CS 461 - Artificial Intelligence
Term Project

2048 with Min-Max Tree Search, MCTS, and DDQN

Group 12
Ceren Akyar, Deniz Mert Dilaverler, Berk Çakar, Elifsena Öz, İpek Öztaş

Contents

Introduction

- Single-player sliding tile puzzle video **game**
- Use **arrow keys** to move the tiles
- When two tiles with the same number touch, they merge into one
- 4x4 **grid**
- **Tiles:** 512, 1024, 2048
- **Scores:** final game score
- **Game over:** no adjacent tiles have the same value [1]

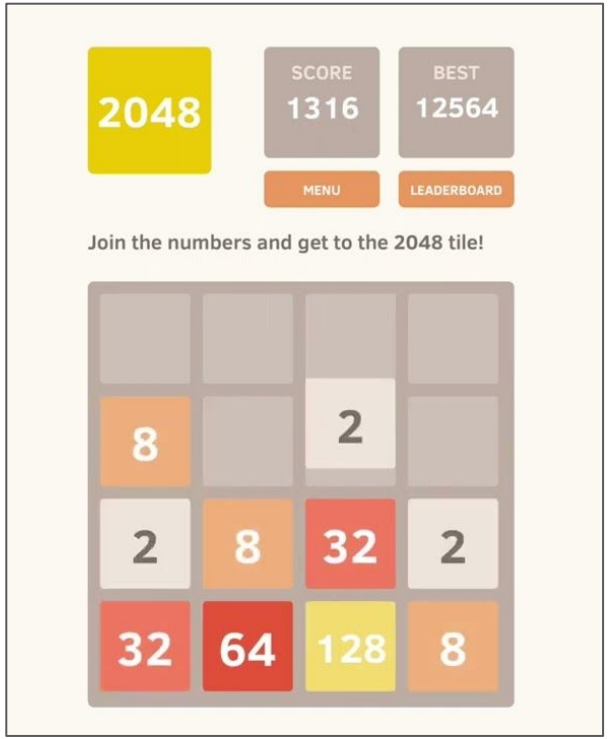


Figure 1. 2048 Game

What is our goal?

- Implement and test **algorithms**:
  - Min-Max Tree Search
  - Monte Carlo Tree Search
  - Double Deep Q-Network (DDQN) (it is possible to implement DDQN to play 2048 [2])
- Improve the **success rate** (previously it was 7% for achieving 2048 [3])
- **Comparative** analysis
- **Metrics**: Average score, maximum score and highest tile

Environment

- OpenAI Gym library wrapper for 2048
- **Reward** = sum of combined tiles
  R => 16 + 16 = 32

- **State Space:** 4x4 matrix
  Contains values of the tiles in log2 form
  Matrix Values:

0	1	0	0
0	4	7	10
0	5	6	4
3	2	2	1

- **Action Space:** up, down, left, and right

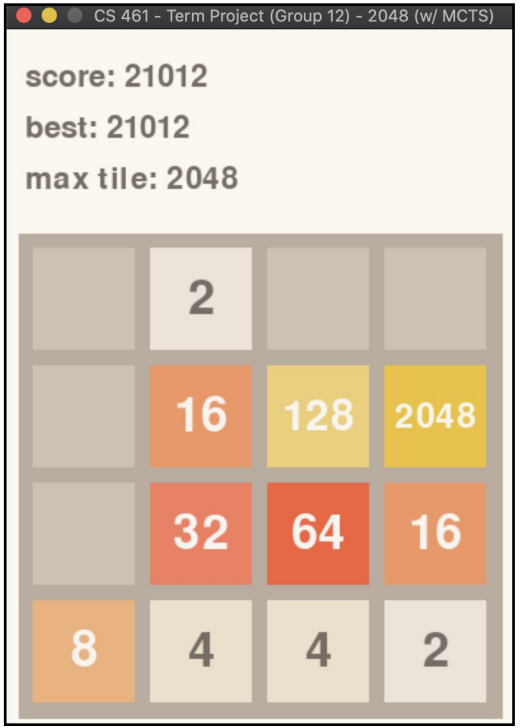


Figure 2. Screenshot of the 2048 environment

Technologies & Libraries

Technologies and libraries used in the project:

- Python
- Numpy
- PyTorch
- Gym
- Matplotlib

Figure 3. Logo of PyTorch

Algorithms

Min-Max Tree Search

- Depth: 3

- Heuristics
  - Corner Heuristic
  - Branch Failure Penalty

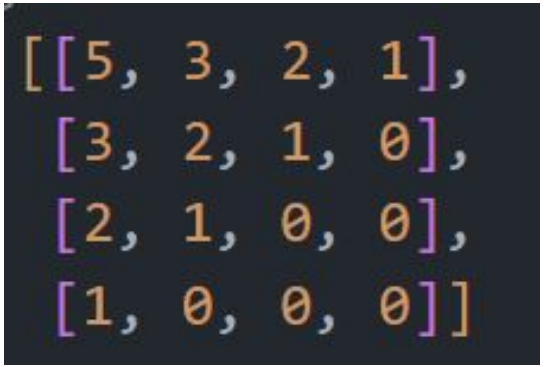


Figure 4. Corner heuristic tile weights

```
BRANCH_FAILIURE_PENALTY = -100000
```

Figure 5. Penalty for game termination

Min-Max Tree Search Results

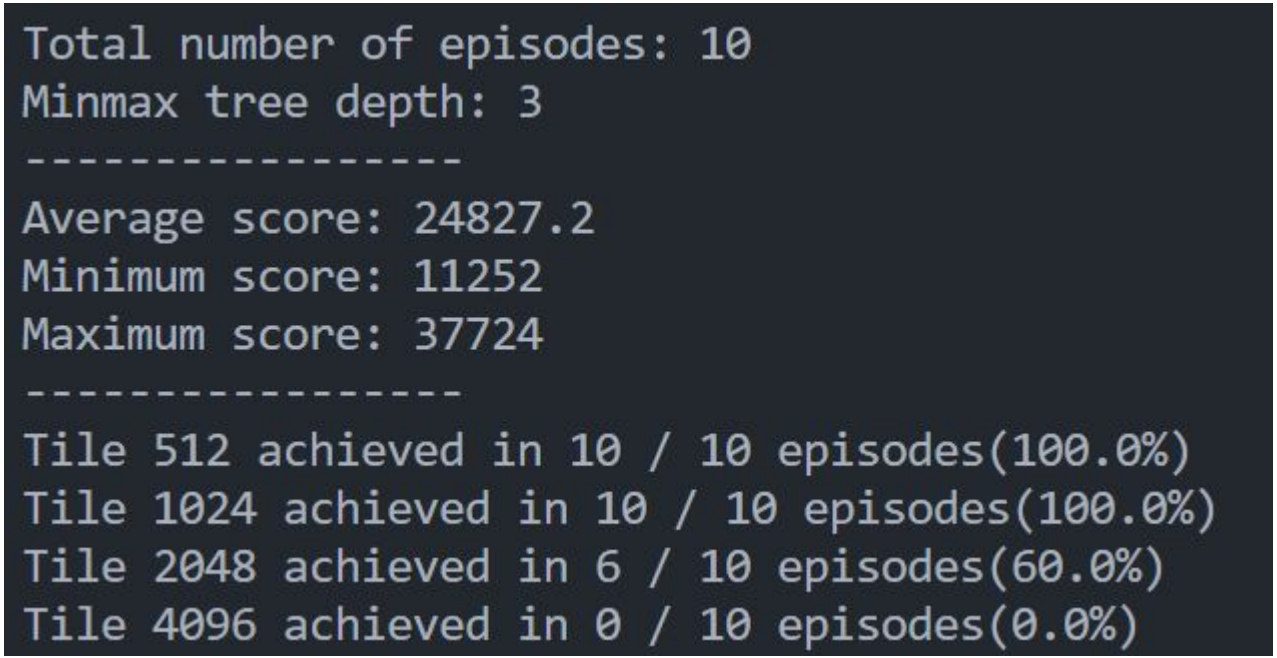


Figure 6. Results for depth 3 Min-Max Search Tree

