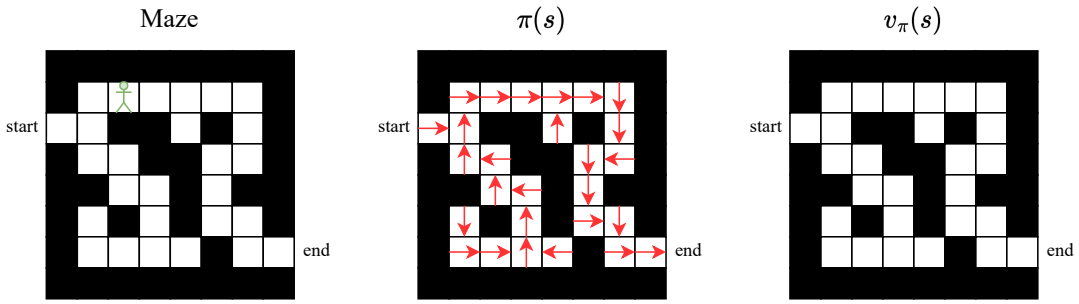


Figure 2.2: An example of an agent solving a maze where each block represents a state s. The optimal deterministic policy $\pi(s)$ and corresponding value function $v_\pi(s)$ are shown.

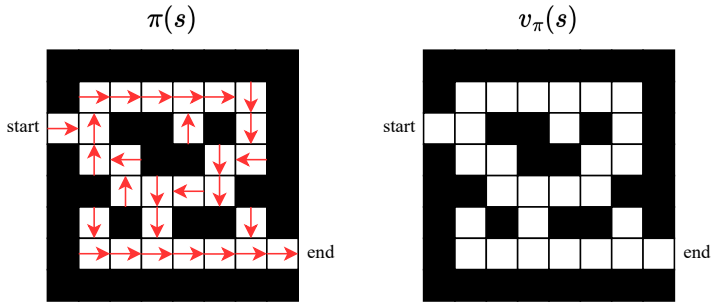


Figure 2.3: A second example of an agent solving a maze. In this example, there are two valid paths and the deterministic policy $\pi(s)$ is sub-optimal.

in Figure 2.2. Here, an agent must solve a maze. The agent is allowed to take one of four actions, $A_t \in \{up, down, left, right\}$. The reward is a constant of $R_t = -1$ for each action taken. In Figure 2.2, we show how a policy $\pi(s)$ defines the state values $v_\pi(s)$. Figure 2.3 shows an example where there are two paths to the goal. In this example, the policy $\pi(s)$ is sub-optimal – it takes the longer path to the goal. Notice how the value function $v_\pi(s)$ changes in Figure 2.3 for the sub-optimal policy.

Tabular functions are effective at modelling the value, action-value, and policy at a small scale. Scalability becomes an issue when there are too many states. Therefore, we approximate these functions. When using deep neural networks to do these approximations, it is often called deep reinforcement learning. We consider two such approaches in this study, deep Q-networks and REINFORCE.

2.1.2. Deep Q-networks

Q-learning is an algorithm used to optimise the action-value function. This is done through iterated sampling of the Bellman Equations, known as temporal-difference control [31]. In deep Q-networks (DQN) [29], we approximate the action-value function $q_\theta(s, a) \approx q_\pi(s, a)$ as a neural network parameterised by θ. We refer to $q_\theta(s, a)$ as a deep Q-network. Mnih et. al. [29] popularised this approach for solving complex tasks, eventually reaching human-level performance on the now popular Atari benchmark [30].

The deep Q-learning loss function is relatively straightforward and hinges on the idea of bootstrapping mentioned before. The loss is calculated as the difference between the predicted action-value $q_\theta(s, a)$ and the actual value of taking that action $R_{t+1} + [\max_{a'} q_\theta(S_{t+1}, a')]$. Deep Q-learning minimises the mean-squared error loss:

$$\mathcal{L}(\theta) = \mathbb{E}\left[\left(R_{t+1} + \left[\max_a q_\theta(S_{t+1}, a)\right] - q_\theta(S_t, A_t)\right)^2\right] \tag{2.5}$$

The term $\max_a q_\theta(S_{t+1}, a)$ estimates the value of the resultant state S_{t+1}. This is known as bootstrapping, as we are updating the weights of the Q-network using an estimation of the actual action value made by that same Q-network. We can also calculate the Q-learning loss gradient with respect to the weights θ [29]:

$$\nabla_\theta \mathcal{L}(\theta) = \left(R_{t+1} + \max_a q_\theta(S_{t+1}, a) - q_\theta(S_t, A_t)\right) \nabla_\theta q_\theta(S_t, A_t) \tag{2.6}$$

We can now perform gradient descent to update the weights θ:

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta) \tag{2.7}$$

where α is the learning rate. With these equations, we want to minimise the error between our predicted value of an action and the observed value gained through experience. If we can minimise this error, our Q-network will provide an accurate estimate of taking each action given the current state. With this, we simply choose the action with the highest predicted value – providing us with a trajectory that will maximise our cumulative reward. We can update our DQN policy as follows:

$$\pi(s) = P(a|s) = \begin{cases} 1 & \text{if } a = \underset{a'}{\text{argmax}}\ q_\theta(S_t, a') \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

An important consideration in DQN is the trade-off between exploration and exploitation. Exploration refers to an agent exploring uncharted territory through random actions. Exploitation refers to an agent acting on what it believes is the optimal trajectory given current knowledge. As an example, imagine you have a favourite restaurant in your area – you know the food there is good. Every time you want to go out for dinner, you visit this restaurant. This is referred to as exploitation, as you are exploiting the knowledge that this restaurant has good food. While you are guaranteed good food, you may be missing out on even better food at another local restaurant. On the other hand, if you visited a different restaurant every time you went for dinner, it would be referred to as exploration. You are exploring multiple possibilities each time you go out for dinner. This comes with both risk and reward; you risk experiencing a bad restaurant or may be rewarded by visiting the best restaurant in town. If we only exploit our knowledge, we often end up with sub-optimal solutions.

There is always a trade-off between exploration and exploitation. Either you exploit your knowledge and visit a restaurant you know is good, or visit a random restaurant and perhaps find an even better restaurant. A common approach to this trade-off in DQN is ϵ-greedy exploration. With some probability ϵ, select a random action at time-step t. Otherwise, greedily select the action with the highest expected reward. Using the same example as before, with probability ϵ, visit a random restaurant. Otherwise, visit your favourite restaurant. With ϵ-greedy, you both exploit your knowledge and occasionally explore to improve your knowledge.

2.1.3. REINFORCE

Deep Q-learning is referred to as a value-based method since the agent policy is updated by the learnt action-value function. Another popular approach RL is policy-based (or

policy gradient) methods, as it is often easier to directly learn the policy without having to learn the value function and then derive the policy afterwards. One of the earliest policy gradient methods, REINFORCE [22], is used almost exclusively in current discrete language emergence literature. In REINFORCE, rather than updating the parameters of a Q-network through gradient descent, we update a policy network through stochastic gradient-ascent. The algorithm is used to optimise the policy: a model that generates a distribution over actions given an input state such that the actions taken maximise the expected reward.

In REINFORCE we approximate the policy function $\pi_\theta(a|s) = P(a \mid S_t = s, \theta)$ as a neural network parameterised by θ. Recall that the policy π defines a probability distribution over actions given the state. We can therefore define the performance measure $J(\theta)$ for a policy parameterised by θ by its value-function as

$$J(\theta) \doteq v_{\pi_\theta}(s) \tag{2.9}$$

where v_{π_θ} is the true value function for the policy π_θ. As mentioned previously, the value function gives an indication of how well the agent is currently doing following its policy. The intuition behind REINFORCE is that we simply want to maximise this value function. To do this, we derive $J(\theta)$ with respect to the weights θ. This is done through repeated unrolling of the value function, finally arriving at the performance gradient [31]:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_a q_{\pi_\theta}(S_t, a)\nabla_\theta(a|S_t)\right] \tag{2.10}$$

We can now update our weights θ through stochastic gradient-ascent:

$$\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta) \tag{2.11}$$

where α is the learning rate as before. The intuition behind REINFORCE is that rather than minimising a loss term, we want to maximise our performance [22].

2.2. Language processing

In this section, we will give a brief overview of various language processing techniques related to deep learning. We first look at language processing for discrete communication, focusing on both the generation and interpretation of discrete sequences. We then move on to speech-related methods, specifically ASR and speech synthesis. These explanations

follow the terminology of Sarker [33].

Recurrent neural networks (RNNs) are particularly useful in language processing for their sequence processing capability. Specifically, gated recurrent units (GRUs) [34] form the basis of all our language processing techniques. The GRU was first proposed by Cho et al. [34] as a simplification to the state-of-the-art long short-term memory (LSTM) units. In the same year, Chung et al. [35] proved GRUs performance to be equivalent to that of LSTM units while being much easier to train and faster to converge due to fewer parameters required. Both LSTMs and GRUs are capable of retaining long term information, unlike standard RNNs, thus making them ideal for sequence generation and sequence-to-sequence models. While transformers with attention [36] may provide better sequence modelling performance, we opt to use GRUs as a simpler, easier-to-train approach with fewer parameters.

2.2.1. Discrete communication

Discrete communication is relatively straightforward and can be split into two parts, encoding (interpreting a sequence) and decoding (producing a sequence). This is performed over a sequence of discrete units, where each symbol consists of a fixed vocabulary $\mathcal{V}$. The following explanation of discrete sequence encoding and decoding is based on the implementation in the Emergence of lanGuage in Games (EGG) toolkit [37].

Discrete sequence decoding

In discrete sequence decoding, we want to go from some fixed-dimensional embedding to a sequence of discrete units. One of the simplest ways to do this is to pass the embedding as the initial hidden state of a GRU. An example is shown in Figure 2.4. Each output of the GRU is passed through a linear layer, reducing the dimension to the size of the vocabulary $|\mathcal{V}|$, referred to as logits. A softmax over these logits defines a distribution over the set of symbols $\mathcal{V}$. These symbols are then either sampled or greedily selected to arrive at the final discrete sequence. We generally use auto-regressive decoding; this means that the model predicts one token at a time, and each predicted token is used as input back into the GRU (dashed line in Figure 2.4) to help predict the next token.

We note the use of <s> and </s> in Figure 2.4. The start-of-sequence symbol <s> indicates to the model when to start decoding. Conversely, the model outputs the end-of-sequence symbol </s> when it is done decoding, allowing it to generate arbitrary length sequences up to a maximum length. It does this by predicting one symbol at a time. If the </s> token is produced, or the maximum length is reached, the sequence is done decoding.

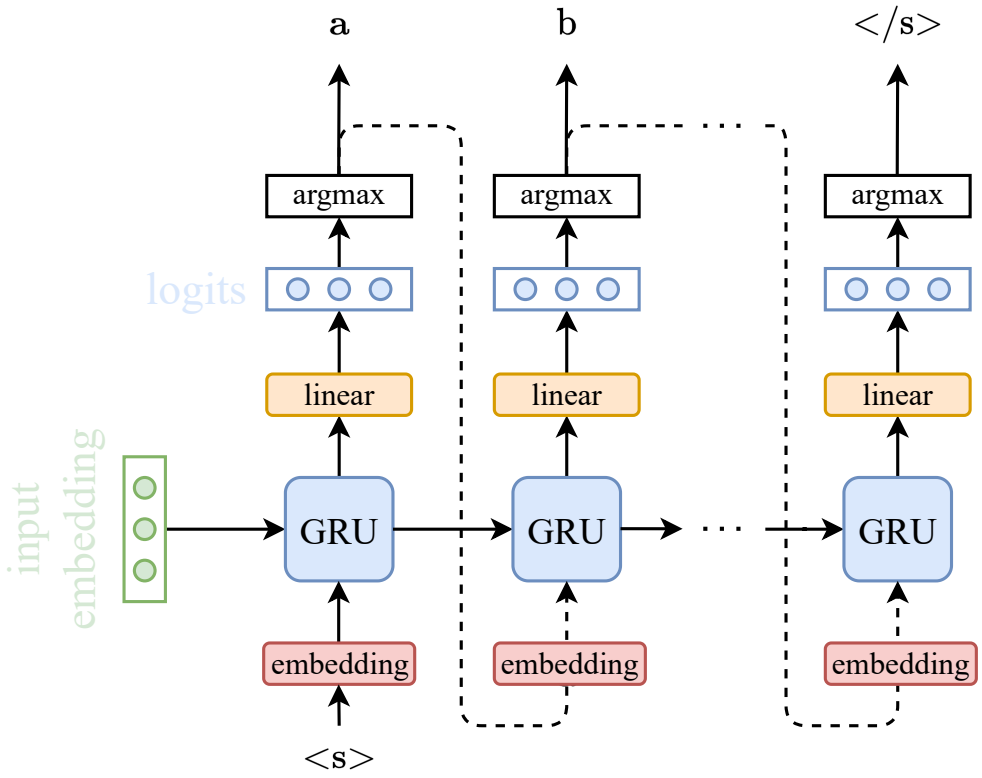


Figure 2.4: Discrete sequence decoding with GRUs. A fixed-dimensional embedding is decoded into an arbitrary-length sequence of discrete symbols. <s> and </s> represent the start-of-sequence and end-of-sequence tokens respectively. The horizontal arrows represent hidden GRU states.

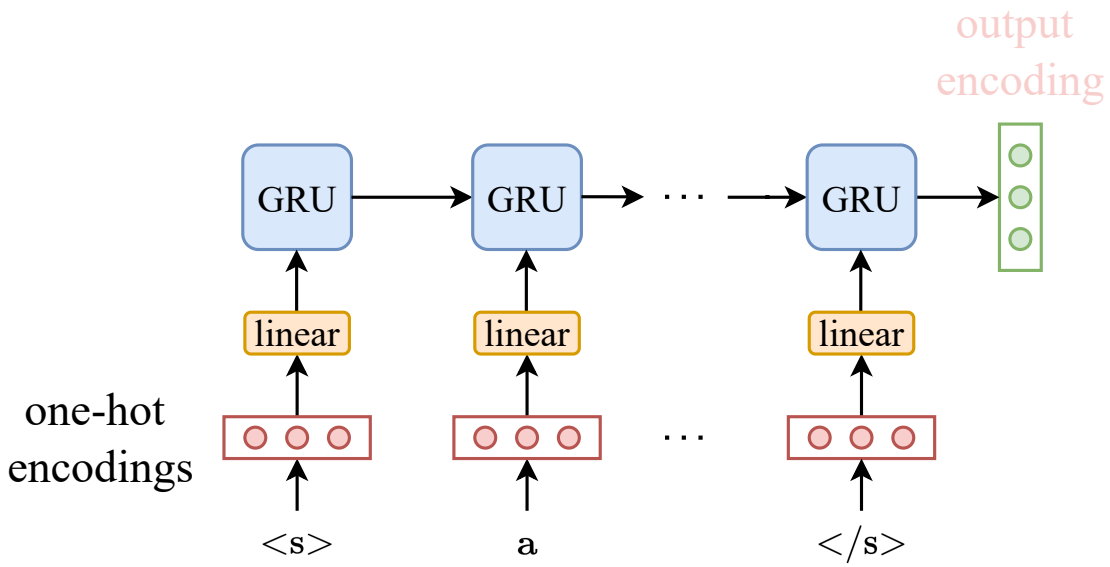


Figure 2.5: Discrete sequence encoding with GRUs. A sequence of discrete symbols (starting with <s> and ending with </s>) is passed through a GRU model, with the final GRU hidden state being used as the fixed-dimensional output encoding.

Discrete sequence encoding

In discrete sequence encoding, we do the opposite; we want to map a sequence of discrete units to a single fixed-dimensional embedding. Figure 2.5 gives an example of discrete sequence encoding. Each symbol in the sequence is first represented with one-hot encoding and passed through a linear layer. In one-hot encoding, each symbol is represented by a vector $\{0,1\}^{|\mathcal{V}|}$. The vector is all 0 except at the index of the symbol in $\mathcal{V}$, where the value is 1. The outputs of this linear layer are fed directly as input to a GRU. The final hidden state of the GRU is then used as the fixed-dimensional embedding.

2.2.2. Automatic speech recognition

When working with spoken language processing, one of the first tasks is how to interpret speech signals. A common approach is to first transcribe speech into discrete natural language, allowing us to apply discrete language modelling techniques. When speech is automatically transcribed with software or machine learning, we refer to it as automatic speech recognition (ASR). This has become a well-studied area if research within the machine learning community [38, 39].

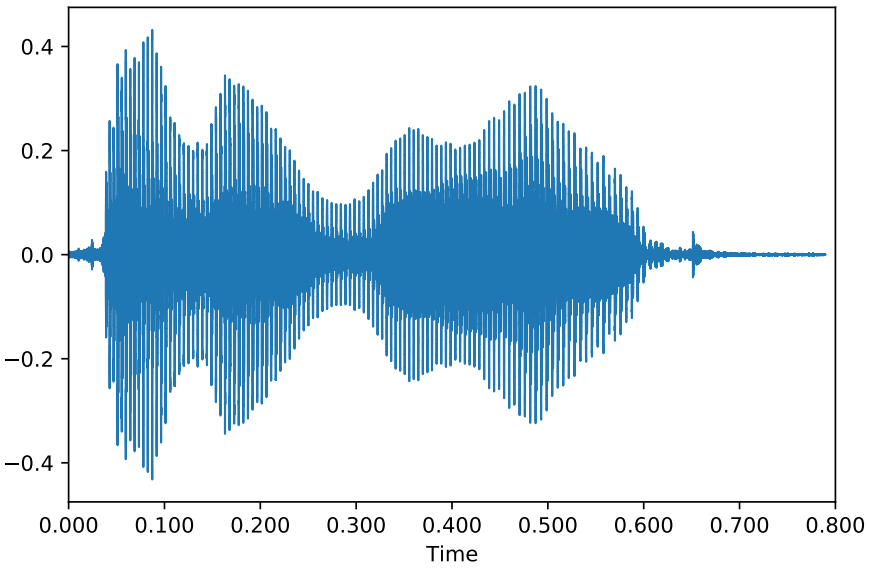
Speech signal representations

One of the most common audio signal representations, and the one used in this study, is the mel-scale spectrogram. A mel-scale spectrogram (or mel-spectrogram) is a sequence of vectors over time representing the frequency content of an audio signal scaled to mimic human frequency perception [40]. Consider the raw audio waveform of the utterance "hello world" in Figure 2.6a. The waveform, sampled at 16 kHz, has the corresponding log mel-spectrogram as shown in Figure 2.6b. Each vector over time in this mel-spectrogram represents a 25 ms window sliding over the raw waveform. The hop-length between each window sample is 10 ms. The frequency content from 64 Hz to 8192 Hz is given in mel-scale on the y-axis. Mel-spectrograms are an effective tool to represent the linguistic content of an audio signal.

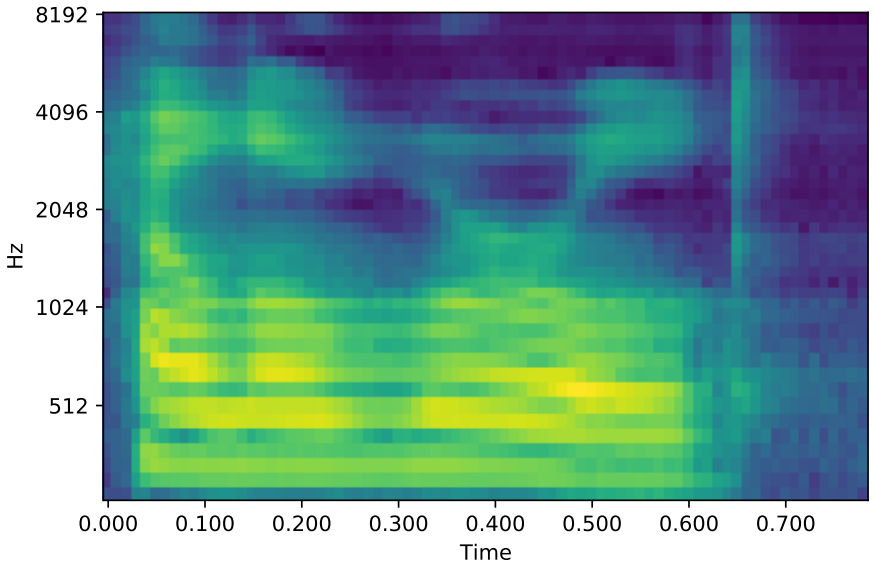
Some recent approaches consider processing the raw audio waveform directly (e.g. [39]). While effective, significantly larger datasets and hundred-million-parameter models are required.

Common model architectures

A simple yet effective end-to-end approach is the model of Deep Speech 2 [38]. Deep Speech 2 applies a 1D convolutional neural network to the log mel-spectrogram of an audio signal in order to perform ASR. The full model architecture is shown in Figure 2.7. The output of the convolutional layers is processed by a bidirectional (the sequence is processed



(a) Raw waveform of the utterance "hello world" sampled at 16 kHz.



(b) Log mel-spectrogram of the utterance "hello world".

Figure 2.6: Raw waveform (a) and the corresponding log mel-spectrogram (b). The mel-spectrogram gives a two-dimensional representation of the frequency content of the raw waveform. Each vector over time represents a 25 ms window, with a hop-length of 10 ms between each vector.

forwards and backwards simultaneously) GRU and a multilayer perceptron (MLP) (a set of linear layers and activation functions).

