Co-evolution of language and agents in referential games

Gautier Dagan
University of Amsterdam
gautier.dagan@gmail.com

Dieuwke Hupkes
University of Amsterdam
dieuwkehupkes@gmail.com

Elia Bruni
University of Amsterdam
elia.bruni@gmail.com

Abstract

Referential games offer a grounded learning environment for neural agents which accounts for the fact that language is functionally used to communicate. However, they do not take into account a second constraint considered to be fundamental for the shape of human language: that it must be learnable by new language learners and thus has to overcome a *transmission bottleneck*. In this work, we insert such a bottleneck in a referential game, by introducing a changing population of agents in which new agents learn by playing with more experienced agents. We show that mere cultural transmission results in a substantial improvement in language efficiency and communicative success, measured in convergence speed, degree of structure in the emerged languages and within-population consistency of the language. However, as our core contribution, we show that the optimal situation is to *co-evolve* language and agents. When we allow the agent population to evolve through genotypical evolution, we achieve across the board improvements on all considered metrics. These results stress that for language emergence studies *cultural evolution* is important, but also the suitability of the architecture itself should be considered.

Introduction

Human languages show a remarkable degree of structure and complexity, and how such a complex system can have emerged is still an open question. One concept frequently named in the context of language evolution is *cultural evolution*. Unlike animal languages, which are taken to be mostly innate, human languages must be re-acquired by individuals (Pinker and Bloom 1990; Hurford 1998). This pressures them to fit two constraints that govern their cross-generational transmission: They must be *learnable* by new language users, and they must allow *effective communication* between proficient language users (see, e.g. Smith and Kirby, 2012).

In the recent past, computational studies of language emergence using *referential games* (see Section for a review) has received a new wave of attention. These studies are motivated by the second constraint, that language is used to communicate. The first constraint, on the other hand, is in

this framework not considered: language is not transmitted from agent to agent and there is thus no need for agents to develop languages that would survive a transmission bottleneck.[1]

In this work, we introduce a transmission bottleneck in a population of agents playing referential games, implicitly modelling *cultural evolution*. However, merely adding a transmission bottleneck is not enough. Since the types of language that may emerge through passing this bottleneck are not just dependent on the *existence* of a bottleneck, but also on the *shape* of the bottleneck, which is determined by the biases of the architecture of the agents playing the game (their genotypical design). If the genotypical design of those agents is not suitable to solve this task through communication, they will – at best – converge to a language that doesn't allow for effective communication or is difficult to learn for every new agent or – at worst – not converge to an appropriate culturally transmittable language at all. Therefore, in this work, we study the *co-evolution* of language and architecture in referential games.

To this end, we introduce the *Language Transmission Engine* to model both cultural and genetic evolution in a population of agents. We demonstrate that the emerging languages benefit from including cultural transmission as well as genetic evolution, but the best results are achieved when both types of evolution are included and languages and agents can co-evolve.

Related Work

Language Emergence Games

Much work has been done on the emergence of language in artificial agents and investigating its subsequent structure, compositionality and morphosyntax (Kirby 2001; Kirby and Hurford 2002). The original computer simulations dealt with logic and symbolic representations (Kirby 2001; Christiansen and Kirby 2003), but with the advent of modern deep learning methods and sequence-to-sequence models

---

[1] In fact, multiple studies, although perhaps unrelatedly, multiple recent studies have shown that the languages emerging in such games do not share even basic properties of human languages (e.g. Bouchacourt and Baroni, 2018).

(Sutskever, Vinyals, and Le 2014), there has been a renewed interest in simulating the emergence of language through neural network agents (Lazaridou, Peysakhovich, and Baroni 2017; Havrylov and Titov 2017).

In the exploration of language emergence, different training approaches and tasks have been proposed to encourage agents to learn and develop communication. These tasks are commonly set up in an end-to-end setting where reinforcement learning can be applied. This is often a two-player referential game where one agent must communicate the information it has access to (typically an image), while the other must guess it out of a lineup (Evtimova et al. 2018; Lazaridou, Peysakhovich, and Baroni 2017). Mordatch and Abbeel (2018) and Choi, Lazaridou, and de Freitas (2018) find that structure and compositionalility can arise in emerged languages in such setups; Kottur et al. (2017) show that natural language does not arise naturally and has to be incentivised by imposing specific restrictions on games and agents.

Evolution of Language

The evolution of human language is a well-studied but still poorly understood topic. One particular open question concerns the relation between two different evolutionary processes: *genetic* evolution of the agents in the population and *cultural* evolution of the language itself (Fitch 2010). Christiansen and Kirby 2003 assert that the question of genetic versus cultural evolution ultimately arises from three distinct but interacting adaptive systems: individual learning, cultural transmission, and genetic evolution.

