

# Learning The Gameplay and Conversations of *Among Us* using Proximal Policy Optimization and Generative Pre-trained Transformers

Govind Pimpale      Garni Gharbani      Nitish Reuben      Sudhanshu Agrawal  
University of California, Los Angeles

## Abstract

In this work, we simulate the popular online multi-player game, ‘Among Us’. Inspired by works like *CICERO* [1], we use multiple models to tackle different parts of a full game. In our first phase, we successfully train a reinforcement learning agent to learn the gameplay of players in the game. In our second phase, we fine-tune GPT-2 [8] on a synthetically generated data-set consisting of conversations between players to generate new conversations appropriate for a given state of the game during simulations. We observe that a Proximal-Policy-Optimization-based model improves performance by 2x on average when simulating the gameplay and finetuning GPT-2 on synthetic data improved performance by 4x on average across various metrics while generating conversations.

## 1. Introduction

‘Among Us’ is an online multiplayer game where players are randomly assigned roles of either *crewmate* or *impostor*. The objective of the crewmates is to complete *tasks* around the spaceship. Meanwhile, the *impostors*’ goal is to kill the crewmates without being caught. If a *crewmate* is killed and a dead body is found, a meeting is called where players enter a chat room to discuss their suspicions and vote on who they think the *impostor*. If all the *impostors* are discovered, the crewmates win. However, if the *impostors* successfully kill all the crewmates, the *impostors* win.

We separate the game into two game phases each with their own deep learning model: a *gameplay phase* and a *conversation phase*. During the gameplay phase, the crewmates complete tasks and avoid *impostors* and the *impostors* try to kill the crewmates and this is modeled using a reinforcement-learning (RL) approach. During the conversation phase, living players chat with each other to decide who the *impostor* is and this is modeled using a generative pre-trained transformer (GPT).

A formal problem formulation of these two phases is discussed in the Methods Section (2) below.

## 2. Methods and Problem Statement

### 2.1. RL Driven Phase

The game consists of a  $5 \times 5$  grid representing the game board. We have two classes of actors, *crewmates* and *impostors*. Additionally, there are *tasks* placed at random locations around the board. We see an example game board in Fig. 1. For an initial attempt, we constrain the game to have 3 *crewmates*, 1 *impostor*, and 1–2 *tasks*, all placed at random initial locations on the board.

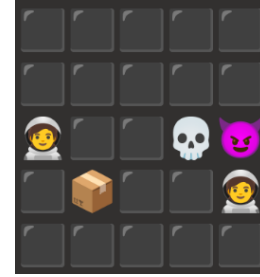


Figure 1. Example game board with *crewmates* represented by the astronaut emoticons, the *impostor* represented by the devil emoticon, a *task* represented by the box emoticon, and a dead *crewmate* represented by the skull emoticon

The gameplay has two basic themes : the *impostors* try to move towards the *crewmates* to kill them and the *crewmates* try to survive and move towards *tasks* to complete them.

To this end, we first define the gamestate as a  $5 \times 5$  board with 4 channels, each representing a different aspect of the gamestate. Each cell contains either a 0 or a 1. Thus the state space is given by  $S = \{0, 1\}^{5 \times 5 \times 4}$ . So for each channel  $x \in \{0, 1, 2, 3\}$ ,  $s.c_x[i, j] \in \{0, 1\}$

$$\begin{aligned} s.c_0[i, j] &= 1 \text{ iff there is an } \textit{impostor} \text{ at position } (i, j) \\ s.c_1[i, j] &= 1 \text{ iff there is a } \textit{crewmate} \text{ at position } (i, j) \\ s.c_2[i, j] &= 1 \text{ iff there is a } \textit{task} \text{ at position } (i, j) \\ s.c_3[i, j] &= 1 \text{ iff the } \textit{current player} \text{ is at position } (i, j) \end{aligned}$$

We define a reward system for the *impostor* as  $R_i(s) : S \rightarrow \mathbb{R}$  by giving it a reward of 1 for each *crewmate* it

overlaps (kills) and a reward of 0 if it is in an empty space.

