How to Catch a Rabbit: Reinforcement Learning in the Stag Hunt Dilemma

Adrian Manhey
amanhey@oxy.edu
Occidental College

1 Introduction and Problem Context

The current technological climate in the United States promotes many of its advancements, such as in artificial intelligence (AI) and machine learning, despite a significant number of people lacking understanding of what these technologies rely and how they operate. Even as a senior Computer Science student, I am unfamiliar with many of the technologies that are gaining increasing importance in the field and in society, as seen by increased public interaction with AI technologies such as OpenAI's ChatGTP [11].

In order to introduce myself to some unfamiliar areas of computer science, I decided to use my senior comprehensive project to explore an area of AI called reinforcement learning. I found interest in this area from a reading group I was part of, which discussed various theories involved in AI, introducing me to reinforcement learning through a research article. Once I had an area I wanted to explore, I decided to apply it to the context of my mentor's research project, stag hunt, discussed in further detail in section 2.

The stag hunt environment focused on the trade-offs between collaboration and competition between actors in an environment, where the choices of one actor directly affects the outcome of the choices of other actors. Considering reinforcement learning and stag hunt together led to the question "Can an agent without prior knowledge of an environment learn to collaborate through experience?" An alternative way to think about this question is "How do we construct a training environment such that an agent can learn the value of collaboration without any explicit representation?" As AI is becoming increasingly integrated with the public, understanding AI's social interactions, such as the ability to balance collaboration and competition, become of vital importance. This idea will be further explored in the section 7.

Since the area of study was new for me, some of the main sources of difficulty of the project were understanding reinforcement learning concepts and understanding where, when, and how those concepts could be applied. There are constraints to different techniques, so understanding which techniques to use for this project was a significant issue, especially in the beginning. In order to solve these issues, my mentor suggested reading a portion of an AI textbook [13], which was an incredible resource in building my own knowledge base on reinforcement learning and the techniques I would ultimately use.

2 Technical Background

Before describing the technical knowledge needed to understand the project, it is important to note that the Russel & Norvig text was used as primary reference material for the project. Thus I attribute some of the ideas and many of the definitions of this section to that text, specifically Section 5, Chapter 21 which covers reinforcement learning.

An **agent** is an entity that can observe and act within an **environment**. For example, a person grocery shopping can be considered an agent with the task of buying groceries in the environment of the grocery store/market. In an environment, there are different **states** $s \in S$ which represent different scenarios of the environment, such as the set of positions for characters on map. When the positions of characters change, the state transitions to another state with the current positions. After these transitions, the agent receives a **reinforcement**, or **reward**, from the environment [13]. For example, if a person buys an expensive or unnecessary amount of groceries, they might receive a negative reward in the form of the cost of the groceries. This reinforcement might lead to the person buying less groceries the next time they're shopping.

An agent operates using a **policy**, which maps a set of possible states to a set of possible actions the agent is able to take. In **reinforcement learning**, the goal is to use the received rewards to find an optimal policy. Additionally, in the field, the set of states and actions are often represented as S and A and referred to as the state-space and action-space, respectively. individual states and actions represented as s and a. The policy is represented as $\pi(s)$ and gives the an optimal action a for a state s. The policy uses a **value function** which estimates the value of a state or state-action pair (s, a).

A policy can use different methods to model the transition between states. **Model-based** methods have a model to specify the probability of moving from state s to s' by action a. However, if the environment is affected by factors other than the agent, explicitly defining a model can be difficult. **Model-free** methods don't require a pre-defined model and can be thought of as using dynamic programming for an expansive brute force search. If the agent can predict the reward for some action before actually performing it thereby planning what it should do, the algorithm is model-based. While if it actually needs to carry out the action to see what happens and learn from it, it is model-free [10].

$$Q(s,a) = R(s) + \gamma * \sum_{s'} P(s'|s,a) * max_{a'}(s',a')$$

Figure 1. shows the updating of a Q-Value without using temporal difference, but using a transition model P.

Despite the method, a policy needs to have a **value function** which calculates or estimates the value, or utility, of a state. In **Q-Learning**, a traditional reinforcement learning technique, the utility of a state is represented as a Q-value, which is adjusted based on the observed reward 1. Q-Learning goes through the entire state- and action-space and stores Q-values $Q(s,a)$ for each state-action pair (s,a) in a table. As a Q-Learning agent is trained, the Q-values are updated using a greedy policy, where the model selected a random action $\frac{1}{\epsilon}\%$ of the time. This greedy policy should be Greedy in the limit of infinite exploration (GLIE), so that as the number of epochs increase the agent should abide by its developed policy more.

$$\hat{Q}(s,a) = \theta_0 + \theta_1 * f_1(s,a) + ... + \theta_n * f_n(s,a)$$

Figure 2. shows an evaluator function for Approximate Q-Learning.