The state-of-the-art for training end-to-end ASR models, including Deep Speech 2, involves the use of connectionist temporal classification (CTC) [41]. CTC is a method to model the probability of a predicted transcript given a target transcript, serving as a loss function. What makes CTC unique is the ability to handle unaligned sequences. Each output feature of the model corresponds to a small window of time and may produce more

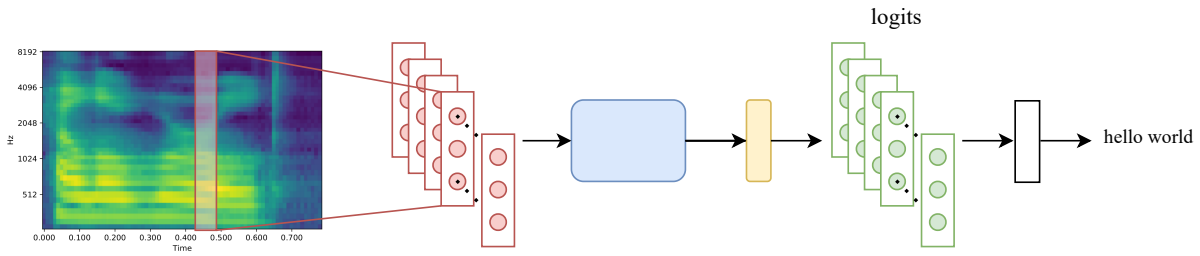


Figure 2.7: Deep Speech 2 model architecture. Given a mel-spectrogram of an utterance, the model produces the corresponding text transcription.

characters than in the target transcript. Therefore, CTC uses a special padding token ϵ, which is removed in the final transcription.

2.2.3. Speech synthesis

The next step in speech processing involves the generation or articulation of discrete sequences into waveforms, referred to as speech synthesis. The goal is to have a system that can convert written language into naturally sounding speech signals. We take a look at three techniques: eSpeak, Festival and Tacotron 2 + HiFi-GAN.

eSpeak and Festival

Both eSpeak [42] and Festival [43] are rule-based formant synthesis systems. In eSpeak, characters are individually synthesised with very limited co-articulation. This means that there is a very limited influence on neighbouring characters in eSpeak. This results in the synthesised waveform sounding less natural or smooth than larger synthesisers, which are typically based on human speech recordings. Festival utilises a much more complex rule schema, allowing for improved co-articulation and more natural-sounding speech.

To perform text-to-speech in both systems, the target text is first converted to a sequence of phonetic units (or phones). Phones are the smallest unit of language and are considered the building blocks of speech. This conversion to phones is done through the lookup of a phonetic dictionary, where every English word has a corresponding phonetic description. For example, in eSpeak, the word “square” becomes the phone sequence $(\mathrm{s}, \mathrm{k}, \mathrm{w}, \mathrm{e}, \mathrm{ə})$. The rule-based systems then convert each phone to a corresponding waveform. For this, eSpeak uses additive synthesis where multiple sine waves of varying frequencies are added together. The prominent frequencies used to determine the sound of a phone are referred to as formant frequencies. Additive synthesis is not as effective for unvoiced consonants (e.g. [s] or [th]). In eSpeak, these are made by playing recorded sounds. These waveforms are then roughly concatenated to generate the final synthesised utterance.

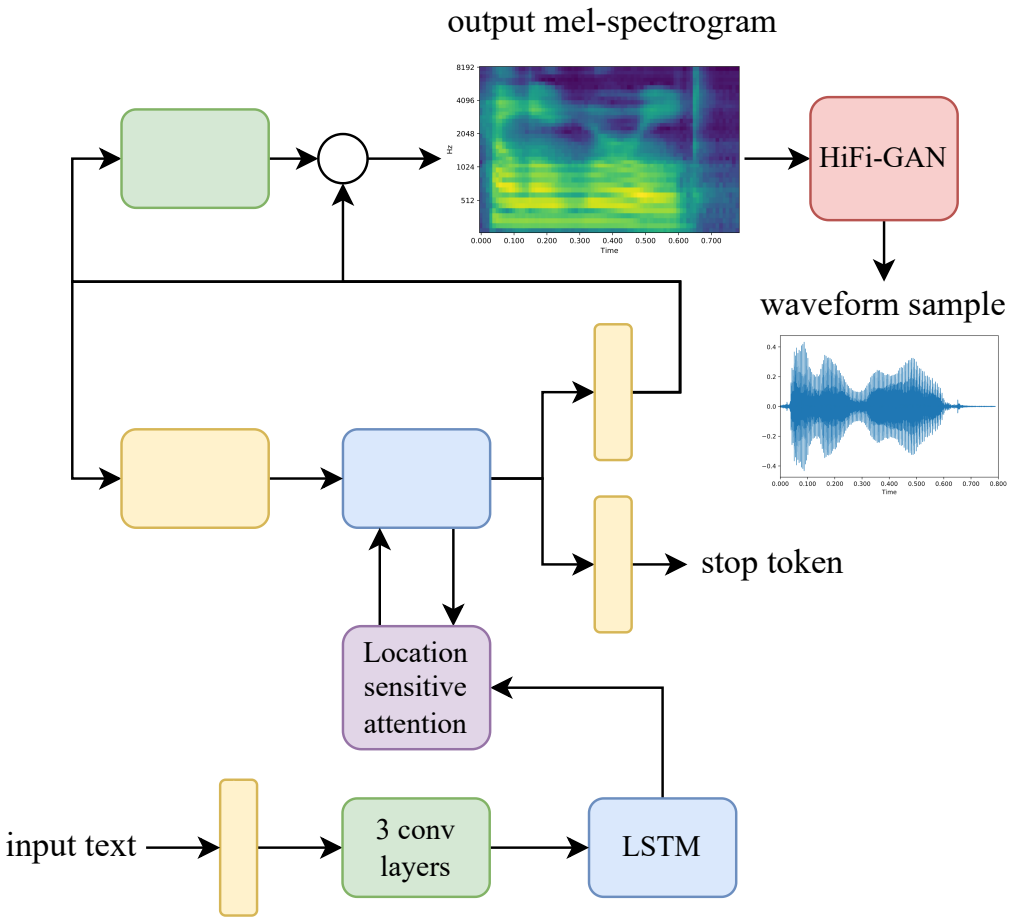


Figure 2.8: Tacotron 2 [44] system architecture with a HiFi-GAN [45] vocoder.

Tacotron + HiFi-GAN

Recently, researchers have turned to deep learning as a tool for training speech synthesis systems. The current state-of-the-art in this area is Tacotron 2 [44], a large neural network model trained to predict mel-spectrogram frames. Tacotron on its own is unable to directly synthesise waveforms, only mel-spectrograms. For this, we use HiFi-GAN [45]. HiFi-GAN, trained as a generative adversarial network (GAN), produces high-quality natural speech from mel-spectrograms. GANs are the process of training two competing neural networks in a zero-sum game. This means that the gain of one network is the loss of another. In HiFi-GAN, one network generates the mel-spectrograms while another tries to differentiate between real mel-spectrogram samples and generated mel-spectrograms output by the generative network. Tacotron and HiFi-GAN are trained independently on hundreds of hours of real audio samples. Combined, they produce synthesised speech almost indistinguishable from human speech [45].

The full system architecture of Tacotron 2 is given in Figure 2.8. Tacotron 2 uses an encoder-decoder architecture. The input text is encoded through an embedding layer, three convolutional layers and a bidirectional LSTM. The resulting hidden feature representation

is consumed by the decoder through location-sensitive attention. Attention is a technique introduced by Vaswani et. al. [36] which improves memory retention in longer sequences. This allows the decoder to produce the output mel-spectrogram frame-by-frame until a stop token is reached; allowing Tacotron 2 to generate arbitrary length mel-spectrograms up to a maximum length. The post-net (Figure 2.8) produces each individual mel-spectrogram frame and the pre-net (Figure 2.8) encodes the LSTM hidden state for the next frame. The final output mel-spectrogram is passed through a vocoder, a network that synthesises a continuous waveform. In our implementation, we use HiFi-GAN as the vocoder.

2.3. Literature summary

In this chapter, we have outlined some of the core concepts used in this study. Specifically, we covered RL and language processing, with specific details pertaining to deep learning. For RL, we introduced the two algorithms compared in this study, DQN and REINFORCE. In the next chapter, we will perform a thorough literature review rather than covering machine learning concepts. We will look at the higher-level motivations surrounding RL as a tool for language acquisition, and the relevant recent work in emergent communication.

Chapter 3

Literature review

We now provide a comprehensive literature review of the work surrounding natural language processing, language acquisition, and the various attempts to model human communication. We first look at the current state of language modelling, and the direction of current trends. This serves as motivation for our work, where we start to consider some aspects ignored by state-of-the-art language modelling. We highlight the disconnect between real human language acquisition and these trends. We proceed by considering language acquisition from a cognitive perspective, alongside tangential attempts to model discrete emergent communication.

We then look at the recent work directly related to this thesis. We first look at various referential signalling games and how they are designed to study discrete emergent languages. Finally, we cover work in spoken language acquisition, which this thesis extends.

3.1. Modelling language acquisition and evolution

Modelling language, and understanding its acquisition and evolution have been studied throughout history [17, 18, 46, 47]. Despite this, we can still only speculate as to how natural language emerged and evolved. In this section, we first take a look at the use of language, the philosophy behind it and the goals of language processing. We then take a look at the two approaches most commonly used in modelling language acquisition and communication emergence.

3.1.1. Language use, philosophy, and goals of language processing

What is language? Dor [17] explains that language is a communication technology, dedicated to the systematic instruction of imagination. We use language to communicate directly to our interlocutor's imagination, being a way of sharing and explaining experience without having them directly experience it. This may be seen as bridging the experiential gaps between speakers.

"You can't learn language from the radio" [18]. Language cannot be learned from linguistic signals alone; with the common belief being that humans rely on non-linguistic knowledge [46, 47]. Bisk et. al. [18] and Linzen [19] reflect on the current direction of

natural language processing (NLP) and argue that many of the assumptions on which communication relies lie outside of text. Yet, recent breakthroughs in NLP focus on training large models on massive text corpora. They are far away from the human-like generalisation to new structures and contexts [19].

Bisk et. al. [18] proposes a notion of world scopes, which go beyond text to consider the foundations of language in grounding, embodiment, and social interaction. They introduce five levels of world scope: corpora and representations, the written world, perception, embodiment and action, and social interaction. The intention of these world scopes is to be used as a lens to audit the progress of NLP.

The first world scope, corpora and representations, focuses on our past in a data-driven manner where datasets of naturally generated language are processed and annotated (stored in written form) for the purpose of learning. The second world scope broadens this idea to large-scale, raw, unlabeled data from the Internet. The majority of current trends in NLP (e.g. [38, 44, 45, 48]) fall within these first two world scopes. The third world scope, perception, considers the world of sights and sound. Language semantics are often based on our perception of the world around us. This is more closely related to multimodal NLP (e.g. [49]), where computer vision is often used to augment NLP. The fourth world scope, embodiment and action, begins to focus more introspectively. Human infants learn to change their perception and understanding through environment manipulation. We find this world scope to relate well to the field of reinforcement learning (RL). The fifth and final world scope, the social world, is hinged on the idea that language is inherently a tool for interpersonal communication. We hope communication between multi-agent reinforcement learning (MARL) agents is able to reflect the core ideas of this world scope, while additionally combining the ideas of perception and embodiment. MARL agents are able to observe their environment (perception), while also being able to take action (embodiment).

Human language also has a fundamental constraint, the Now-or-Never bottleneck [50]. Language is processed using a "chunk-and-pass" strategy: it is impossible to store all the fine-grained acoustics, so we chunk groups of speech sounds together. This poses a challenge for language processing, we either rapidly process information or it is lost forever. Christiansen and Chater [50] argue that this Now-or-Never bottleneck has fundamental implications in studying language emergence. This constraint is common with MARL communication.

NLP encapsulates a wide range of goals and applications. This includes anything from creating dynamic scalable speech recognition systems, generating human-like synthetic speech, and the documentation of low-resource endangered languages [49]. Other applications include anything related to human-robot interaction and the development of real autonomous agents. The application most applicable to this research is the understanding of language origins. We currently do not know for sure where, how and why language

emerged [17].

3.1.2. Language acquisition from a cognitive perspective

The process of language acquisition was initially studied from a cognitive perspective, usually considering vocal development in infants [15, 51, 52]. Human infants acquire their native language by being exposed to speech audio in their environments [15]; by interacting and communicating with their caregivers using continuous signals. The basis for these works stems from behavioural studies of infant development [53].

A popular modelling approach considers imitation learning: a type of social learning where new behaviours are learnt through imitation [51, 52]. Asada [53] claimed that the speech produced by infants and caregivers may be too different to learn purely through direct acoustic matching. Instead, Asada argued that the interaction itself between the infant and caregiver may have an important role to play. A more recent approach by Howard and Messum [16] used a computational model of an infant, they named Elija. They conducted experiments where a caregiver would interact with Elija by responding to the produced utterances. These interactions would serve as a feedback or reward signal to the unsupervised online learning model, with the goal of encouraging natural-sounding utterances. The authors found Elija was able to associate its motor patterns with caregiver responses. This led to Elija being able to pronounce simple words.

Some work, such as Kirby [7], focus more on the learned communication protocol. Kirby introduced an iterated learning model (ILM) to study the evolution of linguistic behaviour. This ILM showcased the ability to learn discrete compositional communication protocols at a small scale with rule-based methods.

3.1.3. Discrete emergent communication

More recent literature has started considering discrete emergent communication modelled with RL agents. Foerster et. al. [54] introduced the first successful communication based on deep multi-agent RL. Foerster et. al. introduced two methods, Reinforced Inter-Agent Learning (RIAL) and Differentiable Inter-Agent Learning (DIAL). The former method, RIAL, allowed agents to communicate through discrete symbols to solve a simple cooperative task. Conversely, DIAL uses a centralised approach where gradients are passed directly through agents. An overview of the RIAL architecture is shown in Figure 3.1. Here, agent 1 makes an observation S_t^1 of the environment and generates a message M_t^1 and action A_t^1. In the next time-step $t+1$, agent 2 consumes M_t^1. The notation used here relates to that used in Section 2.1. The superscript refers to agent 1 and agent 2 respectively. *Q-net* refers to a DQN value network, and *action select* refers to the process of selecting an action given a value-function output. In this case, the Q-net outputs a value for a message M_t and action A_t.

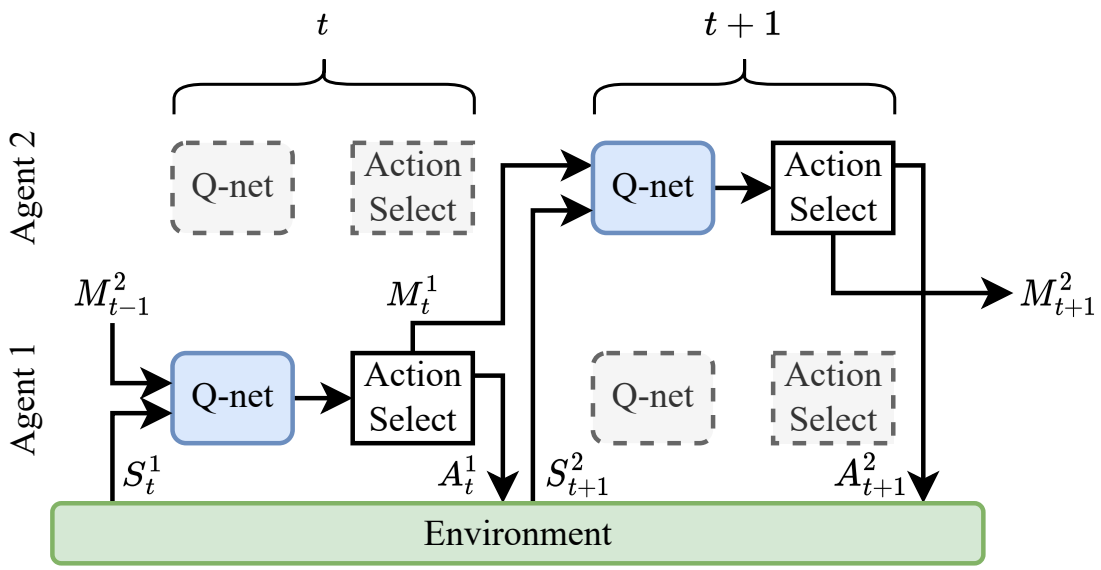


Figure 3.1: RIAL [54] architecture showing two agents interacting with a discrete message M^1_t. Superscripts indicate each respective agent.

Significant advances in discrete language emergence have since been made [2, 3]. Chaabouni et. al. [4] (and more recently [6]) took an in-depth look at the learnt communication protocol, showing the emergence of compositionality and generalisation. Lazaridou [5] and Moulin-Frier [24] both provide comprehensive reviews of how MARL may be used as a computational tool for evolution research.

3.2. Recent work in emergent communication

We now give a quick overview of some recent advances in emergent communication studies. First, we look at the most common type of environment used to study discrete language emergence – referential signalling games. We then consider a few exceptions to the discrete communication trend, where spoken language acquisition is studied.

3.2.1. Referential signalling games

Referential signalling games have become a staple for studying discrete emergent communication. Originally proposed by Lewis [27], the game considers two players, a sender and a receiver. The core idea is that only one player, the sender, is allowed to observe the environment state. The sender must communicate the environment state to the receiver through signalling. With this information, the receiver takes actions within the environment. Generally, both players have a common goal, although some studies consider more competitive interaction between agents [55].

The referential game itself may take on many variations, but the core principle remains the same – a sender communicating with a receiver to solve a common task. The simplest

form of the game considers a reconstruction task, and proceeds as follows [4]:

1. Sender receives $\boldsymbol{s}$ and generates a message.
2. Receiver consumes the message and produces $\hat{\boldsymbol{s}}$.
3. Agents are successful if $\hat{\boldsymbol{s}} = \boldsymbol{s}$

The receiver has no direct access to $\boldsymbol{s}$ and has to perform reconstruction solely on the sender's message. In this setup, the game becomes analogous to an auto-encoder where the bottleneck is the communication channel. Some studies instead consider a differentiable communication channel, comparing gumbel-softmax estimation and supervised learning to RL [1, 2]. In the case of a differentiable communication, the channel takes the form of a continuous vector allowing gradients to flow. Gumbel-softmax [1] is an approach that approximates a softmax at the communication channel, allowing gradients to flow through a discrete channel during training.

Most recent work consider a variant of this referential game where the message is a sequence of discrete symbols [2, 4, 6, 28]. In these setups, $\boldsymbol{s}$ represents an arbitrary concept. In this study, we take a similar approach to Chaabouni et. al. [4], where $\boldsymbol{s}$ represents a set of arbitrary attribute-values.

Some work considers a variation of the game where $\boldsymbol{s}$ represents an image. One example considers an MNIST digit reconstruction game [37], where the receiver must reconstruct MNIST digits. An example is given in Figure 3.2; here, the sender is provided with an MNIST digit that the receiver must reproduce. Note that the communicated message "6" is not necessarily related to the input digit three. The reward signal used in this game is inversely proportional to the mean squared error between the input and reconstructed digit. With this signal alone, the sender is able to learn how to encode the input digit and have the receiver generate a rough reconstruction. Similarly, other work has the receiver select an image from a set of distractors [2, 13]. Figure 3.3 gives an example, in this variation the receiver is not reconstructing the image and is instead using the communication channel to make informed decisions. For the agents to be successful, the receiver must select the correct target image out of a set of images. This set of images includes the target image and two distractors. The distractors are random images from the dataset and are used as negative samples.

Kajić and Aygün [12] took a slightly different approach and proposed a game in the form of a navigation task. In this example, the receiver was placed in a maze environment with an unknown goal location. The sender agent acted as an observer, knowing the goal location and position of the receiver, but unable to act in the environment. To solve the task, the sender had to instruct the receiver on how to move towards the goal.

In all these examples, the communication took the form of a single or sequence of discrete units. We now take a look at spoken language acquisition.

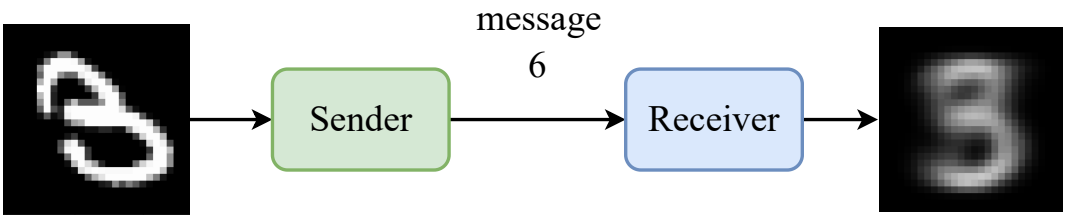


Figure 3.2: Example of the MNIST digit reconstruction game. The sender receives an image of a three, produces message (6,), and the receiver reconstructs a three. The reward used in this game is the inversely proportional to the mean squared error between the input and reconstructed digit.

