Learning Translations: Emergent Communication Pretraining for Cooperative Language Acquisition

Dylan Cope and **Peter McBurney**
King's College London
dylan.cope@kcl.ac.uk

Abstract

In Emergent Communication (EC) agents learn to communicate with one another, but the protocols that they develop are specialised to their training community. This observation led to research into Zero-Shot Coordination (ZSC) for learning communication strategies that are robust to agents not encountered during training. However, ZSC typically assumes that no prior data is available about the agents that will be encountered in the zero-shot setting. In many cases, this presents an unnecessarily hard problem and rules out communication via preestablished conventions. We propose a novel AI challenge called a *Cooperative Language Acquisition Problem* (CLAP) in which the ZSC assumptions are relaxed by allowing a 'joiner' agent to learn from a dataset of interactions between agents in a target community. We propose and compare two methods for solving CLAPs: Behaviour Cloning (BC), and *Emergent Communication pretraining and Translation Learning* (ECTL), in which an agent is trained in self-play with EC and then learns to translate between an emergent protocol and the target community's protocol.

1 Introduction

Creating teams of artificial agents that can communicate and cooperate has been a long-standing area of interest in multi-agent systems research. Advances in multi-agent reinforcement learning have enabled researchers in the field of Emergent Communication (EC) to train such teams in ever more complex domains [Wagner *et al.*, 2003; Foerster *et al.*, 2016; Sukhbaatar *et al.*, 2016; Lazaridou and Baroni, 2020]. These agents are typically trained by allowing discrete messages to be exchanged between agents. Programmers do not assign meaning to the messages, rather, meaning emerges via the training process as communicative conventions are developed in service of solving the task. As such, the mapping between meanings and messages is arbitrary, and any permutation of a learned protocol is equally likely to appear across different training runs [Bullard *et al.*, 2021]. The result is that the learned conventions established within a training community will be very unlikely to work with new agents, and by default, the EC-trained agents will be incapable of adapting.

In response to this, many researchers have become interested in devising methods in which agents learn communicative strategies that can adapt to this *Zero-Shot Coordination* (ZSC) setting [Li *et al.*, 2023; Hu *et al.*, 2021; Hu *et al.*, 2020; Ossenkopf, 2020; Cope and Schoots, 2020; Bullard *et al.*, 2020]. ZSC algorithms typically aim to successfully communicate with an unknown agent on the first encounter, without any prior information. But in many real-world settings, this is an unnecessarily challenging assumption. If someone is injured on a street in London, passing pedestrians can form an ad hoc team and aid the patient by speaking to each other in English to coordinate a response. Indeed, language is arguably the most critical set of conventions that such teams can draw upon to effectively work together.

The study of artificial agents that can form ad hoc teams is known as *Ad Hoc Teamwork* (AHT) [Stone *et al.*, 2010]. Similarly to ZSC, most of these algorithms aim to make as few assumptions as possible about the players that an agent may form a team with. Notably, [Sarratt and Jhala, 2015] applied this minimalist approach to communication. Other work has relaxed this by assuming a prior known communication protocol [Barrett *et al.*, 2014; Mirsky *et al.*, 2020].

In this work, we present a novel AI challenge that we call a *Cooperative Language Acquisition Problem* (CLAP). Here by 'language acquisition' we mean learning the syntax and semantics of a preexisting communication system used by a community. This class of problems is positioned between the challenges of ZSC and AHT. In a CLAP, we are given a dataset of communication events between speakers and listeners in a *target community* as they solve a problem. Our goal is to construct a *joiner agent* that can communicate and cooperate with agents from this community. This problem is also closely related to Imitation Learning (IL), however, most work in IL is confined to the single agent setting [Hussein *et al.*, 2017]. So to the best of our knowledge, this is the first attempt to pose an IL problem for multi-agent communication within a formal cooperative model.

Alongside defining CLAP, we outline two baseline solutions to this problem. The first uses a simple imitation learning method. The second is a novel algorithm called *Emergent Communication pretraining and Translation Learning*

(ECTL). We introduce two environments and train target communities of agents that cooperate via a learned communication protocol. Data is then gathered from these communities and then used to train joiner agents with ECTL and the behaviour cloning (BC) algorithm [Sammut, 2010]. We demonstrate that ECTL is more robust than BC to expert demonstrations that give an incomplete picture of the underlying problem and that ECTL is significantly more effective than BC when communications data is limited. Finally, we apply these methods to user-generated data and show that ECTL can learn to communicate with a human to cooperatively solve a task.

2 Background

2.1 Decentralised POMDPs

A *Decentralised Partially-Observable Markov Decision Process* (Dec-POMDP) is a formal model of a cooperative environment defined as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, r, \mathbf{\Omega}, O)$ [Oliehoek and Amato, 2016], where $\mathcal{S}$ is a set of states, and $\mathcal{A} = \prod_i \mathcal{A}_i$ is a product of individual agent action sets. A *joint action* $\mathbf{a} \in \mathcal{A}$ is a tuple of actions from each agent that is used to compute the environment's transition dynamics, defined by a probability distribution over states $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$. Team performance is defined by a cooperative reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ over state transitions and joint actions. $\mathbf{\Omega} = \{\Omega_i\}$ is a set of observation sets, and $O : S \to \prod_i \Omega_i$ is an observation function.

