Project Presentation on

Playing 2048 Game using Deep Neural Networks

P.Sreelekha Reddy	**19121A05H9**
R.B.Moulya	**19121A05J6**
R.Namratha	**19121A05J7**
R.Pranathi	**19121A05K2**

SREE VIDYANIKETHAN Engineering College (Autonomous)

Computer Science and Engineering

Guide & HOD:
Dr. B. Narendra Kumar Rao,
Professor & Head,
Dept. of CSE

CONTENTS

Abstract

A stochastic puzzle game for one player is called Game 2048. This fascinating and engaging game has gained widespread acclaim and drawn researchers to create gaming software. The game 2048 has evolved into an engaging and difficult platform for assessing the efficacy of machine learning techniques because of its simplicity and complexity. Convolutional neural networks were used to create some computer players,but they performed poorly. In this project, we create a 2048 agent based on the Reinforcement Learning method and neural networks.We want to outperform other neural network based competitors in terms of results. Additionally, a cutting-edge software created using this methodology for 2048 achieves the best performance out of all learning-based algorithms.

Introduction

- Game 2048 is played on a 4×4 grid. It is a slide-and-merge game, easy to learn but hard to master.
- The objective of the original Game 2048 is to reach a 2048 tile by moving and merging the tiles on the board according to the rules.
- In an initial state two tiles are placed randomly with numbers 2 or 4. The player selects a direction (either up, right, down, or left), and then all the tiles will move in the selected direction. When two tiles of the same number collide, they create a tile with the sum value and the player gets the sum as the score.
- The merges occur from the far side and newly created tiles do not merge again on the same move. And the player cannot select a direction in which no tiles move nor merge. After each move, a new tile appears randomly at an empty cell with number 2 or 4 ,when the player cannot move the tiles, the game ends.

Problem Statement

To develop an agent that is capable of playing 2048 Game efficiently. The need for developing such an agent is mainly for the purposes of research.

Objectives

- The primary objective is to train the agent to play the game optimally, by learning from experience and adjusting it's strategy based on the rewards received for making different moves.
- The agent should improve its performance over time
- Agent should explore the game state space.
- The agent should be designed to use computational resources efficiently, so that it can be trained and run on a variety of hardware platforms.

Software and Hardware Requirements

- Memory :4GB (recommended)
- Graphics Card :Intel Media Accelerator 500
- Languages used :Python
- OS :Windows 7,8,8.1,10
- Libraries used : Pandas, Numpy, OS, Tensorflow, Keras etc.,

Existing System

- The existing system for playing the 2048 game typically involves a human player manually making moves by sliding tiles on a grid.
- It was developed based on deep convolutional neural networks trained by supervised learning.

Software Requirement Specification

Functional Requirements:

- Game environment for the agent to interact with and learn from.
- Reinforcement learning algorithm to train the agent to play the game.
- Neural network architecture to estimate the expected reward.
- Training data to train the agent effectively.

Non Functional Requirements:

- Performance of game play with a minimum acceptable game score and win rate.
- Scalabile to handle increase in the size of the training data.
- Compatibility with different operating systems are considered.

Proposed System

- The proposed system is developing an agent to play the 2048 game using deep reinforcement learning involves using a machine learning algorithm to learn how to play the game by observing the rewards received for making different moves in the game.

- In the proposed system, the agent would replace the human player and would observe the state of the game, select actions based on a policy, perform the selected action, receive a reward from the game environment, and update its policy based on the received reward and the current state.

Algorithm

Step 1: The board will have two tiles in random locations, each of which either has a “2” or a “4” on it – each has an independent 10% chance of being a “4”, or otherwise a is a “2”.

Step 2: Moves are performed by shifting all tiles towards one edge – up,down, left, or right. When doing this, any tiles with the same value that are adjacent to each other and are moving together will merge

Step 3: After merging, a new tile equal to the sum of the earlier two will be placed onto the board. This is placed in a random location.

Step 4: The game then continues until there are no more moves possible.

In general, the goal of the game is to reach a single tile with a value of “2048”. However, the game doesn’t stop here, and we can continue playing as far as it is possible to go, aiming for the largest possible tile.