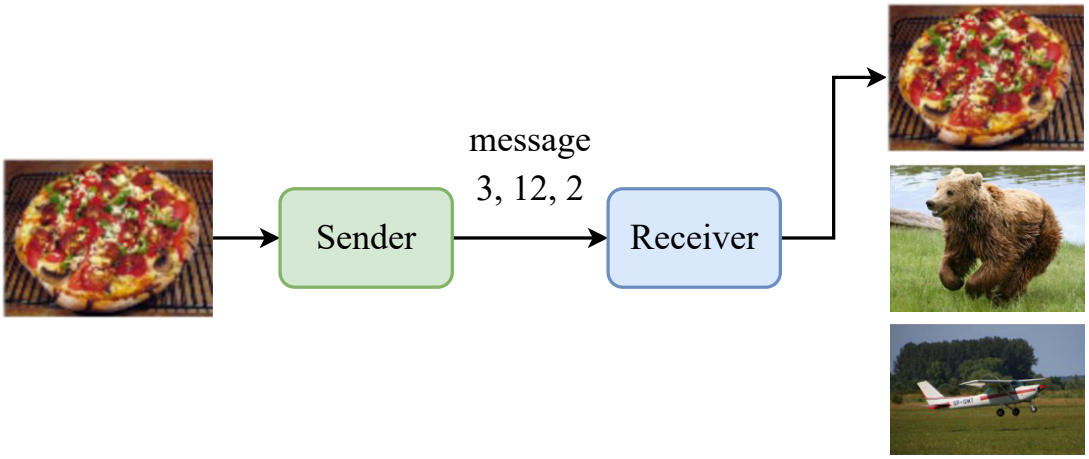


Figure 3.3: Example of a referential game with distractors. The receiver selects the sender's image from a set of images with the target and two distractors (random images used as negative samples). In this example, the sender represented the image of a pizza with the message (3, 12, 2).

3.2.2. Spoken language acquisition

One recent exception which does use continuous signalling within a modern RL framework is Gao et. al. [25] and the follow-up work [26]. The environment combines the navigation task of Kajić and Aygün [12] with spoken acoustic communication. In their approach, a student (the sender) communicates with a teacher (the receiver) by repeating words from a learnt dictionary (see Figure 3.4). This learnt dictionary is constructed with unsupervised word segmentation – a process of splitting a natural language sentence into its component words [56]. The student plays the segmented words to the teacher, which uses a trained automatic speech recognition (ASR) system to classify the words into movement commands in a discrete environment. The student is then rewarded for moving towards the goal position.

Another approach to spoken language acquisition by Rugayan [57] considered vector-quantised autoencoders to perform a similar function of unsupervised acoustic unit discovery. Vector-quantised autoencoders are a type of dimensionality encoder-decoder setup. In this setup, one network must encode speech signals into a discrete vectors, while another

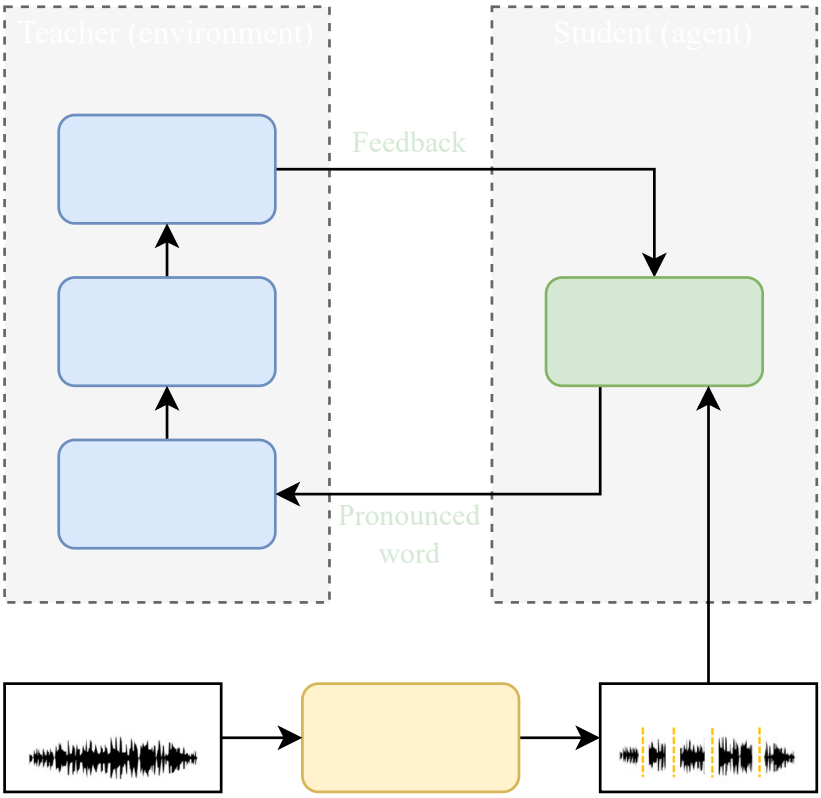


Figure 3.4: Student-teacher system of Gao et. al. [25]. The student communicates with the teacher by repeating words segmented using k-means.

must decode the discrete vectors into the original speech signals. However, instead of a referential game, only a single agent was considered. In this setup, the agent learnt to recite numbers in order from zero to nine.

Both the work of Gao et. al. and Rugayan are limited in the way that the sender is unable to generate novel utterances. In this study, we consider a setup where the agents may generate their own unique audio waveforms rather than just segmenting and repeating words from past observations.

3.3. Literature summary

In this chapter, we provided a review of the related work relevant to this study. We first took a quick look at the history behind language acquisition and evolution, and some attempts at modelling this process. We continued by covering the recent work on which this thesis builds. Specifically, we discussed referential signalling games and spoken language acquisition. In Chapter 4, we will look at combining this work into a single environment, where we may study spoken language as a referential game.

Chapter 4

Environment and Proposed Solution

In this chapter, we first define the environment, which takes the form of a referential communication game. We want to investigate whether we are able to observe emergent language between agents with a continuous communication channel. Alongside this, we want to quantify changes in performance and behaviour between discrete and acoustic communication. The referential communication game, which extends prior work, is designed to help answer the larger research question of whether we are able to learn emergent language between agents communicating with a continuous channel. We then discuss our approach to solving this environment and cover the training and optimisation details.

4.1. Communication game

Our goal is to see whether we can observe emergent language between agents with a continuous communication channel. To this end, we extend a well-defined signalling game environment [27] used to study discrete language emergence. This is advantageous as we may also answer the question of "what changes?" between discrete and acoustic communication.

We base our environment on the referential signalling game from [4] and [28]—which itself is based on [27] and [2]—where a sender must convey a message to a receiver. As a reminder, in these referential signalling games, two players communicate through a one-way channel. One agent, the sender, must communicate the environment input to a receiver. The agents are successful if the receiver is able to reconstruct the environment input (Section 3.2.1). In the case here, communication takes place between a Speaker and a Listener over a continuous acoustic channel, instead of sending symbols directly. In each game round, a Speaker must convey a set of attributes to a Listener agent. The Speaker needs to transmit these attributes using a speech waveform which is transmitted over a noisy communication channel, and then received by a Listener agent. The Listener agent then classifies its understanding of the Speaker's attributes. If the Speaker's target attribute matches the classified attributes from the Listener, the agents are rewarded. The Speaker is then presented with another set of attributes and the cycle repeats.

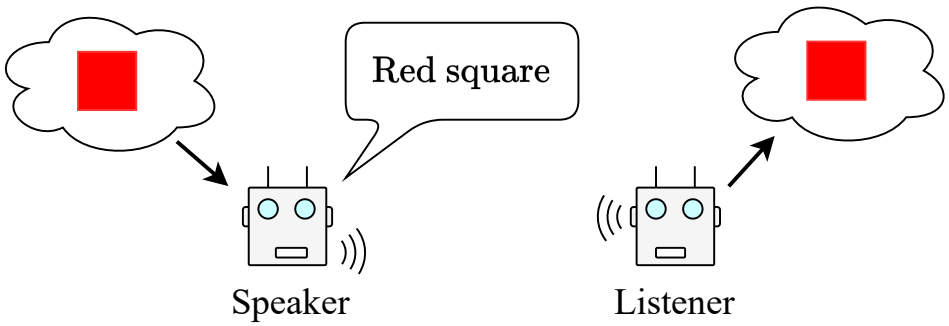


Figure 4.1: Example of a Speaker conveying the concept of a "red square" to a Listener in English.

Figure 4.1 provides a very simple example where a Speaker is conveying two attributes, a colour and a shape, to a Listener. In this example, the Speaker conveys the concept of the *red* and *square* attributes over an acoustic communication channel with the English utterance "red square". The Listener receives the communicated signal and predicts what the Speaker was trying to convey. The Listener correctly identified the colour and shape as *red* and *square*, and therefore the agents are rewarded. It is important to note that the signal passed between the agents takes the form of a continuous acoustic signal, which is passed over a lossy communication channel.

The core difference between our referential game and that of previous work is the inclusion of this continuous acoustic channel. This environment is simple yet sufficient to study the emergence of natural language while providing a platform to investigate what changes between our acoustic communication and prior work. We want to study the effects of lossy communication channels that mimic the real-world scenarios where human language emerged. Because of this, the realism of the communication channel is vital.

In the setup here, the communication is unidirectional from Speaker to Listener. This may limit the social interaction aspect of the environment as the only feedback the Speaker receives is the reward signal. In future work, more conversational communication could be investigated. In Section 7.1 we discuss an extension to this environment to include multiple communication rounds.

4.2. Approach

To implement and solve the above communication game, we break the environment into four components: a Speaker agent, a synthesiser, a channel and a Listener agent. The Speaker agent works with a static synthesiser to produce the audio waveform. This simplifies the task of the speaker agent, as generating natural-sounding raw waveforms directly is a challenging task, and will be considered for future work. The utterance generated by the Speaker and synthesiser is then passed through a realistic lossy communication channel, where the signal is distorted and noise added. The resulting noisy signal is then received by the Listener as a mel-spectrogram.

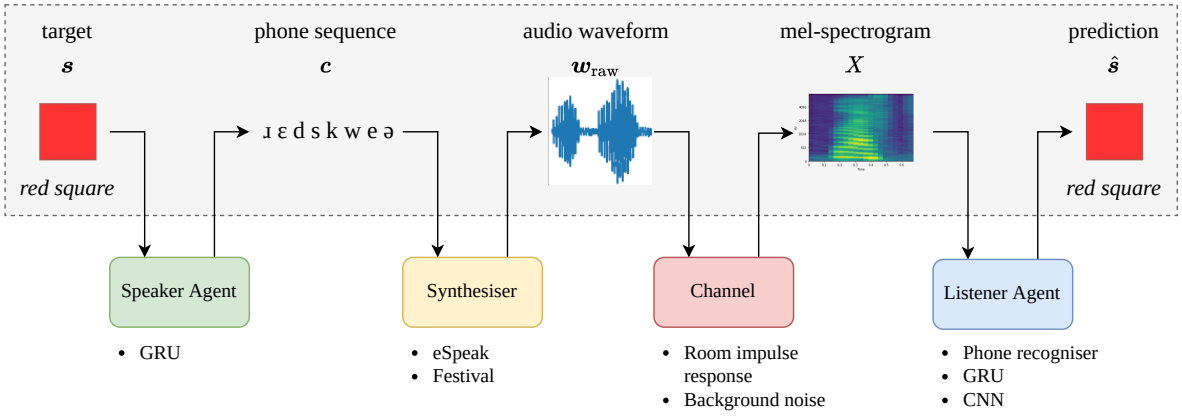


Figure 4.2: Example interaction of each component and the environment in a single round.

Formally, in each episode, the environment generates a set of attributes $\boldsymbol{s} = (s_1, s_2, \ldots, s_N)$, defined as N one-hot encoded vectors each representing one of M attribute values, i.e. $s_i \in \{0,1\}^M$. The Speaker receives $\boldsymbol{s}$ and generates a sequence of phones $\boldsymbol{c} = (c_1, c_2, \ldots, c_L)$, each $c_t \in \mathcal{P}$ representing a phone from a predefined phonetic alphabet $\mathcal{P}$. The phone sequence is then converted into a waveform $\boldsymbol{w}_{\text{raw}}$, an audio signal sampled at 16 kHz. For generating the Speaker's phone sequence, we use a trained text-to-speech model (Section 2.2.3). A channel noise function is then applied to the generated waveform, and the result $\boldsymbol{w}_{\text{in}} = f(\boldsymbol{w}_{\text{raw}})$ is presented as input to the Listener. The Listener converts the input waveform to a mel-scale spectrogram: a sequence of vectors over time representing the frequency content of an audio signal scaled to mimic human frequency perception [40]. Taking the mel-spectrogram sequence $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$ of T acoustic frames as input, the Listener agent outputs a vector $\hat{\boldsymbol{s}}$ representing its prediction of the attribute values. The agents are both rewarded if the predicted vector of attributes is equal to the target vector of attributes $\boldsymbol{s} = \hat{\boldsymbol{s}}$.

To make the environment a bit more concrete, we present a brief example in Figure 4.2. Consider a case where we transmit two attributes ($N = 2$): one colour and one shape. Each attribute can take on one of three values ($M = 3$): $s_1 \in \{red, green, blue\}$ and $s_2 \in \{circle, square, triangle\}$. The concatenated state representation for *red square* would be $\boldsymbol{s} = [1,0,0,0,1,0]^\top$. A possible phone sequence generated by the Speaker could be $\boldsymbol{c} = (\text{r}, \varepsilon, \text{d}, \text{s}, \text{k}, \text{w}, \text{e}, \text{ə}, </\text{s}>)$. This would be synthesised, passed through the channel, and then interpreted by the Listener agent. If the Listener's prediction is $\hat{\boldsymbol{s}} = [1,0,0,0,1,0]^\top$, then it correctly interpreted the message as conveying the attributes *red square*. The environment would then reward both agents.

The environment reward is calculated as $R = \frac{1}{N}\sum_{i=1}^{N} r_i$, where r_i is the per-attribute reward:

$$r_i = \begin{cases} 1 & \text{if } s_i = \hat{s}_i \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

Since we use an existing text-to-speech system for the Speaker's output, the Speaker's task has in effect been converted to a discrete communication problem. However, the combination of both agents and their environment is still a continuous communication task: the noisy channel is based on real acoustic environments and the Listener takes in a noisy acoustic signal (see Section 4.2.3 for full details). What we have done here is to equip the Speaker with articulatory capabilities so that these do not need to be learned by the model. Some studies consider how articulation can be learned [16,52,53], but none of these does so in an RL environment, rather using a form of imitation learning. In Section 7, we discuss how future work could consider learning the articulation process itself within our environment, and the challenges involved in doing so. With this environment setup, if we remove the synthesiser and channel component, we revert to the discrete communication environment of [2,4,28]. This will prove useful in later sections where we compare acoustic communication directly to the discrete case.

Each component is now described in detail.

4.2.1. Speaker agent

The Speaker agent is tasked with generating a sequence of phones $\boldsymbol{c} = (c_1, c_2, \ldots, c_L)$ describing a set of attributes. The set of target attributes is represented by the one-hot input state $\boldsymbol{s}$. We use gated recurrent unit (GRU) [58] based sequence generation as the core of the Speaker agent. The Speaker's GRU generates a sequence of logits. In DQN, these logits represented the predicted action-values $q_\theta(\boldsymbol{s})$ (Section 2.1.2) as shown in Figure 4.3. As a reminder, these action-values represent the expected reward for choosing each action (phone sequence) given the input state $\boldsymbol{s}$. At each output step (from 1 to L), a softmax over the logits defines a distribution over the set of phone symbols $\mathcal{P}$. During training, the symbols are sampled based on this distribution to produce $\boldsymbol{c}$. At test time, $\boldsymbol{c}$ is greedily selected from the logits. The input state $\boldsymbol{s}$ is embedded through a linear layer as the initial hidden state of the GRU. We also make use of start-of-sequence and end-of-sequence tokens, <s> and </s> respectively, appended to the phone set. These allow the Speaker to generate phone sequences of arbitrary length, up to a maximum of L. These phones are then passed to a separate synthesis system, described next.

It is important to note that the Speaker agent is detailed here is not predisposed to communicate using anything close to English. The agents are allowed to develop any arbitrary communication protocol. In Section 6.4.1 we consider grounding the learnt communication to English.

Synthesis implementations

We compare two rule-based synthesis algorithms to the state-of-the-art of deep neural network synthesis (2.2.3). The rule-based synthesis algorithms, eSpeak [42] and Festival [43],

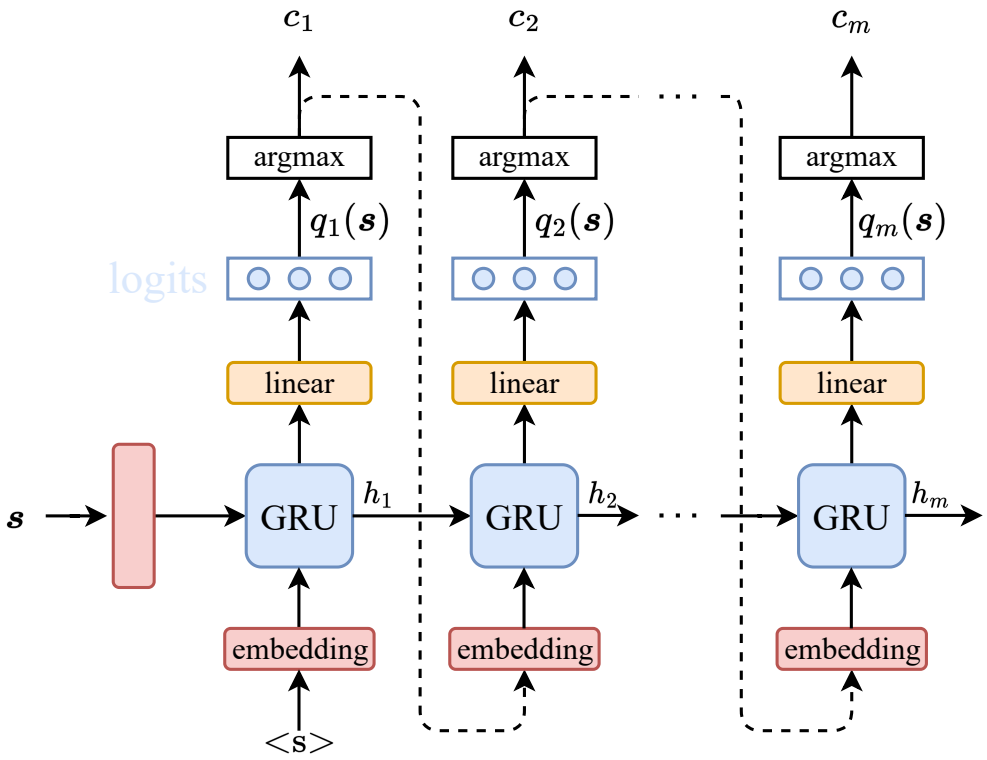


Figure 4.3: Architecture of the GRU-based Speaker. For DQN, the logits represent a set of Q-values for each item in the phonetic vocabulary $\mathcal{P}$.

are both deterministic rule-based synthesis systems. Tacotron 2 [44] and HiFi-GAN [45] combined are used as the state-of-the-art deep neural network synthesis model. All three systems allow for speech synthesis from character or phone sequences.

Both eSpeak and Festival do not have direct Python implementations. Instead, wrappers were implemented that call the respective algorithms from command line using the Python `subprocess` module. The module `miniaudio` [59] is then used to convert the captured byte sequences to a waveform vector. Librosa [60] is used to resample the waveform to a consistent target frequency. The Python implementations for both eSpeak and Festival are given in Listing 4.1 and Listing 4.2 respectively. A list of standard imports for all Listings is given in Appendix A. Festival does not natively support out-of-domain phone sequences. To cater for this, we overwrite the phonetic description of "_" in the Festival lexicon to be our own custom phone sequence. The Festival system then pronounces "_" as our custom sequence.

To use Tacotron 2 and HiFi-GAN, we use PyTorch models trained on a single speaker. PyTorch [61] is a Python deep learning framework used to do automatic differentiation. The models are directly available on PyTorch Hub [61]. These models are loaded and used for inference. Tacotron 2 generates mel-spectrograms given character or phone sequences. HiFi-GAN generates audio given mel-spectrograms. Combined they generate natural-sounding English utterances.

Listing 4.1: Python wrapper for eSpeak. Given a sequence of phones, eSpeak synthesises a corresponding waveform in a target language.

```python
def phones_to_speech(
    phones: str,
    rate: int=16000,
    lang: str='EN',
) -> np.ndarray:
    """ Use eSpeak to convert phones to speech. """

    audio_bytes = subprocess.run(
        ['espeak-ng', f'-v{lang}', '-z', '--stdout', f'[[{phones}]]'],
        stdout=subprocess.PIPE
    ).stdout

    decoded_audio = decode(audio_bytes,
                           nchannels=1,
                           sample_rate=22050,
                           output_format=SampleFormat.FLOAT32)
    decoded_audio = np.array(decoded_audio.samples)
    decoded_audio = resample(decoded_audio,
                             orig_sr=22050,
                             target_sr=rate)

    return decoded_audio
```

Listing 4.2: Python wrapper for Festival.

```python
def phones_to_speech(
    phones: str,
    rate: int=16000,
) -> np.ndarray:
    """ Use festival to convert phones to speech. """
    audio_bytes = subprocess.run(
        f"echo \"_\" | text2wave -eval \
                '(or (lex.add.entry '( \"_\" n ({phones}))) \
                (voice_cmu_us_slt_arctic_hts))' \
                -F {rate}",
        stdout=subprocess.PIPE
    ).stdout

    decoded_audio = decode(
        audio_bytes[-44:] + audio_bytes[36+8:-44],
        nchannels=1,
        sample_rate=rate,
        output_format=SampleFormat.FLOAT32
    )

    decoded_audio = np.array(decoded_audio.samples)

    return decoded_audio
```

A direct comparison of the three synthesis systems is given in Table 4.1. We find Festival difficult to work with due to the complicated phone schema and comparatively slow inference speed. One thing to note about Tacotron 2 + HiFi-GAN is that the model is specifically trained for English words. This means that the model often malfunctions when provided with unseen phone sequences. When this occurs, the model is unable to

Table 4.1: A comparison of various speech synthesis mechanisms. Phone support refers to whether a synthesiser uses a standard phone set (such as IPA). Speed shows the time (mean ± std) taken to infer the audio waveform of the utterance "hello".