Cultural Evolution Cultural transmission is thought to enforce structure and compression to languages, since a language must be used and learned by all individuals of the culture in which it resides and at the same time be suitable for a variety of tasks. Kirby et al. 2015 define those two pressures as compressibility and expressivity and find that structure arises from the trade-off between these pressures in generated languages. The importance of cultural evolution for the emergence of structure is supported by a number of artificial language learning studies (e.g. Saldana et al., 2018) and computational studies using the Iterated Learning paradigm, in which agents learn a language by observing the output produced by another agent from the previous 'generation' (Kalish, Griffiths, and Lewandowsky 2007; Kirby, Cornish, and Smith 2008; Kirby et al. 2015). An alternative way of imposing cultural pressures on agents, is by simulating a large population of them and pairing agents randomly to solve a communicative game (Cogswell et al. 2019). This approach is more naturally aligned with cultural pressures in humans (see e.g. Wray and Grace, 2007) and is the one we use in this paper.

Genetic Evolution While there is much controversy about the selection pressures under which the fundamental traits underlying the human ability to learn and use language evolved in other humans, that genetic evolution played an essential role in endowing humans with the capabilities to learn and use language is generally undebated. Pre-modern humans, for instance, did not have the ability to speak or understand complex structures (Fitch 2010).

There are several approaches to simulate genetic evolution of neural network agents. Neural Architectural Search (NAS) focuses on searching the architecture space of the networks, unlike many traditional evolutionary techniques which often include parameter weights in their search space. Some of the earlier techniques such as NEAT gained considerable traction as a sound way of doing topology search using biologically inspired concepts (Stanley and Miikkulainen 2002). NAS methods however have mostly reverted to optimising solely the neural architecture and using gradient based methods such as SGD for weight optimisation due to the large parameter space of modern architectures (see, e.g., Elsken, Metzen, and Hutter, for a survey).

More recently, state-of-the-art one-shot search techniques such as ENAS (Efficient Neural Architecture Search) and DARTS (Differentiable Architecture Search) have allowed to bring a gradient-based approach to NAS through the use of intelligent weight-sharing schemes (Liu, Simonyan, and Yang 2018; Pham et al. 2018). In this work, we use the DARTS search space, which is constrained but still obtained state-of-the-art performance on benchmark natural language tasks (Li and Talwalkar 2019).

Approach

Sender/Receiver communication

We study language emergence in a referential game inspired by the signalling games proposed by Lewis (1969). In this game, one agent (called the *sender*) observes an image and generates a discrete message. The other agent, the *receiver* of the message, uses the message to select the right image from a set of images containing both the sender image and several distractor images. Since the information shown to the sender agent is crucial to the receivers success, this setup urges the two agents to come up with a communication protocol that conveys the right information.

Formally, our *referential* game is similar to Havrylov and Titov (2017):

1. The meaning space of the game consists of a collection D of K images $\{d_0, d_1, ..., d_K\}$, represented by z-dimensional feature vectors.

2. In each round i of the game, a target item d_i is randomly sampled from D, along with a set C of n distractor items.

3. The sender agent s of the game, parametrised by a neural network, is given item d_i and generates a discrete message m_i from a vocabulary V. The message is capped to a max message length of L.

4. The receiver agent r, also parametrised by a neural network, receives message m_i and uses it to identify d_i in the union of d_i and C.

We use $z = 512$, and $n = 3$ and train agents with Gumbel-Softmax (Jang, Gu, and Poole 2017a) based on task-success.

Language Transmission Engine

We introduce both cultural and genetic evolution to this game through a process that we call the Language Transmission Engine (LTE), which is depicted in Figure 1.[2] Similar

---

to Cogswell et al. (2019), we create a population of communicating agents. In every training iteration, two random agents are sampled to play the game. This forces the agents to adopt a simpler language naturally: to succeed they must be able to communicate or understand all opposing agents. In our setup, agents are either sender or receiver, they do not switch roles during their lifetime.
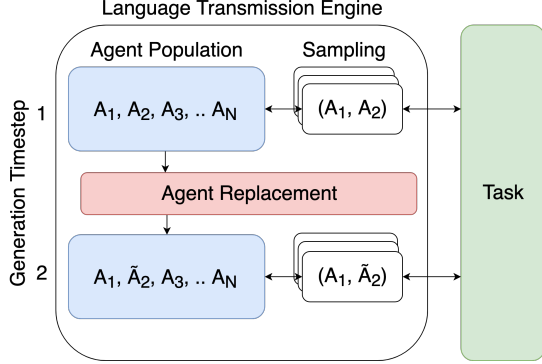

Figure 1: The Language Transmission Engine: Agent pairs are randomly sampled from each population and trained. After l training steps, a portion α of the population is culled.

Cultural evolution To model cultural evolution in the LTE, we periodically replace agents in the population with newly initialised agents. Cultural evolution is implicitly modelled in this setup, as new agents have to learn to communicate with agents that already master the task. Following Cogswell et al., we experiment with three different methods to select the agents that are replaced: randomly (no selection pressure), replacing the oldest agents or replacing the agents with the lowest fitness (as defined in Section). We call these setups `cu-random`, `cu-age` and `cu-best`, respectively.

Genetic evolution To model genetic evolution, rather than periodically replacing agents with randomly initialised new agents, we instead mutate the most successful agents and replace the worst agents with variations of the best agents, as outlined in Section . Note that cultural evolution is still implicitly modelled in this setup, as new agents still have to learn to communicate with older agents. Therefore, we call this setup with the term `co-evolution`.

