# CSE584 HW2

**Janice Ahn (jfa5672)**
**10/15/2024**

1. **Introduction**
    a. I found a piece of code that implements reinforcement learning for Pacman which is a well known game in the United States. There was a github repository that allows people to use and explore the code. https://github.com/Gucci-JRat/ReinforcmentLearning-For-MsPacman/tree/master

2. **Abstract**
    a. The code from github by Jash Rathod implemented the reinforcement learning using a deep Q-network (DQN) approach. for the Pacman game which is a well known game in the United States. The purpose of the code is to train an agent to maximize its score by learning optimal actions through trial and error, leveraging neural networks to approximate Q-values for state-action pairs. The agent's decision-making process is guided by an epsilon-greedy policy, balancing exploration of new actions and exploitation of learned actions. Experiences are stored in replay memory and used to train a neural network, which predicts rewards for different actions. Through Q-learning, the agent improves over multiple episodes, aiming to maximize its score. The code provides real-time performance tracking and saves model weights for testing. It demonstrates how an agent can learn to play a game through trial and error, refining its strategy over time.

3. **Deep code analysis**
    a. All parts of the code is important for the agent training but if we have to choose the core part, then the function for selecting action (get_action) and training the model based on the reinforcement learning method (train_model) can be choosed. Get_action function implements the epsilon-greedy policy, which is crucial for balancing exploration and exploitation—an essential aspect of reinforcement learning. The agent chooses between exploring new actions (randomly selecting) and exploiting its learned knowledge (choosing the action with the highest predicted reward). Train_model function performs the training of the neural network, applying the Q-learning algorithm. The agent learns by adjusting the Q-values based on the Bellman equation, which considers both immediate rewards and future rewards from the next state. This is the heart of the agent's learning process.