Min-Max Tree Search Result Plots

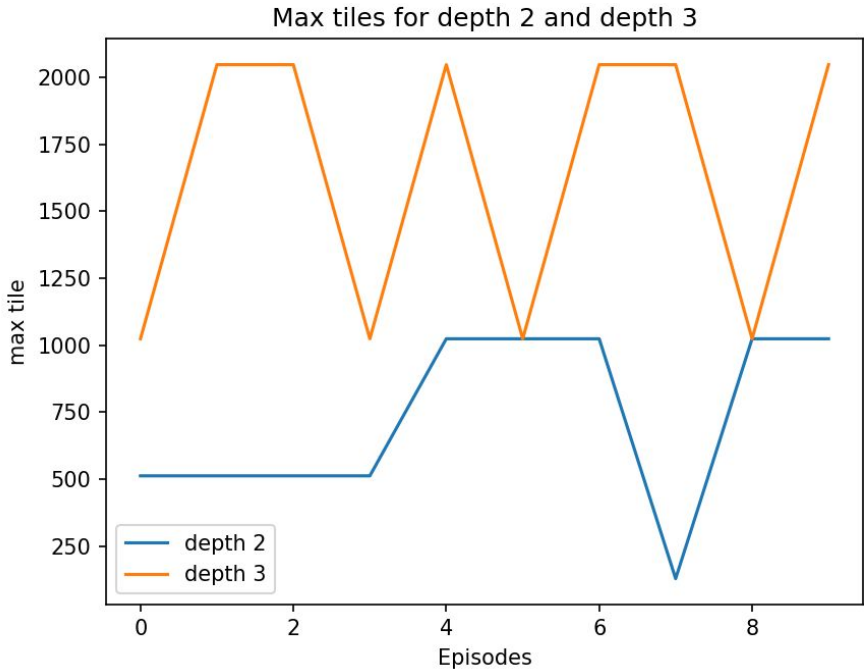


Figure 7. Maximum Tile Reached vs. Number of Episodes

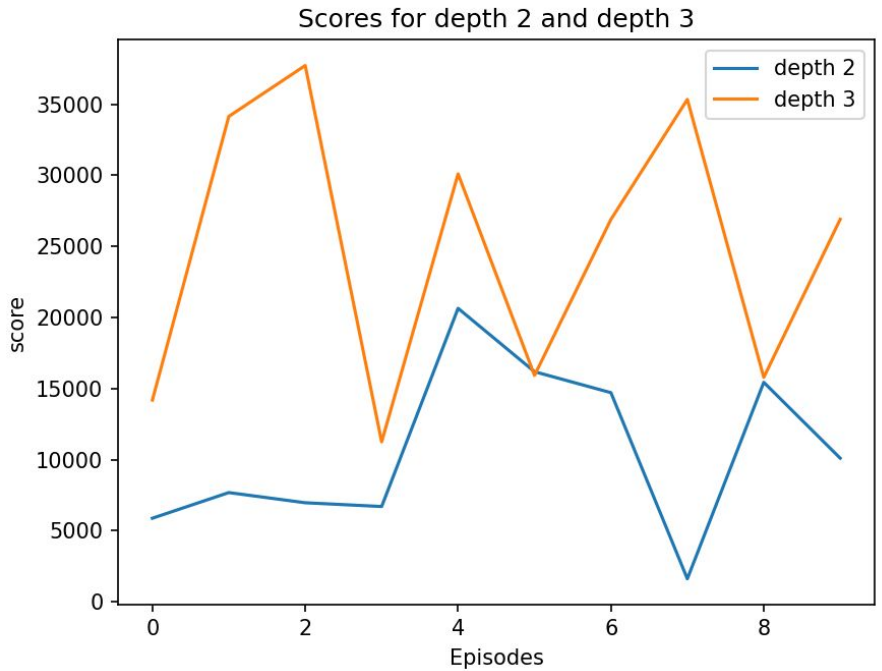


Figure 8. Score Reached vs. Number of Episodes

Min-Max-Tree Search Discussion

- Success rate and highest tile increases with increased depth
  - More foresight
  - Performance problems on higher depths
    - High branching rate despite pruning