Each agent i follows a policy π_i that maps an observation sequence (or a single observation if i is *memoryless*) to a distribution over its actions. A *trajectory* for an agent i is a sequence of observation-action-reward tuples $\tau_i \in \mathcal{T}_i = (\Omega_i \times \mathcal{A}_i \times \mathbb{R})^*$. For a set of policies $\Pi = \{\pi_i\}$, a joint trajectory is $\boldsymbol{\tau} \in \mathcal{T} = (\mathbf{\Omega} \times \mathcal{A} \times \mathbb{R})^*$, and we can denote the distribution of joint trajectories for this set of policies acting in the environment as $\mathcal{M}|_\Pi$. In this work, we will only consider finite-horizon Dec-POMDPs, so the lengths of trajectories will always be bounded. The *total reward* for this trajectory is the sum of rewards along the sequence, denoted $R(\boldsymbol{\tau})$. The expected sum of rewards for a set of policies will be denoted $R(\Pi) = \mathbb{E}_{\boldsymbol{\tau} \sim \mathcal{M}|_\Pi}[R(\boldsymbol{\tau})]$.

2.2 Emergent Communication

Emergent communication is the study of agents that learn (or evolve) to make use of communication channels without previously established semantics. In a typical set-up, each agent's action set in the Dec-POMDP can be expressed as $\mathcal{A}_i = \mathcal{A}_i^e \times \mathcal{A}_i^c$, where $\mathcal{A}_i^c$ is a set of *communicative actions*, and $\mathcal{A}_i^e$ is a set of *environment actions*. The communicative actions can further be written as the product of one-way communication channels from i to $j \in C_i$ using a discrete *message* alphabet Σ, i.e. $\mathcal{A}_i^c = \Sigma^{|C_i|}$. These are cheap-talk channels, meaning there is no cost to communication. This variant of a Dec-POMDP is known as a Dec-POMDP-Comm [Goldman and Zilberstein, 2004; Goldman and Zilberstein, 2008; Oliehoek and Amato, 2016]. The messages have no prior semantics as the transition function of the Dec-POMDP only depends on the environment actions $\mathcal{A}_i^e$, and agents are not programmed to send messages with any prescribed meaning. Rather, the semantics emerge through training. For this paper, we will consider memoryless agents that communicate within a centralised forward pass computing a joint action for the environment (similar to [Goldman and Zilberstein, 2003]).

Policy Factorisation. We suppose that we can factor a memoryless communication agent's policy π_i into an environment-level policy $\pi_i^e : \Omega_i \times \Sigma^{n_s} \to \mathcal{A}_i^e$, where n_s is the number of agents that send messages to agent i, and a communication policy $\pi_i^c : \Omega_i \to \mathcal{A}_i^c$. Therefore, given a joint observation $\mathbf{o} = (o_1, \ldots, o_N) \in \prod_i \Omega_i$, the joint policy $\boldsymbol{\pi}(\mathbf{o})$ is computed by first producing the outgoing messages $M_{out}^i = \pi_i^c(o_i)$ from each sender agent i to each receiver agent $j \in C_i$. A set of incoming messages M_{in}^i is then constructed for each agent i by passing the messages for i through a communication channel function $\sigma_i : \mathcal{A}_i^c \to \Sigma$. The final environment actions are then given by $a_i = \pi_i^e(o_i, M_{in}^i)$.

Self-play. Emergent communication training is often done in self-play, where parameters are shared between agents and centrally optimised [Tesauro, 1994; Hu *et al.*, 2020]. Communication can be hard to learn with Reinforcement Learning (RL) as sending messages cannot provide reward unless listeners already know how to use the information. This chicken-and-egg problem can be solved with centralised training by allowing gradients to backpropagate through communication channels. A Gumbel-Softmax [Jang *et al.*, 2017; Maddison *et al.*, 2017] function is a popular choice as it facilitates backpropagation during training and can be discretised during evaluation.

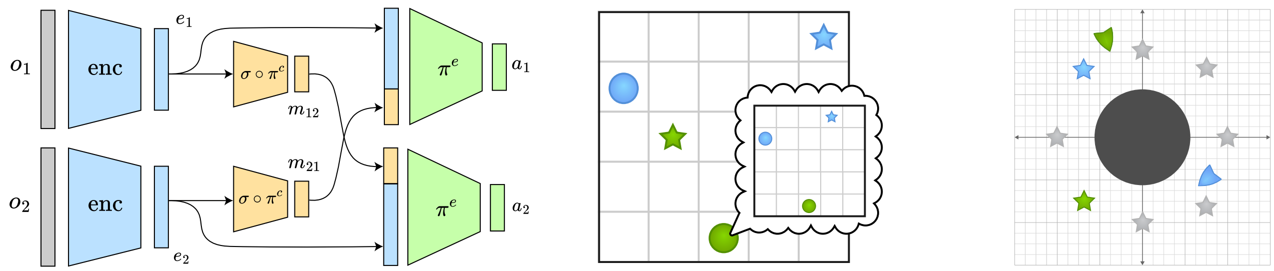
As [Lowe *et al.*, 2019] have shown, measuring whether or not agents trained to communicate are doing so can be tricky for complex environments. To remedy this, they introduced definitions of *positive listening* and *positive signalling*. In short, to be positive listening/signalling; the listening agent should change its actions according to the messages it receives and the signalling agent should change its messages depending on what it observes.

2.3 Imitation Learning

Imitation Learning (IL) is a form of machine learning in which expert demonstration data is used to construct a policy for solving a task. The data is typically in the form of an observation o and action a, and the imitation learning problem is posed as supervised classification learning [Ross and Bagnell, 2010; Rajaraman *et al.*, 2020], e.g. optimising θ to minimise the difference between a and $\hat{a} = \pi_\theta(o)$. See [Hussein *et al.*, 2017] for a comprehensive review of these methods.

3 Cooperative Language Acquisition

In this section, we introduce our definition of a *Cooperative Language Acquisition Problem* (CLAP). A CLAP can be formulated from the simplest case involving a preexisting *target community* of two agents, denoted $\Pi = \{A, B\}$, that achieve some non-trivial performance in a Dec-POMDP-Com $\mathcal{M}$. Consider an interaction at time t in which A is speaking and B is listening. While making the observation o_t^s, the speaker emits a message m_t. Then, while observing o_t^r and m_t, the receiver takes the action a_i in the environment. We are given access to $\mathcal{M}$ and a dataset of such interactions,



(a) Joint agents architecture (two agents) with communication.

