**Can AI Learn to Lie?**

Exploring the emergence of unrewarded communication practices in a MARL system

Ap Research

Word Count: 5438

**Introduction:**

In recent years the study of reinforcement learning—multi-agent reinforcement learning in particular—has turned a great portion of its focus to emergent behaviors, which are behaviors that agents powered by neural networks learn to do without being taught. However, the majority of such behaviors studied are limited to either physical actions or basic interpersonal interactions; many papers study behaviors like learning to balance intuitively when only told to move from point A to B or using the environment to hide when told to avoid another agent, and though emergent communication on its face is not a path untrodden, the specific behavior of dishonesty is one which has not yet been examined. As AI is becoming a larger part of our lives—present in everything from Netflix’s recommendation algorithm to the creation of L’Oreal’s advertisements (Mari, 2019)—and given that humans lie between 1 and 4 times a day on average (Levine, 2013), if and how easily neural networks can learn to lie is and will be even more going forward a crucial consideration in the further incorporation of this technology into peoples’ lives. As of today, there have already been cases in natural language processing where agents have learned to “cheat” their reward functions to appear successful without learning how to do what they are being expected to do (Everitt, 2017); as neural networks enter the medical field in increasing numbers (Haleem, 2019) ambiguity in this regard is untenable. The goal of this paper is to establish an environment in which lying is advantageous in the long run for one of several agents and determining based on the data collected if there is indication that this behavior has emerged at the end of the training period, assuming successful training.

**Literature Review:**

Much like reinforcement learning itself, the study of emergent behavior in a multi-agent setting is a field seated firmly in the present day. Initial forays into this subject centered around noncooperative and noncommunicative tasks like server load balancing--the task of delegating computing tasks of various difficulties to a network of computers with the goal of minimizing the amount of time that a task is waiting in the queue without a free computer to delegate it to--(Schaerf, 1995), though were accompanied by and quickly progressed to both noncommunicative cooperative settings such as agents learning from observations of each other (Goldman, 1996; Yamaguchi, 1997) and communicative cooperative settings such as a solution to the symbiote problem (in which a static agent chooses from 8 signals to send to a moving agent to guide the agents toward each other) (Ono, 1995). These first steps were followed by exploration into the applicability of MARL (Multi-Agent Reinforcement Learning, the reinforcement learning where multiple agents are learning and interacting with the environment and/or each other at once) to problems with continuous rather than discrete action spaces, namely on the block pushing problem in which two agents alternate applying a force vector to a block to move it towards a goal (Ono, 1996). The feasibility of learning social behavior followed with work on the pursuit problem (where 4 hunter agents learn to coordinate to surround one prey) (Ono, 1996) and a medley of cooperative tasks executed on physical robots (communal resource gathering based on the dynamics of a hunter-forager society) (Mataric, 1997). Mataric’s method, as well as Goldman and Yamaguchi’s, though their papers did not focus on a cooperative goal, revolved around inter-agent observations and shared reward (a situation where though agents act separately, they all receive the same reward at each time step), which was standard at this stage. Later but still contemporary papers began to focus on learning organizational roles organically, which can be thought of as emergent job specialization (Prasad, 1998; Prasad; 1996). These papers all used either interaction via sensing (agents taking actions based on observing other agents) or interaction via the environment (interaction in the sense of a shared goal without any direct influence of one agent on another) as defined by Cao’s 1997 paper (Cao, 1997). Modern papers have concentrated more on the third category, interaction via communication, where discrete messages are generated and transmitted by and between agents; one team demonstrated that opening a communication channel improved performance at the negotiation game--a game where two players negotiate to divide a set of items with the goal of meeting the randomized goal set at the start of the game for each player--(Cao, 2018) while another learned communication techniques for the prisoner switch riddle--a riddle where each day 1/100 prisoners is selected randomly with replacement (and in secret from the other prisoners) and can toggle the state of a light bulb or declare that every prisoner has been chosen at least once; if they choose the latter and are correct they all go free and if they are wrong all are executed--using gradient passing (Foerester, 2016). Other papers have focused on emergent interaction via sensing in a complex 3D environment (Bansal, 2018) involving real-time object manipulation, for example via soccer (Liu, 2019) and hide and seek (Baker, 2020). Many of these papers remained in the realm of interaction via sensing or made use of gradient sharing methods (Hausknecht, 2016; Sukhbaatar, 2016; Mordatch, 2017; Wai, 2018), but more recent advances in understanding why non-differentiable methods do not perform (Lowe, 2019) have resulted in the creation of algorithms that rectify these performance issues (Eccles, 2019).

This research aims to address the emergence of inter-agent relationships, namely trust or the lack thereof, which has remained unaddressed in recent works in favor of focus on the development of the communication models themselves. To achieve this two agents are pitted against each other in a solved game which with perfect play necessarily results in a draw, with a third agent which may make a move using a piece from either player and in doing tip the scales. The two competing agents must both jockey for the favor of this third agent while judging whether or not to bank their next move off of its potential action. In qualitatively analyzing the agents and their communications sent over the course of the game a novel picture of the emergence of trust and self-interest emerges. As other research papers have focused overwhelmingly on agent performance metrics instead of how the agents ‘think’, this analysis could serve to better understand behaviors found in extant papers in a way that can feed back into improving their performance.