Property	*eSpeak* [42]	*Festival* [43]	*Tacotron 2* [44] + *HiFi-GAN* [45]
Speed	53.3 ± 0.91 ms	222 ± 5.82 ms	276 ± 14.9 ms
Deterministic	Yes	Yes	No
Quality	Low	Medium	Highest
Phone support	Yes	No	Yes
Multilingual	Yes	No	No
Sample freq	22 050 Hz	Any	16 000 Hz

output an end-of-sequence token, thereby generating 10-second utterances which are way above the average of 0.6 seconds. For this reason, we use eSpeak throughout most of this work, with the exception of the consonant-vowel experiments in Section 6.3.4 where co-articulation is necessary. eSpeak also has a much faster inference speed ($\approx 5\times$) than the other synthesisers.

4.2.2. Listener agent

Given an input mel-spectrogram X, the Listener generates a set of predicted attributes $\hat{\boldsymbol{s}}$. We use two setups for the Listener agent, an end-to-end Listener and a discrete listener combined with a static phone recogniser. The former predicts $\hat{\boldsymbol{s}}$ directly from X, while the latter has an intermediary discretisation step.

End-to-end Listener

The model architecture in Figure 4.4 is roughly based on Deep Speech 2 [38] – a convolution-based ASR model (Section 2.2.2). The model first applies a set of convolutional layers over the input mel-spectrogram, keeping the size of the time-axis consistent throughout. The convolution outputs are then flattened over the filters and feature axes, resulting in a single vector per time step. Each vector is processed through a GRU, with a linear layer applied to the final hidden state to produce logits over attributes. An argmax of these logits gives us a greedy prediction for $\hat{\boldsymbol{s}}$. We call this the "end-to-end acoustic Listener", as there are no intermediary steps going from the mel-spectrogram input to $\hat{\boldsymbol{s}}$.

Phone recogniser Listener

As an alternative to the above approach, we simplify the task of the Listener: we first process X through a pre-trained static phone recogniser as shown in Figure 4.5. The model of the phone recogniser is a direct implementation of Deep Speech 2 [38], as shown in Section 2.2.2. The model is trained to predict phone sequences of an input audio signal represented as a mel-spectrogram. The trained phone recogniser has a phone error rate

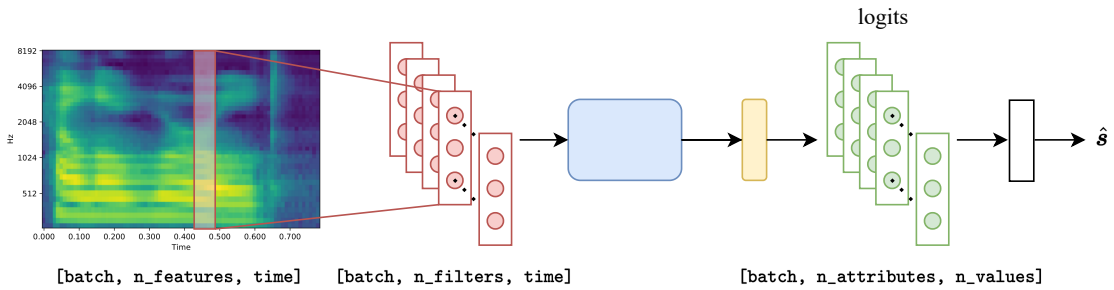


Figure 4.4: Architecture of the end-to-end Listener. The Listener predicts $\hat{s}$ given a mel-spectrogram X.

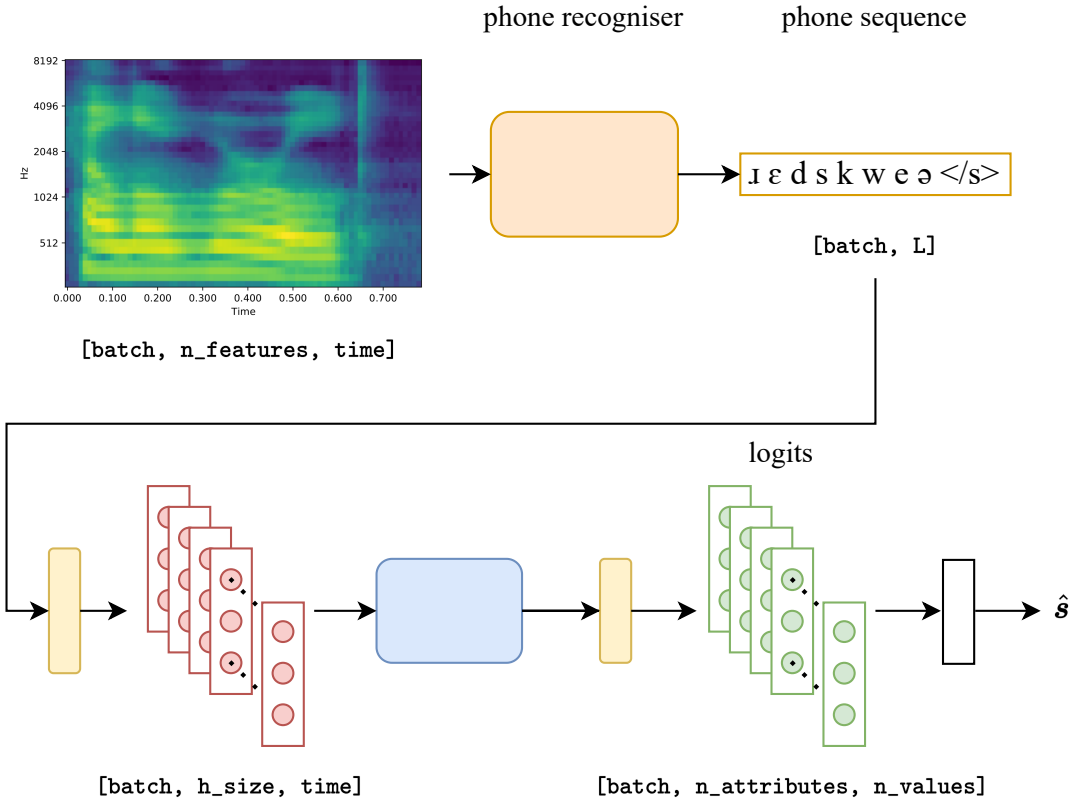


Figure 4.5: Architecture of the phone recogniser Listener. The Listener predicts $\hat{s}$ given a phone sequence produced by Deep Speech 2.

(PER) of 6.42%. These phones are then consumed by the Listener agent, embedded with a linear layer, and then processed by a GRU. The final hidden state of the GRU is passed through a linear layer and an argmax to arrive at our final greedy prediction for $\hat{s}$. The GRU implementation here follows the discrete sequence encoding from Section 2.2.1. In our experiments, we denote this approach with a "+ PhoneRec" label. With this agent, communication is still different to the purely discrete case, since both the Speaker and Listener need to develop a protocol that can compensate for information loss over the continuous communication channel (described next).

4.2.3. Communication channel

The simulated lossy communication channel f consists of two core components: superimposed background noise and convolution with a room's impulse response. We first add background noise directly to the raw audio signal. The background noise is randomly sampled from the Clotho audio captioning dataset [62]. We then convolve the resulting waveform with a room's impulse response from the Aachen Impulse Response (AIR) Database [63]. We use five rooms in total. In training, we use the *booth*, *lecture* and *office* rooms, while for testing, we use the *meeting* and *stairway* rooms separately as unseen evaluation rooms. The full channel function f is shown in Equation 4.2, where $\boldsymbol{n} \in \mathcal{N}$ is sampled from the Clotho dataset $\mathcal{N}$, $\boldsymbol{h} \in \mathcal{H}$ is sampled from the AIR database $\mathcal{H}$, σ is a noise scaling factor, and $*$ denotes convolution.[1]

$$f(\boldsymbol{w}_{\text{raw}}) = (\boldsymbol{w}_{\text{raw}} + \sigma \cdot \boldsymbol{n}) * \boldsymbol{h} \tag{4.2}$$

The channel is sufficient in mimicking both realistic background noise (from Clotho) and realistic channel distortions (impulse response from AIR). We implement the full channel function in PyTorch [61] as shown in Listing 4.3.

Listing 4.3: PyTorch implementation of the realistic communication channel.

```python
def apply_noise(
    wav: torch.tensor, # raw waveform
    noise_wav: torch.tensor, # clotho noise sample
    impulse_wav: torch.tensor, # room impulse response sample
    wav_rms: float = 0.07927095, # measured RMS of eSpeak
    snr: float = 10, # target SNR
) -> torch.tensor:
    """ Apply noise according to Equation 3.1 """

    # calculate RMS of noise sample and find variance to maintain SNR
    noise_rms = np.sqrt(np.mean(noise_wav[:]**2))
    noise_var = self.espeak_rms/((10**(noise_db/20))*noise_rms)
    # extract random segment of the noise sample that fits the original wav sample
    noise_wav = noise_wav[np.random.randint(0,noise_wav.shape[0]-wav.shape[0]):]

    # add scaled noise sample
    wav = wav + noise_wav[:wav.shape[0]] * noise_var

    # pad sequences to length N prevent aliasing when applying convolution
    N = impulse_wav.shape[0] + wav.shape[0] - 1
    N = int(2**(np.ceil(np.log2(N)))) # Round to nearest power of 2

    # pad with zeros
    impulse_wav = torch.nn.functional.pad(impulse_wav, pad=(0,(N-impulse_wav.shape[0])))
    wav = torch.nn.functional.pad(wav, pad=(0,(N-wav.shape[0])))

    # apply FFT multiplication and then inverse FFT
    wav = torch.fft.irfft(torch.fft.rfft(wav) * torch.fft.rfft(impulse_wav))
    return wav
```

[1] In this equation we are overloading the notation, with all vectors representing discrete-time sequences.

In this implementation, we can set the signal-to-noise ratio (SNR) of the input waveform relative to the noise sample from Clotho. To do this, we supply the typical root-mean-square (RMS) of the waveform (measured as 0.0793 for eSpeak) and the target SNR. Using the measured RMS of the noise sample we calculate a scaling factor σ to apply to $\boldsymbol{n}$ to reach the target SNR:

$$\begin{aligned} \text{SNR} &= 20 \log_{10} \left(\frac{\boldsymbol{w}_{\text{rms}}}{\sigma \cdot \boldsymbol{n}_{\text{rms}}} \right) \\ \sigma &= \frac{\boldsymbol{w}_{\text{rms}}}{\boldsymbol{n}_{\text{rms}} \cdot 10^{\frac{\text{SNR}}{20}}} \end{aligned} \tag{4.3}$$

To apply discrete-time convolution, we first convert the waveform and impulse response to the frequency domain. Recall that frequency-domain multiplication equates to time-domain convolution. We use the built-in fast Fourier transform functions in PyTorch. To prevent aliasing, we pad both signals to $|\boldsymbol{w}_{\text{raw}}| + |\boldsymbol{h}| + 1$.

Real communication channel

As an extension to the simulated realistic communication channel, we also consider using a real communication channel to train and evaluate our agents. This would allow us to test the generalisation of our agents to a real-world environment. In such a setup, the generated audio waveforms of the Speaker agent would be played over a physical loudspeaker in various scenarios (such as a postgraduate student lab). A microphone would simultaneously record the audio which would then be passed to the Listener agent. Such a setup could be performed in multiple rooms with varying levels of noise.

As a robust implementation of the real communication channel, a Raspberry Pi is used to run a Flask [64] REST web interface as shown in Figure 4.6. A Raspberry Pi [65] is a small Unix-based microprocessor with sufficient functionality to run Python-based web servers. The local device used to train and evaluate the agents would send and receive audio waveforms over the network to the Raspberry Pi using a `POST` request. This is equivalent to uploading an input audio file to a web page and receiving the resultant audio file in return. The Flask server implementation is given in Listing 4.4.

The biggest constraint with this real communication channel is training time. In our simulated realistic communication channel, we can process audio waveforms much faster than in real time. With the real communication channel, the audio playback would of course be in real-time – resulting in a slow-down of up to 100×. For this reason, we restrict the usage of the real communication channel to the small-scale experiments in Section 6.4.2. As future work, we could consider using multiple Raspberry Pi's in multiple room environments (see Section 7.1 for discussion).

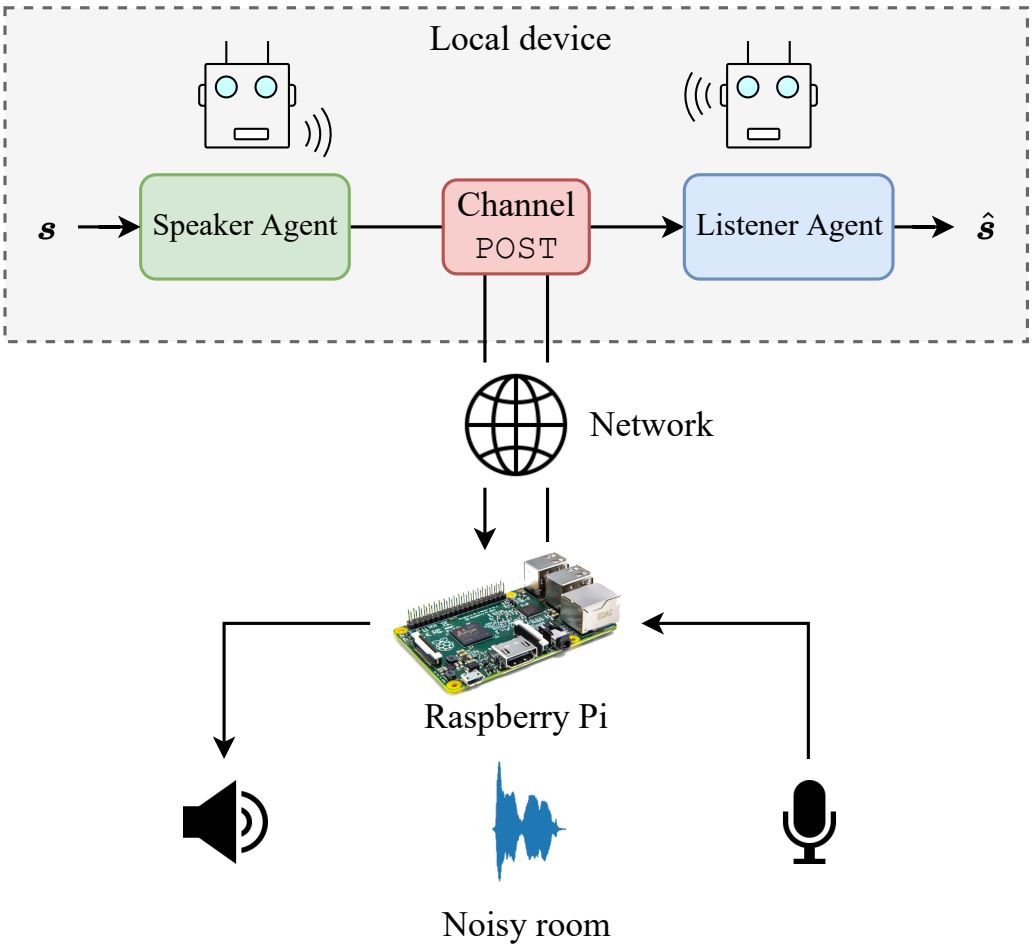


Figure 4.6: Real communication channel using a microprocessor and web interface. The channel sends a `POST` request to a Raspberry Pi web server.

Listing 4.4: Flask web server for audio recording and playback.

```
app = Flask(__name__)

@app.route('/play',methods = ['POST'])
def play():
    if request.method == 'POST':
        wav = np.array(request.json['audio'])

        rec = sd.playrec(wav, samplerate=request.json['sr'], channels=1)

        sd.wait()

        rec = np.array(rec)[:,0]
        return jsonify({'recording': rec.tolist()})
    return 'Error'

app.run(host="192.168.196.1", debug=True)
```

4.3. Training and optimisation

To train our agents we follow the approach of Chaaubouni et. al. [4], which differs slightly from standard deep RL (e.g. [29, 30]). In standard RL, a replay buffer is used to store past experiences when acting in an environment. A replay buffer [29] stores state transitions. Each state transition contains the state S_t, the action taken A_t, the reward received R_t, and the resultant state S_{t+1}. During training, a batch of state transitions is sampled and used to update the weights θ according to the gradient update equation of either DQN (Section 2.1.2) or REINFORCE (Section 2.1.3).

Due to the nature of our referential game and finite number of states, a replay buffer is not necessary. To train our agents, we simply iterate over every possible combination of attribute values $\boldsymbol{s}$. For each $\boldsymbol{s}$, the agents produce an action A_t and receive a reward R_t. We use these values to update the weights as described in Section 2.1.2, without having to store state transitions. We do not need to store the resultant state as our environment consists of only a single step.

We implement our acoustic environment and DQN in the EGG toolkit [37], which already has support for the discrete referential game and REINFORCE optimisation. All models are implemented in PyTorch [61], which is used to perform automatic differentiation and network parameter updates.

4.3.1. Optimisation

Most previous emergent language studies use REINFORCE [22], a policy-gradient algorithm, for Speaker optimisation. The algorithm is used to optimise the policy: a model that generates a distribution over actions given an input state such that the actions taken maximise the expected reward – see Section 2.1.3 for details. REINFORCE is known to have high variance and often struggles to consistently converge [23]. Therefore, we compare REINFORCE to DQN [29].

In DQN, the Q-network is used to estimate the value of actions given the current environment state. The Q-network of the Speaker generates a sequence of phones $\boldsymbol{c}$ in every communication round until the end-of-sequence token is reached. The sequence of phones may be seen as predicting an action sequence per environment step, while standard RL generally only predicts a single action per step. To train such a Q-network, we modify the general DQN gradient-descent update given in Equation 2.6. Since we only have a single communication round, our environment is analogous to a one-arm bandit problem, meaning our target is simply the reward R_t (the S_{t+1} term falls away). A one-armed bandit is a slot machine game where each round is independent of the previous [31]. Therefore, we can calculate the Q-learning loss gradient with respect to the weights θ as follows:

$$\nabla_\theta \mathcal{L}(\theta) = \left(R_t - \frac{1}{L} \sum_{i=1}^{L} q_i(S_t, A_t; \theta) \right) \nabla \boldsymbol{q}(S_t, A_t; \theta), \tag{4.4}$$

where R_t is the environment reward, S_t is the environment state, A_t is the action, and $\boldsymbol{q} = (q_1, q_2, \ldots, q_L)$. For the Speaker, q_i is the value of performing the action c_i at output i. For the Speaker, the environment state would be the desired concept $S_t = \boldsymbol{s}$ and the actions would be $A_t = \boldsymbol{c} = (c_1, c_2, ..., c_L)$.

We can also use DQN for Listener optimisation but found it to be slightly less stable than the approach from previous studies where the cross-entropy between $\boldsymbol{s}$ and the Listener's prediction of $\hat{\boldsymbol{s}}$ is optimised. Therefore, to be consistent with prior work, we also use cross-entropy to optimise the Listener. Using cross-entropy to optimise the Listener means we are essentially treating the Listener as a policy network, where the output is a distribution over actions.

For REINFORCE optimisation, the approach is more standard. We simply treat the whole sequence as a single action and calculate the gradient ascent loss following the equations in Section 2.1.3. As a reminder, the derived performance gradient is calculated as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_a q_{\pi_\theta}(S_t, a) \nabla_\theta (a|S_t) \right] \tag{4.5}$$

In our setup, $q_{\pi_\theta}(S_t, a)$ is the observed reward for taking action a in state S_t. We then update our weights θ of the policy $\pi_\theta(a|s)$ through gradient ascent (Section 2.1.3).

4.4. Chapter summary

In this chapter, we first proposed an extension to the referential games used in previous work, where the agents must learn to use spoken communication. This environment provides a platform to study the properties of the spoken emergent language. We continued to discuss our approach to this environment, where we split the task into components. At the end of the chapter, we discussed how we train and optimise our agents.

Chapter 5

Experimental Setup

In this chapter, we detail the experimental setup. We begin by discussing the general implementation details. We then proceed to introduce the metrics used to evaluate the emergent communication developed by the agents. These metrics include generalisation, compositionality and redundancy.

5.1. Implementation details

For our Speaker agent, we use eSpeak as our speech synthesiser. eSpeak is a parametric text-to-speech software package that uses formant synthesis to generate audio from phone sequences. We also experimented with using Festival and Tacotron 2 + HiFi-GAN, but instead favoured eSpeak due to its fast inference, simpler phone scheme, and multi-language support. In Section 6.3.4 we perform consonant-vowel co-articulation experiments using Tacotron 2 + HiFi-GAN. In our experiments, the use of an existing speech synthesiser allows us to focus on the emergence of phonotactic and lexical structure in multi-agent interaction without having to simultaneously focus on articulatory learning (learning to produce individual speech sounds). In future work we may look to relax some of these assumptions: see Section 7 for a complete discussion.

In each communication round, the Speaker is allowed to generate up to L phones. The default phone set used is $\mathcal{P} = \{\mathrm{a}, \mathrm{e}, \mathrm{i}, \mathrm{o}, \mathrm{u}\}$. These are a subset of the available eSpeak phones, denoted here using the international phonetic alphabet (IPA). All experiments are performed with a fixed input size of $N = 4$ (attributes) and $M = 5$ (attribute values), giving a total input size $|S| = M^N$ of 625 attribute combinations. All models are trained until convergence, which always occurs before 50 training epochs. In all experiments, except the grounding experiments of Section 6.4.1, the Speaker and Listener agent are trained simultaneously. All other setup settings match the default configuration of the `compo_vs_generalization` [4] environment in EGG [37], more details are given in Section 5.1.1.

Table 5.1: General training setup and model hyperparameters.