(b) Illustration of the gridworld environment (Section 5.1).

(c) Illustration of the driving environment (Section 5.1).

Figure 1: (a) Agents architecture diagram for two agents (top and bottom) with communication. (b) Illustration of the gridworld toy environment with two agents, depicted with blue and green circles on a 5x5 grid. The green and blue stars indicate the locations of each agent's respective goals. The cloud thought bubble depicts the world as the green agent observes it. Note, each agent does not see their own goal, and they are off-by-one square in their knowledge of each other's goal. (c) Illustration of the driving environment with two agents. The dark circle in the centre is the 'pit'; the region in which large negative penalties are given when agents enter. The stars indicate goal locations, which can spawn in one of eight locations (the unused locations indicated by the greyed-out stars). The continuous state space is indicated with the grid axes. The agents are represented by arrowheads indicating their current position and direction. Again, here agents do not observe their own goals and therefore need to communicate, but unlike the gridworld, they do have perfect knowledge of the other agent's goals.

denoted $(o_t^s, m_t, o_t^r, a_t^r) \in \mathcal{D}$, and our task is to construct a *joiner agent* $\mathcal{J}$. In this paper, we focus on a specific case of the task in which we aim to replace a target agent in Π (so there is always a fixed number of players), which we call CLAP-Replace. The agent $\mathcal{J}$ should be able to take on the role of a target agent (e.g. either A or B) and successfully communicate with the other, while also acting in the environment to maximise the cooperative reward. The joiner will be evaluated zero-shot, i.e. *on the first joining event*, so all learning must be done beforehand.

The emphasis of the dataset is on learning the communication protocol as this is assumed to be the key aspect of the community's strategy that cannot be learned independently from observing their behaviour. As the communication symbols have no prior semantics, any protocol could be equally successful if the same meanings were assigned to a different permutation of the symbols. On the other hand, there may be many 'environment-level' behaviours that can be learned without needing to know the specific strategies of a given community. We will discuss disentangling these factors in Section 3.2.

3.1 Problem Definitions

The cooperative language acquisition task is to construct a *joining agent* $\mathcal{J}$ that learns from observing a target community Π. When $\mathcal{J}$ replaces an agent in Π, we have a CLAP-Replace task:

Problem 1 (CLAP-Replace)**.** *Suppose there exists a Dec-POMDP-Com $\mathcal{M}$ and a set of policies $\Pi = \{\pi_i\}$ trained in $\mathcal{M}$. Given a target policy $\pi_k \in \Pi$ to replace, a dataset $\mathcal{D}_k$ of interactions $(o_t^s, m_t, o_t^r, a_t^r)$ in which π_k is either a speaker or listener, the task is to construct a policy $\pi_{\mathcal{J}}$ such that:*

$$\pi_{\mathcal{J}} \in \arg\max_{\pi'} R(\{\pi'\} \cup \Pi^{-k}) \tag{1}$$

Where $\Pi^{-k} = \Pi \setminus \{\pi_k\}$. In other words, maximise the team rewards when the joiner replaces k.

This can be decomposed into three related problems; a *forward communication* (signalling), a *backward communication* (listening), and *acting*. More precisely, $\mathcal{J}$ needs to learn to send messages that maximise the expected sum of rewards for listening agents and interpret messages to maximise the rewards of its trajectory. But the agent also needs to learn to utilise information, both communicated and observed, for selecting actions.

If we were allowed to interact with the target community Π before evaluation, we could apply standard reinforcement learning tools. However, we are interested in settings in which disrupting the community and jeopardising performance to learn is not acceptable, so $\mathcal{J}$ needs to perform well on the first actual joining episode.

3.2 Disentangling Environment-Level and Communicative Competencies

We assume that teams of agents can achieve a certain level of performance in the environment by two categories of competencies: (1) 'environment-level' skills, and (2) communication strategies that rely on established conventions. The key distinction we aim for is that the former should be learnable independently of observing a target community's behaviour. We start by defining that the team achieves an average total reward of R when acting together in the environment and communicating.

Measuring the *communicative competency* (1) is relatively easy: if communication is blocked and the team achieves a lower average total reward $R' < R$, and by assuming that the agents are engaged in positive listening and signalling, we can attribute this drop in performance to a lack of communication.

We evaluate the influence of *environment-level competencies* (2) by similarly hampering them, and investigating whether this leads to a drop in performance. We will assume that each agent's observation can be partitioned into 'private' (or 'local') and shared (or 'global') components: $o_t^i = (l_t^i, g_t)$, where g_t is observed by both speakers and listeners, but l_t^i is private to i. To assess the contribution of environment-level skills we restrict the environment-level policy by removing private information, $o'_t = (\mathbf{0}, g_t)$, and replacing each π_i with $\pi'_i(\mathbf{o}_t, M_t) = (\pi^e_i(o'_t, M_t), \pi^c_i(o_t))$. We can then attribute a change $R'' < R$ to the agent being prevented from exercising environment-level skills. To understand this intervention, consider the following points:

1. We hamper environment-level skills by blocking private information from π^e, but an agent must still be allowed to receive messages as this may provide vital information for succeeding based on communication skills. Thus, π^c for the speaker must not be intervened with.
2. Communication may rely on shared information, so removing the context by blocking this information may interfere with measuring environment-level skills.
3. Information private to a recipient cannot have been used by the speaker to create the messages, so if a drop in performance is seen when it is removed from π^e, this can only be due to a loss in the recipient's capacity to use that private information to contribute to the team's performance through its actions (as opposed to its contributions through sending messages). Recall, we are only obfuscating this information from π^e; π^c still sees it.
4. If no drop from blocking private information to π^e is observed, it implies either: (a) all of an agent's contribution to the collective performance is the result of following direct orders from a speaker. Or, (b) an agent's contribution is entirely contingent on global information.

