Towards Procedurally Generated Languages for Non-playable Characters in Video Games

Joshua Sirota
Computing Science
University of Alberta
Edmonton, AB, Canada
sirota@ualberta.ca

Vadim Bulitko
Computing Science
University of Alberta
Edmonton, AB, Canada
bulitko@ualberta.ca

Matthew R. G. Brown
Psychiatry
University of Alberta
Edmonton, AB, Canada
mbrown2@ualberta.ca

Sergio Poo Hernandez
Computing Science
University of Alberta
Edmonton, AB, Canada
pooherna@ualberta.ca

***Abstract*—Non-playable characters (NPCs) enhance a player's immersion in a video game. Communications among NPCs create atmosphere and, in some games, are a core element of the gameplay. Yet, the majority of games manually script inter-NPC communications creating only an illusion of such exchanges. Doing so is laborious and results in fixed interactions which may not respond to the player's actions or changes in the environment. In this paper we propose procedural content generation (PCG) for emergent inter-NPC languages. Indeed, recent research demonstrated that deep neural networks can be trained to develop an artificial language to communicate with each other. The work used a fixed, handcrafted network architecture identical for both the sending and receiving agents. We extend the work by using neuroevolution for both the sender and the receiver. In doing so we evolve both the architecture and weights of the agents and show that they successfully develop novel languages to communicate among themselves.**

***Index Terms*—Linguistics, Language and Speech; Neural networks/Deep learning; Evolution Strategies; Machine learning**

I. Introduction

Simulation games [1], [2], as well some adventure games [3] involve procedural generation of populations of agents. These agents should be interesting, surprising, and responsive to their environment. That is, the physical and behavioural characteristics of these agents should emerge as a response to their physical and social environment, as this adds consistency and realism to the simulation while increasing the player's sense that their decisions are actually consequential.

Despite the procedural generation of numerous aspects of NPC characteristics in simulation games, language and communication tends to be either scripted [4] or something that agents only pretend to engage in but which is ultimately inconsequential [5]. As such, this scripted or faked inter-NPC communication is a departure from the other desirable emergent characteristics of procedurally generated NPCs. Emergent communication (EC) [6] provides a possible means of solving this problem. With EC, NPCs could develop their own real languages and use these languages to cooperate and communicate. Emergent languages have been shown to vary greatly when the topics of communication change, demonstrating responsiveness even to subtle changes in the environment in which they are developed [6]. As such, EC provides an exciting avenue to procedurally generate NPC languages which have desirable emergent properties, adding depth and richness to game worlds.

Currently, emergent languages are simple, unpredictable, not interpretable by humans, and lack natural language properties (e.g., syntax, grammar). While this may be sufficient for certain games which embrace unpredictability and unusual outcomes [7], it makes EC in its current form a poor choice for AAA titles. This is because these titles require increased control to deliver an experience that is guaranteed to be high quality. Continued progress in EC, though, may be able to tame this unpredictability and create natural emergent languages which can be understood by humans. So, while the work in this paper may only be relevant to certain niche game types, we believe that this line of research could lead to results which may be incredibly useful for a wide variety of game types.

Contemporary work in EC has used reinforcement learning (RL) to train agents. Recent work in evolution strategies (ES) [8], [9], though, has shown that ES can be competitive or even preferable for certain tasks when compared to RL. We believe that ES has many attractive properties for inter-NPC emergent communication. Evolution strategies are comparatively simple and easily implemented, making them desirable for application in industry. They have been shown to outperform RL on tasks which involve noisy or sparse rewards [10],

978-1-7281-1884-0/19/$31.00

which has been shown to be a problem with certain EC tasks [11]. They are easily parallelizable and scale naturally to populations of various sizes. ES requires no differentiable model of the environment. In addition, ES can be combined with RL when training deep neural networks and combining the two has in some cases performed better than using either methodology on its own [8]. As such, there is clear value in exploring the applicability of ES to emergent communication in addition to reinforcement learning.