Monte Carlo Tree Search Algorithm

- The algorithm was tested for depths 100 and 150
- For depth 100:
  - It can reach the 2048 tile with an 80% success rate
  - It can reach 1024 tile with a 100% success rate
- For depth 150:
  - It can reach the 2048 tile with a 100% success rate
  - It can reach 1024 tile with a 100% success rate

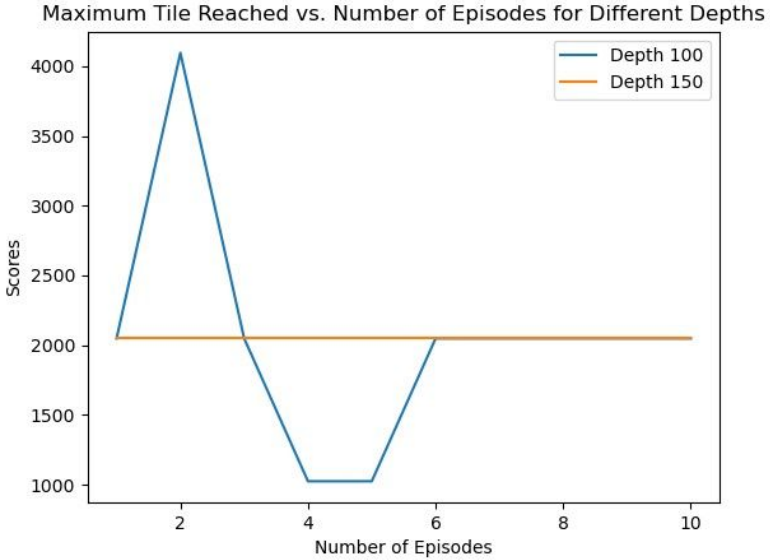


Figure 9. Maximum Tile Reached vs. Number of Episodes

Monte Carlo Tree Search Algorithm

- When the depth increases, the algorithm's overall performance also increases:
  - Because the explored tree depth is increased
  - A more thorough exploration of possible moves and outcomes was allowed