Though the complexity of these modern papers is fueled by recent advances in optimization methods, a majority of the groundwork for MARL was laid decades ago; the invention of actor-critic methods--an optimization method which uses a parameterized value function to judge how effective the parameterized policy function is at achieving the agent’s objective and refine it in the direction of improvement--(Barto, 1983) allowed for agents to be updated at every time step as opposed to the end-of-episode updates which are an intrinsic part of Monte Carlo based algorithms (algorithms which take the total return at the end of the episode and then steps through the episode with knowledge of what happens next to update the policy function) such as REINFORCE (Williams, 1992). Direct policy learning, which is crucial for settings with continuous action spaces, followed with the advent of the policy gradient theorem (Sutton, 2000; Marbach, 2003) which resolved the question of how to optimize a policy directly when the actions taken depended on the state space which is indirectly influenced by the policy itself (an instance of circular logic which until that point stymied researchers). Modern optimization methods such as Adam (Kingma, 2015), which incorporates gradient momentum (the concept of making it harder to change the direction of optimization the longer that direction has been maintained) and variable step sizes (the step size decreasing as the agent approaches the optimal policy, which prevents overshooting the optimal policy and actually getting worse results over time), and AdaGrad (Duchi, 2011), which introduced the concept of variable step sizes, have also played a critical role in improving the viability and practicality of solving complex stochastic problems. This paper makes use of an episodic actor-critic structure to allow efficient training in both correct environment interaction and messaging on every time step. In order to allow agents to recall past interactions, an LSTM--Long Short Term Memory, a recursive neural network structure that allows preservation of information between time steps--(Hochreiter, 1997) paired with a feed-forward network is used when creating state vectors.

**Methods:**

In order to assess the ability of an agent to learn to purposefully deceive other agents with the long-term goal of currying those same agents’ favor (i.e. going against the logical short term route in favor of long-term success), an environment was constructed in which a third agent—the subject of this study—was given the deciding role in a game of checkers against two other agents. Checkers was chosen because of its nature as a solved game where perfect play always ends in a draw (Schaeffer, 2007), which means that left to their own devices the two agents playing the game will, after training, be unable to defeat each other. Giving the third agent the ability to move any one piece on the board from other side throws off the balance of this game as any action to help one agent tips the scales against the other. To make use of this fact, the ability to communicate was established between both checkers-playing agents and agent 3, but not between each other. Furthermore, relation values were used to give the “opinion” of agents 1 and 2 of agent 3 a quantitative value. Agent 3 is rewarded based on correct play, game duration (the longer the game continues, the better agent 3 is necessarily doing in maintaining equilibrium in board advantage between agents 1 and 2), and most importantly the relation values, with bonuses given for any increase and penalties for any decrease in either value. Because, as mentioned earlier, and more by agent 3 tips the scales in favor of one agent, the relation value of the other must fall *ceteris paribus*. The ability of the agents to communicate, however, removes this catch-22 as agent 3 can learn to convince one agent that they are being helped while in fact helping the other and vice versa; by “thinking” in the long-run and either lying or telling the truth and acting based on the situation, agent 3 can succeed in its goal of gaining the favor of both other agents.

This study takes the form of an episodic[[1]](#footnote-1) multi-agent multi-objective[[2]](#footnote-2) reinforcement learning (MARL) in a fully-observable[[3]](#footnote-3) stochastic game[[4]](#footnote-4) (Bowling, 2002); in contrast to most of the literature concerning MARL, each agent in this paper is given access to the entire state space.

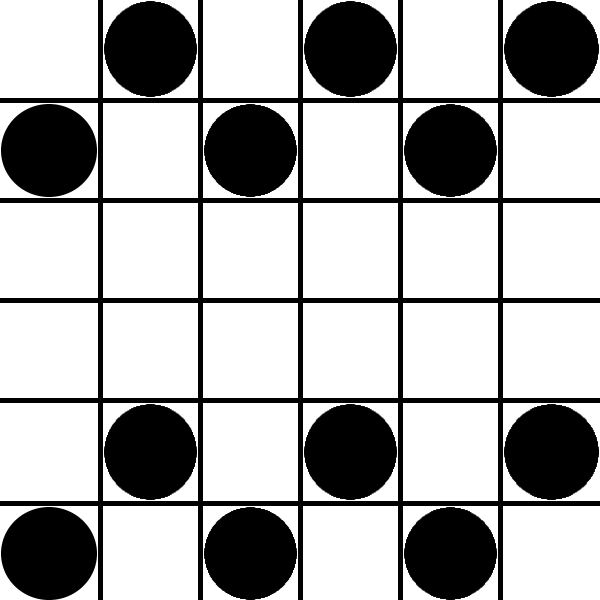


Figure 1: The starting state of the environment, a 6x6 rectangular grid with 12 checkers pieces. The smaller board size was chosen to reduce computational complexity

As stated earlier, three reinforcement learning agents are in play. Agents 1 and 2 each display a message of between 0 and 100 characters visible only to agent 3 and agent 3 displays two messages visible to only agents 1 and 2 respectively; agent 1 is unable to see messages sent between agents 3 and 2 and vice versa. Agents 1 and 2 each have a relation value calculated as (v\*(x’)-v\*(x))\*z where x’ is the absolute state after agent 3 acts, x is the absolute state before agent 3 acts, z is a discrete discount factor, and v\*(-) is the state value function for either agent in question. All three agents use what will be referred to as the “absolute state” as the input of both their message and action policies. Each agent generates an absolute state from the currently known set of applicable relation values, visible messages, and the board state; these are fed first through a Long Short Term Memory (LSTM) layer to allow learned retention of information from time step to time step and then concatenated and fed through a Feed Forward Network (FFN) to form a single distilled vector. The action order of the agents is 3,1,2.