Parameter	*Description*	*Value*
	Training parameters	
framework	Python implementation framework	PyTorch
optimiser	PyTorch model parameter optimiser	Adam
batch_size	Optimiser number of samples per batch	128
learning_rate	Optimiser learning rate	0.001
runs	Runs per experiment across different seeds	5
n_epochs	Maximum number of epochs	50
n_tune_epochs	Maximum number of epochs when using pre-trained models	10
	Model parameters	
h_size	Size of hidden dimension of GRUs and linear layers	256
gru_layers	Number of GRU layers	2
emb_size	Size of embedding layers	30
activation_func	Activation function used on linear layers	ReLU
	Speech processing parameters	
conv_layers	Number of 1D convolutional layers	3
conv_kernel_size	Width of each 1D convolution kernel	3
conv_stride	Stride of each 1D convolution kernel	1
conv_pad	Total padding of each 1D convolutional layer	2
conv_n_filters	Number of filters per convolutional layer	64
n_mels	Number of mel-spectrogram features	40
n_fft	Mel-spectrogram window width	25 ms
hop_length	Mel-spectrogram window stride	10 ms
f_min	Mel-spectrogram minimum frequency	64 Hz
f_max	Mel-spectrogram maximum frequency	8192 Hz
sample_rate	Audio waveform sample frequency	16 kHz
	Hardware setup	
gpu	Graphics processing unit	Nvidia RTX 2080 Super
cpu	Central processing unit	Intel i7-10700k
vram	Available GPU virtual memory (used by models)	8 GB
ram	Available memory (used by datasets and cache)	64 GB

5.1.1. Setup and hyperparameters

Unless otherwise stated, all experiments follow the general setup and hyperparameters as defined in Table 5.1. The training parameters were optimally chosen by trial and error given hardware constraints. Model parameters follow the implementations of previous work [4]. The convolutional and mel-spectrogram parameters follow common implementations for speech processing [38].

5.1.2. Computational complexity and training time

In the majority of our experiments, the agents are trained for 50 epochs and must transmit a total of $|S| = 625$ input attribute combinations ($M = 5$, $N = 4$). This results in a training time per run of around ≈ 3 hours. The computational complexity is linearly proportional to the number of input combinations $|S|$. Early on we performed one experiment with $|S| = 10000$ over 50 epochs and the run took two days. The computational complexity of our system is therefore $O(n)$ in terms of the input dimension. The computational complexity is also linearly proportional to the number of training epochs and independent of message parameters.

Overall, the biggest influence on training time is the speed of the synthesisers discussed in Section 4.2.1. Approximately 80% of an episode's run-time is taken up by the synthesiser.

5.2. Metrics

In this section, we introduce some of the metrics used to analyse the emergent languages. The primary metric used to assess the performance of our models is accuracy. Specifically, we calculate the mean accuracy per attribute over all input combinations. This is equivalent to the mean environment reward R (Section 4.2) for all environment states. The accuracy for random predictions is inversely proportional to the number of attribute values. For our models we therefore have a random baseline accuracy of $\frac{1}{M} = \frac{1}{5} = 20\%$.

In addition to accuracy, we are concerned with the quantification of three other properties – generalisation, compositionality, and redundancy.

5.2.1. Generalisation

Generalisation refers to a model's ability to handle novel input combinations. For example, a model is trained to classify two attributes, a colour and a shape. A *red triangle* and a *blue square* are used as training samples. At test time we ask it to classify *red square* and a *blue triangle*. The attribute combination of *red square* and *blue triangle* has never been seen by the model, but the individual attributes have appeared during training. We say the model is able to generalise if it is able to classify these novel inputs by learning to understand an input could have any combination of attributes.

To test this, we uniformly sample a set of attribute combinations that will be hidden during training – referred to as a uniform holdout set. During test time, we will evaluate the accuracy of the model in classifying these unseen attribute combinations. If the learnt communication protocol is not general, the agents will be unable to classify the unseen attributes and achieve low accuracy.

5.2.2. Compositionality

Compositionality in our context refers to a language or communication protocol. A compositional language is one in which we reuse symbols to denote the same attributes of a communicated concept. For example, we have three colours *red*, *green* and *blue*, and three shapes *triangle*, *square* and *circle*. We are tasked with labelling all nine combinations of colour and shapes. A naive approach would be to use a unique symbol for every combination, ignoring the fact that there are repeated attributes. This approach would use a total of 9 symbols. A compositional approach would be to have a symbol for each attribute value (one for each colour and one for each shape). This would total 6 symbols for the same number of concept combinations. The naive approach uses multiplicatively more symbols as the number of attribute values increases. If we had x colours and y shapes, we would require $x \cdot y$ total symbols, whereas a compositional approach would only require $x + y$ symbols. In human language, these symbols take the form of words.

A model may be able to generalise and not necessarily be compositional. For example *red triangle* may be represented by "AB", and *blue square* may be represented by "AC", the model may generalise and learn a unique protocol where "CB" represents the unseen *blue square*. The communication protocol may be arbitrary as long as it is still able to communicate the information correctly. But this does not mean the protocol is compositional. Although, Auersperger and Pecina [6] showed that there is a correlation between generalisation and compositionality, which contrasts prior work which found no correlation [4].

We use three metrics to quantify compositionality: topographic similarity, positional disentanglement, and bag-of-symbols disentanglement.

Topographic similarity

The most widely-used metric in the discrete emergent language literature is topographic similarity [2, 66]. Topographic similarity, or *topsim*, studies the structural similarity of the emergent communication in terms of Spearman ρ correlation between the input and message space. We calculate this by taking the correlation between all pairwise edit distances in the input and message spaces. This metric gives an indication of how compositional a learnt communication protocol is by finding the correlation between which message symbols are reused for the same input combinations. For this metric, higher is better and the maximum value is 1 when the input and message space are perfectly correlated.

In addition to topographic similarity, [4] introduced positional disentanglement and bag-of-symbols disentanglement as alternative measures of compositionality. In contrast to topographic similarity, which is correlation-based, these methods use entropy to measure compositionality.

Positional disentanglement

Positional disentanglement, or *posdis*, measures the positional contribution of symbols to meaning. Suppose we have a message $\boldsymbol{c} = c_1, c_2, \ldots, c_L$ of fixed length L, where each symbol c_j represents a character from alphabet $\mathcal{V}$ of size K. Let $\boldsymbol{i}^j = \underset{a}{\text{sort}}\ I(c_j; a)$ be the mutual information between the j-th symbol and each input attribute a sorted in descending order. i_1^j is therefore the highest mutual information between symbol c_j and all attributes, while i_2^j is the second highest. Denoting $H(c_j)$ as the entropy of the j-th position (used as a normalising term), we define the positional disentanglement as:

$$\text{posdis} = \frac{1}{L}\sum_{j=1}^{L}\frac{(i_1^j - i_2^j)}{H(c_j)}$$

Similar to *topsim*, higher is better with a range of values between 0 and 1. Intuitively, to do well in positional disentanglement, the change of a single input attribute should correspond to only a single symbol change in the message $\boldsymbol{c}$. This would maximise the difference between the mutual information between the symbol changed and all other symbols.

Bag-of-symbols disentanglement

Bag-of-symbols disentanglement, or *bosdis*, measures distinct symbol meaning but does so in a permutation-invariant way. We adjust the equation of positional disentanglement to now consider the mutual information between attributes and symbol occurrences regardless of position. Let n_j be a counter of the occurrences of the j-th symbol in $\boldsymbol{c}$. We let $\boldsymbol{i}^j = \underset{a}{\text{sort}}\ I(n_j; a)$ be the mutual information between the number of occurrences of the j-th symbol and each input attribute a sorted in descending order. We define the bag-of-symbols disentanglement as:

$$\text{bosdis} = \frac{1}{K}\sum_{j=1}^{K}\frac{(i_1^j - i_2^j)}{H(n_j)}$$

Similar to *topsim* and *posdis*, higher is better with a range of values between 0 and 1. Unlike positional disentanglement, where the change of a single input attribute should correspond to the change of a single symbol, bag-of-symbols disentanglement is related to the number of symbol occurrences in $\boldsymbol{c}$. Therefore, the change of a single input attribute should correspond to a single change in the occurrences of a specific symbol in the message $\boldsymbol{c}$. This would maximise the difference between the number of symbols used per attribute combination, making the metric invariant to the symbol position.

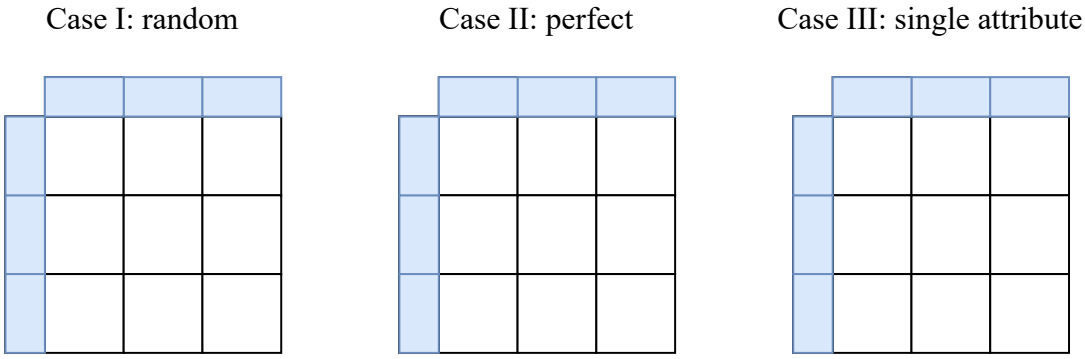


Figure 5.1: Examples of three learnt communication strategies. The metrics for *topsim*, *posdis*, and *bosdis* are also shown.

Analysis of compositionality metrics

We have introduced three metrics to quantitatively measure the compositionality of a message protocol relative to an input space. While these metrics are effective, they do not come without their flaws. In Figure 5.1 we present three cases as examples of a learnt communication strategy. In this example, there are two attributes that can take on three values each. The message space is discrete with a vocabulary $\mathcal{V} = \{\text{'a'}, \text{'b'}, \text{'c'}\}$ and a maximum length of two. For illustrative purposes, we vary each attribute from 0-2 along the rows and columns respectively. The intersection between each attribute value is then the learnt utterance of that attribute combination. For example, "aa" in Case II is the utterance for the attribute combination (0,0).

In Case I of Figure 5.1, the learnt communication protocol is completely random. As we would expect, all three compositionality metrics for this case are very low. This is because a random communication protocol is very unlikely to be compositional. In Case II, we present a perfect communication protocol. Here, the first character of the utterance corresponds to the first column attribute, and the second character corresponds to the row attribute. As expected, the compositionality for Case II is perfect for both *topsim* and *posdis*. Surprisingly, the value of *bosdis* is 0.000. This is likely due to the way *bosdis* is related to the number of symbol occurrences. In Case III, we show a simple example where both characters are related to the same column attribute. Here we find both *posdis* and *bosdis* reach a value of 1.000, while *topsim* reaches 0.577. This is interesting, as both *posdis* and *bosdis* indicates a perfectly compositional message while it actually only has limited compositionality (also indicated by the *topsim*).

The maximum accuracy of each example is proportional to the number of unique utterances per attribute combination. Therefore, the maximum accuracies in Figure 5.1

are 67%, 100% and 33% for Case I, Case II and Case III respectively. These maximum accuracies and the metrics of Case III (in Figure 5.1) suggest that there is a lower bound of accuracy for which these metrics hold.

Overall, the most consistent metric is topographic similarity or *topsim*. The most inconsistent metric is *bosdis*, which agrees with recent work [6]. For this reason, we can generally ignore the *bosdis* metric in our experimental results.

5.2.3. Redundancy

Human language is not always entirely compositional. While we do have a very high level of compositionality, we often use superfluous words to denote the same meaning depending on context. For example, in English, we use different words for denoting gender such as *he* vs *him*, when the meaning remains relatively the same. Or in Afrikaans, we use the word "nie" twice in one sentence to indicate a negative. It is possible that these are used to aid in understanding and improve clarity when communicating. We refer to this behaviour as redundancy.

To measure redundancy we perform quantitative comparisons of the symbols used in the emergent protocol. We compare two metrics, the number of unique symbols and the number of repeats. If a communication protocol is redundant, it will likely use fewer unique symbols per utterance, decreasing the likelihood that an important symbol is lost during transmission. Similarly, it will likely repeat phonetic unigrams, bigrams and trigrams. A trigram is a contiguous sequence of three units, a bigram a sequence of two units, and a unigram is a single unit. Therefore, for a redundant communication protocol, we would expect fewer unique symbols and more repeated symbols per utterance.

5.3. Chapter summary

In this chapter we first discussed the details on implementing the approach proposed in Section 4.2. This included an overview of the five main system variants, including a discrete baseline. We then outlined the metrics that will be used to evaluate the emergent communication protocols.

Chapter 6

Experiments

In this study, we propose two contributions that expand on previous work. The first approach compares REINFORCE to DQN (Section 6.1). We do this in contrast to previous language emergence work, which solely focuses on REINFORCE optimisation. Next, we focus on the core contribution of this study – learning to speak and hear through multi-agent communication over a continuous acoustic channel (Section 6.2). We then systematically analyse the properties of the emergent languages and compare them for each setup (Section 6.3). Finally, we include two additional experiments performed earlier in this work – language grounding and real-world performance experiments.

6.1. REINFORCE vs DQN

We first explore DQN as an alternative optimisation technique to REINFORCE for language emergence. As mentioned in Section 4.3.1, REINFORCE is said to have relatively high variance and often struggles to consistently converge [23]. We investigate this claim and compare it to the alternative, DQN.

Experimental setup. For this experiment, we use an identical setup to Chaaubouni et. al. [4], which is essentially a purely discrete version of our proposed acoustic environment. In these experiments, we only switch out the optimisation method to DQN from REINFORCE (Section 4.3.1). We only consider discrete communication here, as we want an environment where we can compare DQN directly with previous work. The agents are trained for 50 epochs and must transmit a total of 625 attribute combinations ($M = 5$, $N = 4$). Five experiments are performed. We compare DQN and REINFORCE with the metrics of Chaaubouni et. al. [4]. This includes accuracy per-attribute and three measures of compositionality (Section 5.2). For the first compositionality metric, we study the structural similarity of the emergent communication in terms of Spearman ρ correlation between the input and message space, a metric known as topographic similarity or *topsim* [2, 66]. This metric gives an indication of how compositional a learnt communication protocol is by finding the correlation between which message symbols are reused for the same input combinations. For example, if we had to communicate the attributes "red square" and "red triangle" and used the sequence "aaa" to do so in both cases, we would get

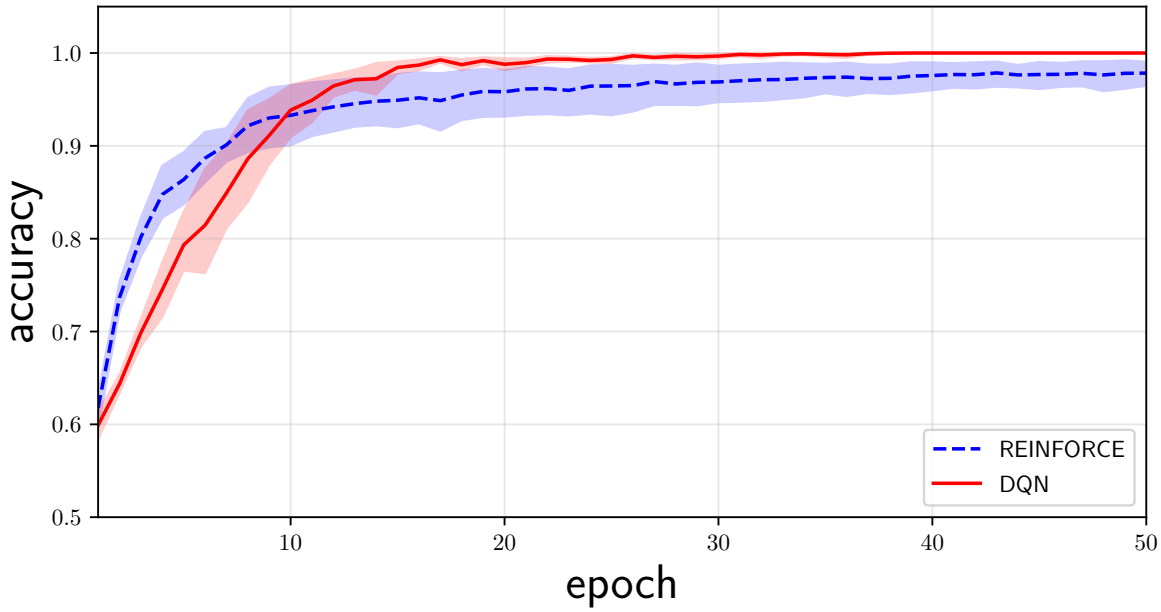


Figure 6.1: Mean accuracy per attribute of REINFORCE and DQN agents throughout training. The 95% confidence bounds are shown over five runs.

a high topsim. If, however, we used "aaa" in the first case and "eee" in the second to indicate the colour, we would get a low topsim. We also include positional disentanglement (*posdis*) and bag-of-symbols disentanglement (*bosdis*) as secondary compositionality metrics (Section 5.2). For these metrics, higher is better and the maximum value is 1.

Experimental results. The accuracy per attribute throughout training is presented in Figure 6.1. It is immediately clear that DQN has a much lower variance than REINFORCE, indicated by the narrower confidence bounds. In fact, DQN consistently converges on 100% accuracy for all five runs. This is not the case for REINFORCE, where the accuracy converged on $97.8 \pm 1.5\%$ (mean±std). Interestingly REINFORCE seemed to learn faster initially with slightly lower variance, and is the first to reach an accuracy of 90% after approximately 6 epochs, which DQN only reaches after approximately 9 epochs.

The compositionality metrics for both optimisation techniques is shown in Table 6.1. As a reminder, higher is better with a maximum value of 1 for all compositionality metrics. We find that DQN outperforms REINFORCE for both *topsim* and *posdis*. For *bosdis*, REINFORCE does slightly better. As mentioned previously in Section 5.2.2, *bosdis* is quite inconsistent and can likely be ignored for our setup. The metric we are most concerned with is *topsim*. We measured the *topsim* of the REINFORCE communication protocols to be 0.387 ± 0.012 (mean±std), which aligns with the results of Chaabouni et. al. [4]. DQN reaches almost double the *topsim* performance of REINFORCE reaching 0.691 ± 0.083. This is much higher than the topographic similarity of previous work [2, 4, 28]. In the rest of this study we will focus on *topsim* as the compositionality metric due to its consistency.

Overall, DQN seems to be better suited as a tool to optimise agents when studying language emergence.

Table 6.1: Compositionality metrics of the learnt message protocol for both REINFORCE and DQN. The metrics are reported as mean ± std over five runs.

Optimiser	*topsim*	*posdis*	*bosdis*
REINFORCE	0.387 ± 0.012	0.095 ± 0.048	0.175 ± 0.035
DQN	0.691 ± 0.083	0.386 ± 0.174	0.083 ± 0.014

6.2. Learning to speak and hear

In this section, we investigate if our agents can learn to speak and hear within a lossy communication channel. As a baseline solution, we train the Speaker and Listener models of Chaaubouni et. al. [4] to solve the discrete communication task (identical to the setup in Section 6.1). During training, the discrete units generated by the Speaker are directly consumed by the Listener. During the evaluation, we follow the same procedure as our acoustic implementation (Section 4.2). We first map the discrete units of the discrete Speaker to the same phone set $\mathcal{P}$ used by our acoustic Speaker, thereafter utilising an identical synthesis and channel noise setup. A pre-trained phone recogniser, based on Deep Speed 2 [38] (Section 4.2.2), is then used to interpret the waveform as discrete units which are fed to the discrete Listener. This baseline thus represents the setting where discrete-only agents are trained and then employed in an acoustic environment using fixed symbol-to-waveform and waveform-to-symbol models.

In the experiments of this section, we compare five main system variants:

1. DISCRETE: Our baseline follows the discrete implementation described above, where we first train the agent to solve the discrete communication task. The agent is then evaluated in the acoustic communication channel of Section 4.2.3. To do this, the discrete units are mapped to phones, which are synthesised and passed over the channel. A pre-trained phone recogniser (Section 4.2.2) then converts back to discrete units for the Listener.

2. ACOUSTIC E2E: This approach combines our acoustic Speaker model (Section 4.2.1) with the end-to-end (E2E) Listener model (Section 4.2.2). As a reminder, the end-to-end Listener converts directly from a mel-spectrogram to the predicted attributes $\hat{\boldsymbol{s}}$.

3. ACOUSTIC* E2E: We consider a variant of the above model where the agents are pre-trained; in this case, the Speaker agent is initialised with the weights of the DISCRETE Speaker. We denote system variants that use pre-training with a '*'.

4. ACOUSTIC + PHONEREC: Our fourth approach combines the same acoustic Speaker model with the phone recogniser (PHONEREC) Listener (Section 4.2.2). The phone recogniser used here is identical to that of the DISCRETE approach. We use this

variant as a direct comparison to the DISCRETE model, as neither are able to update the weights of the static phone recogniser.

5. ACOUSTIC* + PHONEREC: The final approach uses pre-training with ACOUSTIC + PHONEREC. In this approach, both the Speaker and Listener are initialised with the weights of the DISCRETE agents.

6.2.1. Approaches in noisy environments

We start by comparing different system variants in noisy environments, with the goal of seeing how discrete-only training compares with training an acoustic agent. As a reminder, the discrete agents are trained exclusively for symbolic communication, while the acoustic agents are trained in an environment with channel noise.