Culling We refer to the selection process and subsequent mutation or re-initialisation step as *culling*. In biology, culling is the process of artificially removing organisms from a group to promote certain characteristics, so, in this case, culling consists of removing a subset of the worst agents and replacing them with variations of the best architecture. The proportion of agents from each population selected to be mutated is determined by the culling rate α, where $\alpha \in [0, 1)$. The culling interval l defines the number of iterations between culling steps. A formalisation of the LTE can be found in appendix .

Mutation Algorithm We base potential mutations on the RNN cell search space DARTS, defined by Liu, Simonyan, and Yang (2018). This space includes recurrent cells with up to N nodes, where each node $n_1, n_2, ..., n_N$ can take the output of any preceding nodes including n_0, which represents the cell's input. All potential connections are modulated by an activation function, which can be the identity function, Tanh, Sigmoid or ReLU. Following Liu, Simonyan, and Yang (2018) and Pham et al. (2018), we enhance each operation with a highway bypass (Zilly et al. 2016) and the average of all intermediate nodes is treated as the cell output.

To sample the initial model, we sample a random cell with a single node ($N = 1$). As this node must necessarily be connected to the input, the only variation stems from the possible activation functions applied to the output of n_1, resulting in four possible starting configurations. We set a node cap of $N = 8$. We mutate cells by randomly sampling an architecture which is one edit step away from the previous architecture. Edit steps are uniformly sampled from i) changing an incoming connection, ii) changing an output operation or iii) adding a new node; the mutation location is uniformly sampled from all possible mutations.[3] Note that while we use the DARTS search space to define potential mutations, contrary to Liu, Simonyan, and Yang, we do not use differentiation to sample new architectures based on a selection criterion.

Fitness Measure

The fitness criterion that we use in both the `cu-best` and `co-evolution` setup is based on task performance. However, rather than considering agents' performance right before the culling step, we consider the age of the youngest agent in the population (defined in terms of the number of batches that it was trained on) and for every agent compute their performance up until *when they had that age*. For any agent a_j in population **A** this is defined as:

$$\text{fitness}(a_j) = \frac{1}{\mathcal{T}_A} \sum_{t=0}^{\mathcal{T}_A} \mathcal{L}(a_j^t) \tag{1}$$

where $\mathcal{T}_A = \min_{a \in A} \mathcal{T}(a)$ is the age $\mathcal{T}(a)$ of the youngest agent in the population, and $\mathcal{L}(a_j^t)$ is the loss of agent a_j at time step t. This fitness criterion is not biased towards older agents, that have seen already more data and have simply converged more. It is thus not only considering task performance but also the *speed* at which this performance is reached.

Experiments

We test the LTE framework on a compositionally defined image dataset, using a range of different selection mechanisms.

---

[3]For a formal description of the mutation process, we refer to Appendix .

Dataset

In all our experiments, we use a modified version of the Shapes dataset (Andreas et al. 2015), which consists of 30 by 30 pixel images of 2D objects, characterised by shape (circle, square, triangle), colour (red, green, blue), and size (small, big). While every image has a unique symbolic description – consisting of the shape, colour and size of the object and its horizontal and vertical position in a 3x3 grid – one symbolic representation maps to multiple images, that differ in terms of exact pixels and object location. We use 80k, 8k, 40k images for train, validation and test sets, respectively. Some example images are depicted in Figure 2.
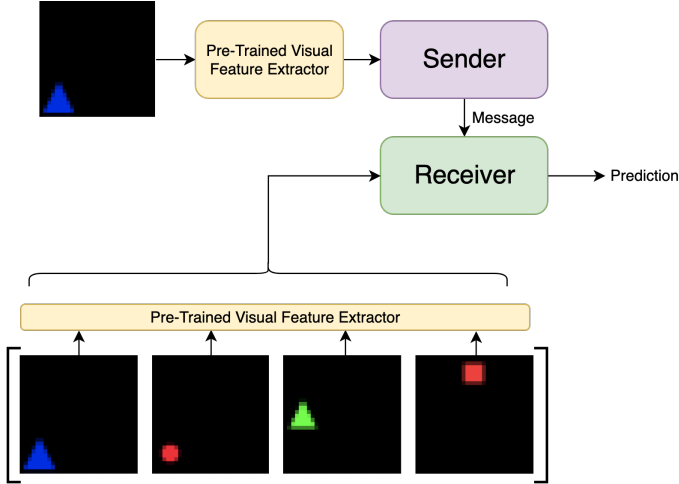

Figure 2: The modified Shapes task consists of showing an image the sender, and then letting the receiver deduce from the sender's message which image out of the target and k distractors is the correct one.

We pre-train a CNN feature extractor for the images in a two-agent setting of the task (see Appendix for more details).

Architecture and Training

For our co-evolution experiments, we use the DARTS search space as described above. For all cultural evolution approaches, we use an LSTM (Hochreiter and Schmidhuber 1997) for both the sender and receiver architecture (see Appendix for more details). Unless otherwise specified, we use the same sizes and hyper-parameters for all models. The sender and receiver models have a hidden size of 64 for the recurrent layer and an embedding layer of size 64. Further, we use a vocabulary size V of 4, with an additional bound token serving as the indicator for beginning and end-of-sequence. We limit the maximum length of a sentence L to 5.