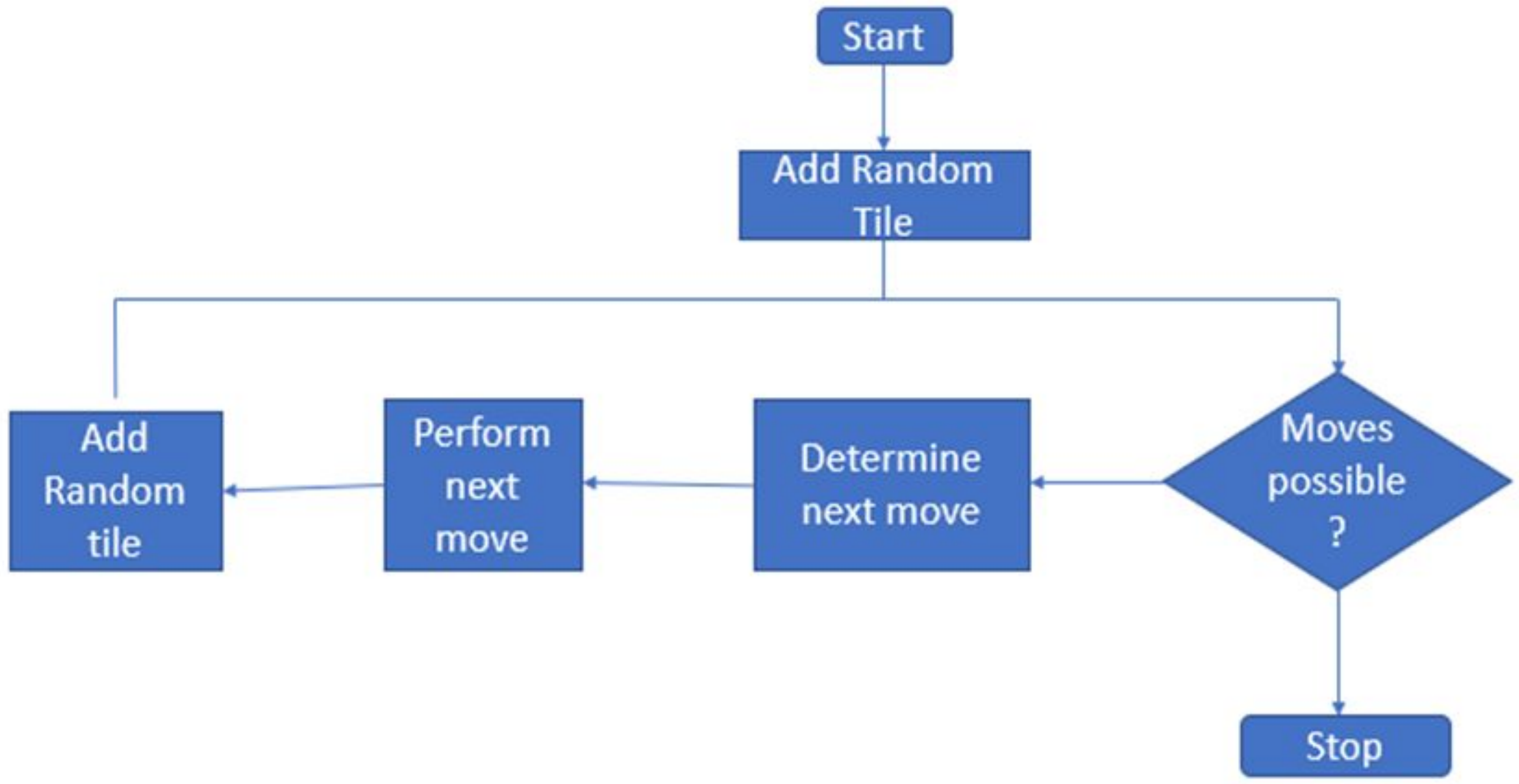