Note that following 4b, the absence of a drop does not imply that the team does not possess environment-level competencies, but the presence of a drop does.

4 Methods for Constructing Joiners

In this section, we introduce two methods for constructing joiners for the CLAP-Replace task. The first applies a standard imitation learning algorithm, and the second pretrains agents using emergent communication and then translates the learned communication protocol to the target community's protocol. Each method is decomposed into forward and backward problems, posed as supervised learning tasks.

4.1 Behaviour Cloning (BC)

In the setup for CLAP-Replace we are given a dataset $(o^s_t, m_t, o^r_t, a^r_t) \in \mathcal{D}_k$ of speaker and listener observations, messages, and actions taken by the receiver. The simplest imitation learning baseline solution is to independently apply the *behaviour cloning* algorithm [Widrow and Smith, 1964; Michie *et al.*, 1990; Sammut, 2010] to the signalling and listening problems. The dataset is partitioned into two datasets: a signalling dataset $(o^s_t, m_t) \in \mathcal{D}^s_k$ where k is the speaker and a listening dataset $((o^r_t, m_t), a^r_t) \in \mathcal{D}^r_k$ where k is the receiver. These datasets are structured as input-label tuples to learn communicative and environmental-level policy factors (π^c_{il} and π^e_{il}) of the overall imitation policy π_{il}. These are learned with a categorical cross-entropy (CCE) loss between predicted and actual labels. In Figure 2, π^c_{il} and π^e_{il} are composed of encoder and communication/action heads, with the same architecture as in Figure 1a.

4.2 Emergent Communcation Pretraining and Translation Learning (ECTL)

A well-known issue with imitation learning agents is that they can be brittle given the natural biases in the expert demonstration data [Kumar *et al.*, 2022]. Especially in more complex domains, experts typically stay in the regions of state space in which they get high rewards, which may only be a small portion of the possibilities. If a BC agent makes a mistake when attempting the task itself, it may enter into unseen territory. Thus errors can compound as the agent has not learned what to do and makes more mistakes, leading to degraded team performance in a multi-agent cooperative setting.

This leads to the idea that the joiner could explore the state space of $\mathcal{M}$ before joining Π, and thereby become a more reliable cooperator. In general, pretraining may allow the agent to learn environment-level skills that could be transferable to cooperating with any new team. To this end, we propose the method *Emergent Communication pretraining and Translation Learning* (ECTL). The first step of ECTL is to (pre)train a set of agents Π' from scratch in $\mathcal{M}$ that we refer to as the *EC training community*, where for each agent in Π there is a corresponding agent in Π'.

The agents in Π' are composed of three components, illustrated in Figure 1a: an *observation encoder* enc_θ, a *communications head* π^c_θ, and an *action head* π^e_θ, parameterised by θ, and shared amongst the policies Π'. For brevity and to be consistent with the notation in previous sections we will omit the encoder from our notation, and write $\pi^c_\theta(o)$ instead of $\pi^c_\theta(enc_\theta(o))$. This training process could use any viable multi-agent reinforcement learning algorithm, and which is most suitable depends on the properties of the underlying environment and the number of agents. In Section 5.2, we will outline the specific methods we used for our experiments.

Agents in the EC training community learn to cooperate via an emergent communication protocol over their message alphabet Σ'. The next step in ECTL towards building the joiner agent for Π is to *translate* this protocol to the one used by the target community. For a CLAP-Replace task with target agent $\pi_k \in \Pi$, we select the equivalent pretrained $\pi'_k \in \Pi'$ to use a starting point for *translation learning*. We will refer to π'_k as the *EC pretrained agent*.

Once again, the target community data $\mathcal{D}_k$ is transformed into a signalling dataset $(o^s_t, m_t) \in \mathcal{D}^s_k$ and a listening dataset $((o^r_t, m_t), m'_t) \in \mathcal{D}^r_k$. Note that the ECTL signalling dataset is identical to the signalling dataset used for imitation learning, but the listening dataset is different. Instead of the label being an action, it is a message from the pretrained message space Σ'. To construct these labels, we use the speaker's observation from $\mathcal{D}_k$ and compute the message that pretrained agent would have emitted in that situation, i.e. $m'_t = \pi^c_\theta(o^s_t)$.

From these data, we learn *translation functions* for signalling and for listening. A separate translation function can be learned for each communication channel if there are multiple possible senders and/or receivers for the target agent, or two models (one for each CLAP sub-problem) can be used by providing the sender/receiver identifiers as input. For our experiments, we will use the latter approach. However, for simplicity in the following descriptions of how these func-

