

**WiDS ‘22 - ‘23 Final Documentation**

## **Playing 2048 game using reinforcement Learning**

**Polamreddy Mohit Krishna**

|  |  |  |
| --- | --- | --- |
| **Team Member Name** | **Roll Number** | **Email-Id** |
| B S V SRI TEJA | 22N0459 | 22N0459@iitb.ac.in |

**Introduction to Problem Statement**

|  |
| --- |
| Playing 2048 game using deep reinforcement learning. Using deep learning to learn about strategies of the game. At any stage there are four moves available for the user to make using reinforcement learning we need to select the move which gives higher chance to score highest in the game. |

**Existing Resources**

|  |
| --- |
| Previous papers have explored various ways to win 2048 without human knowledge.  Amar, et al. 2017, uses policy network to learn the best strategy to play 2048 without human  knowledge. The policy network is a neural network that contains two convolutional layers with ReLU as activation function. The model takes the state of the game as input, and outputs a policy vector of length 4 with each entry representing how likely choosing a policy at the current state will lead to thehighest score. To encourage exploration of different strategies, the paper also uses e-greedy with e as a decreasing function of the number of games played. However, the model doesn’t serve to win the game: it reaches 1024 most of the time and reaches 2048 occasionally. Szubert, et al. 2014, models the 2048 game as an MDP, and applied temporal difference learning (TDL) on the MDP to learn the best policy. The model dwarfs the human performance by reaching 2048 with a rate of 97%. However, the model still has a hard time reaching larger tiles (never reached 32768 or higher), so Yeh, et al. 2015, improves the model by applying multi-stage temporal difference learning with 3-ply expect Imax search, and the new model reaches 32768 31.75% of the time |

**Proposed Solution**

|  |
| --- |
| Our baseline is to play the game by choosing a valid action at random at each state until we reach the end of the game. The strategy rarely reaches any tile above 128. We have implemented Deep Q-learning Algorithm with e-greedy exploration strategy to win the 2048 game. Deep Q-learning is the same as Q-learning except we use deep learning to evaluate the Q value. e-greedy search is a search algorithm that encourages exploration. e is a number between 0 and 1. When selecting the best action to perform, we select a random number between 0 and 1. If the number is smaller than e, we choose a random action regardless of the action’s Q value. Otherwise, we choose the action with the highest Q value. We used the epsilon value 0.3 in both of our models and next we used Beam search where Beam search is a greedy search algorithm that explores a tree by always choosing the k best nodes. In the game of 2048, each node in the tree is a state, and each edge is an action |

**Methodology & Progress (Mention the work done week-wise)**

|  |
| --- |
| Week 1-4 :- Studied basics of machine learning ,CNN, reinforcement learning, deep learning using the resources provided the mentor and basics of java script  Week 5:- Made the game using java script  Week6 - Now: Using the research paper implemented the solution. |

**Results**

|  |
| --- |
| Please add the link to drive folder/ github page consisting of code files and reports  GitHub link: - https://github.com/teja1729/2048-using-reinforcement-learning |

**Learning Value**

|  |
| --- |
| Learnt Convolutions Neural Networks, Reinforcement learning and deep reinforcement learning and some basics of machine learning. Java script basics to make a web game. |

**Tech-stack Used.**

|  |
| --- |
| **Python**  **Java Script**  **Python libraries – Random, numpy, Lasagne,Theano,Selenium** |

**Suggestions for others**

|  |
| --- |
| This project need understanding of java script for making the game in webpage and understanding of reinforcement learning, deep learning and it will take good amount of your time to complete but you will enjoy the journey. |

**Contribution by each Team Member**

|  |
| --- |
| It is individual project |

**References and Citations**

|  |
| --- |
| * Amar, Jonathan, et al. "Deep Reinforcement Learning for 2048." (2017) http://www.mit.edu/∼amarj/files/2048.pdf * Shilun Li, Veronica Peng,” Playing 2048 With Reinforcement Learning.” (2021) https://arxiv.org/pdf/2110.10374v1.pdf |