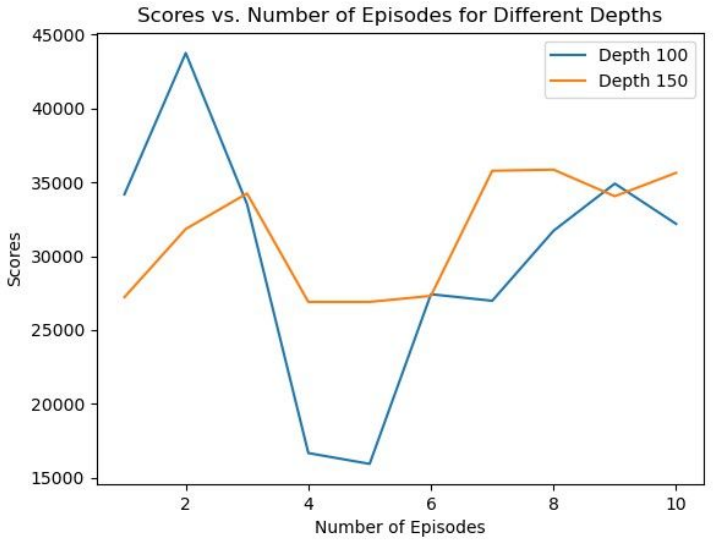


Figure 10. Scores vs. Number of Episodes

Monte Carlo Tree Search Algorithm Comparison

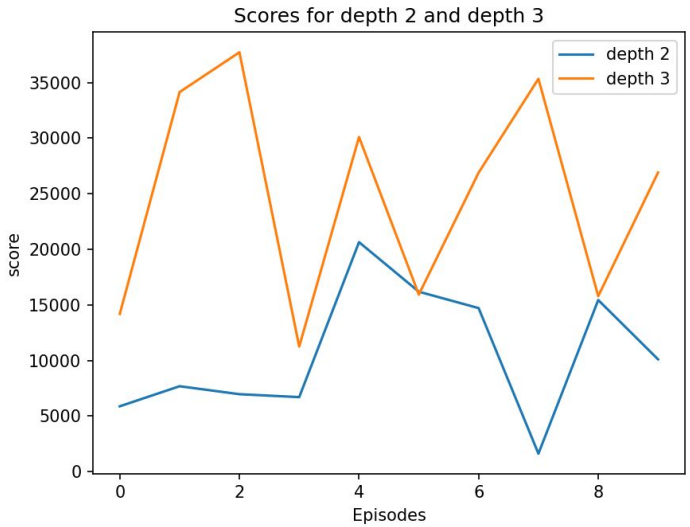


Figure 11. Min-Max Tree Scores

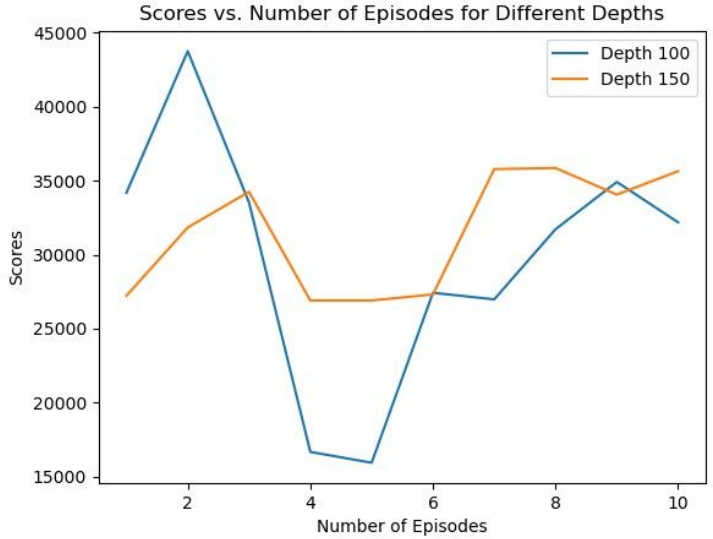


Figure 12. MCTS Scores

- The average performance scores and success rates are higher than the Min-Max tree search algorithm's performance
- Explores promising branches of the search tree
- The Monte Carlo Tree Search algorithm is more suitable for games with uncertainty or stochastic elements

DDQN

- The DDQN Algorithm was tested for:
  - Learning Rates: 0.0001, 0.00025, 0.0005
  - Gamma Values: 0.1, 0.5, 0.8
  - Buffer Sizes: 10000, 50000, 100000
- Results are shown over 1000 episodes (~one million steps) for this algorithm.
- Otherwise, the default values are:
  - LR: 0.0001, Gamma: 0.99, Buffer Size: 10000
  - Epsilon: 1, Epsilon Decay: 0.995, Minimum Epsilon: 0.01

DDQN - Learning Rate Experiments (Max Tiles)