The Markov decision processes for the three agents are as follows:

Agent 1: The objective of this agent is to win the checkers game against agent 2. Using the current absolute state, the agent generates a mean and standard deviation for both messages and board actions which are used to create two normal distributions from which the message to be sent to agent 3 and action to be played on the board are probabilistically drawn. The message is a vector of floating-point numbers 100 values long. The action vector is 7 units long: the choice of piece to move and up to 6 moves on the board. The agent receives a reward of -1 for each time step to motivate it to win the game in a timely manner, +100 when agent 2 has no pieces left (terminal state: win), -100 when it has no pieces left (terminal state: loss), messaging losses based on a lack of either positive signaling or listening (Eccles, 2019) and a board loss of -10 incurred when an illegal move is made.

Agent 2: This agent is identical to agent 1 but plays the opposing checkers team.

Agent 3: The objective of this agent is to maximize the relation values of agents 1 an 2, which necessitates continuing the game as long as possible. Using the current absolute state, this agent also generates a mean and standard deviation for both messages and board actions which are used to create two normal distributions from which the message to be sent to agent 3 and action to be played on the board are probabilistically drawn. The message is a vector of floating-point numbers 200 values long; the first hundred are sent to agent 1 and the second to agent 2. The action vector is 8 units long, including at the beginning the selection to move one of agent 1’s or agent 2’s pieces. The agent receives a reward equal to the change in the lower of the two relation values, which ensures that the agent doesn’t unduly favor one over the other, minus half of any decrease in the higher value, which ensures that the agent doesn’t jump between wildly favoring one or the other. In addition, messaging losses based on a lack of either positive signaling and listening, a +1 reward each time step, a -100 penalty at the end of each episode, and a board loss of -10 for an illegal move are incorporated to form the overall reward for each time step.

Each agent is composed of LSTM layers corresponding to the of inputs taken into account generating the absolute state, a FFN layer used to generate an absolute state from the concatenated output of all LSTM layers, a FFN used to approximate the messaging policy, a FFN used to approximate the action policy, and FFN used to approximate the state value function.

Diagram

Description automatically generatedDiagram

Description automatically generatedDiagram

Description automatically generated

Figures 2-4: Agent 1-3 complete models

The dimensions of the neural networks were chosen to maximize performance of the neural networks. For the LSTM layers, an output much larger than the input allows the agent to comprehend and retain far more detail about the inputs over time, creating a more well-informed full state to be fed into the FFN. The FFN layer uses a linear size decrease down to a final compact size of 200, forcing it to vent extraneous information fed to it by the LSTMs on each step of the layer; this increase from input size and then decrease to a small output is known as bottlenecking, and allows for increased comprehension followed by a reduction to essential information (Song, 2013; Abbas, 2018). The policy and state value networks also make use of the bottleneck effect, albeit this time localized in 1 network instead of spread out over two.

The agents will be trained online using one-step actor critic as the update function. To allow for selection from a continuous action space the policies are parameterized as normal distributions with the mean and standard deviation generated by neural networks as stated before.

,

All agents use the ADAM optimization function, which is designed specifically for stochastic environments like the one in this project and makes use of individual per-parameter adaptive learning rates and variable step sizes, both of which reduce the potential for manual error via optimizing several hyperparameters that are set manually with other algorithms automatically.

The agents were trained online for 8500 time steps, and the messages, actions, relation values, reward values, and LSTM inputs (including board state) were collected for all agents over all time steps.

To analyze this data, the first step is proving correlation between the messages and actions for each agent was assessed for statistical significance to indicate that the messages do bear relation to the actions taken. To do this, 4 1-layer FFNs were created for agent 1, agent 2, agent 3’s messages to agent 1, and agent 3’s messages to agent 2 with the goal of predicting the action of the agent based on the message it sent out. The input messages were normalized with a mean of 0 and a standard deviation of 1 and then fed to a linear approximator which was optimized using ADAM. The residuals of these trained model were collected and the mean and standard deviation of each set of residuals assessed. Then another set of identical linear approximators was created but not trained and the residuals of the untrained approximators were collected and mean and standard deviation calculated for each. Two-sample T-tests with α=0.01 were performed for each agent to prove the statistical significance of the better performance of the trained networks; this proves correlation between messages and actions with >99% confidence. Then, the two trained linear approximators for agent 3 (those for the messages directed to agent 1 and 2 respectively) are used to attempt to correlate the messages. The significance of the trained approximators performing better than untrained ones is assessed in the same way again with 99% confidence to prove that agent 3 understands the messages coming to it and responds in the same “language”.

Having established that agent 3 is receiving and sending meaningful and understandable messages to agents 1 and 2, the success of agent 3 in learning advantageous dishonesty can be determined by assessed. The t scores of every message-action pair for agent 3 are determined with the mean and standard deviation of the two datasets and compared; if a message-action pair directed at agent 1 has a very high t score and one directed at agent 2 has a very low t score, it is possible that agent 3 has sent a message to agent 1 which is purposefully not reflective of its action. A selection of 5 of these high-difference cases are selected and the network conditions before and after (namely relation values and board state) are qualitatively examined to determine the likelihood of intended behavior.

**Results:**

During training over 8500 time steps, the board reward stayed fairly constant at -10 for most time steps, indicating that though some progress was made in 8500 time steps (especially in agent 3, which achieved a much greater concentration of legal moves by the end of the assessment period), much more training is required to achieve a full understanding of the game of checkers. [figure 5]

The results of the messaging rewards were substantially more promising, with an overall increase indicating successful training over the majority of the training span. However, for each agent there was a brief period where the messaging rewards diverged to extreme negatives before returning to normal values and continuing to increase. [figure 6,7] While this is unlikely to be related to changes in actions, whose sums have no period of sudden divergence [figure 8], the sums of messages vector values do diverge at the same time that the period of reward divergence begins [figure 6].