$$R_i(s) = \sum_{i,j} (s.c_3[i,j]) \cdot (s.c_1[i,j])$$

We define a reward system for the *crewmate* as  $R_i(s) : S \rightarrow \mathbb{R}$  by giving it a reward of +0.5 for each *task* it overlaps (completes), a reward of -0.5 for each *impostor* it overlaps (gets killed by), and a reward of 0 if it is in an empty space

$$R_i(s) = (0.5) \sum_{i,j} (s.c_3[i,j]) \cdot (s.c_2[i,j] - s.c_0[i,j])$$

Further, both classes of players are subject to an action space  $A = \{L, R, U, D, W\}$ , representing moving left, right, up, down, and waiting in position respectively.

The objective is to learn two policies, one for the *crewmates*,  $\pi_{c,\theta}(s)$  and one for the *impostors*  $\pi_{i,w}(s)$ . At each state, these policies produce a probability distribution on  $A$ , corresponding to the best possible action for an *impostor* or a *crewmate* at that state.

$$\pi_{c,\theta}(s) \in \mathbb{R}^5 \quad \pi_{i,w}(s) \in \mathbb{R}^5$$

The next move is then sampled from this distribution for all players and the game progresses.

This policy is learned through a Proximal Policy Optimization (PPO) algorithm [9], a standard approach to reinforcement learning. We apply PPO to a Advantage-Actor-Critic (A2C) setup [5], where both the *Actor* and *Critic* are formulated as convolutional neural networks to better learn the multi-channel gamestate. The architecture is described in detail in Section 6.1

Finally, we employ a self-play approach [10] to train both the *impostor* and *crewmate* policy networks concurrently by playing them against each other as well as past (presumably weaker) versions of themselves to maximize the reward function described above. We also play the policy networks against opponents based on purely random policies and engineered policies to further improve performance.

## 2.2. Transformer Driven Phase

The conversation phase of the game consists of a chat room consisting of *crewmates* and an (unidentified) *impostor*. In a traditional game of ‘Among Us’, the objective of the chat room is to discuss who the *impostor* is and vote them out of the game. For our purpose, we simply aim to use a generative model to produce conversations that would be *likely* in such a chat room.

We first *synthetically generate a dataset* of conversations between players according to the method described in 6.2.1. This data consists of sequences of the form  $\{d_i\}$  where each  $d_i$  is the dialogue of some player  $p$  with a *class* field designating it as an *impostor* or a *crewmate*.

$$d_i = p\_name < tab > p\_class < tab > p\_dialogue$$

We constrain the chat room in our problem to consist of only 3 living players. We define 3 rounds in a single conversation, where each player speaks once in turn, for a total of 3 times per player in an entire conversation. Thus a chat-log for a conversation phase would consist of a sequence  $\{t_{p,i}\}$ , for dialogues  $i \in \{0, 1, 2\}$  of players  $p_n \in \{a, b, c\}$ :

$$[t_{a,0}, t_{b,0}, t_{c,0} \quad ; \quad t_{a,1}, t_{b,1}, t_{c,1} \quad ; \quad t_{a,2}, t_{b,2}, t_{c,2}]$$

We apply an auto-regressive approach to this problem where, given a prompt  $T$ , we first generate  $t_{a,0}$ , then use  $T$  and  $t_{a,0}$  to generate  $t_{b,0}$ , then use all three previous generations to generate  $t_{b,1}$  and so on.

$$t_{p_n,i} = \text{model}(\theta ; T, t_{a,0}, \dots, t_{p_{n-1},i-1})$$

E.g.

Prompt = “A dead body was found”

Response 1 = “That’s really suspicious!”

Response 2 = “You’re right, something was going on in the Engine room!”

Given the ubiquity and high performance of transformer-based models in natural text generation, we chose to *fine-tune GPT-2* [11] on our synthetic dataset. The output of our model is generated word-by-word and is valued based on similarity to sequences in our dataset. Details of the architecture are provided in 6.2.2.

Thus, our fine-tuned model, when prompted with a message, is expected to produce a response for a given player that is similar to the conversations that one would expect in a real-life game of ‘Among Us’. In particular, we predict that this fine-tuning process will produce better results than a zero-shot approach to generation using the baseline GPT-2 model on the same prompts.