We back-propagate gradients through the discrete step outputs (message) of the sender by using the Straight-Through (ST) Gumbel-Softmax Estimator (Jang, Gu, and Poole 2017b). We run all experiments with a fixed temperature $\tau = 1.2$. We use the default Pytorch (Paszke et al. 2017) Adam (Kingma and Ba 2015) optimiser with a learning rate of 0.001 and a batch-size of 1024. Note that the optimiser is reset for every batch.

We use a population size of 16 senders and 16 receivers for all multi-agent experiments. The culling rate α is set to 0.25 or four agents, and we cull (re-initialise or mutate) every $l = 5k$ iterations (batch). We run the experiments for a total of $I = 500k$ iterations, and evaluate the populations before each culling step.

Evaluation

We use a range of metrics to evaluate both the population of agents and the emerged languages.

Jaccard Similarity We measure the consistency of the emerged languages throughout the population using *Jaccard Similarity*, which is defined as the ratio between the size of the intersection and the union of two sets. We sample 200 messages per input image for each possible sender-receiver pair and average the Jaccard Similarity of the samples over the population. A high Jaccard Similarity between two messages is an indication that the same tokens are used in both messages.

Proportion of Unique Matches We compute how similar the messages that different agents emit for the same inputs by looking at all possible (sender, message) pairs for one input and assess whether they are the same. This metric is 1 when all agents always emit the same messages for the same inputs.

Number of Unique Messages We compute the average *number of unique messages* generated by each sender in the population. An intuitive reference point for this metric is the number of images with distinct symbolic representations. If agents generate more messages than expected by this reference point, this demonstrates that they use multiple messages for the images that are – from a task perspective – identical. A smaller number of unique messages, on the other hand, indicates that the agent is using a simpler language which is underspecified compared to the symbolic description of the image. Higher numbers of unique messages

Topographic Similarity Topographic similarity, used in a similar context by Lazaridou et al. (2018), represents the similarity between the meaning space (defined by the symbolic representations) and the signal space (the messages sent by an agent). It is defined as the correlation between the distances between pairs in meaning space and the distances between the corresponding messages in the signal space. We compute the topographic similarity for an agent by sampling 5,000 pairs of symbolic inputs and corresponding messages and compute the Pearson's ρ correlation between the cosine similarity of the one-hot encoded symbolic input pairs and the cosine similarity of the one-hot encoded message pairs.

Average Population Convergence To estimate the speed of learning of the agents in the population, estimate the *average population convergence*. For each agent, at each point in time, this is defined as the agent's average performance from the time it was born until it had the age of the current youngest agent in the population (analogous to the fitness criterion defined in Section). To get the average population convergence, we take we average those values for all agents in the population.

Average Agent Entropy We compute the average certainty of sender agents in their generation process by computing and averaging their *entropy* during generation.

Results

We now present a detailed comparison of our cultural and co-evolution setups. For each approach, we average over four random seeds and display the standard deviation error in plots. To analyse the evolution of both agents and languages, we consider the development of all previously outlined metrics over time. We then test the most successful converged languages and architectures in a single sender-receiver setup, to assess the impact of cultural and genetic evolution more independently. In these experiments, we compare also directly to a single sender-receiver baseline, which is impossible for most of the metrics we consider in this paper. Finally, we briefly consider the emerged architectures from a qualitative perspective.

Task performance

We first confirm that all setups converge to a solution to the task. As can be seen in Figure 3, all populations converge to an almost perfect solution to the game. The `cu-age` approach slightly outperforms the other approaches, with an accuracy that surpasses the 95% accuracy mark. Note that, due to the ever-changing population, the accuracy at any point in time is an average of both 'children' and 'adults', that communicate with different members of the population.

Figure 3: Average Population Accuracy of final populations.

Agent behaviour

To assess the behaviour of the agents over time, we monitor their average message entropy convergence speed. As can be seen in Figure 4, the `co-evolution` setup results in the lowest average entropy scores, the messages that they assign

to one particular image will thus have lower variation than in the other setups. Of the cultural evolution setups, the lowest entropy score is achieved in the `cu-best` setup.
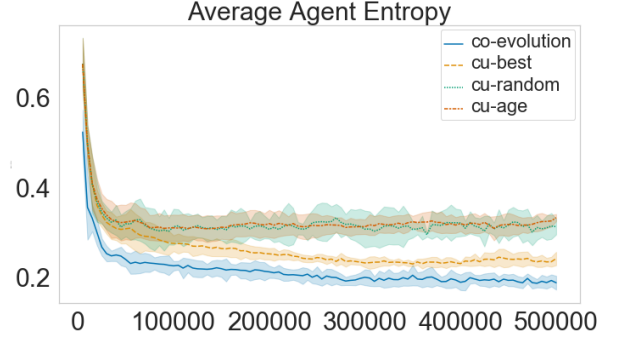

Figure 4: Average Agent Entropy over time.