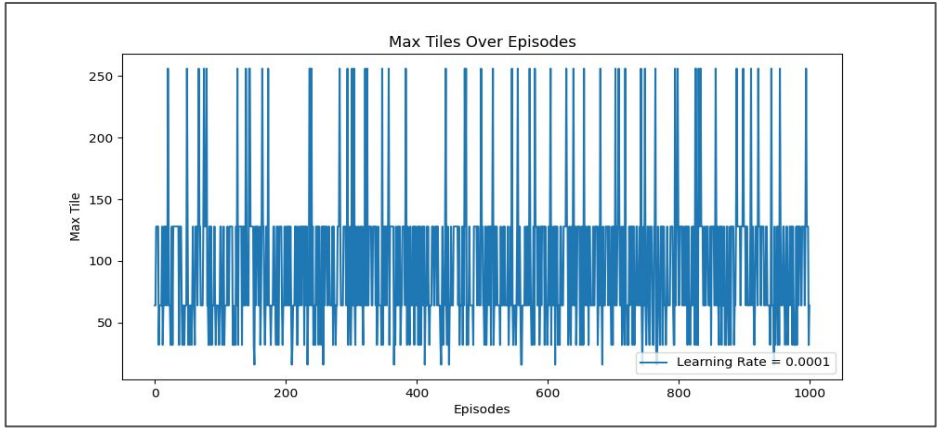


Figure 13. Max Tile vs. Episode (learning rate=1E-4)

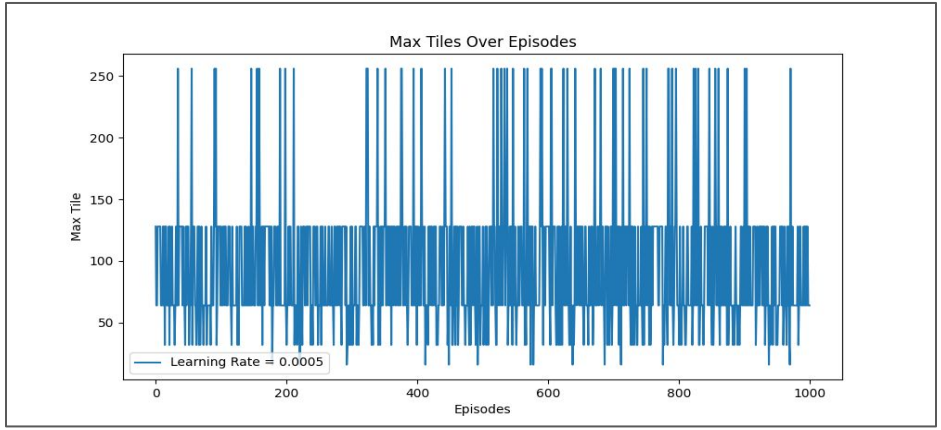


Figure 14. Max Tile vs. Episode (learning rate=5E-4)

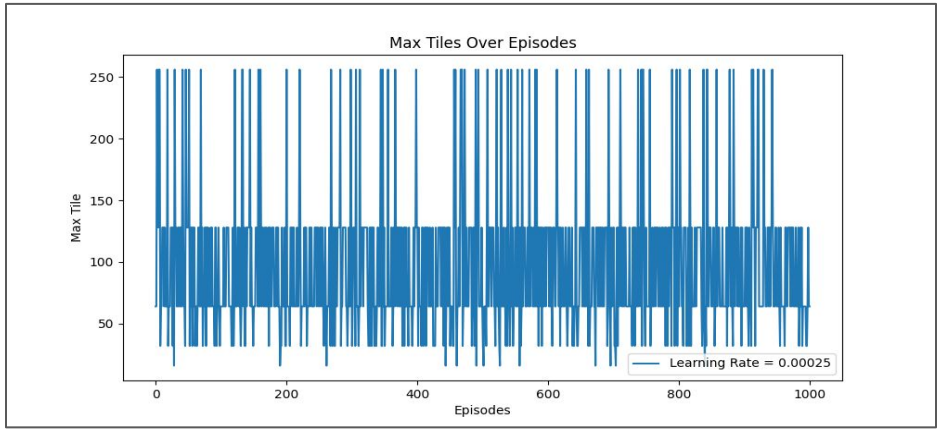


Figure 15. Max Tile vs. Episode (learning rate=2.5E-4)

DDQN - Learning Rate Experiments (Scores)