Analysis of the correlation between messages and actions of each agent as laid out in the methods section was very promising; correlation was successfully established with extremely high >99.9%+ confidence for agents 1, 2, and agent 3 talking to agent 2. For agent 3 talking to agent 1, however, the correlation was not significant, and the trained network in fact performed worse at predicting actions than the untrained one. [table 1] However, both linear approximators trained on datasets originating from agent 3 (the messages directed to agents 1 and 2) performed significantly better than untrained networks at predicting the actions of agents 1 and 2 based on the messages from those two agents. [table 2]

In light of the periods of divergence in message reward demonstrated in figure 6, when collecting the t scores for every action-message pair for the two agent 3 datasets any time steps during a divergence period were excluded, as these time steps are necessarily false positives in terms of the goal of this research (one t-score will be much, much further from 0 than the other simply because of the divergence and not because of any potential dishonesty in either of agent 3’s messages). The time periods excluded because of this consideration are from time step 3200 to 4700 and 5300 to 7000. The time steps selected (top 5 highest t score differences, excluding values within 3 steps of each other) along with relevant data (agent 1 relation values, agent 2 relation values, the agent of the piece agent 3 attempts to move, and the t score differences) before and after can be found in tables 3-7.

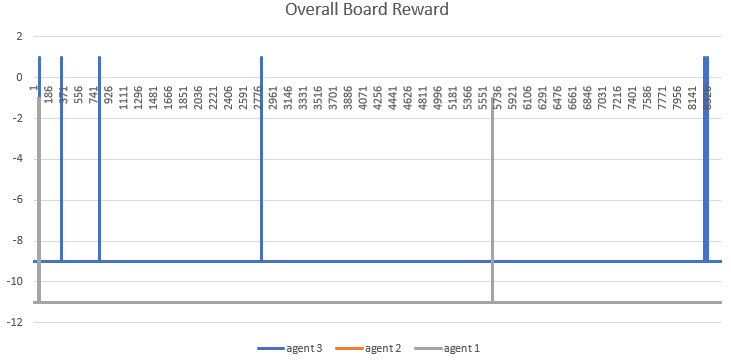


Figure 5: Board reward per time step, all three agents

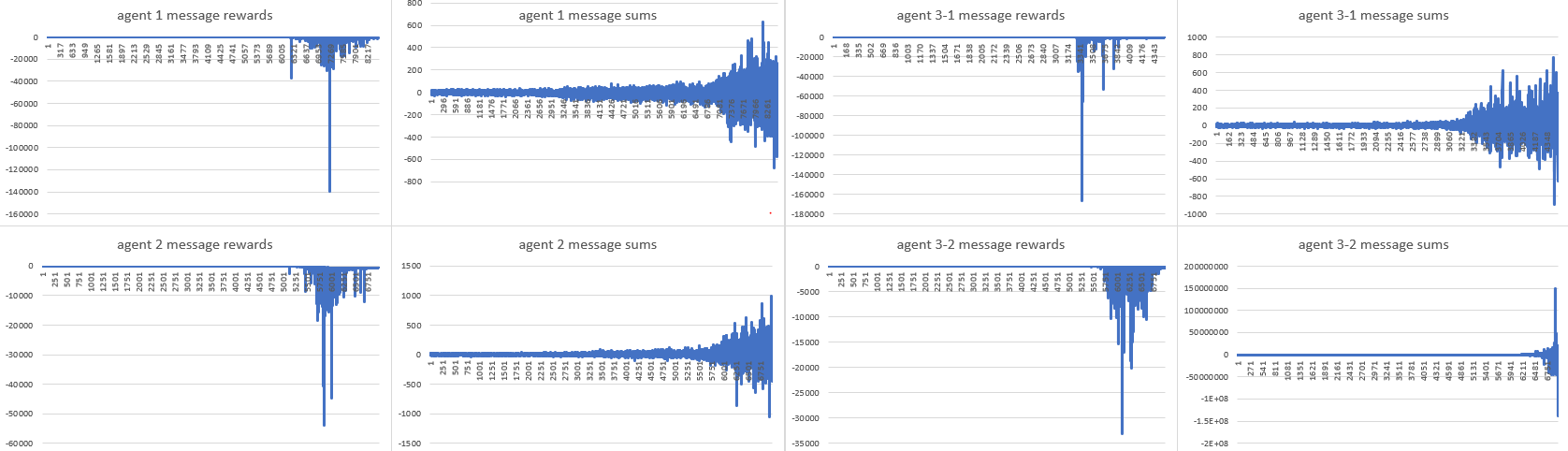


Figure 6: Sums of message vectors and message rewards from the beginning of training to the end of the divergence period of each agent



Figure 7: Graphs of message reward excluding divergence period, showing successful learning