Figure 5 shows the average population convergence over time. We again observe a clear difference between cultural evolution only and co-evolution, with an immediately much lower convergence time for co-evolution and a slightly downward trending curve.
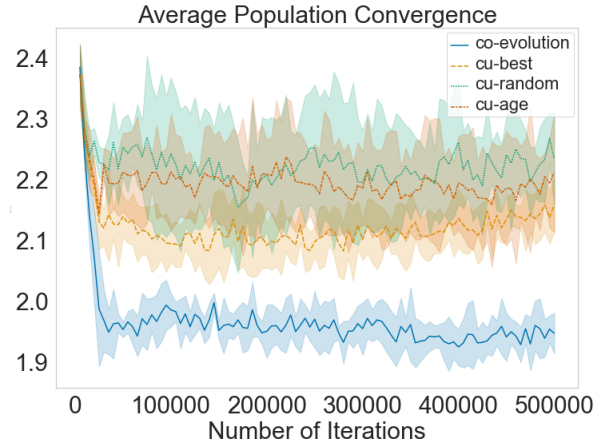

Figure 5: Average convergence for all cultural transmission modes and evolution.

Language Analysis

To check the consistencies of languages within a population, we compare the Jaccard Similarity and the Average Proportion of Unique Matches, which we plot in Figure 6. This shows that, compared to cultural evolution only, not only are the messages in co-evolution more similar across agents (higher Jaccard Similarity), but also that agents are considerably more aligned for the same inputs (less unique matches).

To assess the level of structure of the emerged languages, we plot the average Topographic Similarity and the Average Number of Unique Messages generated by all senders (Figure 7). The co-evolution condition again outperforms all cultural only conditions, with a simpler language (the number of the unique messages closer to the symbolic reference
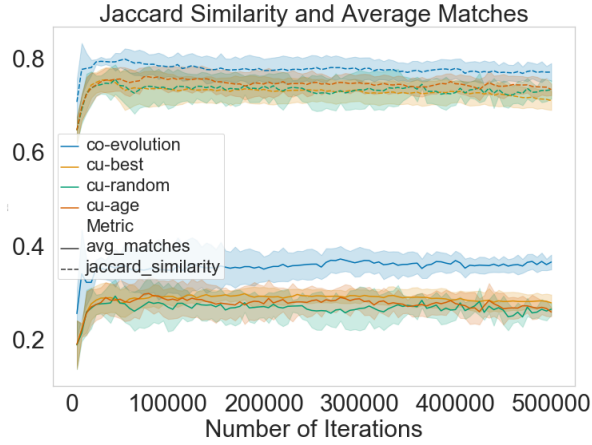
Figure 6: Average Jaccard Similarity and proportion of message matches for all cultural transmission modes and evolution
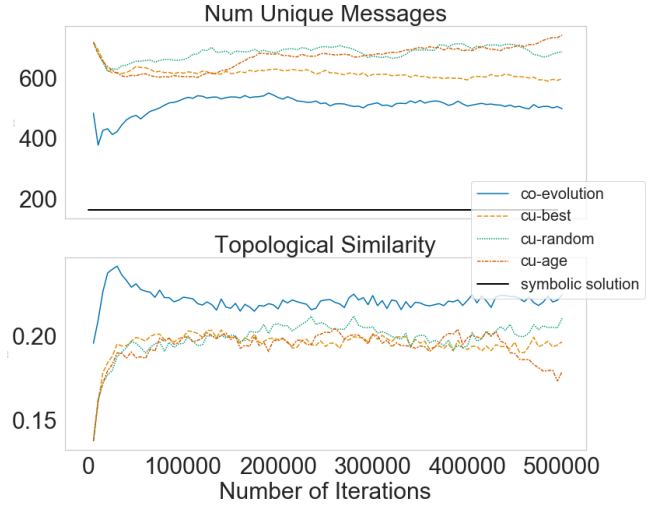

Figure 7: Average Number of Unique Messages and Topographic Similarity for all cultural evolution modes and co-evolution. For comparison, we also plot the number of unique messages for a symbolic solution that fully encodes all relevant features of the image (since we have three possible shapes and colours, two possible sizes, and a 3×3 grid of possible positions, this symbolic reference solution has $3 \times 3 \times 2 \times 9 = 162$ distinct messages.

point) that is structurally more similar to the symbolic representation of the input (higher Topographical Similarity).

Architecture Analysis

In Figure 8 we show the co-evolution of an agent and a sample of its language during three selected iterations in the co-evolution setup.

Strikingly, the best sender architecture does not evolve from its original form, which could point towards the limitations of our search strategy and space. On the contrary, the receiver goes through quite some evolution steps and converges into a significantly more complex architecture than its original form. While this is a single instance of the evolution mechanism, we find the preference for simpler senders and complex receivers consistent for different seeds. The total number of seen receiver architectures during training was 264, while only 50 for the sender architectures. Throughout training, twenty different sender architectures were selected as the best, whereas only five were ever selected for the sender. A wider exploration of the search space by receivers suggests that more architectural complexity is preferred in the receiver role compared to the sender role.

Qualitatively, we also observe a unification of language throughout evolution in Figure 8, which is also supported by Figure 7. The population of senders starts with eleven different unique messages and ends with only two to describe the same input image. We will leave a more detailed analysis of the evolved architectures for future work.

Frozen Experiments

With a series of experiments we test the a priori suitability of the evolved languages and agents for the task at hand, by monitoring the accuracy of new agents that are paired with converged agents and train them from scratch.