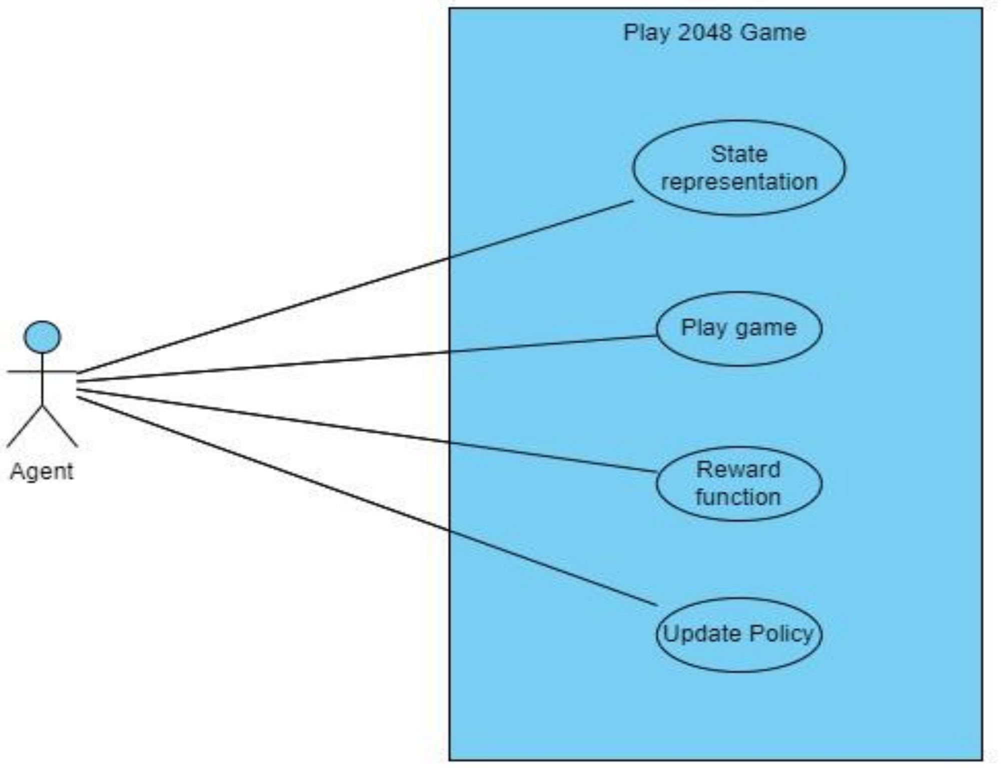