We propose to procedurally generate NPCs implemented as deep neural networks that actually communicate with each other using their own language. In this paper we take a first step towards this task by extending existing recent work [6] with neuroevolution [8], [9], which is a class of ES algorithms tailored to evolving neural networks. This neuroevolution is twofold: we evolve the weights of communicating agents, as well as their architecture. Evolving the architecture of communicating agents has two benefits: (i) evolution of architecture further reduces handcrafting of agents, incorporating increased procedural generation of these agents and (ii) architectural evolution has been shown to improve performance when compared to fixed architecture [12], so this allows us to explore whether or not architectural evolution is useful for emergent communication.

II. Problem Formulation

In this paper we address the problem of training agents to develop their own languages so as to cooperate with each other. In line with previous work [6], we formalize the problem as a repeated referential game played by agents. In these games a speaker agent A_S is shown some target object. A listener agent A_L is shown the target and some distracting objects. The speaker then sends messages to the listener which the listener uses to attempt to choose the target object rather than a distracting object.

Our objective in this paper is to maximize the referential game success rate of agent pairs when playing the referential game where the objects being referred to are previously unseen. For a repeated referential game over T rounds, we define the success rate of an agent pair (A_S, A_L) as the proportion of rounds in which A_L correctly chooses the target object. While the behaviour of these layers changed as agents learned, each layer had a predefined function and use.

III. Related Work

Recent work showed that it is possible to train agents to develop language from scratch so as to succeed at a referential game [6], [13]. The agents played the referential game with both human-annotated and pixel data. The agents learned with deep reinforcement learning and had *a priori* handcrafted architectures.

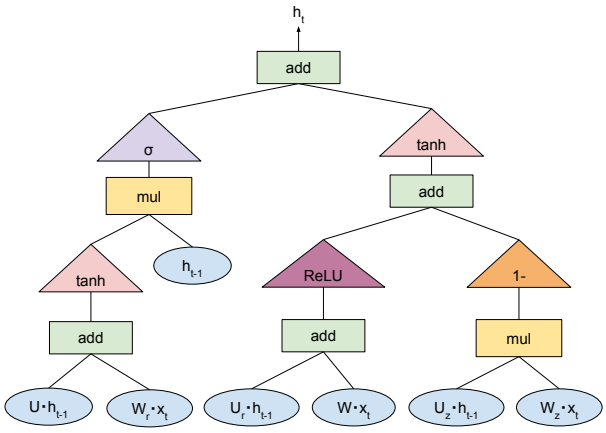

Fig. 1. Example of a possible initial network architecture when using random architecture.

Emergent communication has been utilized to facilitate cooperation between agents in multi-agent environments [11]. These agents had fixed, handcrafted architecture, and were provided with extra information about other agents in their environment during training (e.g., rewards, observations). Our agents have no information about each other beyond messages received and communicative success.

Early work in emergent communication [15] trained robots to play a referential game using physical objects. Agents had various pre-defined cognitive layers (e.g., sensory, conceptual) to learn representations and communicate. Our agents have no such pre-defined layers. All cognitive functionality was learned from scratch.

IV. Proposed Approach

Speakers A_S and listeners A_L are both recurrent neural networks (RNNs). While we periodically mutate the architecture of A_S and A_L, an initial architecture for the first generation is needed. We test two initial RNN architectures: a gated recurrent unit (GRU) [16], and an RNN with partially randomized architecture. Figure 1 shows an example of a randomized RNN in tree form - the structure of any other randomized initial RNN will be the same but the orders of inputs, as well as the various operators, will be randomized. We selected these initial architectures to test, respectively, whether neuroevolution could improve the performance of a well

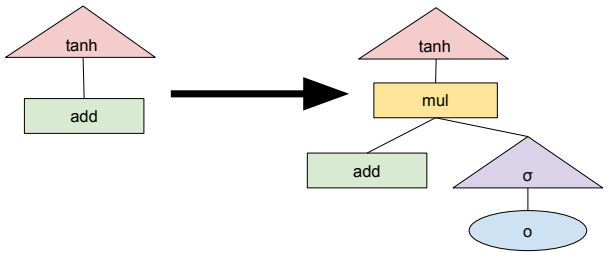

Fig. 2. Insertion mutation mechanism. Example of a multiply node being added as a parent to the add node. A second child is added as part of the addition.