One alternative to Q-Learning, is to use an **evaluator function** to approximate the value of states rather than storing the values in a table. Evaluator functions calculate use **features** to calculate the value of a state. For example, if an agent was being trained to play virtual soccer, an important feature might be the distance of the ball to the goal, since if the distance decreases enough, the ball enters the goal and the agent scores a goal. The features can then be used by a technique such as **linear function approximation** to calculate the combined value of a set of features. In this technique, the sum of features multiplied by weights is calculated 2. The resulting value may vary by implementation but must follow the general organization seen in figure 3.

$$R_{step} < R_{hare} < \frac{1}{2} * R_{stag}$$

Figure 3. Shows the relationship of rewards where R_{step} represents the penalty of taking a step, R_{hare} represents the reward for catching a hare, and R_{stag} represents the reward for catching a stag.

In order to update without a transition model, we can use **temporal difference**. This method uses the fact that if a state is good, then states close to it should also be good. The method for updating a Q-value consists of two parameters: $\alpha \in (0,1]$ is the learning rate and $\gamma \in [0,1]$ is the discount factor, which makes short-term or long-term rewards more valuable. The equation for this relationship can be written as

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(r + \gamma max Q_a(s',A)),$$

where r is the reward of an action and s' is the next state. So we are learning the proper action by taking the old Q-value and adding the learned value of the immediate reward and possible future rewards.

In terms of optimizing the environment after training, **grid search** can be used to find optimal hyper-parameters (e.g. α, ϵ, γ). This generates a range of values for each of the parameters and then trains the agent using those values. As each game is completed, variables to track the set of hyper-parameters that resulted in the best performance are updated.

The environment of these virtual agents is the stag hunt game 5, a stochastic prisoner's dilemma-style game. In stag hunt, the goal is to accumulate the highest number of points by capturing high-value mobile targets (stags) or low-value static targets (hares). Agents, whether virtually created or human players, control hunters which can hunt stags or hares, but stags require two or more agents to capture them. All moves happen simultaneously and agents are allowed to occupy the same spaces. Although the rules of the game are simple, there are hidden complexities to finding an optimal policy for this sequential decision making process in a multi agent system.

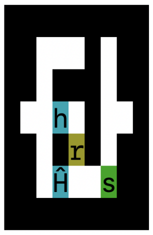


Figure 4.

An example stag hunt scenario generated by the environment.

In this project's implementation, it is important to note that the map of each state has a surrounding border wall, so the matrix representation is padded with zeros. When referring to the hunter being trained I will use the term **subject**, while using companion hunter(s) to refer to the other hunters in the environment.

3 Prior Work

3.1 Game Theory

At the moment, many of the solutions to stochastic games involve finding the Nash equilibrium but these methods do not provide strong explanation for human behavior in these environments. Battalio et al. [4] utilize a framework, based on the assumption that agents (human subjects) are highly rational, that can be explained using Nash equilibrium, wherein an agent changing their behavior does not benefit them if the behavior of other agents does not change.

They changed the value of the expected earnings difference between two actions, termed optimization premium, and found evidence that, for individual agents, the larger the optimization premium, the more responsive they are to beliefs and the quicker they are to converge to an equilibrium. However, they also found that the model failed to capture some important aspects of behavior, such as the observed bias of the payoff-dominant action (the action that results in the highest payoff for each agent).

3.2 Reinforcement Learning

One line of research explores this problem from a reinforcement learning perspective. A notable example of this comes from the Microsoft Collaborative AI Challenge (MCAC) [16], an Minecraft mini-game where participants must train an agent to play *Pig Chase*. This is a version of stag hunt which has two hunters and one stag, represented as a pig, and the other hunter, the Blue Agent, can be either random or guided by a heuristic. Note the Blue Agent was programmed to chase the target 75% of the time and moved randomly 25%. Additionally, the low value targets are represented as exits rather than hares. The winning implementation of this challenge, HogRider, proposed by Xiong et al. [18], was created using an adapted Q-Learning approach.

Traditional Q-Learning becomes unreasonable for large state-action spaces, such as in stag hunt. Although Pig Chase is a simplified version, it still suffers from the same challenges as stag hunt, seen by its 4,057,200 possible number of states. In order to account for this, they created evaluator functions 2 in order to abstract the state and action spaces. For example, one of the evaluator functions used calculated the relative distance between the agent and the pig.

