**ENGINEERING DESIGN DOCUMENT**

**MACHINE LEARNING FOR GHOSTBUSTERS**

**A picture containing indoor, building, telescope

Description automatically generated**

**Team Members**

Aravind Jyothi

Ilanchezhian Iniya Nehru

Roshini Chakrapani

Sankareswari Govindarajan

Vipul Singh

**Table of Contents**

|  |  |
| --- | --- |
| Goal of the project | 3 |
| Background and Overview | 3 |
| Prior Research | 5 |
| Architecture Flow Diagram | 6 |
| Environment Setup | 7 |
| Methodology undertaken | 7 |
| Future Work | 9 |
| References | 10 |

**GOAL OF THE PROJECT**

Ghostbusters is an asymmetric hide-and-search board game with incomplete information. It is a double-sided game with searchers, called ghostbusters trying to catch the hider, ghost. The ghost will try to escape the ghostbusters and stay hidden so that it doesn’t get caught. The ghost busters’ goal is to catch the ghost by moving onto the playing board area where it is in the hiding.

The goal of the project is to develop neural network agents in which one will play the ghostbusters and the other will play the ghost. We are going to improve the performance of both the sides simultaneously by generating a neural network model.

The agents are modelled using the deep-Q-network based neural network, which is an improvement on Q-learning. It helps model a large state space efficiently.

**BACKGROUND AND OVERVIEW**

Ghostbusters is a hide-and-search board game with two sides. A group of players, ghostbusters cooperate to track down a hiding player, say ghost which is on the playing board. The board is represented by the underground of an abandoned haunted city.

The board game is played by a total of 6 players: 5 players called Ghostbusters and one hiding player caller Ghost. The game is a double-sided game in which the five ghost busters work as a team to capture the ghost in a haunted city which has different underground routes presented on the underground city map.

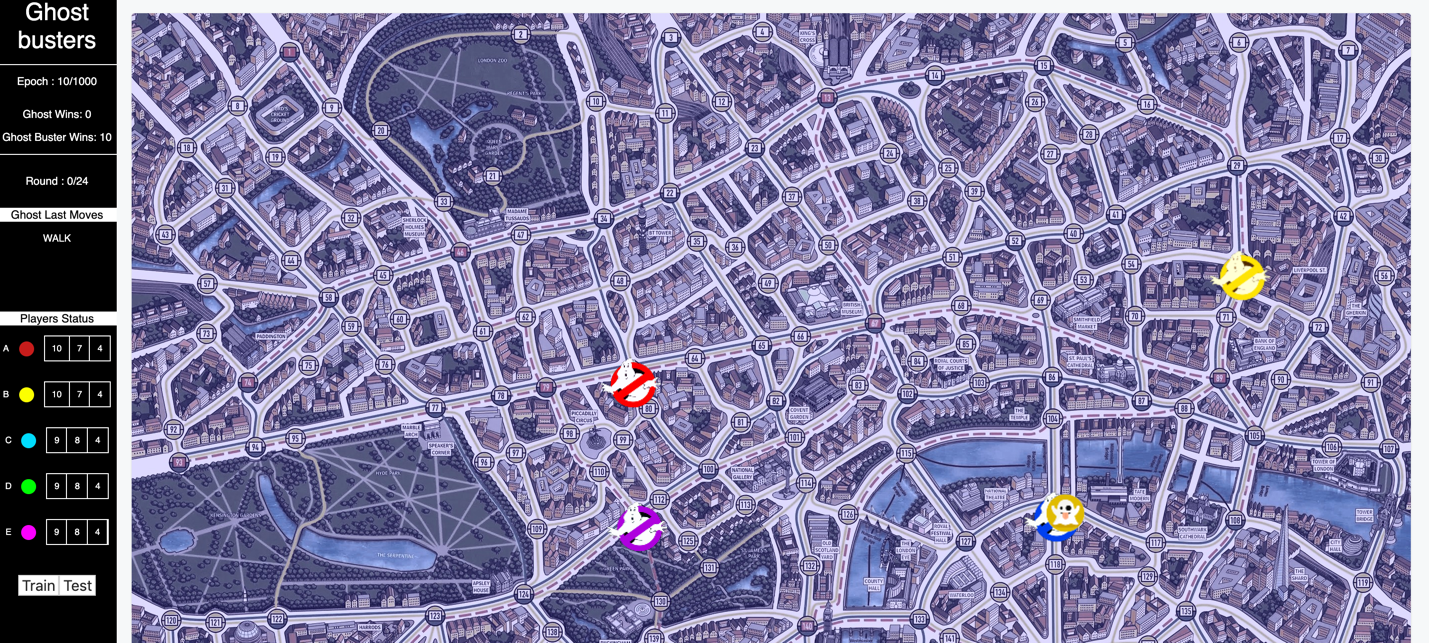
The game’s asymmetric property is contributed by the fact that both sides of the game do not stand on equal ground. The information that is available is incomplete because of the fact that the ghostbusters are unaware of the location of the ghost and they are allowed to know the location only at some specific times.

