# **CSE 472 (Machine Learning Sessional) - Project Proposal**

### 

### **Problem Definition**

### **Deep Reinforcement Learning to play Tetris**

Reinforcement learning is a type of machine learning where an agent learns to make decisions in an environment by performing actions and receiving rewards or punishments. To apply reinforcement learning to playing Tetris, we can define the environment as the game, the actions as the moves the player can make (e.g. rotating or moving the falling block), the state as the current layout of the blocks on the playing field, and the rewards as points for clearing lines.

### **Dataset**

We don’t need a dataset for deep reinforcement learning to play a game automatically.Instead of using pre-existing data, the agent learns from its own interactions with the game environment.

According to the results reported in the “Playing Atari with Deep Reinforcement Learning”, the DQN algorithm was able to learn how to play Tetris and achieved a score of approximately 100,000 points.

### **Papers**

Here are some research papers in the area of deep reinforcement learning applied to playing tetris and our work will be based on this papers:

1.[Learn to Play Tetris with Deep Reinforcement Learning](https://openreview.net/pdf?id=8TLyqLGQ7Tg) (2020)

2.[Playing Tetris with deep reinforcement learning](http://cs231n.stanford.edu/reports/2016/pdfs/121_Report.pdf) (2016)

3.[The Game of Tetris in Machine Learning](https://arxiv.org/pdf/1905.01652.pdf) (2019)

### **Architecture**

A general architecture could look like this:

1. **Input layer**: The input layer of the neural network would take the current game state as input and it could be an image of the current state of the game, or features that describe the state.
2. **Hidden layers**: The hidden layers of the neural network would consist of one or more dense or convolutional layers. These layers would apply transformations to the input to extract relevant features.
3. **Output layer**: The output layer of the neural network would produce a probability distribution over possible actions for the agent to take. The actions could be moving the block left or right, rotating it, or dropping it. The output layer can use a softmax activation function to ensure that the probabilities sum to 1.
4. **Reinforcement learning algorithm**: The reinforcement learning algorithm takes the output of the neural network and the reward signal as input and updates the network parameters to improve performance. **Deep Q-learning** can be suitable as a reinforcement learning algorithm.
5. **Reward function:** The reward function would determine the reward the agent receives for each action it takes.The reward function could be based on the number of lines cleared, the height of the blocks, and other factors that affect the score.

**Deep Q-learning:**Deep Q-Learning (DQL) is a reinforcement learning algorithm that combines deep neural networks with Q-Learning. It uses a neural network to approximate the Q-function, which maps states to action values, instead of a table-based approach.

### **Performance Metrics**

Some performance metrics for Tetris can be:

1. **Score**: The most straightforward metric would be the score achieved by the agent over a series of games. We can track the average or maximum score.
2. **Blocks cleared:** Another metric could be the number of blocks cleared by the agent. This could give us insight into the efficiency and skill of the agent in playing the game.
3. **Game Duration:** We can also track the length of time the agent can play the game before losing. This could give us insight into the durability of the agent.
4. **Block placement**: Another metric could be the quality of block placement by the agent. This could be assessed by measuring the height of the blocks and the number of gaps or holes in the stack.
5. **Speed:** The speed at which the agent can clear lines could also be a useful performance metric. It can give us insight into the speed and agility of the agent.