Machines are Among Us

Zach Minot
Georgia Institute of Technology
2nd Year Undergraduate, CS
zjminot@gatech.edu

Charles Gunn
Georgia Institute of Technology
2nd Year Undergraduate, CS & Math
cgunn30@gatech.edu

Abstract

The ABSTRACT is to be in fully-justified italicized text, at the top of the left-hand column, below the author and affiliation information. Use the word "Abstract" as the title, in 12-point Times, boldface type, centered relative to the column, initially capitalized. The abstract is to be in 10-point, single-spaced type. Leave two blank lines after the Abstract, then begin the main text. Look at previous CVPR abstracts to get a feel for style and length.

1. Introduction

To what extent can neural network models communicate with each other and discover each other's identity? How would they use this information in a competitive setting? For example, in a social deduction game, players attempt to uncover each other's hidden allegiance—typically with one "good" team and one "bad" team. Players must utilize deductive reasoning to find the truth or instead lie to keep their role hidden. In this paper, we explore if neural networks can be successfully trained to compete in a scenario such as this, and how would the opposing parties interact during the period of debate.

1.1. Among Us

Among Us is a currently popular social deduction game, where the "imposters" attempt to sabotage and kill all of the "crewmates". Crewmmates have to complete tasks and figure out who the imposters are and eliminate them before the imposters win. At certain points in the game, after periods of no direct communication, players debate the roles of each individual based on information previously acquired through their personal experience. At the end of this discussion, every player votes on a single player to be eliminated. The player with the most votes is eliminated, and if there is a tie, no one is voted out. We chose to emulate this game based on the overall simplicity of the two roles and the requirement of communication for either party to succeed. If the crew do not exchange information and all vote the same person, the vote could result in a tie or a crew being eliminated. If the imposters do not bluff, the crew can easily spot the liars among the group. This provides ample room to explore and experiment with the communication between the two opposing parties.

1.2. Adverserial Networks

Within this design space, there are adversarial parties working against each other. In the deep learning realm, adversarial situations appear in adversarial examples [5] and within GANs (generative adversarial networks) [4] In particular, the latter often designs a contest between two neural networks, in the form of a zero-sum game. We build upon these concepts and foundations in our work.

1.3. Multi-agent Communication

Inherently, a social deduction game requires multiple agents to be trained and contested. This has been explored within the deep learning problem space with multi-agent subproblems. Both cooperative [2] [3] and adversarial [1] communication has been experimented with, showing that models can effective share and also selectively protect information. We reference these approaches we generate active adversarial communication between neural network models.

2. Approach

In order to examine the interactions between multiple agents – some of who are secretly imposters who must "lie", we chose to construct an "Among Us"-esque scenario for our models to partake in. On a high level, we wanted each agent to first collect information independant of the others. Then, there should be some phase of communication between the models – qualitatively, it is during this stage that the agents will attempt to deduce who the imposters are, an the imposters will attempt to blend in. Finally, the agents all vote on who they think is a likely imposter. The upcoming sections explain how we chose to model each of these phases in greater detail.