We focus, in particular, on training receivers with a frozen sender from different setups, which allows us to assess 1) whether cultural evolution made languages evolve to be more easily picked up by new agents 2) whether the genetic

evolution made architectures converge more quickly when faced with this task. We compare the accuracy development of:

- An LSTM receiver trained with a frozen sender taken from `cu-best`;

- An evolved receiver trained with a frozen evolved sender.

For both these experiments, we compare with two baselines:

- The performance of a receiver agent trained from scratch along with a receiver agent that has either the `cu` architecture or the evolved `co` architecture (`cu-baseline` and `co-baseline`, respectively);

- The performance of an agent trained with an agent that is *pretrained* in the single agent setup, with either the `cu` architecture or an evolved architecture (`cu-baseline-pretrained` and `co-baseline-pretrained`).

Each experiment is run 10 times, keeping the same frozen agent. The results confirm cultural evolution contributes to the learnability and suitability of emerging languages: the `cu-best` accuracy (green line) converges substantially quicker and is substantially higher than the `cu-baseline-pretrained` accuracy (orange line). Selective pressure on the language appears to be important: the resulting languages are only easier to learn in the `cu-best` setup.[4] They also show that the agents

---

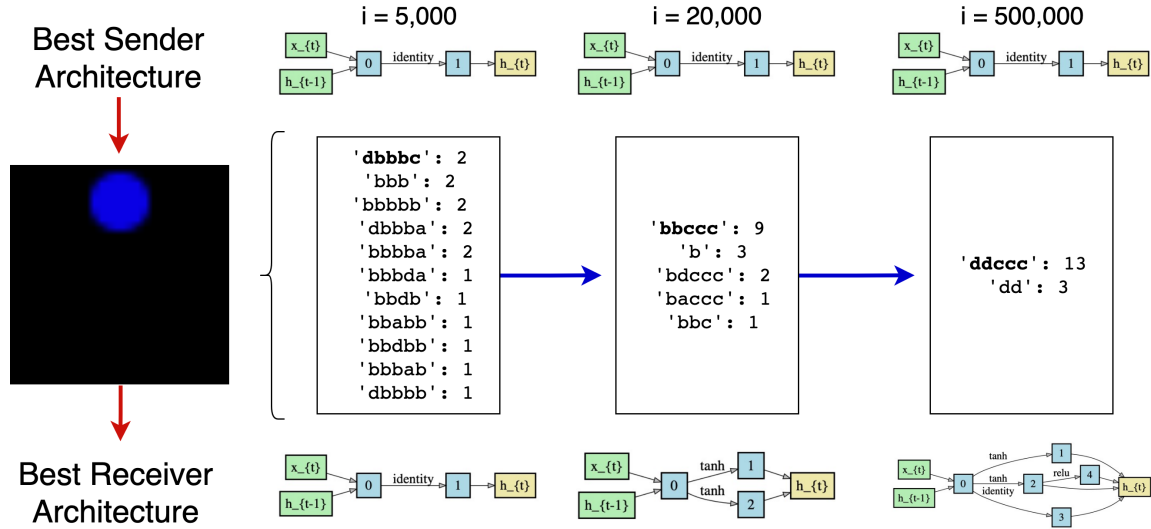[4] `cu-age` and `cu-random` are ommitted from the plot for clarity reasons.

Figure 8: Evolution of the best sender and receiver architecture according to convergence, and the evolution of the population's message description of the same input through iterations. The bold messages represent the message outputted by the best sender whose architecture is pictured above. The count of each message represents the number of agents in the population which uttered this exact sequence.

benefit also from the genetic evolution: the best accuracies are achieved in the co-evolution setup (red line). The difference between the `cu-baseline` (blue) and the `co-baseline` (brown) further shows that even if the evolved architectures are trained from scratch, they perform much better than a baseline model trained from scratch. The difference between the `co-baseline-pretrained` (only genetic evolution, purple line) and the co-evolution of agents and language line (red line) illustrates that genetic evolution alone is not enough: while a new evolved receiver certainly benefits from learning from a (from scratch) pretrained evolved sender, without the cultural transmission pressure, it's performance is still substantially below a receiver that learns from an evolved sender whose language was evolved as well.

Conclusion

In this paper, we introduced a language transmission bottleneck in a referential game, where new agents have to learn the language by playing with more experienced agents. To overcome such bottleneck, we enabled both the cultural evolution of language and the genetic evolution of agents, using a new *Language Transmission Engine*. Using a battery of metrics, we monitored their respective impact on communication efficiency, degree of linguistic structure and intra-population language homogeneity. While we could find important differences in between cultural evolution strategies, it is when we included genetic evolution that agents scored best. In a second experiment, we paired new agents with evolved languages and agents and again confirmed that, while cultural evolution makes a language easier to learn, co-evolution leads to the best communication.