## 3. Results

### 3.1. RL Model Performance

We find that a PPO-based approach successfully learns policies for the players in the game using a CNN as the underlying model architecture.

In particular, the trained policy network for the *crewmate* achieves an average reward / episode that is **3.01x** better than a purely random *crewmate* and an average reward/episode that is **0.8x** as good as a *carefully engineered heuristic policy* for the small game. Similarly, the trained policy network for the *impostor* achieves an average reward/episode that is **1.6x** better than a random policy and **0.8x** as good as the engineered policy. These results are summarized in Table 1.

We also observe in the reward curves for our networks in Figs. 3a, 3b, that the rewards consistently increase as

training progresses and would likely increase significantly with larger model sizes trained for extended periods of time on better hardware.

### 3.2. Transformer Performance

We find that our fine-tuned transformer was successfully able to generate highly believable conversations with a similar distribution to our training data.

Using a cross-entropy loss on next-word prediction as our training metric, we observe that the average loss goes down over training, albeit noisily, as seen in Fig. 4.

We calculate several metrics on a separate test dataset, and find that the model achieves impressive results on the BLEU [7], NIST [12], and METEOR [2] auto-evaluation metrics as compared to the base model. Our trained model achieved **4.64x**, **2.09x**, and **9.96x** higher BLEU, METEOR, and NIST scores respectively against the base GPT-2 model, as seen in Table 2.

We see an example chat generation in Fig. 2, which is strikingly similar to a real ‘Among Us’ conversation.

```
Yellow crewmate I'm not a member of this team.
I was in the room with Blue and the other players.
I was just there to discuss the situation with Blue.
But I'm not sure what to think about it. I don't know
who this person is. I didn't see anyone else around.
I think we should just not discuss anything because we
didn't know anything about each other. Let's not try
to keep anyone from seeing us. Let's just keep an eye
on each other and not start any more fights.

Red crewmate I'm not sure if they're dead or alive,
but I think they were in the area, so I'm not sure.

Green imposter I was in the area, so I was in the
area, but I didn't see anyone else around. I don't know
who it was or what it was doing, but I was in the area
with my friends, so I was in a good spot. I didn't see
anyone around, so I didn't see anything suspicious.
```

Figure 2. Example output of the fine-tuned model.

## 4. Discussion

### 4.1. RL Insights

We have two separate reward curves, one for the *crewmates*, and another for the *impostors*. The *impostor* tends to learn notably faster than the *crewmates*, which causes instability during training. To alleviate this, we periodically save the *impostor*’s weights and train against these past (weaker) versions. We hypothesize that the *crewmates* learn slower due to higher variance in the maximum possible reward they can achieve as compared to the *impostor*.

We kept the size of our model small (65k params), which produced strong results given that we had a relatively small,  $5 \times 5$  grid. We also note that having a more aggressive discount factor of  $\gamma = 0.8$  performed better on this smaller

board. PPO produced strong results, but was limited by our computational ability to tune its hyperparameters. It also seemed to prioritize exploitation, which was mitigated by using entropy regularization. We also discovered that using a convolutional neural network as the backbone of our model architecture successfully helped encode the spatial information in our  $5 \times 5 \times 4$  game state.

We observe that the *crewmates* naturally develop a strategy to stay on the *task* square until the end of the game to keep collecting rewards. This led to strong policy updates in PPO which caused instability in training. To incentivize movement, we constantly shift the position of the *task* squares throughout the game, though this may also be remedied in the future by having a more complex reward function.

### 4.2. Transformer Insights

We considered several sources for our dialogue dataset, such as scraping chat logs from YouTube videos of ‘Among Us’ games and online datasets. We found that *generating a synthetic dataset* by prompting OpenAI’s ChatGPT API as described in 6.2.1 produced a richer dataset for us to train on. During training, we observe that our loss goes down quickly, but noisily, even with a low learning rate. We attribute this to the data distribution being complex, given that it was generated using a black-box API.