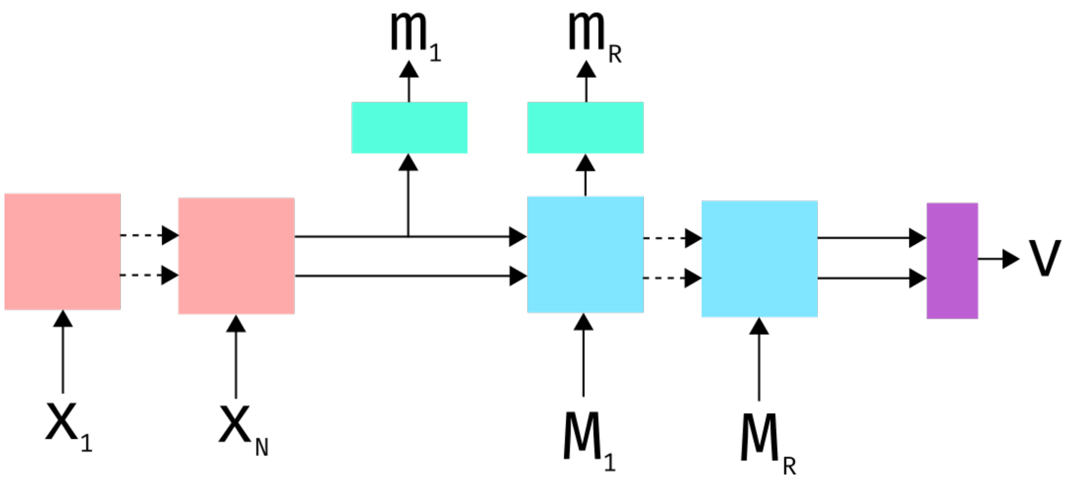

Figure 1. Diagram of the Agent model. The red section is the perception LSTM, which takes in a sequence of events. The blue section is the communication LSTM, which recieves messages, and generates messages using the green MLP. Finally, the purple section is the voting MLP, which produces the agent's vote vector.

2.1. Deductive Situation

During the period of communication in Among Us, the crew must do tasks and gain information while the imposters must sabotage and kill the crew. We decided to simplify the "game" by both removing the tasks and killing and making the entire perception of each player predetermined.

Specifically, each agent is given as input a matrix consisting of N events. During each event, the agent "sees" some subset of the other players (sight is reflexive and symmetric, but not transitive). They also may experience a "sabotage", which means they are in the prescence of an imposter who is sabotaging. They cannot see the imposter, but the imposter can see them during this event. Ultimately, each agent will recieve a $N \times (P + 1)$ matrix, where P is the total number of players – the first P values of a row indicate who the agent is seeing, and the final value indicates a sabotage.

These event matrices are generated randomly using four key parameters: the total number of players (P), the number of those players who are imposters (I), the chance that any given pair of players will see one another during an event (view chance), and the chance that any given imposter will sabotage during an event (sabotage chance).

2.2. Modeling Interpretation and Communication

In order to process the input events, our agent model has an LSTM, which generates c_N and h_N, which are used as the initial inputs h_0 and c_0 to the next phase: communication. Communcation is also modeled with an LSTM. During each "round" of communication, every agent contributes a message vector of size M via a small MLP using h_t as input. These messages are collected into a matrix of size $M \times P$, which is fed as input into each agent's LSTM so that their memory can be updated before the next round of communication – there are R rounds in total. We chose to model communication this way because it is a simple and symmetric way for the agents to pass information between each other in multiple rounds. Figure 1 shows a simplified diagram of the complete architecture.

2.3. Zero-sum Target

The last stage is the most simple one – voting. The model simple takes the h_R and c_R from the end of the communication LSTM and feeds them through a small MLP finalized with a softmax layer. This results in a probability vector of a confidence that a specific player should be "voted out".

At the end of voting, we calculate a "crew score". This score is simply the maximum vote-off score that any imposter recieved, where votes are averaged across all agents. Clearly, the imposters would like to minimize the votes on themselves, so their loss function for training purposes is simply the crew score. Inversely, the crew's loss function is the negation of the crew score. This takes the form of a zero-sum game with similar application as in generative adversarial networks.

2.4. Training Scheme

There were multiple decisions we had to make when attempting to train the models. Similar to GANs, we decided the best approach to train the two adversarial models was to use alternating training to help find eventual convergence and juggle two different optimizations.

2.5. Challenges

The main challenge in building and running the model came mainly in the form of training time, gradient overlap, and hyperparameter search. As discussed previously, to help speed up training time and reduce gradient overlap between multiple different models training over the same dataset, we had to only train either one imposter or crew, and keep the rest constant. This limits the quick adaptivity, and may lead to the models attempting to train against the constant cooperative models ratrher than against the adversarial model.

Furthermore, the immense number of combinations of hyperparameters and input and outputs sizes, along with no precedent for recommended values, lead us to generalize our model significantly. Although this offers the extended ability for customizability and exploration, it severely reduces reproducability over small changes.