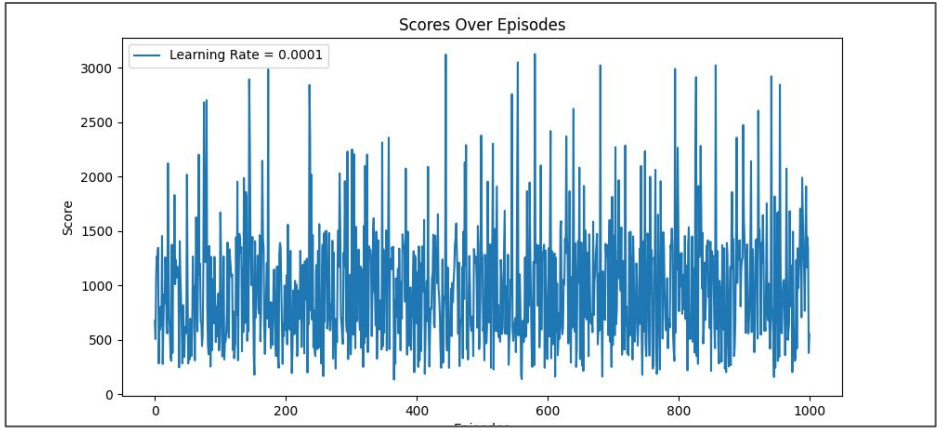


Figure 16. Score vs. Episode (learning rate=1E-4)

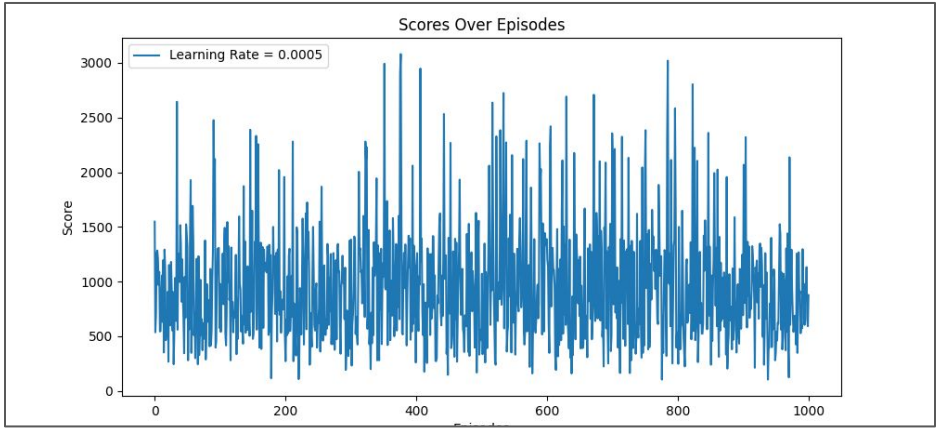


Figure 17. Score vs. Episode (learning rate=5E-4)

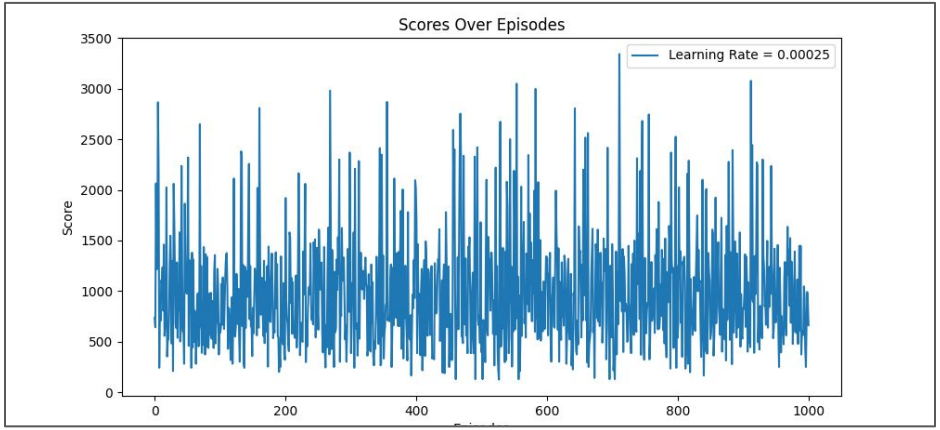


Figure 18. Score vs. Episode (learning rate=2.5E-4)

DDQN - Learning Rate Experiments (Scores)