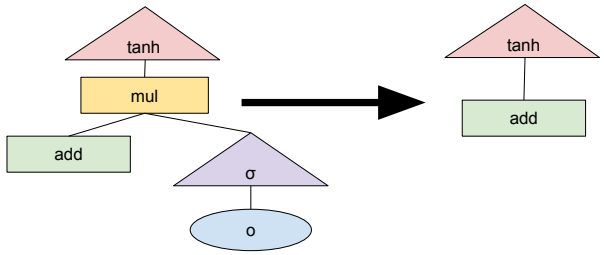

Fig. 3. Shrinking mutation mechanism. Example of a multiply node being removed, along with one of its child subtrees.

known RNN architecture, and whether an RNN with no specific engineering could be evolved to communicate effectively. These RNNs are implemented as program trees (Figure 1) to facililitate architectural mutation.

Agents A_S and A_L are each equipped with their own feedforward neural networks F_S and F_L, respectively. These feedforward networks are used to preprocess target and distractor objects before they are inputted to their respective RNN. The networks F_S and F_L both have a single hidden layer and use sigmoid activation.

In this paper we use evolutionary search [8], [9] to generate the architecture of A_S and A_L as well as update the weights of A_S, A_L, F_S, and F_L. For this evolutionary search, populations of speaker-listener pairs (A_S, A_L) are evaluated on their performance at a repeated referential game. At each generation, the best performing pairs of that population reproduce asexually to create the next generation. For a given agent pair (A_S, A_L), reproduction consists of two parts: (i) the weights of A_S, A_L, F_S, and F_L are mutated with Gaussian noise, (ii) the architecture of A_S and A_L is mutated. Possible architectural mutations are (i) operator swapping (e.g., changing one of the RNN's addition gates to a multiplication gate), (ii) operator insertion (e.g., inserting a tanh gate or a multiplication gate (Figure 2)), (iii) operator removal (e.g., removing an addition gate and replacing it with one of its arguments (Figure 3), or removing a sigmoid gate).

V. Empirical Evaluation

We used the Visual Attributes for Concepts Dataset [17]. This dataset contains 500 concepts (e.g., dog) in 16 categories (e.g., animals). For each concept, there are human-generated annotations describing its characteristics (e.g., has_fur).

We compared the random RNN and GRU initial architectures, each with and without architectural mutation. This resulted in four experimental conditions. We call the two conditions involving a random RNN **RT+E** and **RT**, denoting respectively the condition for which architectures are evolved and the condition for which they are not. Similarly, we call the GRU conditions **GRU+E** and **GRU**.

To evaluate our experimental conditions we randomly partitioned the dataset into a training set, validation set, and test set. For each experimental condition we executed an evolution run with this partitioning for a total of 4 runs, each executing for 420 generations. For a given run, each generation, all agent pairs played the referential game with concepts chosen from the training set. The top agents of a given generation, in addition to reproducing, were chosen to play the referential game again using objects from the validation set. At the end of a given evolution run, the agent pair which performed best on the validation set was chosen to play the referential game with the test set. The performance measure by which we compared the four experimental conditions was the test set performance of the agent pair chosen from that condition's evolution run.

TABLE I
Success rates of each experimental condition on the referential game with objects drawn from the test set.

Success rate	Experimental condition			
	RT+E	**RT**	**GRU+E**	**GRU**
Test	38.4%	20.5%	**44.8%**	44.0%

Results

The set with which we tested each run's best-performing agent pair consisted of 8000 samples, where each sample consisted of 5 objects drawn from the test set: one target and four distractors. A success rate of 50%, for example, denotes that for agent pair (A_S, A_L), A_L correctly guessed on 4000 of the 8000 test samples.