Experimental setup. In these experiments, all agents use a fixed sequence length of $L = 5$. The acoustic models are trained with the full lossy communication channel described in Section 4.2.3, including background noise samples from Clotho that are scaled to correspond to a 10 dB signal-to-noise-ratio (SNR). We study the results when training with both REINFORCE (used in previous work) and our DQN optimisation approach (Section 4.3.1). All models are evaluated in both the training and evaluation environments (Section 4.2.3). We study the accuracy per attribute: the average number of attributes correctly communicated out of all attribute combinations (Section 5.2). We also study the *topsim* as a quantitative measure of compositionality. For this metric, higher is better and the maximum value is 1.

Experimental results. The results for models optimised with REINFORCE and DQN are presented in Table 6.2 and Table 6.3 respectively. All the models optimised with DQN outperform their REINFORCE counterparts, with particularly large improvements in compositionality (as measured by the *topsim* metric). A detailed analysis of REINFORCE versus DQN in the discrete setting is given in Section 6.1. When looking only at the DQN results, we see that the acoustic models (rows 2 to 5) are consistently able to adapt to the noisy environment and outperform the DISCRETE model (row 1). The end-to-end models (with and without pre-training, rows 2 and 3) outperform the models using the phone recogniser (rows 4 and 5). This is to be expected as the Listener can directly adapt to the noisy environment by updating the weights of the "hearing" portion of the model – this is not possible when using a fixed phone recogniser. Nevertheless, by comparing the system in row 1 with those in rows 4 and 5, we see that the acoustic models that use the same phone recogniser as the DISCRETE model are able to adapt their communication protocol in order to mitigate the effects of the channel noise. While the DQN models (Table 6.3) are not as reliant on discrete pre-training as the REINFORCE models (Table 6.2), we still find it to ease the learning process and improve results – illustrated by the relative performance difference between rows 2 and 3 for REINFORCE vs DQN. The best-performing model

Table 6.2: Per attribute accuracy and topographic similarity of various models trained with REINFORCE in both the training and evaluation environments. ACOUSTIC* uses the discrete baseline for pre-training. Each model was trained five times. We observe a maximum result standard deviation of 0.02 over all seeds.

		REINFORCE		
	Model	*train rooms*	*eval. rooms*	*topsim*
1	DISCRETE	0.621	0.612	0.387
2	ACOUSTIC E2E	0.611	0.566	0.275
3	ACOUSTIC* E2E	0.973	0.950	0.373
4	ACOUSTIC + PHONEREC	0.539	0.535	0.358
5	ACOUSTIC* + PHONEREC	0.609	0.591	0.387

Table 6.3: Per-attribute accuracy and topographic similarity of various models trained with DQN in both the training and evaluation environments. ACOUSTIC* uses the discrete baseline for pre-training. Each model was trained five times. We observe a maximum result standard deviation of 0.02 over all seeds.

		DQN		
	Model	*train rooms*	*eval. rooms*	*topsim*
1	DISCRETE	0.649	0.649	0.691
2	ACOUSTIC E2E	0.956	0.789	0.519
3	ACOUSTIC* E2E	**0.986**	**0.958**	0.707
4	ACOUSTIC + PHONEREC	0.682	0.674	**0.747**
5	ACOUSTIC* + PHONEREC	0.726	0.710	0.745

overall is the ACOUSTIC* E2E model, where the Speaker is pre-trained (row 3 of Table 6.3). While pre-trained models already have an established communication protocol, the models without pre-training have to learn to deal with the communication loss at the same time, increasing the difficulty of learning. This is much more apparent in the REINFORCE models. In terms of compositionality, the acoustic DQN models tend to have slightly higher topographic similarity than the DISCRETE model, reaching a similar performance to that achieved in the discrete-only study of [6].

In the experiments that follow, we will focus on a subset of the different DQN-optimised models from Table 6.3. Specifically, we will compare the discrete model (row 1) to both the pre-trained ACOUSTIC* E2E (row 3) and ACOUSTIC* + PHONEREC (row 5) approaches. We choose the ACOUSTIC* + PHONEREC models as a direct comparison to the discrete models, as both are restricted by the performance of the static phone recogniser. We also chose the best performing model overall (ACOUSTIC* E2E) to see how the learnt communication protocol varies based on the Listener dynamics.

6.3. Emergent language characteristics

Previously in Section 6.2, we have shown that we are able to observe emergent language between agents communicating through a continuous acoustic channel. We also found that our acoustic models outperformed the discrete baseline in the acoustic communication task. To better understand these differences in performance, we now look towards investigating the emergent communication protocols directly. We first look at the effects of increased noise and sequence length and then analyse the compositionality and redundancy of the emergent language. We then perform experiments on the choice of vocabulary.

6.3.1. Increasing noise and sequence length

To better understand the difference in emergent language between discrete and acoustic communication, we evaluate the DISCRETE and ACOUSTIC* + PHONEREC models with various configurations of the lossy communication channel. While ACOUSTIC* + PHONEREC does not perform as well as ACOUSTIC* E2E, we choose to focus on ACOUSTIC* + PHONEREC in this set of experiments as it allows for a fairer comparison to the DISCRETE model. Both these models are restricted by the imperfect phone recogniser. The ACOUSTIC* + PHONEREC model must adapt without updating the weights of the listening component. Therefore, it must develop a communication strategy to compensate for transmission errors over the noisy channel and static phone recogniser. These properties allow us to better see the differences in the communication protocols resulting from discrete vs acoustic training.

Experimental setup. Concretely, we consider performance in settings with a lossless communication channel (*no room*), a set of rooms with no background noise, and the same set of rooms with 10 dB SNR background noise sampled from Clotho. Three room setups are considered: an evaluation of the rooms seen during training, an evaluation of the unseen *meeting* room, and an evaluation of the unseen *stairway*. The models are trained and evaluated with a maximum phone length L of both 5 and 8 in order to see the effect of varying sequence lengths.

Experimental results. The results are shown in Table 6.4 and Table 6.5 for no background noise and 10 dB SNR background noise respectively. We see that for a maximum phone length $L = 5$, the ACOUSTIC* + PHONEREC models perform marginally better than the DISCRETE models in all the cases (with and without noise), as also was the case in Table 6.3. Somewhat surprisingly, when background noise is present (Table 6.5) and the DISCRETE model is allowed to use $L = 8$ symbols instead of $L = 5$, the performance drops. This is likely due to an increased probability that one or more phones are lost over the communication channel in longer sequences. Despite this, due to the increased channel capacity when $L = 8$, the ACOUSTIC* + PHONEREC model is able to counteract the

Table 6.4: Accuracy of the discrete and acoustic agents evaluated in various acoustic environments and no background noise. ACOUSTIC* + PHONEREC refers to the pre-trained acoustic + phone recogniser model. *Meeting* and *stairway* refer to the two evaluation rooms, where the *stairway* has more echos than the *meeting* room. Each model was trained five times. We observe a maximum standard deviation of 0.03 over all runs for all results.

			No background noise		
Model	L	*no room*	*training rooms*	*meeting*	*stairway*
DISCRETE	5	0.992	0.780	0.794	0.594
ACOUSTIC* + PHONEREC	5	0.945	0.808	0.798	0.634
DISCRETE	8	0.998	0.728	0.801	0.546
ACOUSTIC* + PHONEREC	8	0.950	0.806	0.826	0.635

Table 6.5: Accuracy of the discrete and acoustic agents evaluated in various acoustic environments and 10 dB SNR background noise. ACOUSTIC* + PHONEREC refers to the pre-trained acoustic + phone recogniser model. *Meeting* and *stairway* refer to the two evaluation rooms, where the *stairway* has more echos than the *meeting* room. Each model was trained five times. We observe a maximum standard deviation of 0.03 over all runs for all results.

		10 dB SNR background noise		
Model	L	*training rooms*	*meeting*	*stairway*
DISCRETE	5	0.651	0.703	0.604
ACOUSTIC* + PHONEREC	5	0.726	0.751	0.664
DISCRETE	8	0.564	0.666	0.544
ACOUSTIC* + PHONEREC	8	0.677	0.731	0.640

information loss and retain performance comparable to its $L = 5$ counterpart. This causes the ACOUSTIC* + PHONEREC model to do significantly better for longer sequences, with 13.3% better relative performance over the DISCRETE case in the *meeting* and *stairway* rooms with background noise (Table 6.5). We see similar results in Figure 6.2, where we plot evaluation accuracy as a function of L: again we observe the DISCRETE model drops in performance after $L = 4$, while the ACOUSTIC* + PHONEREC plateaus without a performance drop. The performance drop at $L = 2$ is due to the channel capacity falling below the total number of attribute combinations ($|S| < |\mathcal{P}|^L$). This means the agents are unable to express all possible input combinations given the limited channel capacity.

Both the discrete and acoustic models perform best in the *meeting* evaluation room, where the acoustic dynamics are relatively clean. The models perform worst in the *stairway* evaluation room, where there are a lot of surfaces for sound to echo off of. The models also experience a large performance decrease when significant background noise is present, as can be seen by comparing the no background noise results of Table 6.4 to the 10 dB SNR background noise of Table 6.5.

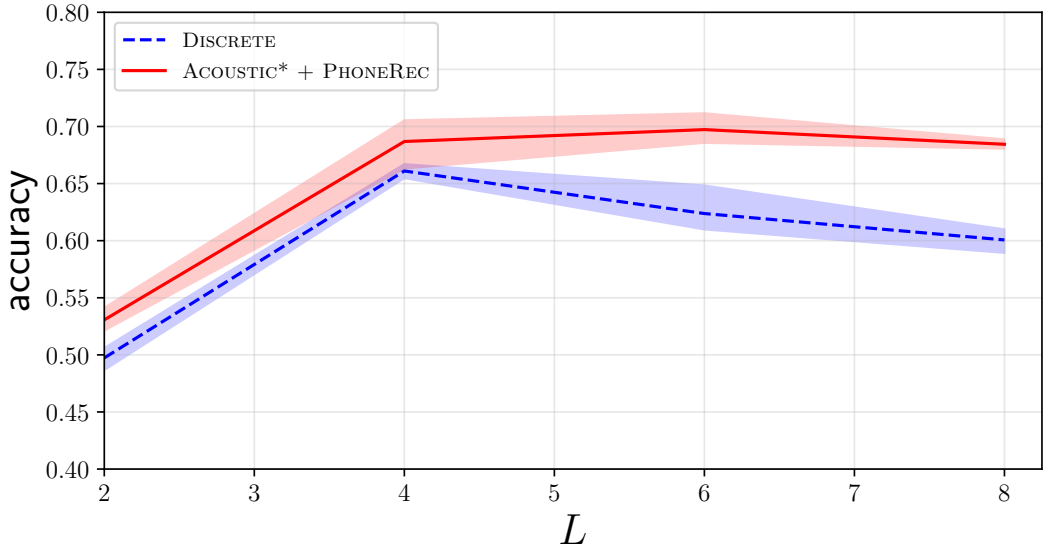


Figure 6.2: Per-attribute accuracy as a function of maximum phone length L. The 95% confidence bounds over five runs are shown.

6.3.2. Emergent compositionality and redundancy

The learnt communication protocols of each agent from the previous section are now investigated qualitatively. The goal here is to take a deeper dive into the emergent communication protocols of each model, in an attempt to understand how and why the acoustic models outperform the discrete ones. To do this, we take samples of the phone sequences produced by the Speaker agent of each model.

Table 6.7 and Table 6.6 respectively show samples of the learnt communication strategy of the Acoustic* + PhoneRec and Discrete case with $L = 8$. For comparison, we also show samples of the Acoustic* E2E Speaker in Table 6.8. Each entry in the tables corresponds to the phone sequence generated by the Speaker for different attribute values. s_1 and s_2 are varied from 0 to 5 across the columns and rows, respectively, while s_3 and s_4 are fixed. Following the example from Section 4.2 where s_1 represents a colour, each column could represent a colour value (e.g. *red*, *green*, *blue*).

By comparing the phones used per utterance in Table 6.6 to Tables 6.7 and 6.8, it is immediately clear that both acoustic Speakers tend to use fewer unique phones per utterance, and also tend to repeat phonetic unigrams, bigrams and trigrams. A trigram is a contiguous sequence of three units, a bigram a sequence of two units, and a unigram a single unit. For instance, as shown in Table 6.7, to communicate this specific combination of s_3 and s_4, the Speaker tends to use the repeated bigram [oi]. The same behaviour is observed in Table 6.8, with both acoustic Speakers repeating this bigram, despite being trained independently and with a completely different Listener setup. This is interesting, as the two acoustic models have no direct motivation to learn similar protocols, other than overcoming the transmission errors of the noisy communication channel.

Table 6.6: Sample of phone sequences produced by the DISCRETE Speaker for $L = 8$. Each entry corresponds to a combination of four attributes: where we vary the first two attributes ($\boldsymbol{s}_1$ and $\boldsymbol{s}_2$), while keeping the last two attributes fixed ($\boldsymbol{s}_3$ and $\boldsymbol{s}_4$).

$\boldsymbol{s}_2$ \ $\boldsymbol{s}_1$	0	1	2	3	4
0	auieueuo	uiieueoo	oiaeueou	aiaeueou	iiaeeoeo
1	aiiuoooi	uiiuoooi	oiiuoooi	aiiuoooi	iiiuoooi
2	uuieueou	uuieoeoi	ouieueou	auieoeoi	iuieueou
3	aiieueoe	eiieueoo	oiieueoo	aiieueoe	iiieueoe
4	aaieeueo	eaieoeoi	oaieeouo	aaieeoui	iaieeouo

Table 6.7: Sample of phone sequences produced by the ACOUSTIC* + PHONEREC Speaker for $L = 8$. Each entry corresponds to a combination of four attributes: $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ varying, while $\boldsymbol{s}_3$ and $\boldsymbol{s}_4$ are fixed. The bigram [**oi**] has been highlighted in bold.

$\boldsymbol{s}_2$ \ $\boldsymbol{s}_1$	0	1	2	3	4
0	auaa**oioi**	eaao**oioi**	oaa**oioi**o	aea**oioi**o	ioa**oioi**i
1	aoa**oioi**o	uoao**oioi**	ooa**oioi**o	aoa**oioi**o	ioa**oioi**o
2	auaao**oi**o	uuaaoo**oi**	uuaaoo**oi**	auaoo**oi**i	iuaao**oi**o
3	eeauio**oi**	eea**oioi**o	oea**oioi**o	aea**oioi**o	iea**oioi**o
4	aaaao**oi**o	eaaoo**oi**i	oaaoo**oi**i	aaaoo**oi**i	iaaao**oi**o

Table 6.8: Sample of phone sequences produced by the ACOUSTIC* E2E Speaker for $L = 8$. Each entry corresponds to a combination of four attributes: $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ varying, while $\boldsymbol{s}_3$ and $\boldsymbol{s}_4$ are fixed. The bigram [**oi**] has been highlighted in bold.

$\boldsymbol{s}_2$ \ $\boldsymbol{s}_1$	0	1	2	3	4
0	auaee**oi**o	eeaoe**oi**e	oeae**oioi**	aeae**oioi**	ieae**oioi**
1	aiau**oioi**	eiao**oioi**	**oi**au**oioi**	aiao**oioi**	iia**oioi**e
2	uuaeeoio	uuae**oioi**	ouaee**oi**o	auae**oioi**	iuaee**oi**o
3	aeaee**oi**o	eeaoe**oi**e	oeaee**oi**o	aeae**oioi**	ieaee**oi**o
4	aaaee**oi**o	eaaoe**oi**e	oaaee**oi**o	aaae**oioi**	iaaee**oi**o

The average number of repeated phones in the ACOUSTIC* + PHONEREC Speaker's utterances is 2.89, while the DISCRETE Speaker has 3.19 repeated phones per utterance. This is an indication that the acoustic Speaker is learning a redundant communication protocol, assisting the coherency of the Listener through repetition. Table 6.9 shows the number of repeated bigrams and trigrams for each model. Both acoustic models tend to repeat bigrams and trigrams, with the ACOUSTIC* E2E model having fewer repeats. This is likely due to repetition not being as necessary when the network weights of the "hearing" portion may be updated. This repetition is not as present in the discrete case, where the

Table 6.9: Number of repeated bigrams and trigrams per utterance for each model ($L = 8$).

Model	*bigrams*	*trigrams*
DISCRETE	1.623	0.277
ACOUSTIC* + PHONEREC	2.680	0.935
ACOUSTIC* E2E	2.073	0.554

communication strategy uses more unique phones per utterance: on average, the DISCRETE Speaker uses 4.074 unique characters per utterance, with the ACOUSTIC* + PHONEREC Speaker using fewer phones at 3.649.

All the acoustic and discrete Speakers exhibit high levels of compositionality. For example, all three cases in Tables 6.6 to 6.8 begin with [a] where $s_1 = 3$ and [i] where $s_1 = 4$. Another example can be seen in that the second phone tends to correspond with s_2 in all cases. These tables qualitatively show the levels of topographic similarity measured in Section 6.2.1 (specifically Table 6.3).

6.3.3. Choice of vocabulary

Another way that the acoustic models are able to improve over the discrete case is in their choice of vocabulary. The distribution over phones for the DISCRETE and ACOUSTIC* + PHONEREC models are shown in Figure 6.3 ($L = 5$). Figure 6.4 shows the phone error rate (PER) per phone of the static phone recogniser (Section 4.2.2) over the noisy communication channel (Section 4.2.3). The discrete model has no knowledge of the acoustic dynamics and therefore defaults to a (roughly) evenly distributed usage of all phones. In the case of the ACOUSTIC* + PHONEREC models, the agent learns to

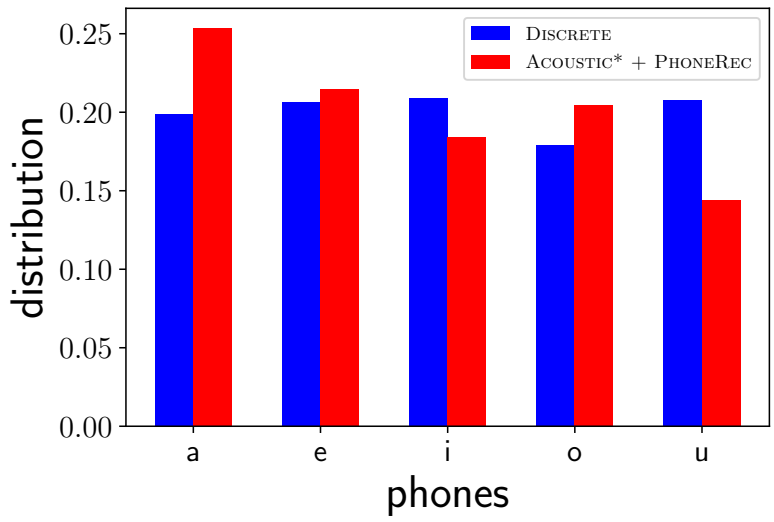


Figure 6.3: Distribution over phones for the learned communication protocols of DISCRETE and ACOUSTIC* + PHONEREC models.

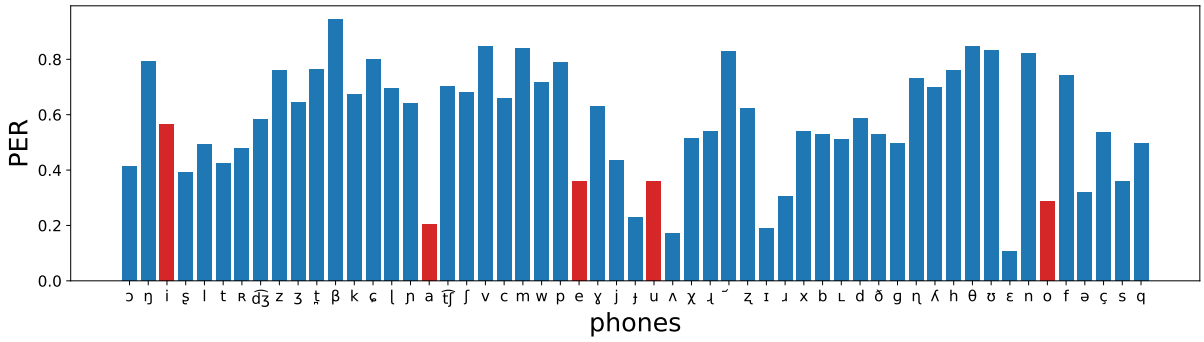


Figure 6.4: PER per phone when using the static phone recogniser over the noisy acoustic channel. The subset of phones used in our experiments is highlighted in red.

use phones that have a lower PER – the phone distribution is inversely proportional to the PER shown in Figure 6.4. We can quantify this relationship using the Spearman ρ correlation between the phone distribution and the phone accuracy ($1 - \text{PER}$). We find the Acoustic* + PhoneRec model has a positive correlation 0.64 ± 0.17 (mean $\pm$ std over five runs), while the discrete case has no significant correlation 0.17 ± 0.59.

6.3.4. Consonant-vowel experiments

As an extension to the experiments on vocabulary choice, we now update the vocabulary to include both consonant and vowel phones. This allows us to study the effects of co-articulation, and whether the Speaker is able to develop a protocol similar to human language. In English, we regularly switch between vowels and consonants in a single word, and rarely repeat two vowels or two consonants after each other.

Experimental setup. Concretely, we use the phone set $\mathcal{P} = \{\text{a}, \text{k}, \text{i}, \text{o}, \text{s}\}$. The Acoustic* E2E model is trained in the same manner as Section 6.3.2, where $L = 8$. Since eSpeak has limited co-articulation, we also train the setup with the Tacotron 2 + HiFi-GAN synthesiser (Section 4.2.1). We then analyse the emergent communication protocol, counting the number of consonant-vowel (CV), vowel-consonant (VC), consonant-consonant (CC), and vowel-vowel (VV) pairs. We compare these numbers to a reference set of 58 110 common English words.