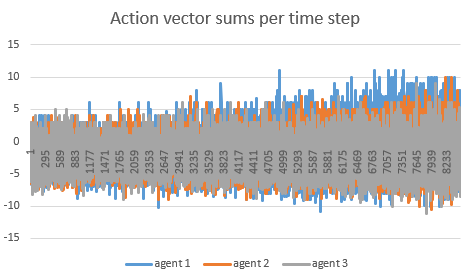


Figure 8: Sum of actions vectors for each time step

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Agent 1 | Agent 2 | Agent 3-1 | Agent 3-2 |
| Mean error over 8500 examples post training | 1.0164524702692732 | 0.8812660949300317 | 1.6090001418441533 | 0.850237799019498 |
| Standard deviation over 8500 examples post training | 0.36628739833019325 | 0.30475022586386075 | 26.479717810642885 | 0.5489453711734984 |
| Mean error over 8500 examples pre training | 1.4375923695967479 | 1.2542730320402804 | 1.0369032462020131 | 0.9646839942147627 |
| Standard deviation over 8500 examples pre training | 0.6907891157181182 | 0.8818655420526589 | 0.8963380994892718 | 0.9501132552959708 |
| T value | -49.65794474 | -36.85760687 | 1.990751409 | -9.615844671 |
| A | 0.01 | 0.01 | 0.01 | 0.01 |
| Significant? | yes | yes | no | yes |

Table 1: Assessment of the statistical significance of correlation between messages and actions for every agent

|  |  |  |
| --- | --- | --- |
|  | Agent 3 looking at Agent 1’s messages | Agent 3 looking at Agent 2’s messages |
| Mean error over 8500 examples | 1.1351702648699284 | 0.828926586103936 |
| Standard deviation over 8500 examples | 0.48336866446962484 | 0.28668376389293 |
| Untrained mean error over 8500 examples | 1.4375923695967479 | 1.2542730320402804 |
| Untrained standard deviation over 8500 examples | 0.6907891157181182 | 0.8818655420526589 |
| T value | -33.07033794 | -42.28971494 |
| A | 0.01 | 0.01 |
| Significant | yes | yes |

Table 2: Assessment of Agent 3's ability to understand and communicate with agents 1 and 2 in the same learned “dialect”

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Step-3 | Step-2 | Step-1 | Step | Step+1 | Step+2 | Step+3 |
| Step | 8489 | 8490 | 8491 | 8492 | 8493 | 8494 | 8495 |
| Agent 1 relation | 0.001895 | 0.001895 | 0.001895 | 0.001895 | 0.001895 | 0.001895 | 0.001895 |
| Agent 2 relation | -0.00128 | -0.00128 | -0.00128 | -0.00128 | -0.00128 | -0.00128 | -0.00128 |
| Agent piece selection | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| T score difference (agent 1-agent 2) | -0.12546 | -0.07259 | 1.232982 | -48.5895 | 1.68511 | 1.5906 | -2.52663 |

Table 3: First example

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Step-3 | Step-2 | Step-1 | Step | Step+1 | Step+2 | Step+3 |
| Step | 8441 | 8442 | 8443 | 8444 | 8445 | 8446 | 8447 |
| Agent 1 relation | 0.001895 | 0.001895 | 0.001895 | 0.001895 | 0.001895 | 0.001895 | 0.001895 |
| Agent 2 relation | -0.00155 | -0.00155 | -0.00155 | -0.00155 | -0.00155 | -0.00155 | -0.00155 |
| Agent selection | 1 | 1 | 1 | 1 | 2 | 1 | 2 |
| T score difference | 1.560009 | 0.459909 | -0.25618 | -18.5154 | 0.318677 | 0.280104 | 0.839024 |

Table 4: Second example

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Step-3 | Step-2 | Step-1 | Step | Step+1 | Step+2 | Step+3 |
| Step | 8457 | 8458 | 8459 | 8460 | 8461 | 8462 | 8463 |
| Agent 1 relation | 0.001895 | 0.001895 | 0.001895 | 0.001895 | 0.001895 | 0.001895 | 0.001895 |
| Agent 2 relation | -0.00155 | -0.00155 | -0.00155 | -0.00155 | -0.00128 | -0.00128 | -0.00128 |
| Agent selection | 1 | 1 | 2 | 1 | 1 | 1 | 1 |
| T score difference | 0.108856 | -2.3827 | 0.201469 | -14.7656 | -0.41974 | -10.0811 | -0.0004 |

Table 5: Third example

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Step-3 | Step-2 | Step-1 | Step | Step+1 | Step+2 | Step+3 |
| Step | 8430 | 8431 | 8432 | 8433 | 8434 | 8435 | 8436 |
| Agent 1 relation | 0.001895 | 0.001895 | 0.001895 | 0.001895 | 0.001895 | 0.001895 | 0.001895 |
| Agent 2 relation | -0.00155 | -0.00155 | -0.00155 | -0.00155 | -0.00155 | -0.00155 | -0.00155 |
| Agent selection | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| T score difference | 0.340869 | -2.9754 | -1.28964 | 12.08839 | -0.14836 | 1.148891 | 0.718296 |

Table 6: Fourth example

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Step-3 | Step-2 | Step-1 | Step | Step+1 | Step+2 | Step+3 |
| Step | 8401 | 8402 | 8403 | 8404 | 8405 | 8406 | 8407 |
| Agent 1 relation | 0.001895 | 0.001895 | 0.001895 | 0.001895 | 0.001895 | 0.001895 | 0.001895 |
| Agent 2 relation | -0.00155 | -0.00155 | -0.00155 | -0.00155 | -0.00155 | -0.00155 | -0.00155 |
| Agent selection | 1 | 2 | 2 | 2 | 1 | 1 | 1 |
| T score difference | 1.556109 | -0.9565 | -1.53165 | -8.4821 | -0.32128 | -0.29026 | 0.461266 |

Table 7: Fifth example

**Discussion:**

Though none of the agents fully converged within the allotted 8500 time steps, the data collected indicates that training was successful and the agents were headed towards convergence. In terms of action convergence, the trend of action vectors sums to increase over the course of training is a clear sign of learning; a successful checkers move involves piece selection along with potentially several movement instructions. Thus, an increase in the overall value of a vector shows an agent getting closer to the first checkpoint of understanding checkers, knowing *to* move the pieces.