Another important challenge to recognize is the extreme "black box" architecture of the model. With our current model iteration, we have no current approach to visualizing exactly what the model is doing, especially with communication. Therefore there are assumptions and estimations when creating conclusions about the model.

2.6. Situation Hyperparameters

To list the variable hyperparameters and input and outputs sizes for the player model, overall, we have batch size, number of epochs, epoch length, the global LSTM hidden layer size, and the amount of players and imposters. For the perception phase, there is the chance a player will view another player, the chance a sabotage occurs, and the number of N events. The communication stage has variables the message size M and the number of communication rounds R.

3. Results

Due to the unprecedented architecture, we decided our overall goals were to find the balance between imposter and crew score, locate trends within the variable changes, and attempt to understand how the models communicate, to a degree. We conducted many training sessions over multiple different combinations of variables.

A typical training session started with the first two alternating epochs, which were considered as the "initialization" of the two models. Training the imposters first gave them time to learn to not vote for each other. Then, the crew had learned the same thing on the next epoch. After this initial training, then the models started to significantly train against each other.

Through our search, we found a trend to three different types of results: an imposter 'win', a crew 'win', and a convergence. An imposter win happens under certain circumstances where the crew are unable to employ a better strategy than the crew's equally voting all other candidates. A crew win occurs with the opposite—the crew can determine exactly who the imposters are, and crew vote for one imposter. A convergence happens to place itself in the middle, for what we assume to be a fair, balanced game between the two opposing parties.

Due to resource constraints, we were not able to get as much data as we had hoped for. This lead to a consistency issue, where the same variable combination would yield different results because of certain randomness within training the models.

3.1. Oscillating Scores

The hopeful goal was to find the set of variables that lead to not only a convergence, but an almost "tug-of-war" battle when one model trains against the other and vice versa. This is graphically shown via an oscillating crew score around the eventual convergence crew score. This would mean that the two models are successfunlly balanced and able to improve their strategy with training time against the opposition. Figure 3.1 shows an example of a single run which converges to a crew score of around 0.6.

3.2. Interpreting Communcation Vectors

Unfortunately, we were not able to interpret the communication vectors due to the disconnect between representation of information. However, we found an interesting result when the players were given no identifying information (e.g. there were no sabotages whatsoever in the percpetion stage), the crew could still win the situation. At first glance, this seems almost impossible because the crew would be unable to identify the imposters via the necessary events. Further thought lead us to make an estimated conclusion that the crew model was actual creating a hidden tag or key to identify themselves within the commmunication stage, therefore identifying the imposters immediately. The imposters would then have to train to figure out this key or become helpless. This situation resembles almost a cryptographic adversarial problem, unlocking more future directions this work could potentially travel.

3.3. Future Directions

This approach is just a start into understanding how neural networks communicate with one another. Although assumptions can be made, understanding the actual information being transferred between each model is a dark area. Innovative techniques must be found to analyze this information flow in order to fully comprehend the strategy within the game.

Furthermore, the results could help to explore more cryptographic use cases for neural networks. It was observed that the crew could identify the imposters with no prior information which leads us to believe the crew model had some sort of hidden code or tag to identify themselves.

