ML 2023 - Exercise 3
Reinforcement Learning

Group 26
Börner, Direktor, Mayer

Implementation: Overview

- We implemented MC **Exploring Starts (ES)** and MC **First Visits (FV)**
- We used **python** for implementation
- We relied on the following existing libraries:
  - numpy (mathematical functionality and utility)
  - random (mathematical functionality)
  - pygame (game visualization)
- File structure:
  - Breakout_Class (game logic)
  - Monte_Carlo_Agent (agent logic)
  - train_MC_Agent (training logic)
  - RL_run (main script)
  - renderer, visualizer (displaying the live game and game information)

Implementation: Training

- Hyperparameter:
  - Gamma = 0.9 (to emphasize long term decision making)
- State: A specific state of the game, consisting of ball position, direction & speed, paddle position & speed and brick layout.
- At each timestep, the agent chooses an **action** (i.e. paddle movement) that maximizes the expected reward
- The reward decreases by 1 for each timestep (no other rewards are given)
- We update rewards and epsilon each time the agent finishes the game **successfully**, or after the agent **runs out of time**.

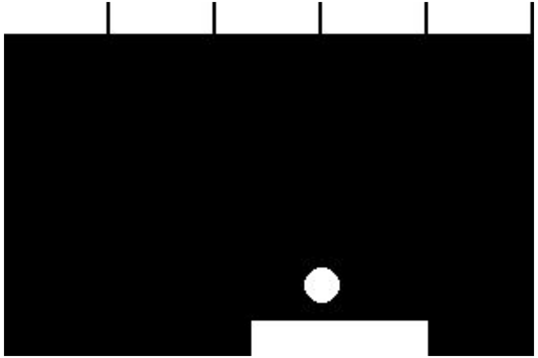
Implementation: Agent Versions

- Exploring starts (ES):
  - At the start of each episode, we **randomly** set the **state of the game** and the **first action** of the agent
  - Each available action has the same probability to be selected in the beginning
  - For the rest of the episode the agent behaves Greedy
- First visit (FV):
  - We estimate value functions
  - Optimal policies are determined by **averaging returns** from the **first visit** of each state-action pair in an episode
  - An **epsilon-greedy strategy** is incorporated to **balance exploration and exploitation** and ensures efficient learning by the agent
  - Epsilon is reduced after each episode (down to 0.05), making agent less likely to explore

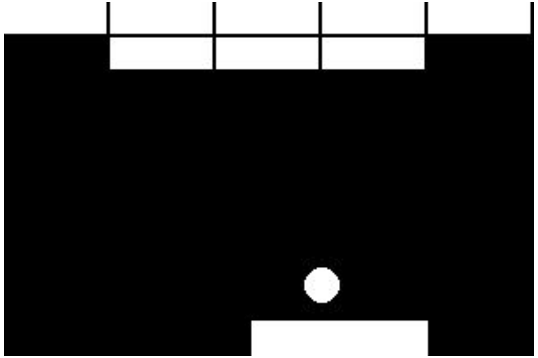
Experiments: Variations

- We used (15,10) as default grid size
- We experimented with the following layouts:
  - TopRow (*classic* Breakout layout)
  - MiddleRow (equally sized rows shifted towards the middle of the grid)
  - ReversePyramid (reversed pyramid that gets narrower towards the middle of the grid)
- We also experimented with the following parameters:
  - Number of episodes (100, 1000, 10000)
  - Maximum timesteps (100, 1000, 10000)
  - Number of bricks (5, 8, 9)
  - Starting direction (-2, -1, 0, 1, 2)