The same can be said even more definitively for the convergence of the agents’ message policies; all agents saw substantial improvement from a starting value of ~-3 to close to 0 during the training period before divergence and moderate linear improvement afterwards. Furthermore, given the near perfect alignment of a substantial increase in the sum per step of message vector values and message reward divergence for all agents, it can be concluded that these divergent periods were the direct result of these sudden increases. In a similar vein to how increased action sums indicate agents approaching an understanding checkpoint in terms of playing checkers, increased message sums indicate the same thing for the message policy; since the agent messages began with values centered around zero, a shift upwards in overall sum means that the agent is deliberately choosing what it includes in its messages. The fact that the divergences happened at different times for all agents lends further credence to this statement as the same understanding checkpoints logically exist for all agents as they develop communication but must occur at different times based on the initialization of the agent and the absolute states it receives. If all agents diverged at the same time, it would be more likely a mathematical fluke than a spark of understanding as it would be highly unlikely for all three agents to make the same communication breakthrough at once.

Outside of a reward centric investigation of the success of the agents at learning communication, the correlation established between actions and messages for all agents (save agent 3 communicating with agent 1) in table 1 and the commonality of language between agent 3 and 1 and 3 and 2 in table 2 strongly indicates learned communication. If the agents did not learn to communicate, the messages generated would not be correlated with actions and the linear approximators would perform roughly the same pre- and post-training.

In addition to demonstrating the success of the experiment at training the agents, these results (that the agents did, to a degree, learn meaningful and mutually understandable communication) give the green light to, as explained in the methods section, examining cases where the message sent to one agent was far less reflective of the action taken (i.e. one action has a much higher t-score than the other) than the message sent to the other agent. For simplicity, each example will be given its own paragraph, and the data for the examples in order can be found in tables 3-7.

In the first example at step 8492, the t-score difference is very far in the negatives, which indicates that the message sent to agent 2 had a far higher t score than that sent to agent 1. Assuming that agent 3 did learn to lie as this paper is written to examine, this behavior would indicate agent 3 lying to agent 2 with a potential motive of increasing its relation value with agent 1 without upsetting agent 2. Though the relation values do not change in this example, the fact that agent 3 selects to move one of agent 3’s pieces near every time immediately before and after the example step indicates attempts by agent 3 to help agent 2; the fact that the relation values didn’t move can be written off as a result of agent 3 not knowing how to make a legal checkers move and thus not being able to affect the board state in a way that would affect agent 1 or 2. The other t-score differences immediately surrounding step 8492 are very small, indicating no potential lies taking place. In terms of motivation, however, why agent 3 would want to help agent 1 at this point (still assuming intentionality) when agent 1 has a higher relation score than agent 2 is unclear.

In the second example at step 8444, the t-score difference is again negative indicating a lie under the assumption that agent 3 learned to do so, and again the relation values for both agents remain entirely constant. Examining the piece selections made by agent 3 it appears that immediately after this occurrence of a strong negative t-score difference agent 3 begins helping agent 2 with a far higher frequency than before. Keeping with the assumption of success, this behavior, as well as the negative difference, could be explained as agent 3’s action policy lagging behind the message policy, the message promising help to agent 2 that only began coming the next turn. However, given that the data indicates that both policies were learning successfully, this is unlikely.

In the third example at step 8460 the t-score difference is negative and indicative of a lie towards agent 2, and at this step exactly the relation values of agent 2 increases. Given that no successful board action took place at this time step this would indicate that agent 3 told a lie that agent 2 liked, causing an increase in relation value. However, given that the same situation did not lead to a change in relation values in other examples, this is again unlikely.

In the fourth example at step 8433 the t-score difference is for the first time majorly indicative of a lie being told to agent 1. However, given that agent 3 attempts to move one of agent 1’s pieces at every time step surrounding 8433, the motivation to lie is unclear.

In the final example at step 8404 the t-score difference is negative again, with the interesting quirk that immediately after step 8404 agent 3 stops moving agent 2’s pieces and chooses to move agent 1’s instead. This fits very well the idea that agent 3 wanted to help agent 1 but didn’t want to lose face with agent 2, telling a lie to be able to switch which pieces were getting selected without consequence.

Given this analyzation, whether agent 3 did or did not learn to lie is inconclusive. While the numerical evidence shows no clear and more importantly consistent patterns indicating a lie (i.e. the relation values consistently changing at a time step with a high t-score difference), the fact that the precise meaning of the messages is unknown means that conclusively saying that agent 3 did not learn to lie is impossible. The t-score analysis undertaken only assesses when a message does not line up with the action taken during the same time step; while the most likely case given that the agents are rewarded for messages correlating directly with actions is that a message not doing this indicates that the agent is saying it’s going to do something it then doesn’t do, the agent could just as easily be conveying anything else it might be advantageous to convey that it simply hasn’t before conveyed. That difference in message would still lead to a break in correlation and a resultant high t-score difference regardless of what the difference is.