The game has three properties:

1. The ghost busters perform open moves while the ghost can perform both open and closed moves. Hence, this incomplete information is available to the ghost busters.
2. There are a total of five ghostbusters, and it requires that all of them should cooperate and must follow a strategy to catch the ghost.
3. The game is asymmetric, and hence, the game is imbalanced in its goals that make automatic adaptation to certain situations harder.

These three properties are what's making it a challenging problem to solve using adversarial learning techniques.

*Game board*  
There are 199 locations (stops) numbered from 1 through to 199. These numbered locations are connected by 4 different kinds of ways that represent different passages: walk, sewage, tunnel, and portal. Each numbered location can hold a maximum of one player. In turn each player can occupy a maximum of one location at one time. The point occupied by a player at a specific round is called the location of the player in that round. The six players can start at a random location which can be one of the 18 predefined numbered locations. The game UI is as follows:



*Rules*  
Each ghostbuster will be given a total of 22 tokens: 10 walk tokens, 8 sewage tokens, and 4 tunnel tokens. The ghost will start with a total of 12 tokens: 4 walk tokens, 3 sewage tokens, 3 tunnel tokens, and 2 portal tokens. The movement of the players with the ghost starting the game. A series of moves by all the six players (5 ghostbusters and ghost) is called a Round. The Ghostbusters board game has a maximum of 25 rounds.

Each player loses a stamina token upon moving from one stop to another based on the type of the chosen passage. When a ghost buster loses a token, it gets added to the ghost’s token pool which can be used by the ghost for later rounds. However, when the ghost plays a token, the token is removed from the game and can never be used again.

In the game, the ghost keeps its location a secret, and only in the rounds 3, 8, 13, 18, and 24, it announces its location. The ghost informs the ghost busters about the token used by the ghost in its movement from one location to another. The ghost is caught when any one of the ghostbusters moves onto the location which the ghost is occupying. The goal of the ghost is to avoid being caught by the ghostbusters until no ghostbuster has any ticket left or none of them can perform a move. A ghostbuster cannot perform the move when it does not have a valid token to move from its presently occupied stop.

**PRIOR RESEARCH**

Search-based heuristics and Learning-based heuristics are the two most popular game-play methodologies. Search-based heuristics can be carried out in a tree-like structure say, game tree or graph. Its usage depends on the properties of the game. Learning-based heuristics’ most successful application is the deep neural network to master the game of Go that also used tree search for enumerating all possible moves. It is suggested that the reinforcement of tree search can make the search more efficient and applicable in reality.

In recent times, Hide and Search Game was modeled by using heuristic search algorithms. One of such techniques was attempted in which Monte Carlo Tree Search (MCTS) technique was used as a heuristic search algorithm. In a different development over the study of this kind of game, a complexity analysis has been performed by Sevenster, where it has been shown that the generalized version of the game is PSPACE-complete.

It is important that in a double-sided game, each side attempts to win by using its strategy. The strategy of one side becomes an adversary for another side and vice-versa. If both the sides are modeled using mathematical approximators, the decisive strategy developed by neural networks in one side is adversarial to the decisive strategy developed by the other side. Such a combination of neural network models is an adversarial neural network that learns through reinforcement, and therefore, the methodology could be called as deep reinforcement learning.

In a recent survey, it has been discussed that neuro-evolution can be a better alternative to the existing search-based heuristics. Furthermore, hyper-heuristic, which is possibly a combination of smaller heuristics, could solve the game problems in a more efficient way.

**ARCHITECTURE FLOW DIAGRAM**

**A screenshot of a cell phone

Description automatically generated**

**ENVIRONMENT SETUP**

**A picture containing screenshot

Description automatically generated**

**METHODOLOGY UNDERTAKEN**

In order to implement automated ghostbusters and ghost as adversarial agents, we can implement Deep Reinforcement based algorithms. The players are modeled as neural network agents that try to approximate the Q values that are to be obtained at each stage of the game. In traditional Q learning based approach, Q values at each state of the game are stored in the Q table. But since there is an exponential increase in the state space and the possible actions at each stage, it is not possible or advised to use traditional Q learning based algorithm. To replace the Q table to output Q values for each action at every stage, we use neural networks which are the best approximators. In order to explore the behavior of a bot as a neural network agent we are implementing the ghost as Deep Q-learning bot. The ghost busters generate random moves to train the ghost.