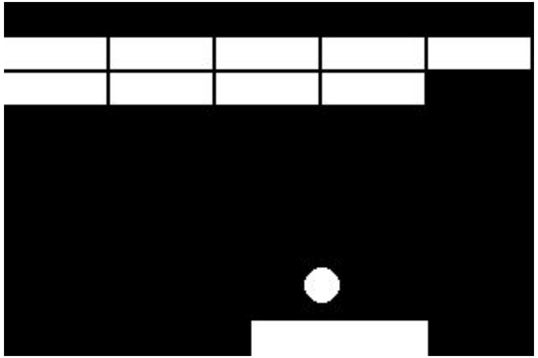
Experiments: Trajectories



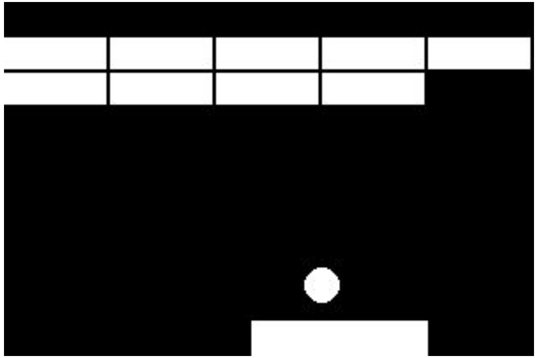
ES
5 bricks
TopRow
Left 2



ES
8 bricks
ReversePyramid
Middle



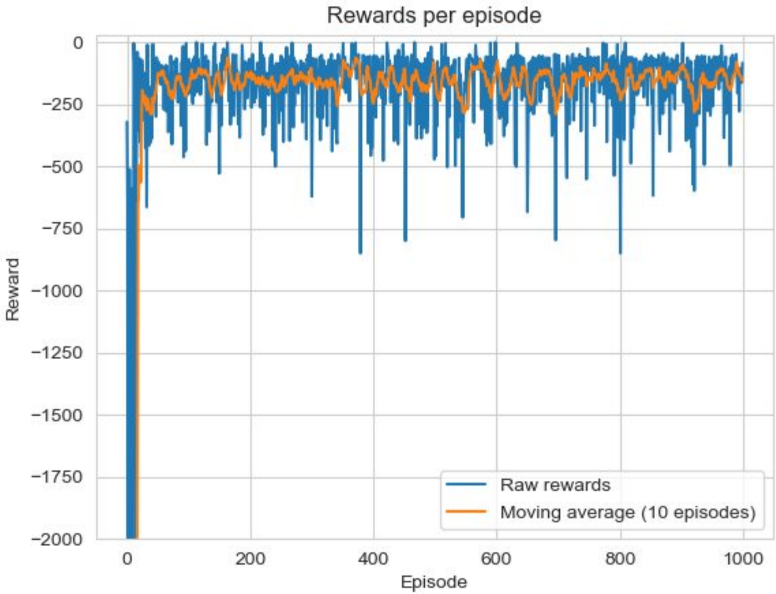
ES
9 bricks
MiddleRow
Middle



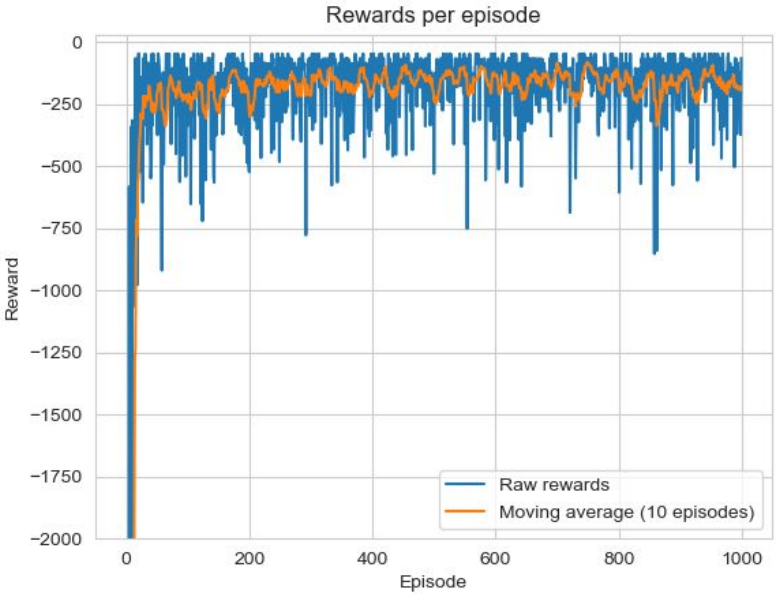
FV
9 bricks
MiddleRow
Middle

Experiments: Exploring starts vs. First Visit

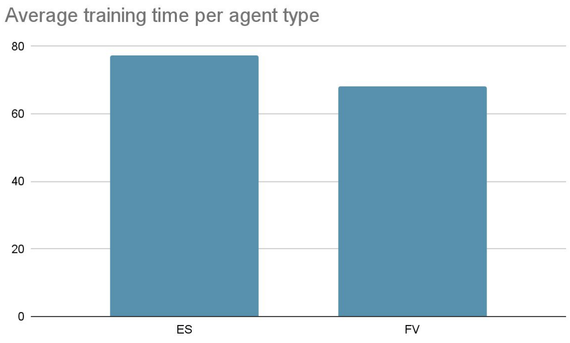
Exploring starts (both TopRow, 5 bricks, 1000 episodes, 10000 timesteps)

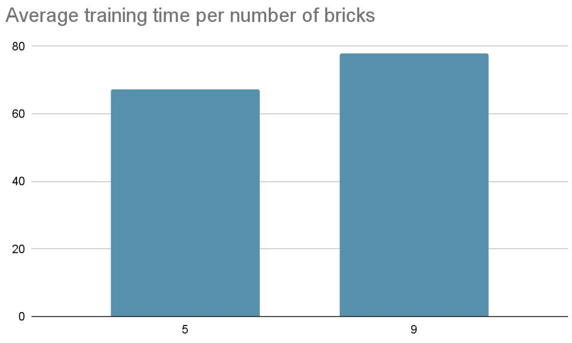


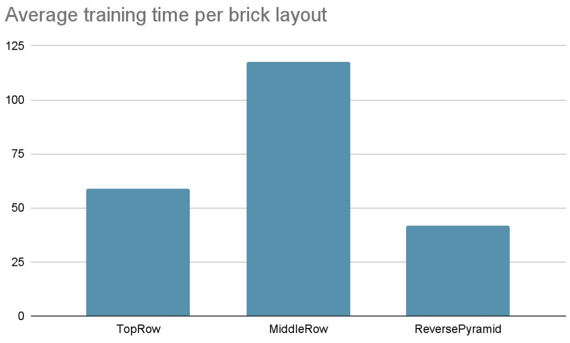
First Visit

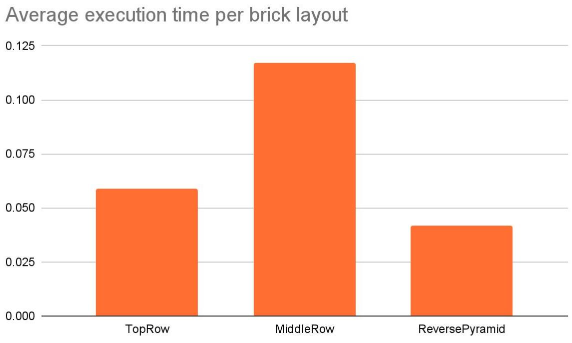


Experiments: Execution and training time









Findings

- Agents do not improve when trained on less than 10000 timesteps
- The agent learns to utilize the upper boundary bounce:
  - Deflecting the ball on the upper boundary making it hit the bricks on the top, bouncing against the upper boundary again
  - Very efficient in beating the game quickly, since it effectively removes several bricks with one action
- Runtime:
  - ES Agents takes longer to train than FV agents
  - Training and execution time for **MiddleRow** layout takes the longest
  - Number of bricks **increases** training time
- Agents struggle to beat the MiddleRow layout, because they can't utilize the upper boundary bounce