The functions are used to estimate the utility of different states and actions, without having to store data for every combination of states and actions, and are able to do this with features that are highly correlated with the chance of winning. Due to the stochastic nature of stag hunt, the changes to states are not only affected by the agent but also the other agents in the game, since both the hunters and stags are able to move around the game space.

This approach blended approximation and traditional Q-Learning by using approximation to reduce the possible number of states and then storing tabular Q-values for the resulting set of states. The result was 240,000 possible states, a 94.1% reduction and the model was shown to outperform other models and even human players, although the number of participants, 10, was small.

Another line of research studies multi-agent systems using neural networks in order to train agents to play stag hunt. Nica et al. trained agents using a convolutional neural network (CNN) with reinforcement learning called a DQN [9]. The approach taken is based on variations of the on-line policy gradient methods REINFORCE and Actor-Critic algorithms with auxiliary tasks. Unlike the manually-defined evaluator functions used by Xiong et al., DQN trains a CNN to find evaluator functions that are able to make good approximations. Additionally, this method does not store any tabular Q-values rather solely relying on the resulting CNN to estimate the values of states and actions.

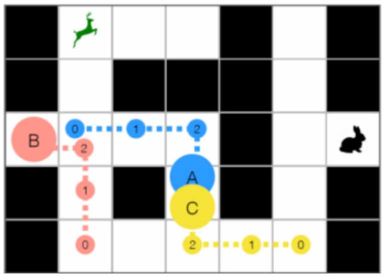
This technique, while effective, does not translate the trust dilemma of stag hunt to the agent. Although using a CNN is a powerful tool, it obstructs the underlying model the agent develops. Rather than training an agent to recognize the value of different states, what if the agent could be taught to consider the trust dilemma by accounting for the perspective of the other hunters. This can be done using theory of mind reasoning, which can be described as the ability to make inferences about others' mental states (e.g. their knowledge, goals, etc.). By considering the goals of another hunter, an agent may be able to make inferences about that hunter's future behavior. For example, if an agent knows that the other hunters have a preference towards catching the hares, then the agent should prioritize catching a hare as well.

3.3 Theory of Mind

One of the methods developed from this approach was Shum et al., in which they created a model for group collaboration called Composable Team Hierarchies (CTH)[15]. This method uses Baysian inference to provide agents with a structure for collaboration. Although this method is able to solve stag hunt for small groups, it still suffers as the number of characters increases in the game. Unlike the DQN which does not require an explicit representation, CTH requires a representation but no training. It is important to note that their model had strong correlations with the predictions made by human subjects in their experiments.

In a similar vein, Rabkina et al. proposed the Analogical Theory of Mind (AToM) model, which posits that a combination of analogical processes and feedback lead to the

Figure 5.



An example stag hunt scenario, taken from scenario (d) of figure 1 of Rabkina et al. (2019), where a pair of agents (A and B) cooperate to capture a stag and another agent captures a hare individually. The circles represent hunters position at a given time-stamp and the dotted lines represent movement.

development of theory of mind reasoning without an underlying model of cooperation[12]. From the experiments run, they found that the AToM model is more accurate than the Bayesian model at most time stamps and does not require a predefined underlying hierarchy of team cooperation. Furthermore, the ability to make predictions about the future actions of agents is an additional effect of reasoning about an agent's cooperation with others.

The AToM model relies on a Many Are Called/Few Are Chosen (MAC/FAC) model of analogical retrieval and Structure Mapping Engine (SME) in order to make inferences. MAC/FAC takes the dot-product of content vectors to find similar previously-observed cases between a probe case (an inference) and a case library (case data from simulations) then uses SME to create a set of inferences based on the structural similarities of two cases, a base case and a target case.

In order to solve the stag hunt dilemma, reinforcement learning without neural networks was chosen. Although the theory of mind approaches allow the consideration of trust by accounting for the perspective of others, the reinforcement learning methods were more accessible and better addressed the guiding questions. The non-technical motivation for using Q-Learning was it's establish history of usage, especially through MCAC. There were many articles, tutorials, and resources available that detailed Q-Learning for a wide array of applications. As a beginner to reinforcement learning, I wanted to choose a method that I could understand, implement, and also improve upon, which was done with the approximation methods.

The technical motivation for using Q-Learning was it's off-policy and model-free properties, meaning that the agent will continue to explore the environment even as a policy is developed and it does not need a pre-defined model for the environment in order to learn. The implementation used a constant Greedy-Epsilon policy and temporal difference, to reduce the time complexity of the algorithm. The utility of each state was calculated by the reward the subject would receive, so if the state had the subject and a rabbit in the same position, the reward of that state would be the value of the rabbit. The Q-Learning agent was trained using $\alpha = 0.1, \epsilon = 0.4, and \gamma = 0.8$ on 100000 episodes. A majority of these properties were touched upon by Xiong et al. [18], best seen by the influence on the final version of the companion hunter.