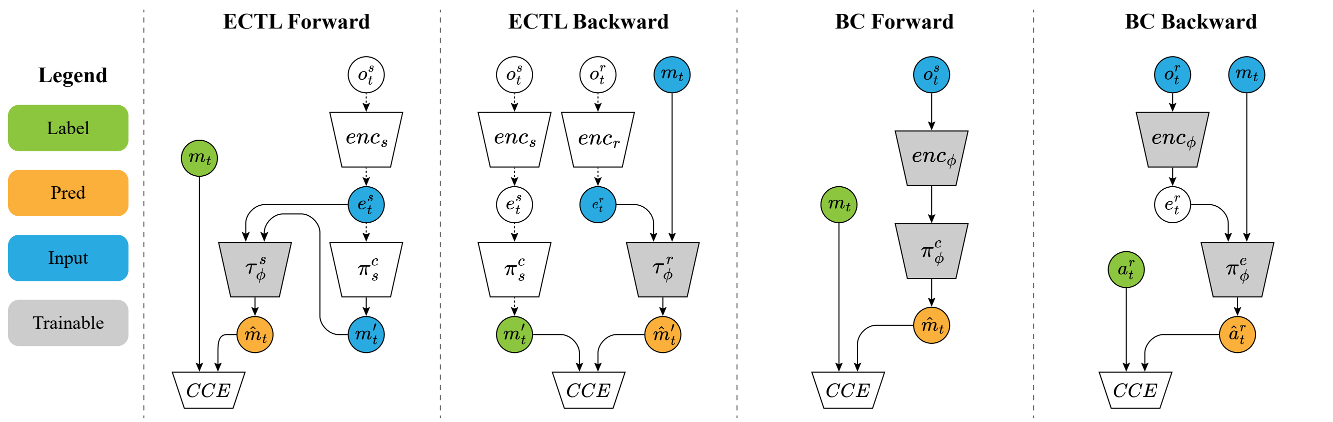
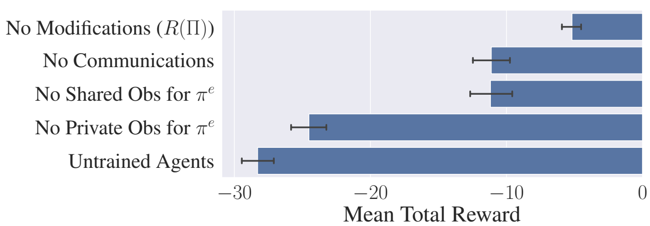
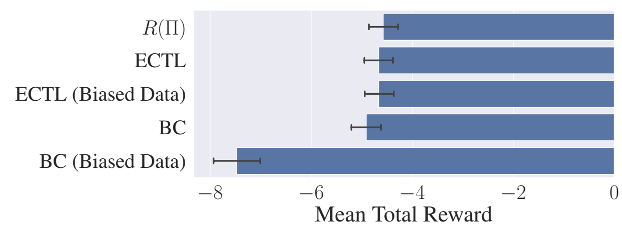


Figure 2: Training architecture diagrams for the *Behaviour Cloning* (BC) and *Emergent Communication pretraining and Translation Learning* (ECTL) methods. The forward and backward problems for each method are solved with supervised learning. Circles are variables and trapezoids are functions. Dotted lines indicate that gradients are blocked from backpropagating along a path. For the ECTL diagrams the enc and π^c functions are learned during the emergent communication pretraining phase. The variables o_t^s, o_t^r, m_t, a_t are the sender/receiver agent observations, the sender's message, and the receiver's action from the dataset of interactions collected from the target community.



(a) Ablated agents in the target community.



(b) Mean reward for different teams.

Figure 3: Performance results from cases in which (a) the team is formed of a variety of agents, comparing ECTL to behaviour cloning (BC) with unbiased and biased data from the gridworld environment. And (b) the target community agents ablated in different ways. All results are from 500 evaluation episodes and the error bars show means within 95% confidence intervals.

tions are trained, we will assume one sender and receiver for the target agent and omit agent identifiers. Furthermore, the training architectures for the translation functions are illustrated by the two diagrams to the left of Figure 2.

A signalling translation function $\tau_\phi^s : \Omega_k \times \Sigma' \to \Sigma$, parameterised by ϕ, maps the communication protocol of the EC training community to the target community protocol. Given an observation $o_t^s \in \Omega_k$ and a message $m_t \in \Sigma$ from $\mathcal{D}_k^s$, we can compute the message $m_t' = \Sigma'$ that the agent $\pi_k' \in \Pi'$ would send in that situation. The translation function is trained to predict the demonstrator agent's message:

$$\phi^* \in \arg\min_{\phi} \sum_{(o_t^s, m_t) \in \mathcal{D}_k^s} CCE(m_t, \hat{m}_t) \tag{2}$$

$$\text{where } \hat{m}_t = \tau_\phi^s(o_t^s, m_t'), \quad m_t' = \pi_\theta^c(o_t^s) \tag{3}$$

A listening translation function $\tau_\psi^r : \Omega_k \times \Sigma \to \Sigma'$ maps the target community's communication protocol to the EC training community's protocol and is parameterised by ψ. Note that, in general, τ^r and τ^s need not be inverses of one another, as two agents can use arbitrarily different protocols in each direction.

To train τ_ψ^r, we use the message that was received by the listener (the target agent) m_t, the listener's observation o_t^r, and the speaker's observation o_t^r. The translation function takes the observation and the message and produces a new message in the EC training community's message space: $\tau_\psi^r(o_t^r, m_t) = \hat{m}_t' \in \Sigma'$. To get the ground-truth label for this prediction, we find the message that the EC pretrained agent would have sent in the target community speaker's situation. The result is the following optimisation criterion:

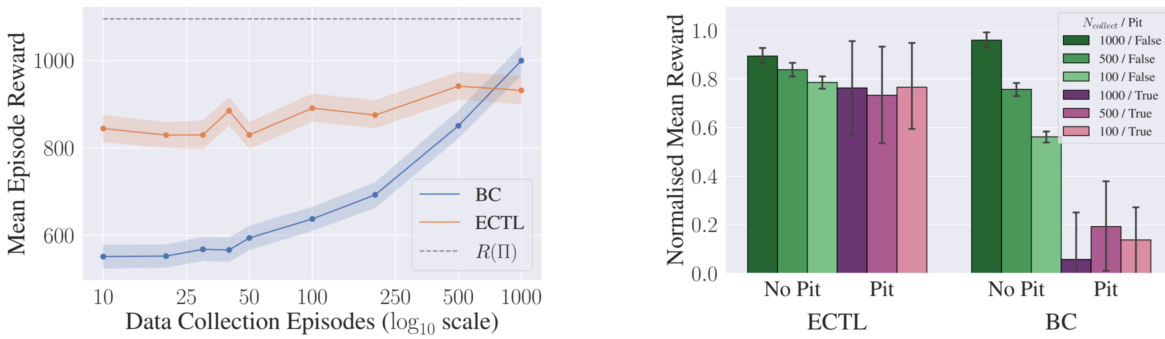
$$\psi^* \in \arg\min_{\psi} \sum_{((o_t^r, m_t), m_t') \in \mathcal{D}_k^r} CCE(m_t', \hat{m}_t') \tag{4}$$

$$\text{where } \hat{m}_t' = \tau_\psi^r(o_t^r, m_t), \quad m_t' = \pi_\theta^c(o_t^s) \tag{5}$$