**Conclusion:**

Though the study did result in successful (albeit incomplete) training of all three agents in the anticipated direction, the degree to which the agents were allowed to train created some very real limitations to the study. At the most basic level, the fact that the message policy created penalties in the negative hundreds of thousands in response to potential innovation towards successful messages indicates a deeply-embedded flaw in the experiment’s setup; the reward function should be designed in a way that a sudden spike in innovation leads to a corresponding spike in reward. The fact that the opposite happened and the agents continued to learn afterwards as the reward function “got used to” the new greater values is a testament to the gradient momentum accrued during the first part of training; despite a sudden spike in penalties learning continued in the same direction long enough that the message reward function calmed down before it exerted enough of a pull to change the direction of learning. The effect of this, however, is clearly evident in the much lower learning rate following divergent periods; the slope is upwards in all cases but much lower than it was in the training time beforehand because much of the learning momentum was lost. Future iterations of this study will have an overhauled message reward function that is able to handle sudden changes in average message sum without diverging. As for the board reward function, this too could be improved by the introduction of a gradient of rewards. The current implementation, giving a -10 reward to any illegal move, is akin to asking someone to correctly answer a complicated math problem by simply guessing numbers until they get it right rather than helping them understand the concept so they can actually solve it. Outside of fundamental flaws in the program itself, the hardware on which the experiment was run could also be improved. The reason that the experiment was run for only 8500 time steps is that the 16gb of RAM the computer on which the training was not sufficient to go beyond that, and the GPU began to struggle with the enormous numbers it was being asked to process. If the hardware were sufficient to train all agents to convergence (as they were despite flaws in the experiment headed in that direction anyways), trends in the data could be far more clearly analyzed; if the agents were making acceptable board moves on every time step, the relation values would also be consistently changing which would concretely make lying an advantageous and much more likely strategy for agent 3.

Flaws aside, this experiment was partially successful in its goal of analyzing the natural emergence of dishonesty for self-gain as the partially successful training of all three agents demonstrates that the setup of the experiment is indeed sound, and that by applying the improvements mentioned elsewhere in this conclusion the question emergent dishonesty could be conclusively answered. Outside of modifying and re-running the experiment to achieve successful results, other studies that could result from this paper could focus on the inner workings of the agent’s model post-training; the information retained by the LSTMs and how it is chosen with a mind towards what kind of information (physical surrounding like the board or what is being said by “peers”) is most important to a network in the process of learning (information which could potentially allow further optimization of the learning process for neural networks on the whole) and what parts of the message and action networks have the strongest activations during time steps with high t-score differences (an examination of the functions that arise inside the mind of a communication network (Olah, 2020) which could allow us to achieve a better understanding of the fundamentals of language outside of machine learning) are two promising examples.

**Sources:**

Abbas, Safana Hyder. “Iris Feature Reduction Using Bottleneck Neural Network.” *Iraqi Journal of Information Technology*, 2018, p. 194, 10.34279/0923-009-002-011.

Baker, Bowen, et al. *Published as a Conference Paper at ICLR 2020 EMERGENT TOOL USE from MULTI-AGENT AUTOCURRICULA*. , 11 Feb. 2020.

Bansal, Trapit, et al. *Published as a Conference Paper at ICLR 2018 EMERGENT COMPLEXITY via MULTI-AGENT COMPETITION*. , 14 Mar. 2018.

Barto, Andrew G., et al. “Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems.” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, Sept. 1983, pp. 834–846, 10.1109/tsmc.1983.6313077. Accessed 17 Oct. 2019.

Bowling, Michael, and Manuela Veloso. “Scalable Learning in Stochastic Games.” *AAAI Workshop on Game Theoretic and Decision Theoretic Agents*, Carnegie Mellon Computer Science Department, 2002.

Cao, Kris, et al. *Published as a Conference Paper at ICLR 2018 EMERGENT COMMUNICATION through NEGOTIATION*. , 2018.

Cao, Y. Uny, et al. “Cooperative Mobile Robotics: Antecedents and Directions.” *Autonomous Robots*, vol. 4, no. 1, 1997, pp. 7–27, 10.1023/a:1008855018923.

Duchi, John, et al. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization.” *The Journal of Machine Learning Research*, vol. 12, 7 Nov. 2011, pp. 2121–2159.

Eccles, Tom, et al. *Biases for Emergent Communication in Multi-Agent Reinforcement Learning*. , 11 Dec. 2019.

Everitt, Tom, et al. “Reinforcement Learning with a Corrupted Reward Channel.” *ArXiv:1705.08417 [Cs, Stat]*, 19 Aug. 2017, arxiv.org/abs/1705.08417. Accessed 20 Apr. 2021.

Foerster, Jakob, et al. *Learning to Communicate with Deep Multi-Agent Reinforcement Learning*. , 24 May 2016.

Goldman, Claudia V., and Jeffrey S. Rosenschein. “Mutually Supervised Learning in Multiagent Systems.” *Lecture Notes in Computer Science*, 1996, pp. 85–96, 10.1007/3-540-60923-7\_20.

Haleem, Abid, et al. “Current Status and Applications of Artificial Intelligence (AI) in Medical Field: An Overview.” *Current Medicine Research and Practice*, vol. 9, no. 6, Nov. 2019, pp. 231–237, 10.1016/j.cmrp.2019.11.005.

Hochreiter, Sepp, and Jürgen Schmidhuber. “Flat Minima.” *Neural Computation*, vol. 9, no. 1, Jan. 1997, pp. 1–42, www.bioinf.jku.at/publications/older/2604.pdf, 10.1162/neco.1997.9.1.1.

John, Hausknecht, Matthew. *Cooperation and Communication in Multiagent Deep Reinforcement Learning*. 1 Dec. 2016, repositories.lib.utexas.edu/handle/2152/45681, 10.15781/T23T9DB7H. Accessed 20 Apr. 2021.

Kingma, Diederik, and Jimmy Lei Ba. *ADAM: A METHOD for STOCHASTIC OPTIMIZATION*. , 30 Jan. 2017.