4 Methods

4.1 Creating the Environment

This project began with a tutorial from OpenAI which walked the reader through the process of training a virtual taxi cab to pick up people at different locations. At first, this project seemed similar to stag hunt as it was a grid-based game with a discrete action-space that used reinforcement learning to train an agent to locate a target. However, the taxi cab problem lacked some of the essential complexity that comes with stag hunt, such as multiple mobile characters that affect the options available. Accounting for a multi-agent environment, where there is more than one agent whose decisions affect the environment, changed the methods from focusing on Q-learning to using approximation methods.

The tutorial provided a loose framework when the environment was being created. Although the initial plan was to use OpenAI's Gym environment [1], as in the tutorial, a new environment was created from scratch since this project required a custom environment and the labor to define one in Gym seemed comparable to creating one. The structure of the custom environment has similarities to the taxi cab environment, however, a notable difference is that the new environment queries and updates agents rather than vice-versa. This change made creating multiple kinds of agents easier, as different ones were needed at different steps of the development process. For example, at the beginning of development I was able to test out functionality with a static, non-mobile agent without having to create the agent logic. This made the development process simpler, allowing for simultaneous development of the environment and the agents instead of completing the environment and then the agents.

The final environment mainly consisted of a map, a game length (30), a state, an interaction manager, an encoder, and a character registry, the last 4 being their own classes and the last 2 having their own test classes. The environment's map was a 7x7 grid seen in 5 where the rabbits and stags were valued at 5 and 20, respectively. For each agent, the cost of taking one step was -1.0. However, to promote collaboration with the companion hunter and exploration in the environment, the step cost was reduced to -0.5 and the value of the stag was increased to 50, as in MCAC [16]. The encoder converted the map between a string representation and a matrix representation and characters between numerical and string ids. The interaction manager handled group interactions and calculated the value of a group interaction (e.g. assign hunters points for successful capture. The character registry stored the position and agent behind every character in the game, also handling CRUD operations on the registry. Each of the states are represented by an instance of the State class, which stores data about the string id, map, character positions, current step count, and the position of the subject. Any state representation is done by this class and the string ids are the values stored in the Q-Table during learning.

4.2 Accounting for State-Space

The initial environment consisted of a 4x4 map with one hunter and one rabbit, used to test out game mechanics. Then different agents were developed to test out game mechanics or to act as companion hunters. The agents that were developed were 1) StaticAgent: a non-mobile agent, 2) RandomAgent: a mobile StaticAgent which moves randomly, 3) StaghuntAgent: a RandomAgent with stag hunt attributes (e.g. type, reward), 4) PreyAgent: a StaghuntAgent representing the stag which moves away from approaching hunters within 2 spaces or moves randomly, and 5) BasicHunter: a StaghuntAgent that acts as a companion hunter. The last class, BasicHunter, had different iterations, but the two most important were a version that moved randomly and the final version which had a 75% chance to target the stag and 25% for the rabbit, and used a Breadth-First-Search (BFS) to locate its target. The final version was inspired by the Blue Agent from MCAC, and was implemented to promote more collaboration between the subject and the companion hunter after tests revealed the subject had a bias towards competition and claiming the rabbit.

After developing the companion hunter and the environment, training began with a Q-Learning agent. This method worked well for smaller environment and a lower number of characters, however as the size of the map and number of character increased the amount of memory required for the dictionary used by the Q-Learning agent became unreasonable. The size of the state-space can be calculated using,

$$S(n,m) = \frac{n!}{(n-m)!}$$

, where n is the number of possible positions and m is the number of characters. As seen in 1, the state space can grow to be quite large, so for a more complex version of the stag hunt game on a 7x7 grid with 7 characters, Q-Learning becomes an inefficient solution.

In order to account for the large state space, I tried to reduce both the number of characters and the map size. The characters considered went from three hunters, two stags, and two rabbits to two hunters, one stag, and one rabbit. This reduction allowed for a smaller state-space while also preserving the fundamental properties of collaboration and competition in the game. The map was changed from an open 7x7 grid to a layout defined in [15], which added additional walls to the interior of the map, reducing the number of possible positions from 49 to 20. This led to a much more manageable state space of 116280, which allowed for the training of a Q-Learning agent to compare the approximate learning agent to.

n	m	S(n,m)
4	2	12
49	2	2352
49	7	$4.33 * 10^{11}$
20	7	$3.91 * 10^{8}$
20	4	116280

Table 1. Sizes of state-space for different sizes of maps and number of characters.