In future research, we would like to apply the Language Transmission Engine on new, more complex tasks
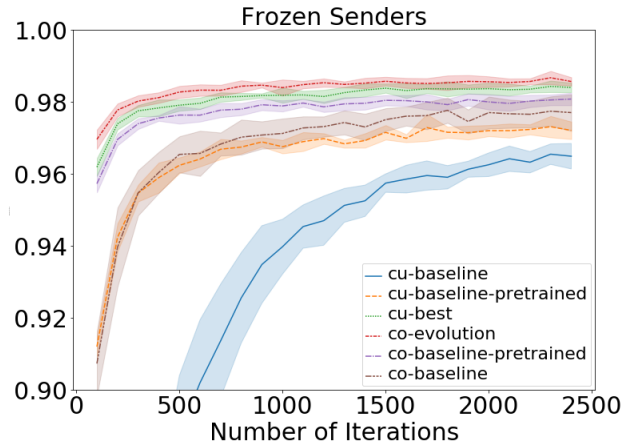

Figure 9: Receiver accuracies trained with different types of frozen senders.

and further increase our understanding of the properties of the emerged languages and architectures. Additionally, we would like to investigate other neuro-evolution techniques and apply them to different search spaces.

References

Andreas, J.; Rohrbach, M.; Darrell, T.; and Klein, D. 2015. Deep Compositional Question Answering with Neural Module Networks.

Bouchacourt, D., and Baroni, M. 2018. How agents see things: On visual representations in an emergent language game. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, 981–985.

Choi, E.; Lazaridou, A.; and de Freitas, N. 2018. Compositional obverter communication learning from raw visual input. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Christiansen, M. H., and Kirby, S. 2003. Language evolution: consensus and controversies. *Trends in Cognitive Sciences* 7(7):300–307.

Cogswell, M.; Lu, J.; Lee, S.; Parikh, D.; and Batra, D. 2019. Emergence of compositional language with deep generational transmission. *arXiv preprint arXiv:1904.09067*.

Elsken, T.; Metzen, J. H.; and Hutter, F. 2019. Neural architecture search: A survey. *Journal of Machine Learning Research* 20:55:1–55:21.

Evtimova, K.; Drozdov, A.; Kiela, D.; and Cho, K. 2018. Emergent communication in a multi-modal, multi-step referential game. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Fitch, W. T. 2010. *The evolution of language*. Cambridge University Press.

Havrylov, S., and Titov, I. 2017. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In *Advances in neural information processing systems*, 2149–2159.

Hochreiter, S., and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation* 9(8):1735–1780.

Hurford, J. R. 1998. The evolution of language and languages. In *The evolution of culture*. Edinburgh University Press.

Jang, E.; Gu, S.; and Poole, B. 2017a. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Jang, E.; Gu, S.; and Poole, B. 2017b. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Kalish, M. L.; Griffiths, T. L.; and Lewandowsky, S. 2007. Iterated learning: Intergenerational knowledge transmission reveals inductive bias. *Psychonomic Bulletin & Review* 288–294.

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

Kirby, S., and Hurford, J. R. 2002. The Emergence of Linguistic Structure: An Overview of the Iterated Learning Model. In *Simulating the Evolution of Language*. London: Springer London. 121–147.

Kirby, S.; Tamariz, M.; Cornish, H.; and Smith, K. 2015. Compression and communication in the cultural evolution of linguistic structure. *Cognition* 141:87–102.

Kirby, S.; Cornish, H.; and Smith, K. 2008. Cumulative cultural evolution in the laboratory: an experimental approach to the origins of structure in human language. *Proceedings of the National Academy of Sciences of the United States of America* 105(31):10681–6.

Kirby, S. 2001. Spontaneous Evolution of Linguistic Structure-An Iterated Learning Model of the Emergence of Regularity and Irregularity. Technical Report 2.

Kottur, S.; Moura, J. M. F.; Lee, S.; and Batra, D. 2017. Natural language does not emerge 'naturally' in multi-agent dialog. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, 2962–2967.

Lazaridou, A.; Hermann, K. M.; Tuyls, K.; and Clark, S. 2018. Emergence of linguistic communication from referential games with symbolic and pixel input. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Lazaridou, A.; Peysakhovich, A.; and Baroni, M. 2017. Multi-agent cooperation and the emergence of (natural) language. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Lewis, D. 1969. *Convention: A Philosophical Study*. Wiley-Blackwell.

Li, L., and Talwalkar, A. 2019. Random search and reproducibility for neural architecture search. *CoRR* abs/1902.07638.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. DARTS: differentiable architecture search. *CoRR* abs/1806.09055.

Mordatch, I., and Abbeel, P. 2018. Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 1495–1502.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Pham, H.; Guan, M. Y.; Zoph, B.; Le, Q. V.; and Dean, J. 2018. Efficient neural architecture search via parameter

sharing. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, 4092–4101.

Pinker, S., and Bloom, P. 1990. Natural language and natural selection. *Behavioral and Brain Sciences* 13(4):707–27.

Saldana, C.; Kirby, S.; Truswell, R.; and Smith, K. 2018. Compositional hierarchical structure evolves through cultural transmission: an experimental study.

Smith, K., and Kirby, S. 2012. *Compositionality and Linguistic Evolution*. Oxford University Press.

Stanley, K. O., and Miikkulainen, R. 2002. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation* 10(2):99–127.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, 3104–3112.

Wray, A., and Grace, G. W. 2007. The consequences of talking to strangers: Evolutionary corollaries of socio-cultural influences on linguistic form. *Lingua* 117(3):543–578.