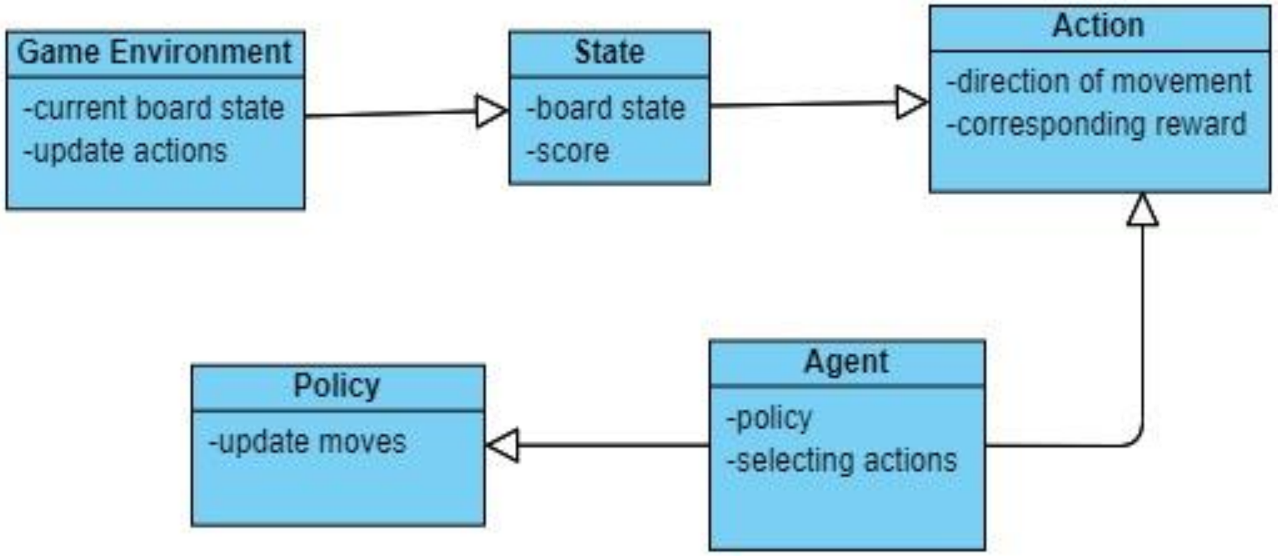
fig - flowchart of gameplay

Design

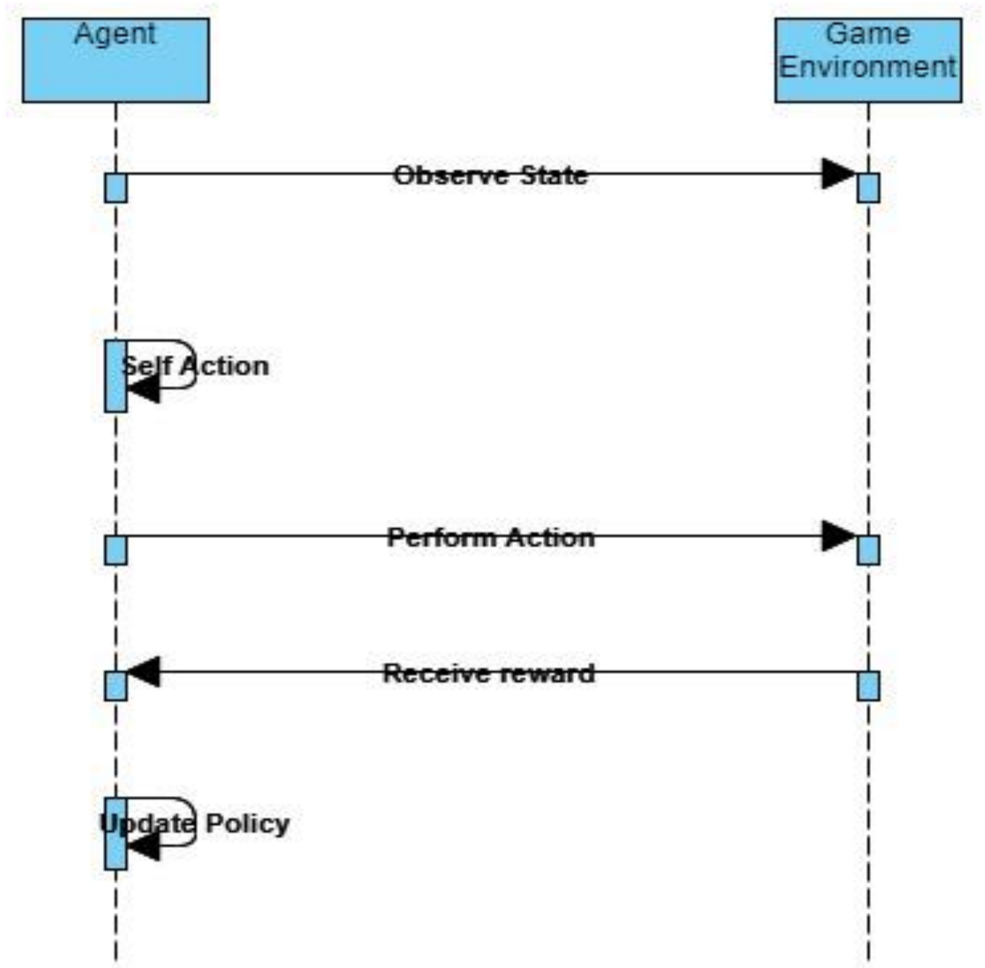
Usecase Diagram:



Class Diagram:

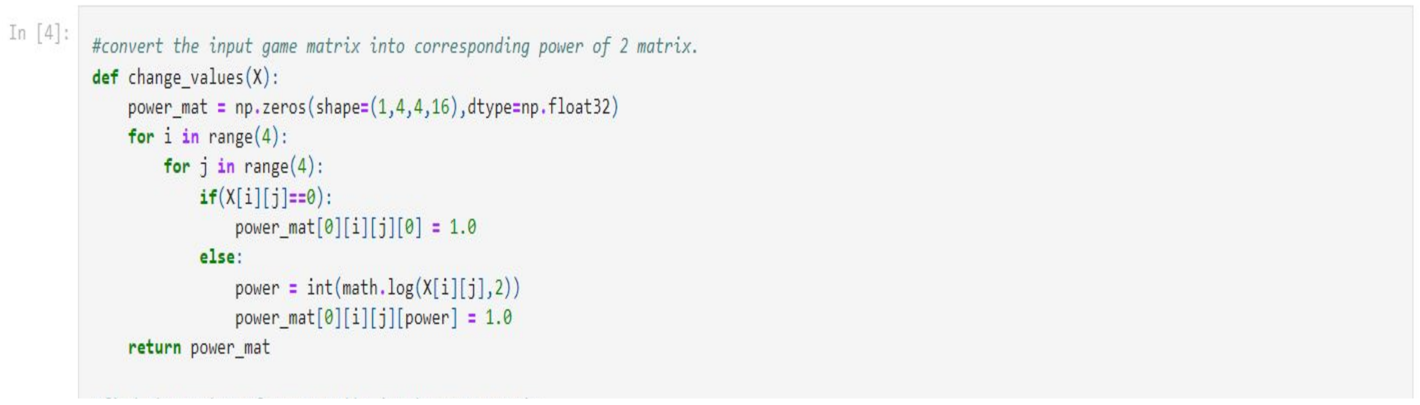


Sequence Diagram:

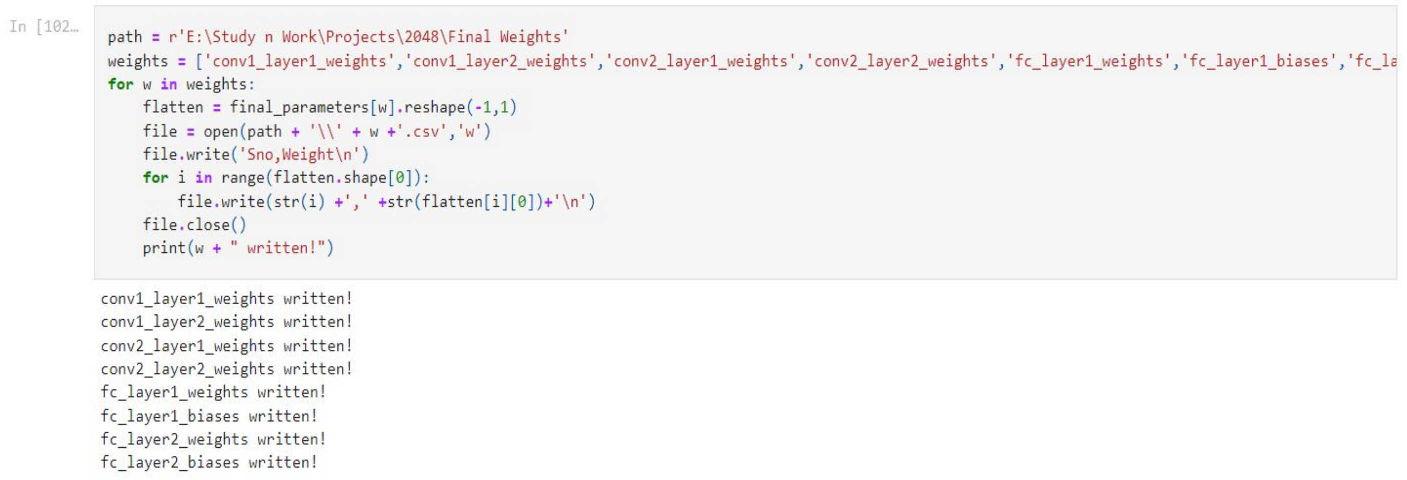


Implementation

The Convolutional Neural Network is implemented using Tensorflow Library. We used tf.train.RMSPropOptimizer() for the optimization algorithm and tf.nn.relu() activation function. Each neural network was trained with 100M episodes for convergence using the reinforcement learning methodology. Every 1M training episodes, or 100k testing episodes, were used to evaluate the learning performance. The dataset is created with the trained weights. The epsilon greedy technique and the hyperparameter gamma for Q-learning are employed. The current reward is determined by the number of tile mergers. The CNN method of playing the 2048 game has generally produced encouraging results, with some models attaining scores much higher than human players.



```
In [4]:
#convert the input game matrix into corresponding power of 2 matrix.
def change_values(X):
    power_mat = np.zeros(shape=(1,4,4,16),dtype=np.float32)
    for i in range(4):
        for j in range(4):
            if(X[i][j]==0):
                power_mat[0][i][j][0] = 1.0
            else:
                power = int(math.log(X[i][j],2))
                power_mat[0][i][j][power] = 1.0
    return power_mat
```



