

INDIAN INSTITUTE OF TECHNOLOGY JODHPUR



॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥

ADVANCED ARTIFICIAL INTELLIGENCE - CSL 7580

COURSE PROJECT - REPORT

PLAYING 2048 WITH REINFORCEMENT LEARNING

Course Instructor:

Dr. Gaurav Harit

Prepared by:

Abhijit S Iyer (M21CS001)

Jayesh Budhwani (M21CS007)

Introduction:

2048 is an addictive mobile application game, where tiles with values corresponding to powers of two are presented to the player, with which the player needs to figure a way out to bring the value '2048' onto a tile by merging tiles containing a similar number. For example, two tiles containing the value '2' can be merged to form a tile containing the value '4', and now this tile containing the value '4' needs to be merged with another tile containing the value '4' to make a tile with the value '8', and so on. This process continues until we create a tile containing the value '2048' by finally merging two tiles containing the values '1024'. The merging operation can be done using one of the following moves - up, down, left, and right. Upon performing the move-up operation, all the tiles in the $n \times n$ grid move upwards and merge with similarly valued tiles. If no tile is there to be merged with, then the tile would settle down at a slot below the existing tile. This rule is applicable to all directions.

Now, the intention of this project is to create an AI agent that would perform a similar task of making moves using tiles, to finally end up with a '2048' valued tile. To facilitate this purpose, we will make use of the 'Deep Q Learning' algorithm that falls under the category of 'Reinforcement Learning'.

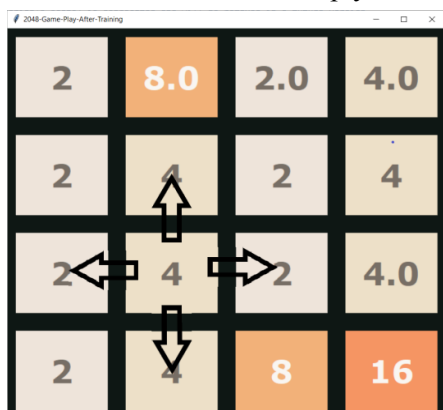
Reinforcement Learning is a machine learning paradigm where rewards are used as a method to propagate the learning of an agent. An agent depicting desired behaviour is rewarded, while an agent depicting undesired behaviour is punished. For our implementation, we have made use of the Q Learning algorithm that comes under the category of reinforcement learning. Q Learning chooses an action at random from its current state and aims to maximise its reward.

Objective:

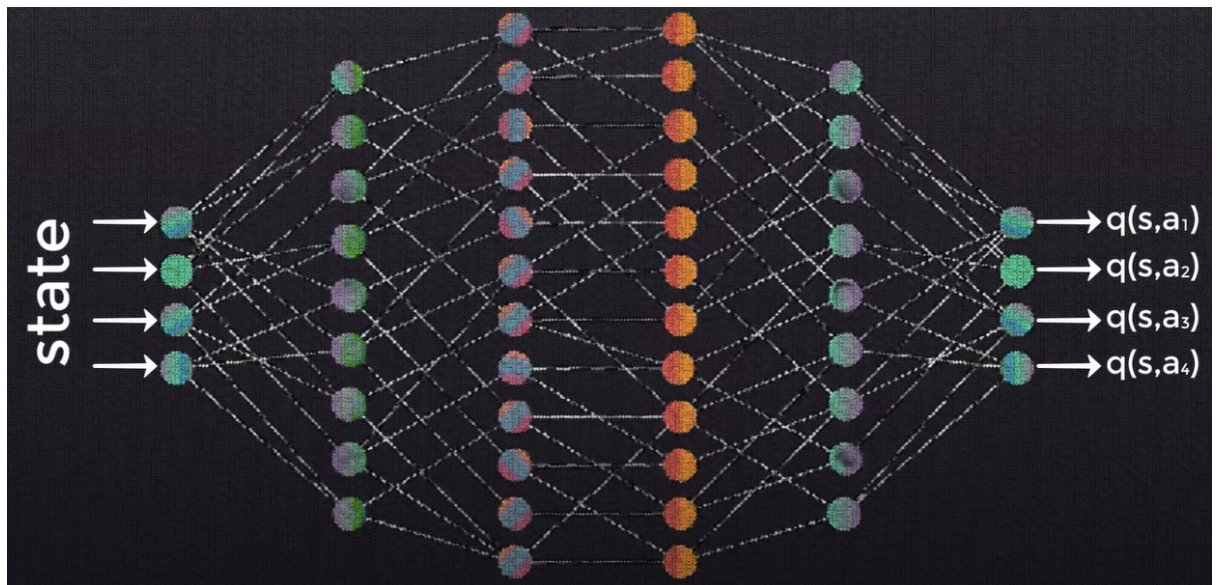
The objective of this project is to play 2048, a classic game using reinforcement learning and to test whether a reinforcement learning model can play the game.