Zilly, J. G.; Srivastava, R. K.; Koutník, J.; and Schmidhuber, J. 2016. Recurrent highway networks. *CoRR* abs/1607.03474.

Appendix

Language Transmission Engine

We formalise our Language Transmission process in the pseudo code shown in Algorithm 1. We select hyper-parameters l as the number of iterations or batches shown between culling steps, and I as the total number of iterations.

Algorithm 1 Language Transmission Engine

$S \leftarrow \{s_0, s_1..., s_N\}$
$R \leftarrow \{r_0, r_1..., r_N\}$
$i \leftarrow 1$
while $i \leq I$ **do**
    for batch b in D **do**
        Sample $\hat{s}$ from S
        Sample $\hat{r}$ from R
        train$(\hat{s}, \hat{r}, b)$
        if $i \bmod l = 0$ **then**
            cull(S, R)
        end if
        $i \leftarrow i + 1$
    end for
end while

Mutation Algorithms Pseudo-code

Algorithm 2 Genotype-level Mutation
---

procedure **mg**$(genotype)$
    $g \leftarrow copy(genotype)$
    $a \leftarrow \mathcal{U}(1, 3)$
    $n \leftarrow \mathcal{U}(1, len(g))$
    if $a = 1$ **then**
        $p \leftarrow \mathcal{U}[ReLU, I, \tanh, \sigma]$
        $n.activation \leftarrow p$
    end if
    if $a = 2$ **then**
        $r \leftarrow \mathcal{U}(1, n)$
        $n.connection \leftarrow r$
    end if
    if $a = 3$ **then**
        $n' \leftarrow new\_node()$
        $p \leftarrow \mathcal{U}[ReLU, I, \tanh, \sigma]$
        $r \leftarrow \mathcal{U}(1, len(g))$
        $n'.activation \leftarrow p$
        $n'.connection \leftarrow r$
        $g.append(n')$
    end if
    return g
end procedure

The genotype mutation is described in pseudo-code by algorithm 2, and takes as input a genotype containing nodes describing the cell. The genotype is mutated by either changing the input connection or primitive (output activation function) for a randomly sampled node n, or adding a new node

altogether. See section for explanations on the workings of the DARTS cell structure.

Algorithm 3 Population-level Mutation
---

procedure **mutate**$(\mathbf{P})$
    $p' \leftarrow \arg\min_{convergence}(P)$
    $\mathbf{p} \leftarrow \pi(P)$
    for p_i in $\mathbf{p}$ **do**
        $p_i.genotype \leftarrow \mathbf{mg}(p'.genotype)$
    end for
end procedure

In order to mutate a population $\mathbf{P}$ using π as a replacement policy, we use the process outlined in algorithm 3.

Agent Architecture

Sender Architecture The sender architecture comprises of a linear layer input mapping the input feature size (512) to the hidden size. The image feature vector is therefore mapped to the same dimension as the RNN layer, where it is used as the initial hidden state. When training, for each step of the sender RNN we apply the cell and use the straight-through Gumbel-Softmax trick to be able back-propagate gradients through the discrete message output. During evaluation however, we sample the categorical distribution at each step to produce each token in the sentence.

Receiver Architecture The receiver architecture is simpler and takes as an input the message outputted by the sender and outputs a vector of input feature size (512). A single embedding matrix is used to encode the sender's message. During training the message is linearly transformed using the embedding matrix, while during the evaluation pass the discrete message outputs of the sender are used to map to the specific embedding dimensions. The embedded message is then passed to the RNN layer, and the final state of the RNN is linearly mapped back to the feature size. Doing so allows us to obtain a prediction for each image feature (distractors and true image), by comparing the alignment between the receiver output and the respective feature vectors.

Feature Extraction

In order to obtain image features, we pre-trained a convolutional model on the task using the raw image as input. Due to the input size requirements of the convolutional model, we resize the images linearly to be 128 by 128 (height, width) by 3 (RGB channels). We used early stopping conditions on the validation accuracy, an embedding size of 256, and hidden size of 512. The two agents are otherwise trained with the same parameters as other experiments: vocab size and max sentence length of 5, Adam optimizer with learning rate of 0.001.

For the visual module itself, we used a similar architecture to that in Choi, Lazaridou, and de Freitas (2018) albeit smaller. We used a five-layer convolution network with 20 filters, and a kernel size and stride of 3 for all layers. For every convolutional layer, ReLU activation was applied on

the output, after a Batch normalization step with no bias parameter. The linear layer which followed the convolutional layers had output dimensions of 512 and a ReLU activation function. This allows us to obtain image features of size 512, which we then used for all experiments.
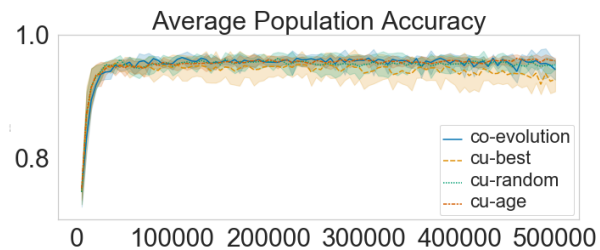
Additional Figures and Analysis

Figure 10: Average Population Accuracy for all Iterations