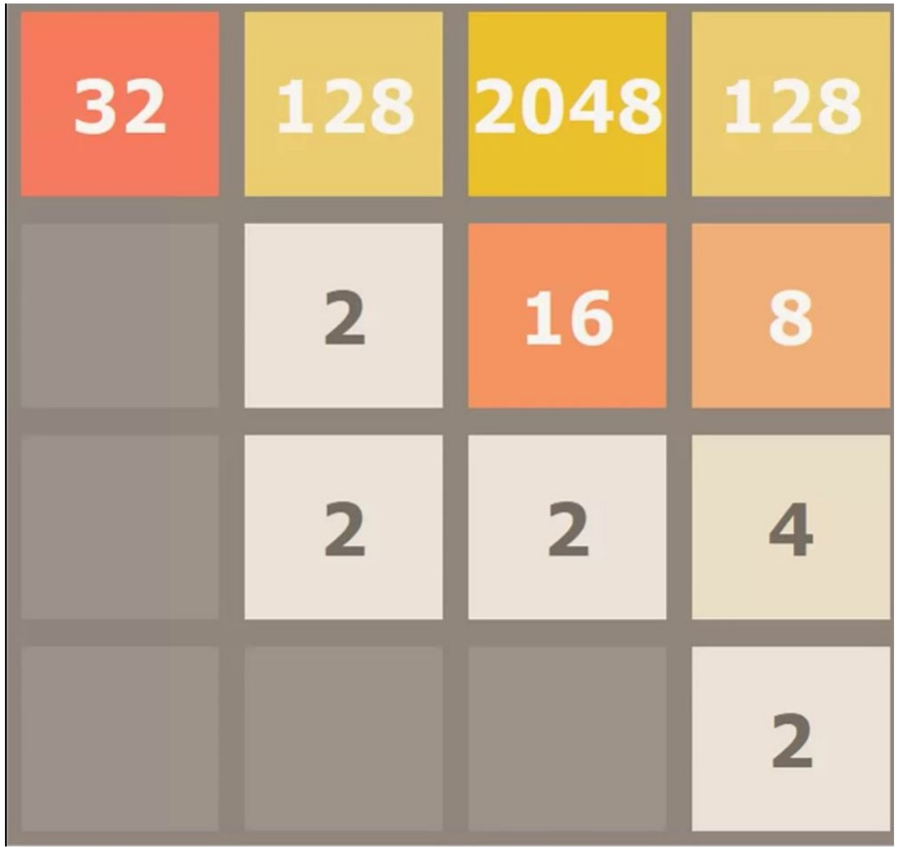
```
In [102…
path = r'E:\Study n Work\Projects\2048\Final Weights'
weights = ['conv1_layer1_weights','conv1_layer2_weights','conv2_layer1_weights','conv2_layer2_weights','fc_layer1_weights','fc_layer1_biases','fc_la
for w in weights:
    flatten = final_parameters[w].reshape(-1,1)
    file = open(path + '\\' + w +'.csv','w')
    file.write('Sno,Weight\n')
    for i in range(flatten.shape[0]):
        file.write(str(i) +',' +str(flatten[i][0])+'\n')
    file.close()
    print(w + " written!")
```

```
conv1_layer1_weights written!
conv1_layer2_weights written!
conv2_layer1_weights written!
conv2_layer2_weights written!
fc_layer1_weights written!
fc_layer1_biases written!
fc_layer2_weights written!
fc_layer2_biases written!
```