As the size of GPT-2 is relatively large (117M parameters), we accumulate our batched sequences together and compute the gradient of the resulting, more-informative sequence to reduce the computational cost. Empirically it appeared a batch size of 4 performed the best.

We considered two approaches for test-time generation. One approach was to utilize the previous history to generate the next dialogue. The second approach was to generate each player’s response independently using the same prompt. We noticed that the second technique performed better and was naturally faster. Moreover, our model tended to generate repetitive sequences using the first technique. In both cases, the model would sometimes generate delusional messages such as players referring to themselves as dead, which is to be expected with limited training.

The ‘temperature’ parameter controls the *novelty* of the generated text (higher temperature = more novel). By controlling this parameter, we were able to produce higher quality results and prevent the model from straying too far away from the desired domain while maintaining novelty.

### 4.3. Conclusion

Thus, in this work we successfully simulated two different aspects of the game, ‘Among Us’ - the decision making aspect as well as the conversational aspect, using PPO and transformers, each a state-of-the art approach.

## References

- [1] Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, Mojtaba Komeili, Karthik Konath, Minae Kwon, Adam Lerer, Mike Lewis, Alexander H. Miller, Sasha Mitts, Adithya Renduchintala, Stephen Roller, Dirk Rowe, Weiyan Shi, Joe Spisak, Alexander Wei, David Wu, Hugh Zhang, and Markus Zijlstra. Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022. [1](#)
- [2] Satantjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. [3](#), [5](#)
- [3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. [6](#)
- [4] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. [6](#)
- [5] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016. [2](#)
- [6] OpenAI. Introducing chatgpt, November 2022. [Online; posted 30-November-2022]. [6](#)
- [7] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. [3](#), [5](#)
- [8] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. [1](#), [6](#)
- [9] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. [2](#)
- [10] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. [2](#)
- [11] Francois St-Amant. How to fine-tune gpt-2 for text generation, May 2021. [Online; posted 8-May-2021]. [2](#), [5](#)
- [12] Krzysztof Wołk and Danijel Koržinek. Comparison and adaptation of automatic evaluation metrics for quality assessment of re-speaking, 2016. [3](#), [5](#)

## 5. Appendix 1

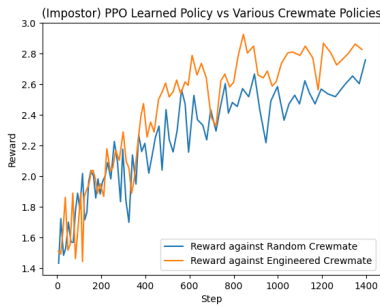
### 5.1. Reinforcement Learning Performance

We compare the results of our trained PPO model as compared to purely random agents and agents that use a engineered policy. ARPE = Average Return Per Episode is the metric used to compare them.

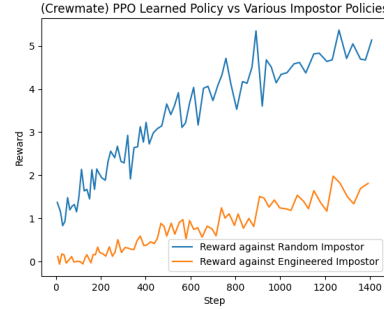
	PPO ARPE	Random Policy ARPE	Engineered Policy ARPE
<b>Impostor</b>	2.26	1.48	2.96
<b>Crewmate</b>	4.34	1.44	5.65

Table 1. Average Return Per Episode for *impostor* and *crewmate* agents, comparing PPO vs random policy vs engineered policy

We also plot the return (discounted reward over the entire episode) of the PPO model through its training in Figs. 3a,3b.