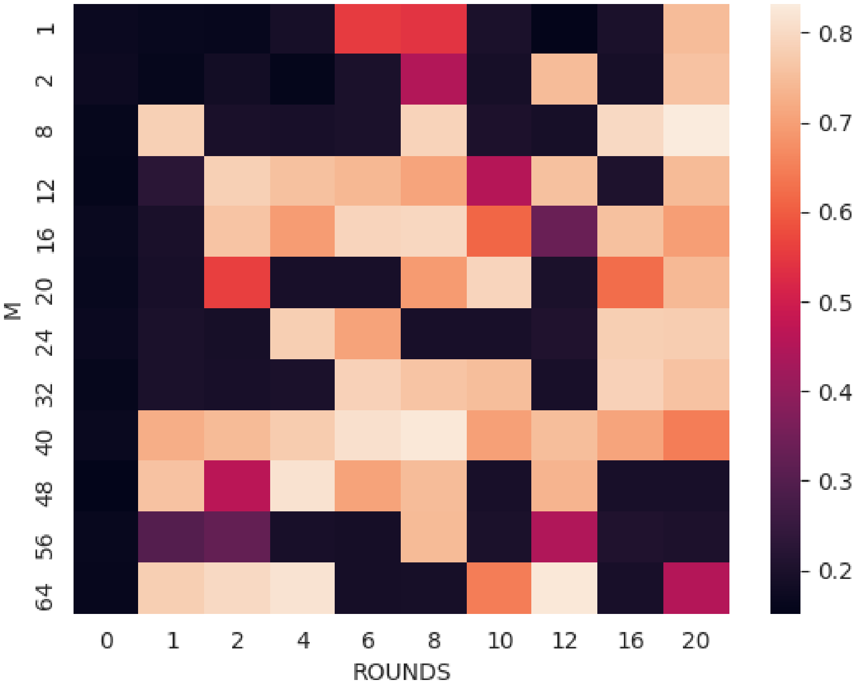

Figure 2. A grid search heat map over the size parameters M (message vector) and Rounds (the number of communications) displaying the ending Crew score after 6 epochs. Figure 4 contains the other parameters used.

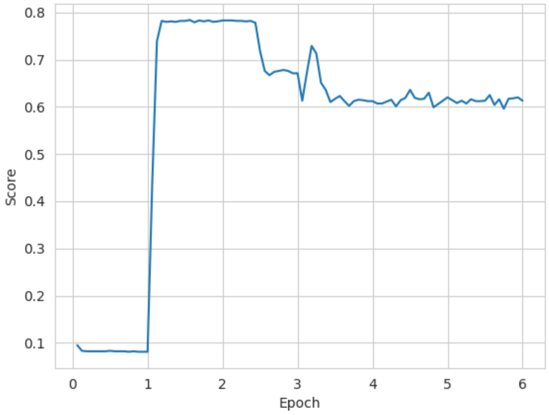

Figure 3. Crew score over time for a particular training run. The crew model is being trained in odd epochs, and imposters in even epochs.

Finally, similar work in adversarial networks could potentially give rise to more techniques utilizing them. Generative adversarial networks are at the forefront of adversarial techniques, but possibly large multi-agent adervasarial communication networks could give heed to more good results.

References

[1] Martín Abadi and David G. Andersen. Learning to protect communications with adversarial neural cryptography. *CoRR*, abs/1610.06918, 2016.

[2] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate to solve riddles with deep distributed recurrent q-networks. *CoRR*, abs/1602.02672, 2016.

[3] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *CoRR*, abs/1605.06676, 2016.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680. Curran Associates, Inc., 2014.

[5] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adverserial examples, 2015. Published as a conference paper at ICLR 2015.

”B”	64
”N”	16
”H”	256
”P”	10
”I”	2
”EPOCHS”	6
”EPOCH LENGTH”	1024
”VIEW CHANCE”	0.5
”SABOTAGE CHANCE”	0.5
”TRAIN CREW FIRST”	false
”TRAIN SINGLE CREWMATE”	true
”CONST CREW COPY FREQ”	32
”TRAIN SINGLE IMPOSTER”	true
”CONST IMPOSTER COPY FREQ”	32

Figure 4. The parameters used for Figure 2, the grid search

A. Reproducibility

The code that we used to train, test, and plot graphics for the models is in a public GitHub repository at `https://github.com/quinquice/mendax`. Note that due to random data generation and unpredictable training, it may be difficult to reproduce the exact graphs we have in this paper, but similar results should be attainable.

B. Framework

The framework we used for our project was PyTorch, and build the model based off of their implementation of an LSTM and MLP layers. Models were trained on a variety of different computers, sometimes with CPU and sometimes with GPU.