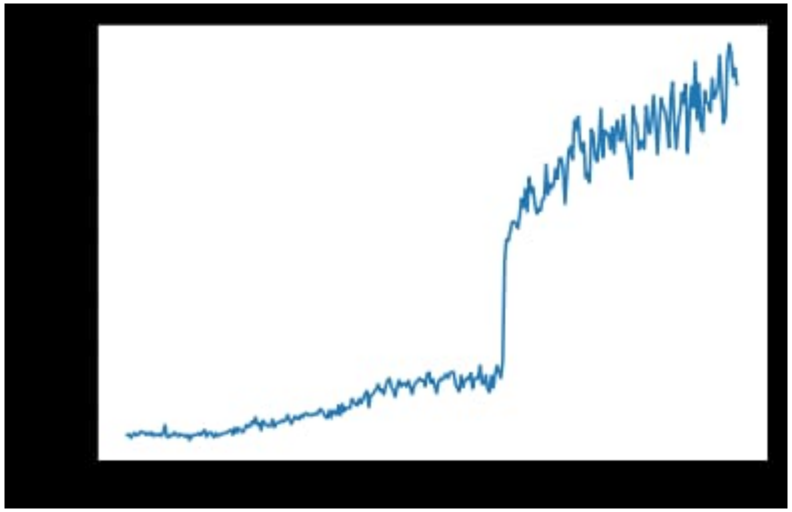
fig-changing the input values and storing the trained data in file

Result

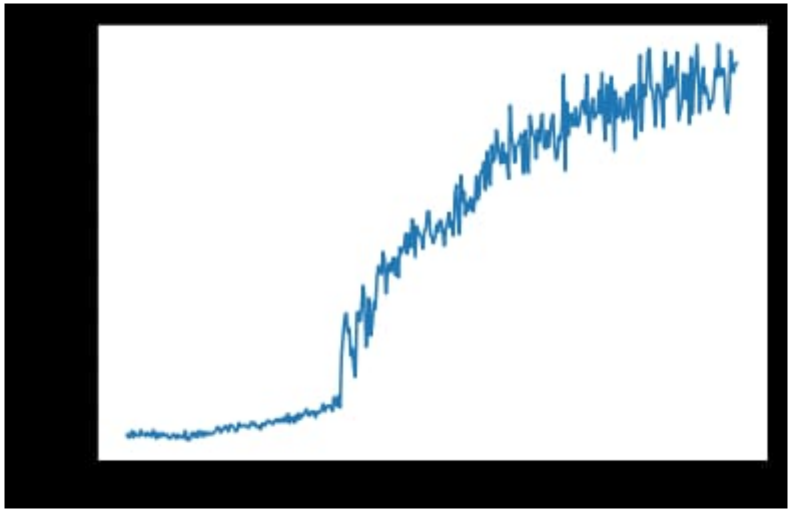
Final Output when max tile is reached:



Graph of various game plays with different outcomes



Graph of moves between agent and player



Conclusion

- In conclusion, creating an agent to play the game 2048 using deep reinforcement learning is an exciting and difficult task in the field of artificial intelligence. We have demonstrated the feasibility of using reinforcement learning to train an agent to play 2048 and achieve a high score through our research.
- Our research builds on previous research in the field by employing deep neural networks to learn complex patterns in the game and make informed decisions.
- Our findings show that deep reinforcement learning can be an effective method for teaching agents to play 2048 and achieve high scores, shedding new light on the application of reinforcement learning to game play.

References

- Naoki Kondo, Kiminori Matsuzaki, ”Playing Game 2048 with Deep Convolutional Neural Networks Trained by Supervised Learning”, Journal of Information Processing, Vol.27, 340-347,(April 2019)
- David, O.E., Netanyahu, N.S. and Wolf, L.: DeepChess: End-to-End Deep Neural Network for Automatic Learning in Chess, Interna-tional Conference on Artificial Neural Networks and Machine Learn-ing (ICANN 2016), pp.88-96 (2016).
- Jaśkowski, W.: Mastering 2048 with Delayed Temporal Coherence Learning. Multi-Stage Weight Promotion, Redundant Encoding and Carousel Shaping, IEEE Trans. Computational Intelligence and AI in Games, Vol. 10, No.1, pp.3-14 (2018).
- Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, L., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T. and Hassabis, D.: Mastering the game of Go without human knowledge, Nature, Vol.550, pp.354-359 (2017).
- Boris, T. and Šuković Goran: Evolving Neural Network to Play Game2048, Proc. 24th Telecommunications forum (TELFOR 2016), IEEE(2016).

Queries

Thank You