Finally, we can put together these pieces into a final *translator* joiner agent $\mathcal{J}_{ectl}$ composed of the following communication and environment-level policy factors:

$$\pi_{ectl}^c = \tau_\phi^s \circ \pi_\theta^c \quad \text{and} \quad \pi_{ectl}^e(o, m) = \pi_\theta^e(o, \tau_\psi^r(m)) \tag{6}$$

So when the joiner agent receives a message, it uses the listening translation function to predict the message that it would have received from the equivalent agent in its EC training community. When speaking, the agent first computes the message that it would have sent in its original message space, and then passes that through the signalling translation function before finally sending it.



(a) The mean CLAP-Replace performance on the Driving Game (No Pit) for ECTL and BC, against the number data collection episodes ($N_{collect}$) from the target community.

(b) Mean reward for each method normalised by the original team mean rewards. Shows the impact of the pit on CLAP performance, highlighting BC's relative fragility.

Figure 4: Comparisons between BC and ECTL on the Driving Communication environment.

5 Experiments

5.1 Environments

In order to empirically investigate BC and ECTL for solving CLAP-Replace tasks, we created two environments. Firstly, a simple gridworld toy environment in which N agents cooperate by communicating goal information. Secondly, a 'driving' communication problem in which agents must navigate a continuous space to reach a goal, while potentially avoiding a pit in their way. Again, the goal location is known by another agent so they need to communicate to solve the problem.

Goal Communications Gridworld. Each agent has a goal square in the grid that they need to reach that changes each episode. No agent observes its own goal unless it is within one tile of it, but at all times it observes a 'close guess' (a location within one tile) of the goal of another of the agents in the game. The environment is illustrated in Figure 1b.

Goal Communications Driving Game. As discussed in Section 4.2, BC agents can be brittle when regions of the state space are not present in the demonstration data. To investigate this problem, we introduced this 'driving' environment in which the agent steers and accelerates a body in a continuous grid (Figure 1c). This environment has two settings defined by the presence or absence of a circular region in the centre of the world where a large penalty is applied for every agent within the region. The agents must navigate to one of eight fixed goal locations, known by their partner and selected at random for each episode.

5.2 Creating Target Communities

Target communities Π of three agents in the gridworld environment and two agents in the driving environment were trained using *Multi-Agent Proximal Policy Optimisation* (MAPPO) [Yu *et al.*, 2022]. For the policy networks we used the architecture in Figure 1a with parameter sharing between agents. The value network used a concatenation of all of the agents' observations as input, meaning that we rely on centralisation for training. The value network did not share any parameters with the policy networks. A Gumbel-Softmax [Jang *et al.*, 2017; Maddison *et al.*, 2017] communication channel function was used on communication channels. Further training details and hyperparameters can be found in the project's GitHub repository[1].

5.3 Ablating Target Community Agents

The gridworld environment was designed such that the agents need to use a combination of communicative and individual skills in order to succeed. To demonstrate how these skills contribute to the performance, we evaluate ablated versions of the agents with the interventions discussed in Section 3.2. Figure 3a shows the results of these experiments. At the top, we show the original unmodified performance for reference ($R(\Pi)$). The next two bars, appearing roughly equal in size and variance, correspond to blocking communications (bar 2) and 'global'/'shared' information from π^e (bar 3).

5.4 Gridworld CLAP-Replace

We train ECTL and BC agents after collecting interaction datasets from $N_{collect} = 100$ episodes of Π acting in the environment. Figure 3b shows the results from forming teams of different compositions in the gridworld environment. We see that ECTL and BC solve the CLAP-Replace task by achieving the equivalent team performance as the original team ($R(\Pi)$).

Both methods have roughly the same performance, which is due to the fact that there are no opportunities for errors to compound. The BC agent will have learned the experts' behaviour from any state as the agent starting positions and goal positions are sampled from a uniform distribution over all grid tiles. Therefore, if a BC agent makes a mistake in any given position, it will have seen during training how to recover in the next position. So to investigate the effects of unseen states we bias the training data by only including cases where the target agent was in one of the first two columns of the grid. The results are also shown in Figure 3b, and we see that while ECTL maintains its performance, BC significantly degrades.