4.3 Q-Learning

One of the difficulties with reinforcement learning is that training time can slow down the development process. In order to monitor the agent during training, a live plot display was added to show the number of epochs and total reward for some percentage of training episodes. The plots would be update live for every k^{th} episode and then could be saved at the end of training. This allowed for a decrease in wasted training time, since if an implementation was exhibiting poor performance, the training could be stopped early and changes could be made. Additionally, a parameter δ was added to the Q-Learning agent to monitor convergence of the values in the subject's Q-Table. Meaning, that for $\delta = 0.001$ if the values were changing at a rate less than 0.1%, it might suggest that the values in the Q-Table have converged and the number of training epochs may be adjusted.

4.4 Approximate Learning

After Q-Learning agent was implemented, development on the approximate learning agent (i.e. ApprxReinforcementAgent) began. The evaluator function consisted of a sum of weights multiplied by their respective features, each stored in a dictionary. In the first iteration of development the approximate learning, hereby referred to as the AL agent, the features selected consisted of closest prey, closest hunter, number of steps taken, direction of hunter, position of character. This set of features was tested in the environment and the results were poor, exhibiting behavior that was seemingly random. In the next iteration of features a similar reinforcement learning tutorial offered by Berkeley [2], motivated me to focus on proximity and distance as factors, leading to using the number of adjacent prey, the number of steps taken, the number of turns between hunter and each character, and the distance between hunter and each character. These features still suffered from similar outcomes and had the additional cost of using a BFS to calculate distances at each step. Although the overall performance was poor, the distance based features seemed to be a worthwhile implementation, so the next set of features used the distance between each of the hunters to each of the rabbits and stags. However, in order for the distance features to be useful I needed to shape them in the correct manor. By reading [13] I learned that the evaluator function must be a linear combination of the probability of winning the game at a state. Furthermore, it must preserve the ordering of the value of the states, meaning that if the evaluator function $Q(s)$ produced a value for a bad state s_1, a moderate state s_2, and a good state s_3, they must have the relationship

$$Q(s_1) < Q(s_2) < Q(s_3)$$

, or in the opposite order. In order to accomplish this, the distance $d_{i,j}$, where i is the i^{th} hunter and j is the j^{th} character, would be represented as a percentage of the maximum possible distance d_m,

$$\text{feature[i-dist-to-j]} = \frac{d_m - d_{i,j}}{d_m}$$

.

These distance-based features led to a breakthrough in the project where the AL agent was able to complete the game. However, the agent was not exhibiting collaborative behavior, in fact not once catching the stag in any of the tests. It seemed that the agent had learned to game the system, shown in the test games consisting of either the subject or companion hunter catching the rabbit. At this point the aforementioned changed to the companion agent and the environment to promote collaboration were made. Additionally, instead of just using distance formula "i-dist-to-j", the percentage of the reward if a prey was captured was included. These features took distance $d_{i,j}$ to define a payoff,

$$\text{feature[i-payoff-to-j]} = \frac{v_j - d_{i,j} * C}{v_j}$$

, where the v_j is the reward for capturing the j^{th} prey character and C is the step cost. The payoff and distance features were effective, discussed further in the 6 section.

4.5 Optimization

One of the core aspects of the project was optimization. The two largest methods of optimization were using a profiler and running a Grid Search on the hyper-parameters α, ϵ, and γ, the results of which can be seen in 6. Each of the hyper-parameters was tested with values of 0.2, 0.4, 0.6, and 0.8 over 10000 epochs each. By finding the optimal hyper-parameters, there is more control on the behavior of the model. Depending on which set of values are selected, different aspects of the environment can be emphasized. Although the first row in 6 results in the lowest epochs and highest reward, other configurations show that one metric can be improved by accepting sub-optimal results for the other.

Kind	α	ϵ	γ	Epochs	Rewards
Optimal	0.2	0.6	0.4	3.0	2.4
	0.8	0.8	0.6	3.0	2.4
Optimal Epochs	0.4	0.8	0.4	2.9	1.9
Optimal Rewards	0.6	0.2	0.6	3.9	2.7

Figure 6. Grid Search for alpha, epsilon, and gamma with each having values 0.2, 0.4, 0.6, and 0.8 with 10000 epochs each.