(a) *Impostor* policy network reward over training episodes



(b) *Crewmate* policy network reward over training episodes

Figure 3. Reward curves for the PPO models against random and engineered opponents

### 5.2. Transformer Performance

We compared the performance of the GPT-2 model from HuggingFace [11] before and after fine-tuning on our dataset for the BLEU [7], METEOR [2], and NIST [12] auto-evaluation metrics and detail these results in table 2. We used the cross-entropy loss for next-word prediction as our training loss and achieved results seen in Fig. 4

	BLEU	METEOR	NIST
<b>GPT-2-Baseline Pretrained</b>	0.102	0.125	0.527
<b>GPT-2-PostTrained</b>	0.473	0.261	5.248

Table 2. A comparison between the baseline GPT-2 and our fine-tuned GPT-2 on various auto evaluation metrics

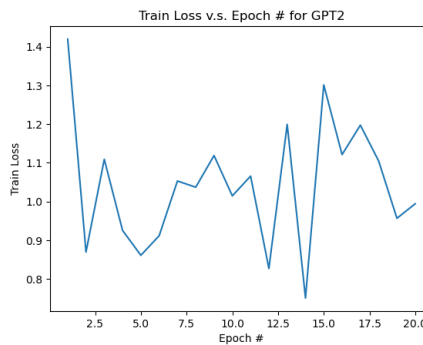


Figure 4. Training loss over time while fine-tuning GPT-2

## 6. Appendix 2

The entire repository for the project can be found at: <https://github.com/pimpale/omegasus>. The transformer training script was inspired by <https://github.com/francoisstamant/lyrics-generation-with-GPT2>.

### 6.1. RL Architectural Overview

We initialize two identical sets of neural networks to learn the policy of the *crewmates*,  $\pi_{c,\theta}(s)$ , and the policy of the *impostors*,  $\pi_{i,w}(s)$ . Each set consists of an *Actor* network and a *Critic* network which are formulated as CNNs :

Actor Network	CONV	ReLU	FC1	ReLU	FC2	SoftMax
Parameters	$3 \times 3$ out channels = 10 $p = 1$	N/A	$250 \times 256$	N/A	$256 \times 5$	N/A

Table 3. Actor Network architecture

Critic Network	CONV	ReLU	FC1	ReLU	FC2
Parameters	$3 \times 3$ out channels = 10 $p = 1$	N/A	$250 \times 256$	N/A	$256 \times 1$

Table 4. Critic Network architecture

The training was done for a total of 400 episodes, with a reward discount factor of  $\gamma = 0.8$  and a learning rate of 0.0001 for both the Actor and Critic networks using the Adam [3] optimizer and PPO loss. We add saved versions of the model to a pool of random agents every 200 episodes and sample from this pool to train via self-play. Thus, initially, we train only against random agents and by the end, primarily against past versions of the model.

### 6.2. Transformer Architectural Overview

#### 6.2.1 Data Collection Methods:

We *synthetically generated training data* by prompting ChatGPT [6] to pretend like it is playing ‘Among Us’. We engineered the prompt to generate data in an easy-to-parse format, giving us data as seen in Fig. 5. The dataset consisting of a total of 4572 dialogues and the prompt script to call ChatGPT’s API can be found in our code repository linked above.

```
Green crewmate "Ayooo crewmates, I found Red's lifeless body in 02...
Yellow crewmate "OMG, that's insane! I was doing my tasks in Shield...
Blue imposter "Wow, that's crazy! I was in Navigation doing my ta...
Green crewmate "Guys, I was in Electrical fixing the wiring when t...
Yellow crewmate "Guys, let's not waste any time. We need to vote of...
Blue imposter "Ha! These crewmates are so sus, they don't even kn...
Green crewmate "Wow, it's getting intense! I'm glad we're all on t...
Yellow crewmate "Let's focus on finding the imposter and completing...
Blue imposter "I have to admit, it's pretty funny watching these ...
```

Figure 5. Example data synthetically generated from ChatGPT used to train our model.

#### 6.2.2 Architectural Choices

We used HuggingFace’s implementation of GPT-2 [8]. The model uses 12 transformer blocks, with 12 attention-heads, an embedding size of 768, and a total of 117M parameters. We downloaded the weights for this model via HuggingFace’s *transformers* library and consider this version to be the baseline model that is then fine-tuned on our dataset.

We found that training with a batch size of 4, learning rate of  $2e^{-5}$ , using the AdamW [4] optimizer with a max input sequence length of 1024 produced consistent results. Prompting the model with a richer initial prompt and a temperature of 0.7 improved the generation quality.