[1]https://github.com/DylanCope/learning-translations

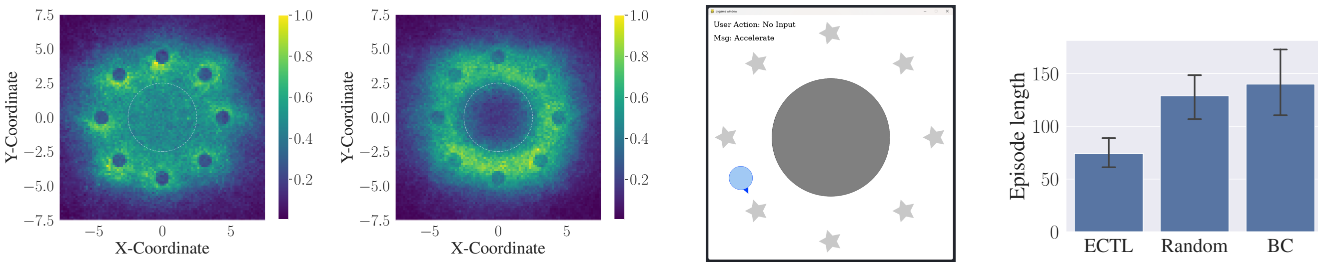


Figure 5: (a,b) Two-dimensional histograms visualising how much time the agents spend in the different regions of the environment before reaching their goals. Left and right show agents trained with/without pit penalties. The radius of the pit is indicated with the dashed circle outline. We can see that when the pit is present agents avoid that region. The goal locations are also visible as we are not plotting positions after the goals are reached. (c) The UI for the human to play the game as a listener. The top-left of the screen shows messages from the other player. (d) The mean episode lengths when we played this game with different agents. The goal is for us to navigate to our goal location as quickly as possible (one of the grey stars) using the messages. We see that only the ECTL agent was able to effectively communicate.

5.5 Driving Game CLAP-Replace

Moving on from the gridworld environment, we then performed a series of experiments in the driving game. First, we trained four teams on the environment, with two agents per training team. Two of the teams with the pit, and the other two without the pit. Figures 5a and 5b visualise where in the environment these agents spend their time (before reaching their goals), illustrating how the expert demonstrations may not cover the pit region of state space.

In Figure 4a we see the effect of the number of data collection episodes on the driving game CLAP-Replace performance without the pit. We find that the ECTL performance starts notably better than BC, and can do well with very little data. However, as more data is gathered, eventually BC scales better than ECTL. Yet, Figure 4b complicates this picture. Here we see the normalised reward comparisons between ECTL and BC showing the effect of the pit on performance. The pit does not have much impact on ECTL, but the performance of BC drops by a large margin, even for the case with 1000 data collection episodes – the setting in which BC outperforms ECTL when the pit is not present.

5.6 Translating to an Interpretable Communication Protocol

After demonstrating that BC and ECTL could successfully solve CLAPs for target communities of artificial agents, we investigated whether or not the same methods could apply to data generated by humans. We developed an interactive UI through which a user could simultaneously control two agents in the driving game while observing the entire game state (i.e. both agent's goals). This was then used to collect data from 70 episodes. Each row of data comprised the actions that the user took for each agent and the agent-specific observations. In its raw form, this data cannot be used to train CLAP agents as it lacks messages. To remedy this we augmented the data by constructing messages on the principle that each agent was following direct instructions from the other, so each agent i's action a_t^i became agent j's message $m_t = a_t^i$ to agent i.

With this data, ECTL and BC agents could now be trained in the same manner as done for the MAPPO target communities. However, due to the lack of a well-defined target community, the agents could not be evaluated in the same way. So instead, we focused on specifically evaluating the signalling capabilities of these agents with another UI that allows a user to view their agent, the possible goal locations, and a message from one of the artificial agents (Figure 5c). As the message space is the action space for the user, it is inherently interpretable and could be rendered to the user as one of the strings 'Accelerate', 'Turn Anti-Clockwise', or 'Turn Clockwise'. We then set up the UI to randomly switch the messenger agent each episode from a pool of agents comprised of an ECTL agent, a BC agent, and an agent that sends a random message. Thus the user was always unaware of which agent they were playing with. Figure 5d shows the results of this experiment, measured by the mean number of timesteps that it took for the user to reach the goal location while using the messages. It shows that only the ECTL agent was able to effectively communicate with the user, with the BC agent conveying the same lack of information as the random policy.

6 Conclusions

In this paper, we have posed a new problem for multi-agent communication learning, namely the *Cooperative Language Acquisition Problem* (CLAP). Positioned relative to Zero-Shot Coordination and Ad Hoc Teamwork, the CLAP challenge is to build an agent that learns the communication strategy of a target community via observational data. But can also leverage general 'environment-level skills' that transfer to any ad hoc team. We have proposed two approaches to this problem: *Behaviour Cloning* (BC) and *Emergent Communication pretraining and Translation Learning* (ECTL).

We have shown that ECTL can perform well in data-scarce scenarios, including learning to communicate with a human user. Additionally, it can effectively compensate for biased expert demonstrations. On the other hand, while BC is brittle, it may scale to large datasets. Further work should investigate the potential for combining the strengths of these methods. Finally, we have only explored CLAPs with a small number of agents and complete, noise-free demonstration data. Future investigations should relax these assumptions.

Acknowledgements

Dylan Cope is supported by the UKRI Centre for Doctoral Training in Safe and Trusted AI (EPSRC Project EP/S023356/1). Thank you to Nandi Schoots, Alex Jackson and Agathe Dorra for helping to proof read and refine ideas. Thank you to Cassidy Laidlaw for implementation tips and help with compute infrastructure.

References

[Barrett *et al.*, 2014] Samuel Barrett, Noa Agmon, Noam Hazon, Sarit Kraus, and Peter Stone. Communicating with unknown teammates. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, AAMAS '14, pages 1433–1434, Richland, SC, May 2014. International Foundation for Autonomous Agents and Multiagent Systems.

[Bullard *et al.*, 2020] Kalesha Bullard, Franziska Meier, Douwe Kiela, Joelle Pineau, and Jakob Foerster. Exploring Zero-Shot Emergent Communication in Embodied Multi-Agent Populations, December 2020. arXiv:2010.15896 [cs].

[Bullard *et al.*, 2021] Kalesha Bullard, Douwe Kiela, Franziska Meier, Joelle Pineau, and Jakob Foerster. Quasi-Equivalence Discovery for Zero-Shot Emergent Communication, June 2021. arXiv:2103.08067 [cs].