Additionally, a profiler was used to assess the time taken for the initialization, reset, step, and render methods of the environment as well as various training and testing methods and the agent logic. The profiler found that the method of encoding the map into a string id was inefficient, so instead of looping through the matrix representing the map, some of Python's built in functions, such as join() and flatten(), were used instead. Additionally, the BFS used by the evaluator function when calculating distances was another source of time complexity. To address this issue, a cache was implemented so that the features calculated for 20 states was saved. Each time a state that wasn't present in cache was encountered, the state with the least number of accesses was replaced by the new state.

5 Evaluation Metrics

The evaluation metrics used for testing out the agents were based on the number of epochs and the total reward at the end of the game. When training, these are the two metrics used in the plot display to indicate the intermediate performance of an agent. When testing an agent, the average number of epochs and rewards is calculated and represents the performance. These metrics were chosen because of their prominence in the literature, including Xiong et al. At the end of each training episode, 10 random test scenarios are generated and the frames 5 and results of each test game are displayed in the terminal.

Due to the emphasis on collaboration in this project, a new metric was added to represent successful cooperation (i.e. catching the stag) or if an agent seemed to attempt to capture the stag and the companion agent chose to capture the rabbit instead. However, since it is difficult to know the

”intentions” of an agent, who is only following a policy, this metric is calculated as the percentage of the test cases where the stag was captured. Another reason for this addition is that although the Q-Learning agent was able to complete the game, it was quite seldom collaborating with the other agent. This issue was discovered after examination of the frames of the test cases, which print at each time step.

With additional time, I would have liked to explore the limits of Q-Learning in stag hunt in more depth, gathering metrics such as an upper limit on the size of the state-space. It is apparent that when the table size increases into billions of entries it is unreasonable, but at what point does it change? Since Q-Learning ensures accuracy, in the long run, and approximation reduces the space complexity, it is important to understand when each should be used.

Furthermore, I feel that the metrics could have been made more robust by attempting to capture when an agent intends to collaborate and is unsuccessful. By measuring the similarities of an agent’s chosen path with a path to a character, that may give insight into which prey and agent was targeting. In a similar fashion, I would want to evaluate the performance of an agent using a relative reward which would account for characteristics such as the distance of each of the hunters to the prey at the starting state of a test game. This would account for situations where the model would exhibit optimal behavior but other factors prevented them from doing so. These metrics would also help to answer questions around whether through play-based experience, an agent ”concludes” whether collaboration or cooperation is more beneficial.

6 Evaluation Results and Discussion

Feature	Weight
bias	-4.26
h1-payoff-for-r1	15.84
h1-dist-to-r1	-14.31
h1-payoff-for-s1	-19.66
h1-dist-to-s1	3.44
h1-dist-to-h2	35.32
h2-payoff-for-s1	-16.85
h2-dist-to-s1	2.03
h2-payoff-for-r1	12.38
h2-dist-to-r1	-12.58
h2-dist-to-h1	35.32

Figure 7. Trained on 100000 epochs this agent chose to try and capture the stag 60% of the time, averaging 6.9 epochs and 1.5 rewards with a step cost was -0.5.

After training the Q-Learning and Approximate Learning Agent, it was surprising to see the comparative performance between the two. In a set of test cases, the Q-Learning agent captured the stag 5/10 times while the AL Agent captured the stag 6/10 times. One caveat is that since the test cases are generated randomly, it is difficult to assess how the decisions in one situation would translate to another. Further, in other tests the Q-Learning Agent ended testing with an average number of epochs of 5.3 and average reward of 10.2, while the AL agent ended with 6.9 and 1.5, respectively. These tests can be seen in the evaluations under the terminal outputs section of the project. These tests suggest that although the features are able to train the agent to play stag hunt, it doesn’t do as well as the Q-Learning agent.

One of the things that stood out was that the percentage of times that the hunter captured the stag in the tests was inconsistent. At times, the proportion was good, as previously mentioned, but there are training sessions that continue to result in the agent prioritizing the rabbit. This suggests that additional features must be added in order to capture the value of the stag. Also, shaping the rewards again might prove to be useful since the decrease in step cost and increase in stag value, seemed to promote ”collaboration”. These results are interesting and suggest that further investigation into different evaluation metrics, such as the relative reward for states, is needed.

7 Ethical Considerations

Social interactions with AI. This project using a Q-learning reinforcement learning algorithm in order to train a Python agent to play stag hunt, a prisoner’s dilemma style game that emphasizes cooperation between agents in order to reach the optimal outcome. In essence, agents are given the option to work independently for a small reward or work cooperatively for a large reward. Developing this technology informs us of methods that may be used to model an agent’s intent recognition [12], meaning their perception of the decisions intended to be made by other agents. Since this game involves autonomous agent control, long-term decision making, interaction between an agent and their environment, consequences for decisions, and temporal reasoning, it’s a good candidate for reinforcement learning. By making the decision to develop the project using reinforcement learning, I must acknowledge that I am also choosing to develop the field of reinforcement learning, AI research, and contributing to the applications of each.