The ideal function approximator is the target neural network. The Policy network tries to recreate the values achieved by the Target network on a closer level. The Target network gets its weight values from the Policy network after every 5 updates. The update of weight occurs using the Q values obtained from the target network in the Bellman equation.

**Deep Q learning model for Ghost:**

* Reward function - Shortest distance between Ghost and any Ghost Buster
* Policy Neural network for Ghost
* Target Neural network for Ghost
* Replay memory size = 1000
* Replay memory sampling size = 256
* Exploration rate - initially set to 0.9 and gradually decreases
* Discount factor = 0.9

**Neural network for the Ghost:**

***Feature Space***

There are 1213 input feature for the neural network of the ghost which comprises of the following:

* Encoded location of Ghost
* Number of resources that can be used by Ghost (Walk, Sewage, Tunnel, Portal)
* Encoded location of each detective
* Number of resources that can be used by each Ghostbuster (Walk, Sewage, Tunnel)
* Current round number
* Current move number

***Layers****​*

|  |  |
| --- | --- |
| ***LAYER*** | ***SIZE*** |
| Input | 1211 x 1 |
| Hidden 1 | 708 x 1 |
| Hidden 2 | 708 x 1 |
| Hidden 3 | 354 x 1 |
| Output | 200 x 1 |

***Hyperparameters***

* ADAM optimizer
* RELU activation
* Learning rate = 0.001
* RMSE loss

For training purpose, the algorithm was executed for 1000 games with each game having a maximum of 24 turns. For each turn, the batch size was updated for the replay memory sampling size. For the purpose of performance evaluation, the metric employed is the difference in the number of games won by the ghost and ghost busters

A close up of a map

Description automatically generated.

For testing purposes, the algorithm was executed for approximately 200 games. Ghost wins 75 percent of the time. The testing output is as follows:



**FUTURE WORK**

As of now, the ghost bot is trained by the ghostbusters whose moves are randomly generated. The accuracy becomes 75 percent. Upon adding heuristics like lesser distance between the ghost and ghostbusters to the randomly generated moves, the accuracy becomes 44 percent. The next part will involve improving this accuracy using better heuristics that will help the ghost win the game.

Additionally, neural network-based agents will be generated for the ghost busters instead of a random move generator. The ghostbusters will be trained by using an already trained ghost bot.

Finally, we are inclined to try using the ghostbusters neural network to train the ghost neural network and vice-versa to improve performance of both the ghost and the ghostbusters.

**REFERENCES**

1. Coulom R. (2007) Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In: van den Herik H.J., Ciancarini P., Donkers H.H.L.M. (eds) Computers and Games. CG 2006. Lecture Notes in Computer Science, vol 4630. Springer, Berlin, Heidelberg
2. Jetal Hunt K, Sbarbaro D, Zbikowski R, Gawthrop PJ (1992) Neural networks for control systems—a survey.
3. Georgeff, M. P., Ingrand François, F. (1989). Decision-making in an embedded reasoning system. In *IJCAI*, pp. 972–978.
4. Nijssen, J.A.M. & Winands, Mark. (2012). Monte Carlo Tree Search for the Hide-and-Seek Game Scotland Yard. IEEE Transactions on Computational Intelligence and AI in Games. 4. 282 - 294.
5. Sevenster, Merlijn. (2006). The complexity of Scotland Yard. Journal of Pharmacology and Experimental Therapeutics - J PHARMACOL EXP THER.
6. Dash, T., Dambekodi, S.N., Reddy, P.N. *et al.* Adversarial neural networks for playing hide-and-search board game Scotland Yard. *Neural Comput & Applic* (2018).
7. Sturtevant, Nathan R. and Richard E. Korf. “On Pruning Techniques for Multi-Player Games.” *AAAI/IAAI* (2000).
8. Wu, Lin and Pierre Baldi. “Learning to play Go using recursive neural networks.” *Neural networks: the official journal of the International Neural Network Society* 21 9 (2008): 1392-400.
9. Zuckerman, I., Kraus, S. & Rosenschein, J.S. The adversarial activity model for bounded rational agents. *Auton Agent Multi-Agent Syst* **24,**374–409 (2012).
10. Rehak, M., Pechoucek, M., & Tozicka, J. (2005). Adversarial behavior in multi-agent systems. In M. Pechoucek, P. Petta, & L. Z. Varga (Eds.) *Multi-agent systems and applications IV: 4th International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2005*, number 3690 in LNCS, LNAI.