Experimental results. Samples of the eSpeak and Tacotron 2 + HiFi-GAN Speakers are presented in Table 6.10 and Table 6.11 respectively. Similar to Section 6.3.2, we show how the utterances change based on the attribute values of s_1 and s_2. In these samples, we find the Tacotron 2 + HiFi-GAN Speaker (Table 6.10) switches between consonants and vowels slightly more often than the eSpeak Speaker (Table 6.11). Notice how the utterances in both tables look slightly more natural than those of Section 5.2.2. This is because English words tend to alternate between consonants and vowels. Audio samples are available online at kevineloff.github.io/learning-to-speak/.

Table 6.12 shows the number of consonant-vowel (CV), and vowel-consonant (VC)

Table 6.10: Sample of phone sequences produced by the ACOUSTIC* E2E Speaker for $L = 8$, $\mathcal{P} = \{\mathrm{a}, \mathrm{k}, \mathrm{i}, \mathrm{o}, \mathrm{s}\}$, and the Tacotron 2 + HiFi-GAN synthesiser. Each entry corresponds to a combination of four attributes: $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ varying, while $\boldsymbol{s}_3$ and $\boldsymbol{s}_4$ are fixed.

		$\boldsymbol{s}_1$ = 0	1	2	3	4
$\boldsymbol{s}_2$	0	soookiik	siooiiik	oookiiki	aiokikik	ikokiiki
	1	aoookiik	sooiiiii	oooiiiik	aoooikii	iooiiiik
	2	asooikii	ssooiiii	osoiiiik	asooiiik	isooiiii
	3	kkookiik	skoiiiik	okoiiiik	kkoikiik	ikoiiiik
	4	iiookiik	siooiiik	oioikiik	aiookiik	iiooiiik

Table 6.11: Sample of phone sequences produced by the ACOUSTIC* E2E Speaker for $L = 8$, $\mathcal{P} = \{\mathrm{a}, \mathrm{k}, \mathrm{i}, \mathrm{o}, \mathrm{s}\}$, and the eSpeak synthesiser. Each entry corresponds to a combination of four attributes: $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ varying, while $\boldsymbol{s}_3$ and $\boldsymbol{s}_4$ are fixed.

		$\boldsymbol{s}_1$ = 0	1	2	3	4
$\boldsymbol{s}_2$	0	aiokssos	kiokosos	oiokssos	aiokssos	iiokssos
	1	iiosooss	siooooss	oiosooss	aiooooss	iiooooss
	2	asooooss	ksooooos	osooooss	asooooss	isooooss
	3	aiokssoo	kiokssos	oiokssoo	aiokssos	iiokssoo
	4	aaokooos	kaokooos	oaokooos	aaokooos	iaokooos

pairs for both synthesisers. We also show the number of consonant and vowel phone pairs for a dataset of 58 110 common English words. This reinforces this idea of English often alternates between consonants and vowels, 61% (CV + VC) of all pairs switch between consonants and vowels. For English, there are also a similar amount of CV and VC pairs. We see similar behaviour in Tacotron 2 + HiFi-GAN, where there are a similar amount of CV and VC pairs. Overall, Tacotron 2 + HiFi-GAN and eSpeak have fewer pairs alternating between consonants and vowels, (43% and 39% respectively). In the case of eSpeak, the number of vowel-consonant pairs is much higher than consonant-vowel pairs.

Table 6.12 also shows the number of consonant-consonant (CC) and vowel-vowel (VV) pairs. We find that English barely ever has vowel pairs, and often has consonant pairs. Tacotron 2 + HiFi-GAN has the opposite behaviour where there are more vowel pairs than consonant pairs. Qualitatively when listening to the audio, the samples of Tacotron 2 + HiFi-GAN tend to collapse repeated vowels into a single phonetic unit. This behaviour does not occur in eSpeak with its limited co-articulation. Overall, the CC and VV columns of Table 6.12 are biased towards the phonetic vocabulary used. English has 26 alphabetic characters, 5 of which are vowels. The phone set $\mathcal{P}$ used in this experiment has 3 vowels and 2 consonants. Thus, we would expect a bias towards vowel pairs being much more frequent than consonant pairs.

Table 6.12: Normalised number of consonant and vowel phone pairs per utterance. The baseline uses 58 110 common English words.

Synthesiser	*CV*	*VC*	*CC*	*VV*
Tacotron 2 + HiFi-GAN	0.22	0.21	0.19	0.38
eSpeak	0.16	0.23	0.39	0.23
English	0.30	0.31	0.36	0.03

Table 6.13: Generalisation accuracy on the uniform holdout set for each model. Each model was trained five times. We observe a maximum standard deviation of 0.03 over all seeds.

Model	*training set*	*holdout set*
DISCRETE	0.649	0.630
ACOUSTIC* E2E	0.958	0.945
ACOUSTIC* + PHONEREC	0.710	0.686

6.3.5. Generalisation

Generalisation is another important property of language. We want our agents to be able to generalise to novel input combinations, as described in Section 5.2.1. To reiterate with an example, two agents could learn to communicate *red square* and *blue triangle* during training. The agents generalise if they are able to successfully communicate *blue square* and *red triangle* as unseen validation combinations.

Experimental setup. In this setup we train the DISCRETE, ACOUSTIC* E2E, and ACOUSTIC* + PHONEREC models as described in Section 6.2 ($L = 5$). Before training the agents, we uniformly sample 10% (63 total combinations) of the attribute combinations to be used as a holdout set. This holdout set is then removed from the training set and only used during evaluation. All results are presented using the evaluation setup of Section 6.2.

Experimental results. The evaluation results on the uniform holdout set are presented in Table 6.13. As a comparison, we also include the evaluation accuracy of the training set. Overall, we find all models generalise very well to the uniform holdout set, with a minimal performance drop compared to the training set. Interestingly, we find no clear difference between each model. This indicates that generalisation is not necessarily influenced by whether the learnt communication is continuous or discrete.

6.4. Additional experiments

In this section we include some of the smaller experiments performed earlier in this work. By smaller, we mean we down-scaled the number of input attribute combinations with two setups: single-attribute ($M = 1$, $N = 4$) and two-attribute ($M = 2$, $N = 4$). In these experiments, we train the ACOUSTIC* E2E model with a slightly simpler noise scheme.

The channel applies Gaussian white noise scaled to a target SNR. We also include other effects such as time and frequency masking, Librosa [60] warping and a band-pass filter. We include two main experiments in this section: language grounding and real-world experiments.

6.4.1. Language grounding

Although the Speaker uses an English phone set, up to this point there has been no reason for the agents to actually learn to use English words to convey the concepts. In this subsection, either the Speaker or Listener is predisposed to speak or hear English words, and the other agent needs to act accordingly. One scientific motivation for this setting is that it can be used to study how an infant learns language from a caregiver [15]. To study this computationally, several studies have looked at cognitive models of early vocal development through infant-caregiver interaction; Asada [53] provides a comprehensive review. Most of these studies, however, considered the problem of learning to vocalise [16, 52, 67], which limits the types of interactions and environmental rewards that can be incorporated into the model. We instead simplify the vocalisation process by using an existing synthesiser, but this allows us to use modern MARL techniques to study continuous signalling.

We first give the Listener agent the infant role, and the Speaker will be the caregiver. This mimics the setting where an infant learns to identify words spoken by a caregiver. Later, we reverse the roles, having the Speaker agent assume the infant role. This represents an infant learning to speak their first words and their caregiver responds to recognised words. Since here one agent (the caregiver) has an explicit notion of the meaning of a word, this process can be described as "grounding" from the other agent's perspective (the infant).

Experimental setup. We first consider a setting where we have a single set of 4 attributes $\mathcal{S} = \{up, down, left, right\}$. Here the agents will be required to use actual English words to convey these concepts. In the setting where the Listener acts as an infant, the caregiver Speaker agent speaks English words; the Speaker consists simply of a dictionary lookup for the pronunciation of the word, which is then generated by eSpeak. In the setting where the Speaker takes on the role of the infant, the Listener is now a static entity that can recognise English words; we make use of a dynamic time-warping (DTW) system that matches the incoming waveform to a set of reference words and selects the closest one as its output label. 50 reference words are generated by eSpeak. In these experiments, we use the full eSpeak English phone set of 164 phones. This means the action-space of the Speaker agent is very large ($|\mathcal{P}|^L$), and would be near impossible to explore entirely. Therefore, we provide guidance: with probability $\epsilon = 0.05$, choose the correct ground truth phone sequence for $\boldsymbol{s}$. We also consider the two-attribute case where either the Speaker or Listener is tasked to speak two English words at a time; DTW is too

computationally expensive for the static Listener in this case, so here we first train the Listener in the infant role and then fix it as the caregiver when training the Speaker.

Experimental results: grounding the Listener. Here the Listener is trained while the Speaker is a fixed caregiver. The Listener agent reached a mean evaluation reward of 1.0, indicating the agent learnt to correctly classify all 4 target words 100% of the time (Figure 6.5). The Listener agent was also tested with a vocabulary size of 50, consisting of the 50 most common English words including the original *up*, *down*, *left*, and *right*. With this setup, the Listener still reached a mean accuracy of 0.934.

Experimental results: grounding the Speaker. We now ground the Speaker agent by swapping its role with that of the infant. The Speaker agent reaches a mean accuracy of 0.983 (Figure 6.5) over five runs, indicating it is generally able to articulate all of the 4 target words. Table 6.14 gives samples of one of the experiment runs and compares them to the eSpeak ground truth phonetic descriptions.

Although appearing very different to the ground truth, the audio generated by eSpeak of the phone sequences is qualitatively similar. The reader can confirm this for themselves by listening to the generated samples. The audio samples are available online (kevineloff.github.io/learning-to-speak/). As seen in Figure 6.5, we find the Speaker struggles to converge in comparison to the Listener. This is likely owed to the difficulty of exploration for the Speaker agent. It is a lot less likely for the Speaker to stumble across the correct target utterance.

Experimental results: grounding generalisation. Analogous to Section 6.3.5, we perform generalisation experiments. We now have infant and caregiver agents in a setting with two attributes ($M = 2$, $N = 4$), specifically $\boldsymbol{s}_1 \in \{up, down, left, right\}$ and $\boldsymbol{s}_2 \in \{fast, medium, regular, slow\}$. For this experiment, we increase $L = 12$, to

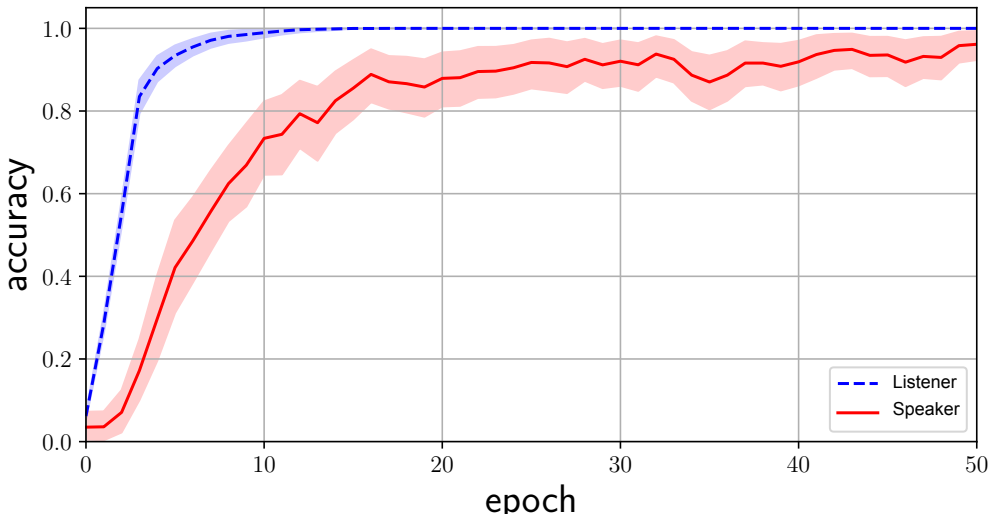


Figure 6.5: Mean evaluation accuracy of Speaker and Listener. The 95% confidence bounds over five runs are shown.

Table 6.14: Table of the target word, ground truth phonetic description, and trained Speaker agent's predicted phonetic description.

Target word	*Ground truth*	*Predicted phones*
up	ʌp	ʌvb
down	daʊn	daʊ
left	lɛft	lɛ
right	ɹaɪt	ɹaɪʃjn

accommodate for the longer ground truth phonetic descriptions. Four combinations are unseen during training (the holdout set): *up-slow*, *down-regular*, *left-medium*, and *right-fast*. Again we consider both role combinations of infant and caregiver. Figure 6.6 shows the results when training a two-word Listener agent. The agent reaches a mean accuracy of 1.0 for the training set and 0.952 for the holdout set. This indicates that the Listener agent learns near-optimal generalisation. As mentioned above, for the case where the Speaker is the infant, the DTW-based fixed Listener was found to be impractical. Thus, we use a static Listener agent pre-trained to classify 50 concepts for each $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$. This totals 2500 unique input combinations. The results of the two-word Speaker agent are shown in Figure 6.7. The grounded Speaker agent does not perform as well as the Listener agent, reaching a mean accuracy of 0.821 for the training set combinations and 0.539 for the holdout set. The grounded Speaker also has a much higher variance (seen by the wide bands in Figure 6.7).

We have replicated the experiments in this subsection using the Afrikaans version of eSpeak, reaching a similar performance to English. The Afrikaans two-word Listener reached 1.0 on the training set and 0.82 on the holdout set. The Afrikaans two-word Speaker performed slightly worse with 0.72 on the training set and 0.41 on the holdout set. This shows our results are generally not language specific.

6.4.2. Real-world performance

An important consideration of this research direction is the application of agents in real-world scenarios. As humans, we are exposed to noisy environments, yet we learn to sufficiently filter signals from noise. From a young age, human infants learn to communicate with their caregivers in the presence of abundant noise. We also want to illustrate concretely the benefit of being able to specify an acoustic channel during training.

Experimental setup. For simplicity, we focus on the grounded Listener agent. Using a similar setup to the grounded listener agent in Section 6.4.1, we generate a set of audio samples by the static speaker agent. These audio samples are played through a speaker in an environment with considerable background noise (a postgraduate student lab). The trained model is then evaluated on these recorded audio samples. We augment the training

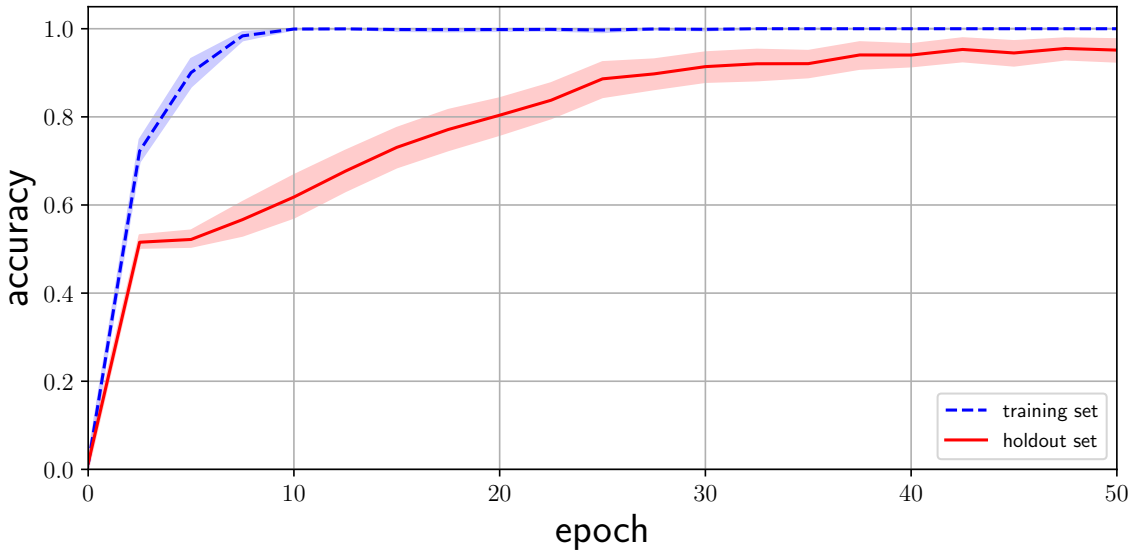


Figure 6.6: Mean evaluation accuracy of grounded two-word Listener on the training and holdout sets. The 95% confidence bounds over five runs are shown.

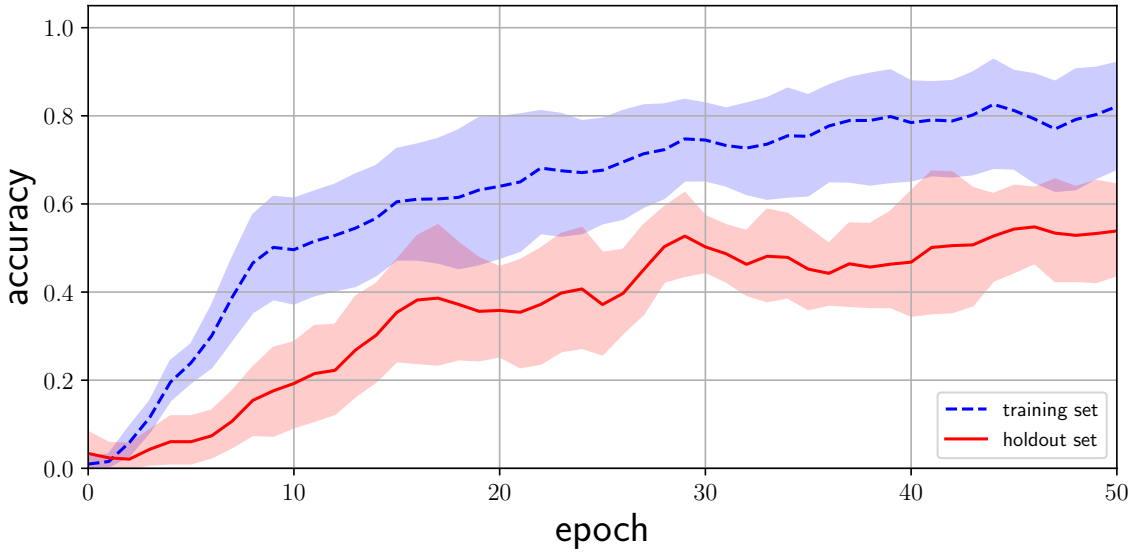


Figure 6.7: Mean evaluation accuracy of grounded two-word Speaker on the training and holdout sets. The 95% confidence bounds over five runs are shown.

of our listener agent by implementing a band-pass filter (BPF) in the lossy communication channel. The filter's lower and upper critical frequencies are uniformly distributed over 1 to 250 Hz and 6500 to 7500 Hz respectively. These values were chosen based on the characteristics of our physical communication channel setup.

Experimental results. Figure 6.8 shows the mel-spectrograms of the raw audio samples generated by the static speaker agent, the noisy recordings of these audio samples, and the simulated channel function. We see similar distortions in the recorded and simulated mel-spectrograms, although the recorded sample has much higher background noise at lower frequencies.

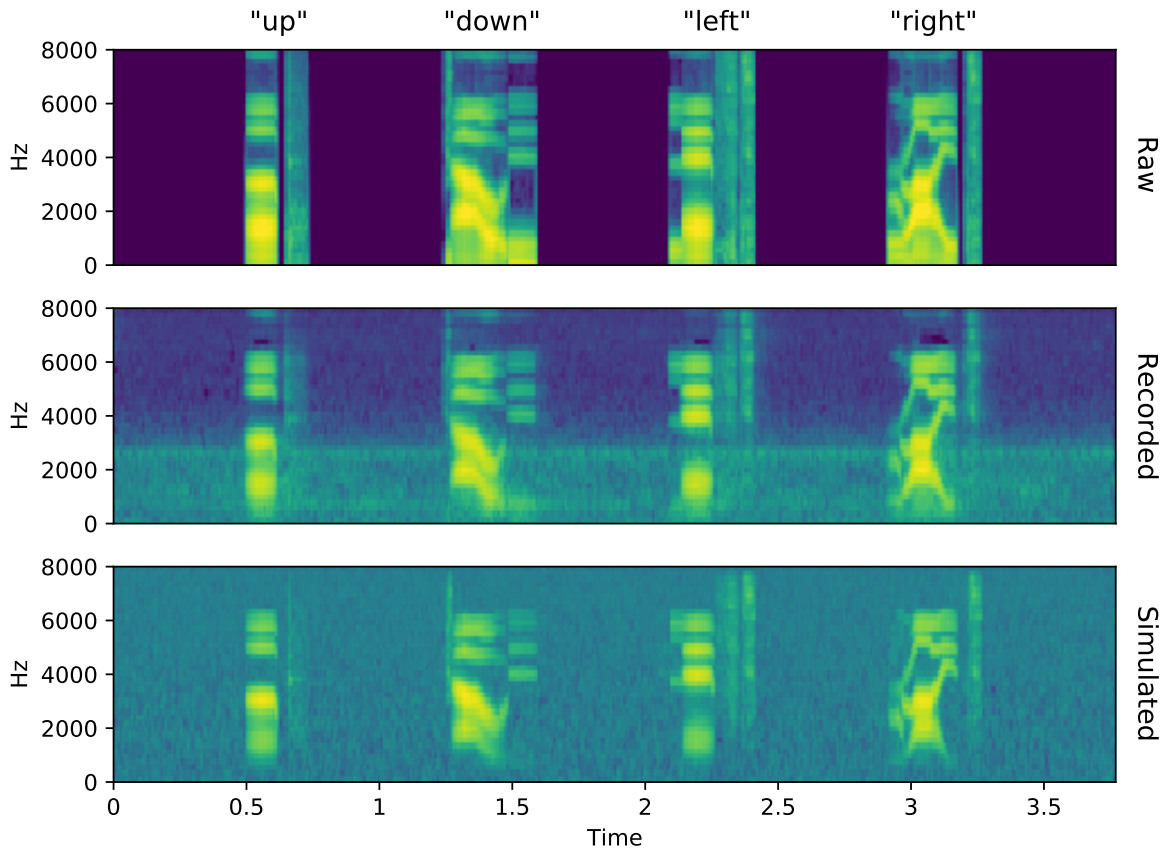


Figure 6.8: Mel-spectrogram samples of the static speaker agent saying "up down left right". The raw noiseless output is shown, along with the recorded and simulated mel-spectrograms excluding time and frequency masking.