7.1 Concentrating Power through Computing Power

Although reinforcement learning seems to be a good fit for this problem, it is important to consider some of the ethical issues surrounding it, specifically the way that this project may concentrate power through the context of its development. Reinforcement learning algorithms are recommended for problems that do not have pre-gathered data since the agent will gather the data through their trial-and-error experience. If designers are not able to access the resources in order to do these computations, a barrier is created between them and the field. This can be seen by the fact that using Q-Learning for the full stag hunt game is infeasible on a student's personal computer but someone with more resource might have more flexibility in dealing with the issue.

Due to the computational nature of AI research in the current era, where available computing power or compute is doubling at a much faster rate than before, the concern is that only a small group of actors will shape the future of AI [3]. This study looked at 171,394 papers from 57 computer science conferences and concluded that the participation of firms has increased in the form of firm-only published papers and firms collaborating with elite universities. As Benjamin 2021 said during a lecture at Occidental College, there is a "mass consolidation of power in the tech field with a small group of people who set the terms of engagement and enforce a new technocracy instead of a multiracial democracy." This means that the network of producers of AI research is getting smaller with prominent figures being academic institutions that have long-standing prestige and funds and technology firms, such as Google and Meta, which have the financial resources to attract talent and invest in hardware dedicated to developing AI technologies (i.e. GPU accelerators). Additionally, it suggests that the consolidation of technological power corresponds to a consolidation of social and political power, taking power from individual actors and concentrating it to technologically powerful entities. Meaning that these large entities not only have the power to control the production of technology in the present, but also the sequence of technologies that can be developed from each other.

A prominent example of technologies produced by a large entity are GPT-2 and GPT-3 [14]. Each are language models released in 2019 and 2020, respectively. Each was built by OpenAi, an AI research and deployment company. According to [5] the cloud computing costs alone for GPT-2 cost an estimated $50,000. Most research research groups outside of large-scale technology companies don't have the means to afford these kinds of costs, resulting in them being excluded in the development of these or comparable technologies. GPT-2 was notable in many natural language studies that used the technology since it was open-source but it's successor, GPT-3, who enjoyed a $1 billion investment from Microsoft to have an exclusive license, charges for its service. This developments demonstrate how once an institution has the means to host computational AI research, they attract more funding and researchers to amass a source of both computational and intellectual power [7].

7.2 Social Impact of AI

On the topic of GPT, OpenAI's ChatGTP has gone viral online as an interesting and cool tool to perform many different tasks. This tool has the capability to generate quality written responses and even write code. A student had even used it to create the code for a startup prototype using code libraries they had never seen before [8]. AI was previously limited in its effectiveness, which rested in more technical tasks, such as data analysis. However, as technologies develop, we are seeing AI branch into more creative fields. Despite the impressiveness of these advancements, AI does not posses consciousness and has no reference for ambiguous concepts such as the truth or a moral compass (e.g. right and wrong), seen by ChatGTP's ability to generate compelling arguments that are based on either uncited or false information.

In the current political climate where "fake news" and misinformation are growing issues, the ability of AI to generate large amounts of high-quality content raises concerns about the spread of false and misleading information online. AI systems can generate seemingly convincing arguments and narratives that are based on false or uncited information, which can be difficult for people to distinguish from genuine content. This can contribute to the spread of misinformation and can have negative impacts, such as eroding trust in institutions and undermining the ability to make informed decisions. For example, an AI system trained on a large data set of news articles and other text could be used to generate a fake news article that looks authentic. This article could contain false information or could be designed to manipulate readers' emotions or opinions. If people believe the information in the article, it could lead them to make decisions based on false or misleading information.

Another way that AI can contribute to the spread of misinformation is by making it easier for people to create and distribute large amounts of false content. AI systems can be used to automate the process of creating fake social media accounts, which can be used to disseminate false information at scale or target vulnerable individuals as the subject of scams or blackmail. At this point, AI is able to generate quality chat and create realistic images, both photos and live videos through "deep fakes" [6]. This can make it difficult for people to distinguish between genuine and fake content, which can erode trust in institutions and undermine the ability to make informed decisions.

One concern for reinforcement learning in particular is that algorithms can learn to take actions that are harmful or unethical in order to maximize their reward. For example, a reinforcement learning algorithm that is trained to control a military drone might learn to take actions that cause civilian casualties in order to maximize its chances of success in a simulated combat scenario. The capabilities of AI, coupled with complex conflicts like in Libya and Syria, have created situations where AI is killing people. One of the biggest examples has to do with drones, such as "loitering munitions", drones that use image recognition to autonomously patrol and dive bomb areas [17].