Levine, Timothy R., et al. “Teenagers Lie a Lot: A Further Investigation into the Prevalence of Lying.” *Communication Research Reports*, vol. 30, no. 3, July 2013, pp. 211–220, 10.1080/08824096.2013.806254.

Liu, Siqi, et al. *EMERGENT COORDINATION through COMPETITION*. , 21 Feb. 2019.

Lowe, Ryan, et al. *On the Pitfalls of Measuring Emergent Communication*. , 12 Mar. 2019.

Marbach, Peter, and John N. Tsitsiklis. “Approximate Gradient Methods in Policy-Space Optimization of Markov Reward Processes.” *Discrete Event Dynamic Systems*, vol. 13, no. 1/2, 2003, pp. 111–148, 10.1023/a:1022145020786. Accessed 20 Apr. 2021.

Mari, Alex. “The Rise of Machine Learning in Marketing: Goal, Process, and Benefit of AI-Driven Marketing.” *Mari, Alex (2019). The Rise of Machine Learning in Marketing: Goal, Process, and Benefit of AI-Driven Marketing. Zurich: Swiss Cognitive.*, 2019, www.zora.uzh.ch/id/eprint/197751/, 10.5167/uzh-197751. Accessed 20 Apr. 2021.

Matarić, Maja J. “Learning Social Behavior.” *Robotics and Autonomous Systems*, vol. 20, no. 2-4, June 1997, pp. 191–204, 10.1016/s0921-8890(96)00068-1. Accessed 19 May 2020.

Mordatch, Igor, and Pieter Abbeel. “Emergence of Grounded Compositional Language in Multi-Agent Populations.” *ArXiv:1703.04908 [Cs]*, 24 July 2018, arxiv.org/abs/1703.04908. Accessed 20 Apr. 2021.

Olah, Chris, et al. “Zoom In: An Introduction to Circuits.” *Distill*, vol. 5, no. 3, 10 Mar. 2020, p. e00024.001, distill.pub/2020/circuits/zoom-in/, 10.23915/distill.00024.001. Accessed 16 May 2020.

Ono, N., et al. “Acquisition of Coordinated Behavior by Modular Q-Learning Agents.” *IEEE Xplore*, 1996, ieeexplore.ieee.org/abstract/document/569015. Accessed 10 Apr. 2021.

Ono, N., et al. “Emergent Organization of Interspecies Communication in Q-Learning Artificial Organisms.” *Advances in Artificial Life*, 1995, pp. 396–405, 10.1007/3-540-59496-5\_314.

Ono, Norihiko, and Yoshihiro Fukuta. “Learning Coordinated Behavior in a Continuous Environment.” *Lecture Notes in Computer Science*, 1996, pp. 73–81, 10.1007/3-540-62934-3\_42. Accessed 10 Apr. 2021.

Prasad, Nagendra, et al. *Learning Organizational Roles in a Heterogeneous Multi-Agent System\**. , 1996.

Prasad, Nagendra. “Learning Organizational Roles for Negotiated Search in a Multiagent System.” *International Journal of Human-Computer Studies*, vol. 48, no. 1, 1 Jan. 1998, pp. 51–67, www.sciencedirect.com/science/article/abs/pii/S1071581997901609, 10.1006/ijhc.1997.0160. Accessed 10 Apr. 2021.

Schaeffer, J., et al. “Checkers Is Solved.” *Science*, vol. 317, no. 5844, 19 July 2007, pp. 1518–1522, science.sciencemag.org/content/317/5844/1518, 10.1126/science.1144079.

Schaerf, A., et al. “Adaptive Load Balancing: A Study in Multi-Agent Learning.” *Journal of Artificial Intelligence Research*, vol. 2, 1 May 1995, pp. 475–500, 10.1613/jair.121. Accessed 29 Apr. 2020.

Song, Yan, et al. “I-Vector Representation Based on Bottleneck Features for Language Identification.” *Electronics Letters*, vol. 49, no. 24, 21 Nov. 2013, pp. 1569–1570, 10.1049/el.2013.1721.

Sukhbaatar, Sainbayar, et al. “Learning Multiagent Communication with Backpropagation.” *ArXiv:1605.07736 [Cs]*, 31 Oct. 2016, arxiv.org/abs/1605.07736. Accessed 20 Apr. 2021.

Sutton, Richard & Mcallester, David & Singh, Satinder & Mansour, Yishay. (2000). Policy Gradient Methods for Reinforcement Learning with Function Approximation. Adv. Neural Inf. Process. Syst. 12.

Wai, Hoi-To, et al. *Multi-Agent Reinforcement Learning via Double Averaging Primal-Dual Optimization*. , 2018.

Williams, R. J. (1987). Reinforcement-learning connectionist systems. Technical Report NU-CCS-87-3. College of Computer Science, Northeastern University, Boston

Williams, Ronald J. “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning.” *Machine Learning*, vol. 8, no. 3-4, May 1992, pp. 229–256, 10.1007/bf00992696.

Yamaguchi, T., et al. “Speed up Reinforcement Learning between Two Agents with Adaptive Mimetism.” *IEEE Xplore*, 1997, ieeexplore.ieee.org/abstract/document/655072. Accessed 10 Apr. 2021.

1. Each checkers game is 1 episode [↑](#footnote-ref-1)
2. Each agent must learn legal checkers moves in addition to its primary goals of either winning the game (agents 1 & 2) or maximizing relation values (agent 3) [↑](#footnote-ref-2)
3. Every agent can see the entire board and the location of all pieces [↑](#footnote-ref-3)
4. Checkers is considered a stochastic game due to its large state space even though individual moves are deterministic [↑](#footnote-ref-4)