[Cope and Schoots, 2020] Dylan Cope and Nandi Schoots. Learning to Communicate with Strangers via Channel Randomisation Methods. In *The Emergent Communication Workshop at NeurIPS*, Vancouver, Canada, 2020.

[Foerster *et al.*, 2016] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In D. D. Lee and M. Sugiyama and U. V. Luxburg and I. Guyon and R. Garnett, editor, *Advances in Neural Information Processing Systems 29*, pages 2137–2145. Curran Associates, Inc., 2016.

[Goldman and Zilberstein, 2003] Claudia V. Goldman and Shlomo Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 03)*, AAMAS '03, pages 137–144, New York, NY, USA, July 2003. Association for Computing Machinery.

[Goldman and Zilberstein, 2004] Claudia V. Goldman and Shlomo Zilberstein. Decentralized control of cooperative systems: categorization and complexity analysis. *Journal of Artificial Intelligence Research*, 22(1):143–174, November 2004.

[Goldman and Zilberstein, 2008] Claudia V. Goldman and Shlomo Zilberstein. Communication-Based Decomposition Mechanisms for Decentralized MDPs. *Journal of Artificial Intelligence Research*, 32:169–202, May 2008. arXiv:1111.0065 [cs].

[Hu *et al.*, 2020] Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "Other-Play" for Zero-Shot Coordination. In *Proceedings of the 37th International Conference on Machine Learning*, pages 4399–4410. PMLR, November 2020. ISSN: 2640-3498.

[Hu *et al.*, 2021] Hengyuan Hu, Adam Lerer, Brandon Cui, David Wu, Luis Pineda, Noam Brown, and Jakob Foerster. Off-Belief Learning. In *Proceedings of the 38th International Conference on Machine Learning*, PMLR 139, March 2021. arXiv: 2103.04000.

[Hussein *et al.*, 2017] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation Learning: A Survey of Learning Methods. *ACM Computing Surveys*, 50(2):21:1–21:35, April 2017.

[Jang *et al.*, 2017] Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *5th International Conference on Learning Representations (ICLR 17)*, Palais des Congrès Neptune, Toulon, France, 2017.

[Kumar *et al.*, 2022] Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. Should I Run Offline Reinforcement Learning or Behavioral Cloning? In *Proceedings of the International Conference on Learning Representations (ICLR 22)*, 2022.

[Lazaridou and Baroni, 2020] Angeliki Lazaridou and Marco Baroni. Emergent Multi-Agent Communication in the Deep Learning Era, July 2020. arXiv:2006.02419 [cs].

[Li *et al.*, 2023] Yang Li, Shao Zhang, Jichen Sun, Yali Du, Ying Wen, Xinbing Wang, and Wei Pan. Cooperative Open-ended Learning Framework for Zero-Shot Coordination. In *Proceedings of the 40th International Conference on Machine Learning*, pages 20470–20484. PMLR, July 2023. ISSN: 2640-3498.

[Lowe *et al.*, 2019] Ryan Lowe, Jakob Foerster, Y-Lan Boureau, Joelle Pineau, and Yann Dauphin. On the Pitfalls of Measuring Emergent Communication. In *The 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2019.

[Maddison *et al.*, 2017] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *5th International Conference on Learning Representations (ICLR 17)*, Palais des Congrès Neptune, Toulon, France, 2017.

[Michie *et al.*, 1990] D Michie, M Bain, and J Mayes-Michie. Cognitive models from subcognitive skills. *Knowledge-based Systems for Industrial Control*, pages 71–99, January 1990.

[Mirsky *et al.*, 2020] Reuth Mirsky, William Macke, Andy Wang, Harel Yedidsion, and Peter Stone. A Penny for Your Thoughts: The Value of Communication in Ad Hoc Teamwork. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 254–260, Yokohama, Japan, July 2020.

[Oliehoek and Amato, 2016] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer International Publishing, 2016.

[Ossenkopf, 2020] Marie Ossenkopf. CoMaze: A cooperative game for zero-shot coordination. In *The Emergent Communication Workshop at NeurIPS*, 2020.

[Rajaraman *et al.*, 2020] Nived Rajaraman, Lin Yang, Jiantao Jiao, and Kannan Ramchandran. Toward the Fundamental Limits of Imitation Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 2914–2924. Curran Associates, Inc., 2020.

[Ross and Bagnell, 2010] Stephane Ross and Drew Bagnell. Efficient Reductions for Imitation Learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 661–668. JMLR Workshop and Conference Proceedings, March 2010. ISSN: 1938-7228.

[Sammut, 2010] Caude Sammut. Behavioral Cloning. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 93–97. Springer US, Boston, MA, 2010.

[Sarratt and Jhala, 2015] Trevor Sarratt and Arnav Jhala. The Role of Models and Communication in the Ad Hoc Multiagent Team Decision Problem. In *Proceedings of the Third Annual Conference on Advances in Cognitive Systems*. Cognitive Systems Foundation, 2015.

[Stone *et al.*, 2010] Peter Stone, Gal A Kaminka, Sarit Kraus, and Jeffrey S Rosenschein. Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[Sukhbaatar *et al.*, 2016] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2252–2260. Neural Information Processing Systems, 2016.

[Tesauro, 1994] Gerald Tesauro. TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play. *Neural Computation*, 6(2):215–219, 1994.

[Wagner *et al.*, 2003] Kyle Wagner, James A. Reggia, Juan Uriagereka, and Gerald S. Wilkinson. Progress in the Simulation of Emergent Communication and Language. *Adaptive Behavior*, 11(1):37–69, March 2003. Publisher: SAGE Publications Ltd STM.

[Widrow and Smith, 1964] Bernard Widrow and Fred W Smith. Pattern recognising control systems. *Computer and information sciences*, 1964.

[Yu *et al.*, 2022] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks.*, New Orleans, USA, 2022.