Another growing concern is the potential for economic disruption. As AI technologies continue to advance, they are becoming increasingly capable of automating many tasks that were previously performed by humans [8]. Although this can be a good thing, such as with hazardous jobs, it can also lead to job loss in a variety of industries, as machines and algorithms can often perform tasks more efficiently and cheaply than humans. This can have a negative impact on workers, as they may lose their jobs and have difficulty finding new employment in the current economic recession. Overall, as the skills of AI begin to approach that of humans, to an extent, it is important for society to develop and implement safeguards and standards to ensure that AI is used in a responsible and ethical manner.

References

[1] URL: https://github.com/openai/gym.

[2] URL: https://inst.eecs.berkeley.edu/~cs188/fa22/projects/proj3/.

[3] Nur Ahmed and Muntasir Wahed. *The De-democratization of AI: Deep Learning and the Compute Divide in Artificial Intelligence Research*. 2020. DOI: 10.48550/ARXIV.2010.15581. URL: https://arxiv.org/abs/2010.15581.

[4] Raymond Battalio, Larry Samuelson, and John Van Huyck. "Optimization Incentives and Coordination Failure in Laboratory Stag Hunt Games". In: *Econometrica* 69.3 (2001), pp. 749–764. ISSN: 00129682, 14680262. URL: http://www.jstor.org/stable/2692208 (visited on 12/15/2022).

[5] Vanya Cohen. *OpenGPT-2: We Replicated GPT-2 Because You Can Too*. Dec. 2019. URL: https://medium.com/@vanya_cohen/opengpt-2-we-replicated-gpt-2-because-you-can-too-45e34e6d36dc..

[6] Jessica Lyons Hardcastle. *Here's how crooks are using deepfakes to scam your biz*. Sept. 2022. URL: https://www.theregister.com/2022/09/28/trend_deepfake_video/#:~:text=Deepfake%5C%20%5C%2B%5C%20existing%5C%20scam%5C%20%5C%3D%5C%20more%5C%20money%5C%20for%5C%20crooks%5C&amp;text=Also%5C%2C%5C%20criminals%5C%20can%5C%20use%5C%20stolen,video%5C%20calls%5C%20to%5C%20verify%5C%20identity..

[7] W Knight. "One of the fathers of AI is worried about its future". In: *MIT Technology Review* (2018).

[8] Ethan Mollick. *Chatgpt is a tipping point for AI*. Dec. 2022. URL: https://hbr.org/2022/12/chatgpt-is-a-tipping-point-for-ai.

[9] Andrei Cristian Nica et al. "Learning to Maximize Return in a Stag Hunt Collaborative Scenario through Deep Reinforcement Learning". In: *2017 19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. 2017, pp. 188–195. DOI: 10.1109/SYNASC.2017.00039.

[10] Elisha Odemakinde. *Model-based and model-free reinforcement learning: Pytennis case study*. Nov. 2022. URL: https://neptune.ai/blog/model-based-and-model-free-reinforcement-learning-pytennis-case-study.

[11] OpenAI. *CHATGPT: Optimizing language models for dialogue*. Dec. 2022. URL: https://openai.com/blog/chatgpt/.

[12] Irina Rabkina and Kenneth D. Forbus. *Analogical Reasoning for Intent Recognition and Action Prediction in Multi-Agent Systems*. 2019.

[13] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. 1995.

[14] Ronald Schmelzer. *What Is GPT-3? Everything You Need to Know*. TechTarget, June 2021. URL: https://www.techtarget.com/searchenterpriseai/definition/GPT-3..

[15] Michael Shum et al. "Theory of minds: Understanding behavior in groups through inverse planning". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 6163–6170.

[16] *The Malmo Collaborative AI Challenge*. Mar. 2022. URL: https://www.microsoft.com/en-us/research/academic-program/collaborative-ai-challenge/.

[17] Gerrit De. Vynck. *The U.S. Says Humans Will Always Be in Control of AI Weapons. but the Age of Autonomous War Is Already Here*. WP Company, Aug. 2021. URL: https://www.washingtonpost.com/technology/2021/07/07/ai-weapons-us-military/..

[18] Yanhai Xiong et al. “HogRider: Champion Agent of Microsoft Malmo Collaborative AI Challenge”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (Apr. 2018). DOI: `10.1609/aaai.v32i1.11581`. URL: `https://ojs.aaai.org/index.php/AAAI/article/view/11581`.