Table I shows the success rate of each experimental condition's chosen agent pair. Given that we used four distractors, if agents guessed randomly then their expected success rate would be 20% (the baseline). All experimental conditions other than **RT** achieved accuracy significantly higher than the baseline (one-tailed z-test, $p = 10^{-16}$ for all conditions other than **RT**). While the success rate of condition **GRU+E** was higher than that of **GRU**, this difference was not signficiant (two-tailed z-test, $p > 0.05$). Conditions **GRU** and **GRU+E** both performed significantly better than conditions **RT+E** and **RT** (two-tailed z-test, $p < 1 \times 10^{-12}$). Condition **RT+E** performed significantly better than condition **RT** (two-tailed z-test, $p < 1 \times 10^{-16}$)

VI. Conclusions & Future Work

As three of the four experimental conditions performed significantly above the baseline, we showed that evolutionary search over the parameters of two neural networks is a viable technique for emergent communication. Interestingly, condition **RT** was unable to learn the task, but the performance of **RT+E** demonstrates that the same initial architecture was able to perform well above baseline when its architecture was periodically mutated. This suggests that architectural evolution could improve the performance of deep neural networks at emergent communication tasks. As such, this suggests that NPCs which learn to communicate could be entirely procedurally generated, using architectural mutation to evolve their structure and a simple genetic algorithm to train them. Given sufficient advances in emergent communication, this technique could be used to automatically generate sophisticated NPCs which evolve coherent languages that are responsive to their surroundings.

The performance of our agents was significantly worse than that achieved with deep reinforcement learning [6], with our agents achieving a best success rate of only 44.8%, while prior work achieved a success rate of 74.2% on the same task. The work in this paper was preliminary, though, and intended to motivate continued exploration of evolution strategies for emergent communication. As the evolution algorithms used in this paper were quite simple, we believe that more sophisticated algorithms could help to bridge this gap.

Future work will evaluate our approach on multi-agent grid environments [11] such as artificial life (A-life) [18]. Evolving AI agents in an A-life environment can approximate generating NPCs procedurally in a video game [18]. In particular, it will be of interest to detect emergent languages and language-based cooperation automatically [19]. Future work will also investigate larger groups which can communicate with each other, rather than the communicating pairs in this paper. Finally, future work will extend our evolutionary search with sexual reproduction and speciation [8] in an attempt to evolve more complex agent designs that may need a longer time to mature [20].

VII. Acknowledgements

We appreciate support from the NSERC, Nvidia, and Compute Canada.

References

[1] Maxis, "Spore," 2008.
[2] Lionhead Studios, "Black and White," 2001.
[3] Hello Games, "No Man's Sky," 2016.
[4] BioWare, "Mass Effect," 2007.
[5] Maxis, "The Sims," 2000.
[6] A. Lazaridou, K. M. Hermann, K. Tuyls, and S. Clark, "Emergence of linguistic communication from referential games with symbolic and pixel input," *CoRR*, vol. abs/1804.03984, 2018.
[7] Tarn Adams, "Dwarf Fortress," 2006.
[8] R. Miikkulainen, J. Z. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat, "Evolving deep neural networks," *CoRR*, vol. abs/1703.00548, 2017.
[9] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," *CoRR*, vol. abs/1712.06567, 2017.
[10] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.
[11] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *CoRR*, vol. abs/1706.02275, 2017.
[12] A. Rawal and R. Miikkulainen, "From nodes to networks: Evolving recurrent neural networks," *CoRR*, vol. abs/1803.04439, 2018.
[13] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *NIPS*, 2016, pp. 2137–2145.
[14] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," *CoRR*, vol. abs/1703.04908, 2017.
[15] L. Steels, *The Talking Heads experiment: Origins of words and meanings*, ser. Computational Models of Language Evolution, 2015.
[16] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014.
[17] C. Silberer, V. Ferrari, and M. Lapata, "Models of semantic representation with visual attributes," in *In ACL (Volume 1: Long Papers)*, vol. 1, 2013, pp. 572–582.
[18] V. Bulitko, M. Walters, M. Cselinacz, and M. R. Brown, "Evolving NPC behaviours in A-life with player proxies," in *EXAG Workshop at AIIDE*, 2018.
[19] E. S. Soares, V. Bulitko, K. Doucet, M. Cselinacz, T. Soule, S. Heck, and L. Wright, "Learning to recognize A-Life behaviours," in *In Poster collection: ACS*, 2018.
[20] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.