Table 6.15: Mean reward of the listener agent evaluated on recorded audio. The 95% confidence bounds are shown.

Channel				
Noise	Masking	Warping	BPF	Reward
-				0.679 (±0.05)
30 dB				0.842 (±0.03)
30 dB	✓			0.892 (±0.04)
30 dB	✓	✓		0.913 (±0.03)
30 dB	✓	✓	✓	0.975 (±0.01)

The performance of the listener agent on the noisy audio recording is shown in Table 6.15. These results show the importance of a noisy communication channel to generalising to a real-world scenario. The agent generalises much better when all channel effects are used during training, reaching a perfect accuracy on most runs.

Simulating agents with a continuous communication channel will prove useful in future work on human-robot interaction [68, 69]. We may be able to train agents to communicate in a simulated environment that translates well to human interaction in a real-world setting. Future work may consider having agents be trained with physical noisy channels, allowing them to gain real-world experience.

6.5. Chapter summary

In this chapter, we covered experiments related to the two core contributions of this study. Firstly, we compared REINFORCE (used in prior work) to DQN and proved that DQN is much better suited for language emergence studies due to the lower learning rate and improved convergence. Secondly, we showed that agents are able to learn to speak and hear when communicating over a lossy acoustic channel. Thereafter, we performed experiments analysing the properties of the emergent languages developed by our agents and compared them to the discrete setup. Finally, we showcased some additional experiments related to language grounding and real-world performance.

Chapter 7

Summary and Conclusion

This thesis has laid the foundation for investigating whether we can observe emergent language between agents using a continuous acoustic communication channel trained through RL. Our acoustic environment allows us to see how the channel conditions affect the language that emerges. In this environment, we observe that the acoustic Speaker learns redundancy which improves Listener coherency. We also observe how the Speaker learns to update the emergent communication protocol to minimise errors due to the imperfect Listener. We proposed two variations of the acoustic setup, one with a static phone recogniser and an end-to-end variation. When analysing the emergent communication protocols, we found that agents trained with a communication channel learn similar strategies of repetition, often repeating the same bigrams. Alongside this, the acoustic agents learn to be more compositional than the purely discrete agents; which implicitly compensate for transmission errors over a noisy channel. These are examples of emergent linguistic behaviour that is not modelled in a purely discrete setting. Additionally, in our small-scale experiments, we demonstrated promising results in language grounding (to English) and effective generalisation to real-world performance.

This work expands on [25, 26], where an agent uses spoken communication to convey meaning to a fixed listener (Section 3.2.2). In their setup, the spoken communication is restricted to fixed audio snippets from a discovered dictionary. In contrast, our Speaker agent has the ability to generate unique audio waveforms. On the other hand, our Speaker can only generate sequences based on a fixed phone set (which is then passed over a continuous acoustic channel). This differs from earlier work [16, 52, 53] that considered a Speaker that learns a full articulation model in an effort to come as close as possible to imitating an utterance from a caregiver; this allows a Speaker to generate arbitrary learnt units. We have thus gone further than [25, 26] but not as far as these older studies. Despite these shortcomings, our approach has the benefit that it is formulated in a modern multi-agent RL setting and can be easily extended. Future work can therefore consider whether articulation can be learnt as part of our model – possibly using imitation learning to guide the agent's exploration of the very large action space of articulatory movements. This speaks to the modularity of our setup, where any number of components may be switched out.

We have also shown that DQN is better suited for this sort of referential game

environment than REINFORCE, reaching similar performance to that achieved in the discrete-only study of [6]. We highly recommend other practitioners working on emergent communication to consider DQN as an alternative method for optimisation – perhaps in combination with the models used in [6].

7.1. Environment extensions and future work

Due to the novelty of combining speech processing and RL, there is much to be explored and discovered. In this section, we will discuss some of the environment extensions, future work, and ideas that were developed throughout writing this thesis. In fact, some of the future work mentioned here has been visited briefly, but due to time and scope constraints did not make it into the final thesis. We first discuss extensions to the real communication channel. We then consider variations of the Speaker used in this study, where the Speaker may develop continuous actions. Finally, we discuss an environment extension with multiple agents (>2) and multiple communication rounds.

7.1.1. Real communication channel

As mentioned previously, an important consideration of this research direction is the application of agents in real-world scenarios. In this study, we included a small subset of experiments related to real-world performance in Section 6.4.2. In future, we should consider a larger focus on these experiments. To do this, we should allocate more time to training and evaluating agents trained in real acoustic environments. We could set up multiple Raspberry Pi web servers in multiple acoustic environments, using some environments for training and some for validation and testing. This would be relatively easy to implement due to our modular Raspberry Pi web server approach.

In future, the latest models should be fully evaluated on the real communication channel in various settings. This would validate the efficacy of our simulated realistic communication channel.

7.1.2. Continuous speaker

One of the biggest concerns in our work is the discretisation of the Speaker agent outputs. As humans, it is a more-or-less continuous muscle actuation system that produces speech. As an alternative to producing phones, we could allow the Speaker to produce continuous actions. Continuous control is not new to RL, algorithms such as Deep Deterministic Policy Gradient (DDPG) [70] have proven effective in solving continuous action-space environments.

We propose two approaches: one using continuous control of a simulated vocal tract, and one using direct control of HiFi-GAN [45]. The first approach would involve the use

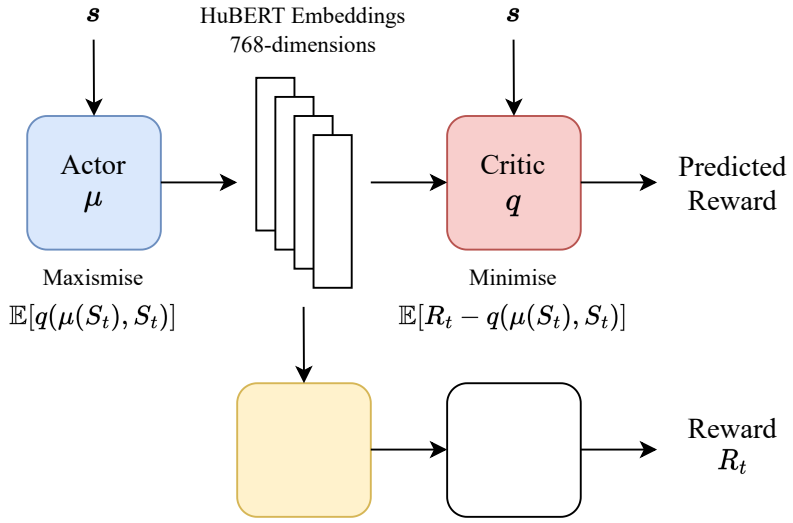


Figure 7.1: DDPG update equations for a continuous Speaker agent.

of a simulated vocal tract, where the agent would be able to control the variables used to produce sound. The second approach is slightly more complex and uses the state-of-the-art in speech synthesis: HiFi-GAN. There is a variant of HiFi-GAN trained on HuBERT [39] codes – a sequence of 768-dimensional speech feature embedding every 20 ms. The goal of the Speaker in this case would be to predict a sequence of continuous HuBERT codes rather than a sequence of discrete phones. Figure 7.1 shows the first attempt at a DDPG actor-critic setup for the continuous Speaker. Initial testing of this approach seemed promising, although difficult to extend to large numbers of input combinations.

7.1.3. Multi-agent multi-round environment

In the experiments carried out in this study, we only considered a single communication round between two agents. This could be expanded to a setup where each agent has both a speaking and listening module, and these composed agents then communicate with one another. Future work could therefore consider multi-round communication games between two or more agents. Such games would extend our work to the full MARL problem, where agents would need to "speak" to and "hear" each other to solve a common task.

A possible realisation of such a game could involve a similar setup to the referential game used in this study. In the current setup, there are two agents with one-way communication. A simple approach to encourage two-way communication is to have both agents be a Speaker and a Listener. This would mean that both agents are provided with a set of attribute values that they require the other agent to predict. As a concrete example, let us refer to the agents as agent A and agent B. At the start of each episode, agent A is given *red triangle*, and agent B is given *blue square*. We refer to this as each agent's internal goal, this goal involves another agent. Now, agent A must predict *blue square* and agent B must predict *red triangle*. To be successful, agent A must communicate with agent B and

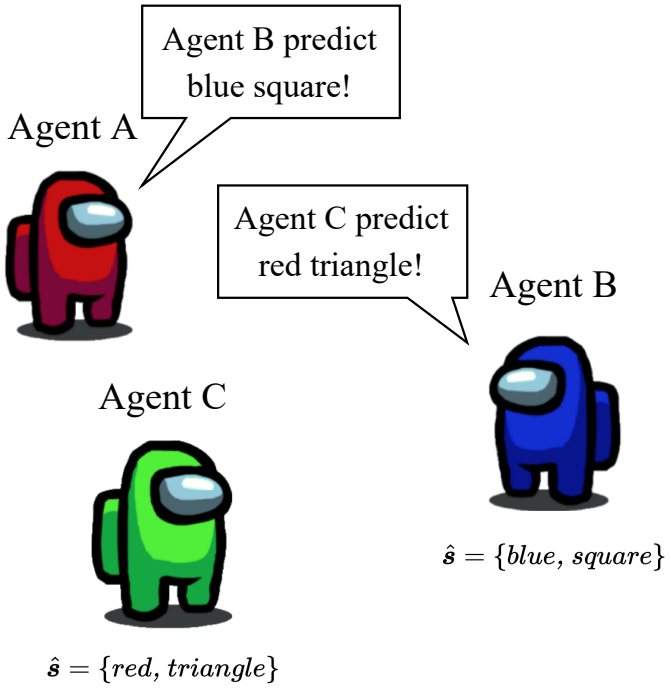


Figure 7.2: Example of a three-agent game where each agent has its own internal goal.

vice-versa.

This can easily be extended to a situation where we have more than two agents. In this case, we would provide each agent with a target agent. At the start of each episode, each agent is given a target agent as well as a set of attribute values that the target agent must output. An example is provided in Figure 7.2, where we have three agents communicating. Agent A wants agent B to predict *blue square* and agent B wants agent C to predict *red triangle*. The agents then have a set of communication rounds, where they explain to each other what their internal goals are. To be successful, agent B outputs *blue square* and agent C predicts *red triangle* once the communication rounds are over.

While the environment itself is simple to extend, this brings some challenges at the same time. In typical multi-agent RL, adding more agents exponentially increases learning difficulty, let alone the need to both do both tasks of sending and receiving messages. Our current optimisation technique would also no longer be sufficient, as there will be multiple communication rounds and the equations would need to be adjusted. We would also now require a replay buffer to store state transitions.

Bibliography

[1] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," in *Proc. AAAI*, 2017.

[2] A. Lazaridou, K. Hermann, K. Tuyls, and S. Clark, "Emergence of linguistic communication from referential games with symbolic and pixel input," *Proc. ICLR*, 2018.

[3] T. Eccles, Y. Bachrach, G. Lever, A. Lazaridou, and T. Graepel, "Biases for emergent communication in multi-agent reinforcement learning," in *Proc. NeurIPS*, 2019.

[4] R. Chaabouni, E. Kharitonov, D. Bouchacourt, E. Dupoux, and M. Baroni, "Compositionality and generalization in emergent languages," in *Proc. ACL*, 2020, pp. 4427–4442.

[5] A. Lazaridou and M. Baroni, "Emergent multi-agent communication in the deep learning era," *CoRR*, 2020.

[6] M. Auersperger and P. Pecina, "Defending compositionality in emergent languages," in *Proc. ACL*, 2022, pp. 285–291.

[7] S. Kirby, "Spontaneous evolution of linguistic structure: an iterated learning model of the emergence of regularity and irregularity," *IEEE Transactions on Evolutionary Computation*, pp. 102–110, 2001.

[8] N. Geffen Lan, E. Chemla, and S. Steinert-Threlkeld, "On the Spontaneous Emergence of Discrete and Compositional Signals," in *Proc. ACL*, 2020, pp. 4794–4800.

[9] J. Andreas, "Good-enough compositional data augmentation," in *Proc. ACL*, 2020.

[10] C. Resnick, A. Gupta, J. Foerster, A. Dai, and K. Cho, "Capacity, bandwidth, and compositionality in emergent language learning," in *Proc. AAMAS*, 2020.

[11] L. Steels, "The synthetic modeling of language origins," *Evolution of Communication*, pp. 1–34, 1997.

[12] I. Kajic, E. Aygün, and D. Precup, "Learning to cooperate: Emergent communication in multi-agent navigation," *arXiv e-prints*, 2020.

[13] S. Havrylov and I. Titov, "Emergence of language with multi-agent games: Learning to communicate with sequences of symbols," in *Proc. NeurIPS*, 2017.

[14] L. Yuan, Z. Fu, J. Shen, L. Xu, J. Shen, and S.-C. Zhu, "Emergence of pragmatics from referential game between theory of mind agents," in *Proc. NeurIPS*, 2020.

[15] P. K. Kuhl, "Early language acquisition: cracking the speech code," *Nature Reviews Neuroscience*, pp. 831–843, 2005.

[16] I. S. Howard and P. Messum, "Learning to pronounce first words in three languages: An investigation of caregiver and infant behavior using a computational model of an infant," *PLOS ONE*, pp. 1–21, 2014.

[17] D. Dor, *The instruction of imagination: language and its evolution as a communication technology*. Princeton University Press, 2014, pp. 105–125.

[18] Y. Bisk, A. Holtzman, J. Thomason, J. Andreas, Y. Bengio, J. Chai, M. Lapata, A. Lazaridou, J. May, A. Nisnevich, N. Pinto, and J. Turian, "Experience grounds language," in *Proc. EMNLP*, 2020, pp. 8718–8735.

[19] T. Linzen, "How can we accelerate progress towards human-like linguistic generalization?" in *Proc. ACL*, 2020, pp. 5210–5217.

[20] P.-Y. Oudeyer, "The self-organization of speech sounds," *Journal of Theoretical Biology*, pp. 435–449, 2005.

[21] L. Steels and T. Belpaeme, "coordinating perceptually grounded categories through language: a case study for colour," *Behavioral and Brain Sciences*, p. 469–489, 2005.

[22] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, p. 229–256, 1992.

[23] L. Weaver and N. Tao, "The optimal reward baseline for gradient-based reinforcement learning," in *Proc. UAI*, 2001, p. 538–545.

[24] C. Moulin-Frier and P.-Y. Oudeyer, "Multi-Agent Reinforcement Learning as a Computational Tool for Language Evolution Research: Historical Context and Future Challenges," in *Proc. AAAI*, 2021.

[25] S. Gao, W. Hou, T. Tanaka, and T. Shinozaki, "Spoken language acquisition based on reinforcement learning and word unit segmentation," in *Proc. ICASSP*, 2020, pp. 6149–6153.

[26] R. Komatsu, S. Gao, W. Hou, M. Zhang, T. Tanaka, K. Toyoda, Y. Kimura, K. Hino, Y. Iwamoto, K. Mori, T. Okamoto, and T. Shinozaki, "Automatic spoken language acquisition based on observation and dialogue," *IEEE Journal of Selected Topics in Signal Processing*, pp. 1–13, 2022.

[27] D. Lewis, *Convention*. Blackwell, 1969.

[28] M. Rita, R. Chaabouni, and E. Dupoux, ""LazImpa": Lazy and impatient neural agents learn to communicate efficiently," in *Proc. ACL*, 2020, pp. 335–343.

[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, pp. 529–533, 2015.

[31] R. S. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

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

[33] I. Sarker, "Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions," *SN Computer Science*, 2021.

[34] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. EMNLP*, Doha, Qatar, 2014, pp. 1724–1734.

[35] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Proc. NIPS*, 2014.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NeurIPS*, 2017.

[37] E. Kharitonov, R. Chaabouni, D. Bouchacourt, and M. Baroni, "EGG: a toolkit for research on emergence of lanGuage in games," in *Proc. EMNLP-IJCNLP*, Nov. 2019, pp. 55–60.

[38] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg *et al.*, "Deep Speech 2: End-to-end speech recognition in English and Mandarin," in *Proc. ICML*, 2016, p. 173–182.

[39] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pp. 3451–3460, 2021.

[40] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pp. 357–366, 1980.

[41] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006, p. 369–376.

[42] J. Duddington, "eSpeak text to speech," 2006. [Online]. Available: http://espeak.sourceforge.net/

[43] P. Taylor, A. Black, and R. Caley, "The architecture of the festival speech synthesis system," 2003.

[44] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, R. A. Saurous, Y. Agiomvrgiannakis, and Y. Wu, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *Proc. ICASSP*, 2018, pp. 4779–4783.

[45] J. Kong, J. Kim, and J. Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Proc. NeurIPS*, 2020.

[46] N. Chomsky, *Aspects of the Theory of Syntax*. The MIT Press, 1965.

[47] C. de l'homme Débat, J. Piaget, N. Chomsky, M. Piattelli-Palmarini, and I. Chomsky, *Language and Learning: The Debate Between Jean Piaget and Noam Chomsky*. Harvard University Press, 1980.

[48] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, 2020, pp. 1877–1901.

[49] K. Olaleye, D. Oneata, and H. Kamper, "Keyword localisation in untranscribed speech using visually grounded speech models," in *Proc. Interspeech*, 2022.

[50] M. H. Christiansen and N. Chater, "The now-or-never bottleneck: A fundamental constraint on language," *Behavioral and Brain Sciences*, p. e62, 2016.

[51] O. Räsänen, "Computational modeling of phonetic and lexical learning in early language acquisition: Existing models and future directions," *Speech Communication*, pp. 975–997, 2012.

[52] H. Rasilo and O. Räsänen, "An online model for vowel imitation learning," *Speech Communication*, pp. 1–23, 2017.

[53] M. Asada, "Modeling early vocal development through infant–caregiver interaction: A review," *IEEE Transactions on Cognitive and Developmental Systems*, pp. 128–138, 2016.

[54] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. NeurIPS*, 2016, pp. 2137–2145.

[55] K. Cao, A. Lazaridou, M. Lanctot, J. Z. Leibo, K. Tuyls, and S. Clark, "Emergent communication through negotiation," *ArXiv*, 2018.

[56] H. Kamper, A. Jansen, and S. Goldwater, "A segmental framework for fully-unsupervised large-vocabulary speech recognition," *Computer Speech & Language*, pp. 154–174, 2017.

[57] J. Rugayan, "A deep learning approach to spoken language acquisition," Master's thesis, Norwegian University of Science and Technology, 2021.

[58] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proc. SSST-8*, 2014, pp. 103–111.

[59] I. de Jong, "miniaudio 1.53," 2019. [Online]. Available: https://pypi.org/project/miniaudio/

[60] B. McFee, A. Metsai, M. McVicar, S. Balke, C. Thomé, C. Raffel, F. Zalkow, A. Malek, Dana, K. Lee, O. Nieto, D. Ellis, J. Mason, E. Battenberg, S. Seyfarth, R. Yamamoto, viktorandreevichmorozov, K. Choi, J. Moore, R. Bittner, S. Hidaka, Z. Wei, nullmightybofo, D. Hereñú, F.-R. Stöter, P. Friesch, A. Weiss, M. Vollrath, T. Kim, and Thassilo, "librosa/librosa: 0.8.1rc2," 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4792298

[61] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[62] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: an audio captioning dataset," in *Proc. ICASSP*, 2020, pp. 736–740.

[63] M. Jeub, M. Schäfer, and P. Vary, "A binaural room impulse response database for the evaluation of dereverberation algorithms," in *Proc. of International Conference on Digital Signal Processing*, 2009, pp. 1–4.

[64] M. Grinberg, *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.

[65] W. Gay, *Raspberry Pi Hardware Reference*, 1st ed. Apress, 2014.

[66] H. Brighton and S. Kirby, "Understanding linguistic evolution by visualizing the emergence of topographic mappings," *Artificial Life*, 2006.

[67] C. Moulin-Frier, J. Diard, J.-L. Schwartz, and P. Bessière, "Cosmo ("communicating about objects using sensory–motor operations"): A bayesian modeling framework for studying speech communication and the emergence of phonological systems," *Journal of Phonetics*, pp. 5–41, 2015.

[68] T. B. Sheridan, "Human–robot interaction: Status and challenges," *Human Factors*, 2016.

[69] V. Renkens and H. Van hamme, "Mutually exclusive grounding for weakly supervised non-negative matrix factorisation," in *Proc. Interspeech*, 2015.

[70] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. ICLR*, 2015.

Appendix A

Code Listings

Listing A.1: Python imports used in code listings.

```
import subprocess # Module used to call CLI applications (eSpeak and Festival)
import numpy as np # Matrix and vector manipulation
from miniaudio import SampleFormat, decode # Used to decode byte strings into audio
from librosa import resample # Used to resample audio signals
import torch # PyTorch automatic differentiation
from flask import Flask, request, jsonify # Flask web server
import sounddevice as sd # audio playback and recording Python module
```