Methodology:

1. To achieve the objective we have defined we have created an environment of 2048 which contain 16 tiles arranged in a 4×4 manner and each tile can be empty or can contain a number which is in powers of 2. The objective of the environment is to create a tile having a score of 2048 or above to win the game.
2. The state of the game comprises the environment and the actions performed on the elements that tile. The actions contain moving a tile up, down, left, or right. And after moving, the number on both the tiles are added. Similarly, the game continues, and when a tile contains a number greater than or equal to 2048 the game is stopped. But not all actions are available to all tiles. When the tile is empty or a corner the number of actions is reduced.



3. Because of these random movements in the game environment we can say that there is uncertainty.
4. After designing the environment and the actions we have used reinforcement learning to learn a model that can play the game and can win the game. For this, we have used a deep Q-learning-based algorithm with an epsilon greedy exploration strategy for training the model.
5. We have created a deep learning net containing an input layer, two convolution layers with hidden layers. We have tested for different values of layer parameters for training to see which parameters provide the best result.



6. After this we have used Q-Learning defined as

$$Q = \text{Reward} + \gamma (\max_{\text{over_previous_action}}(\text{state}))$$
7. Here the reward is calculated as the log to base 2 of the tile value and then the Q value is updated.
8. Epsilon greedy search using the exploration that is the value of epsilon is provided by the user and a random number is selected between 0 and 1 and if the value of a number is less than epsilon then we choose a random action regardless of Q value otherwise we choose an action with highest Q value.



Experiments:

We have conducted different experiments.

Experiment 1: To choose the value of Convolution net parameters in this the number of episodes is fixed and different values of convolution net depths are tested to see which value gives the best performance.

The domain contains depth values (64, 128, and 256). The values of the input, hidden, and output are kept constant at 16, 256, and 4 respectively.

Experiment 2: To choose the values of episodes to get the best-trained model.

In this experiment, we selected a neural network and used different values of episodes to train the model. No. of episodes increases to see how better the model is trained.




Results:

From the experiments, we can say that for values of episodes greater than 1000 the model is trained to provide a better result than with fewer episodes. Further, if we increase the episodes the accuracy will also increase.


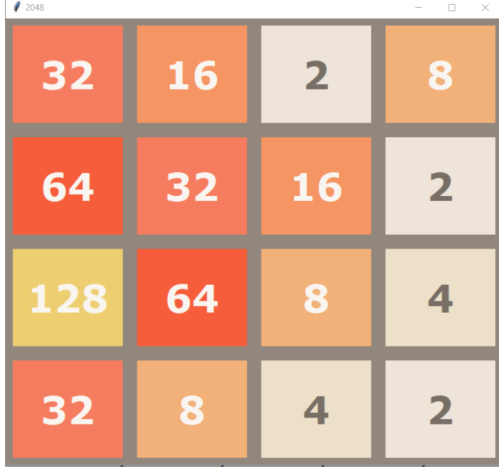

Also for the same number of episodes and different depths of the Convolution network we can say that 128 and 256 are values that may be appropriate choices. Further, if we increase the value of episodes we can see that more accuracy is achieved in cases of 128, 256, and 128, 128.

For the experiments conducted so far, the following are the results obtained:

Experiment 1:

S. No.	Depth of Layer1	Depth of Layer2	Output Model
1.	128	128	
2.	64	128	
3.	128	256	

Experiment 2:

S. No.	Depth of Layer1	Depth of Layer2	No. of episodes	Output Model
1.	128	128	10000	
2.	128	128	5000	
3.	128	128	500	

References:

- [1] S. Li και V. Peng, ‘Playing 2048 With Reinforcement Learning’. arXiv, 2021.