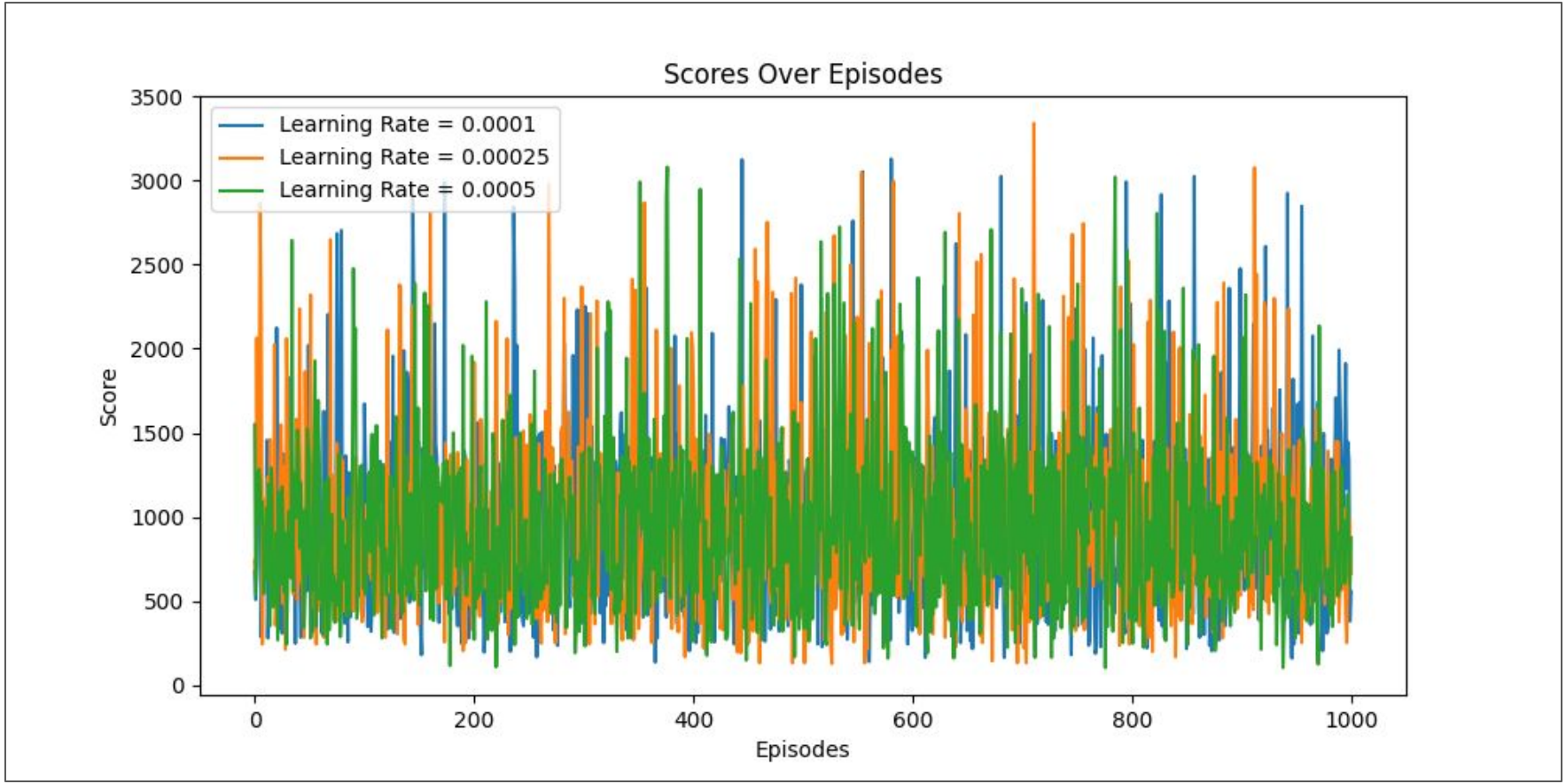


Figure 19. DDQN Score vs. Episode (with different learning rates)

DDQN - Learning Rate Experiments

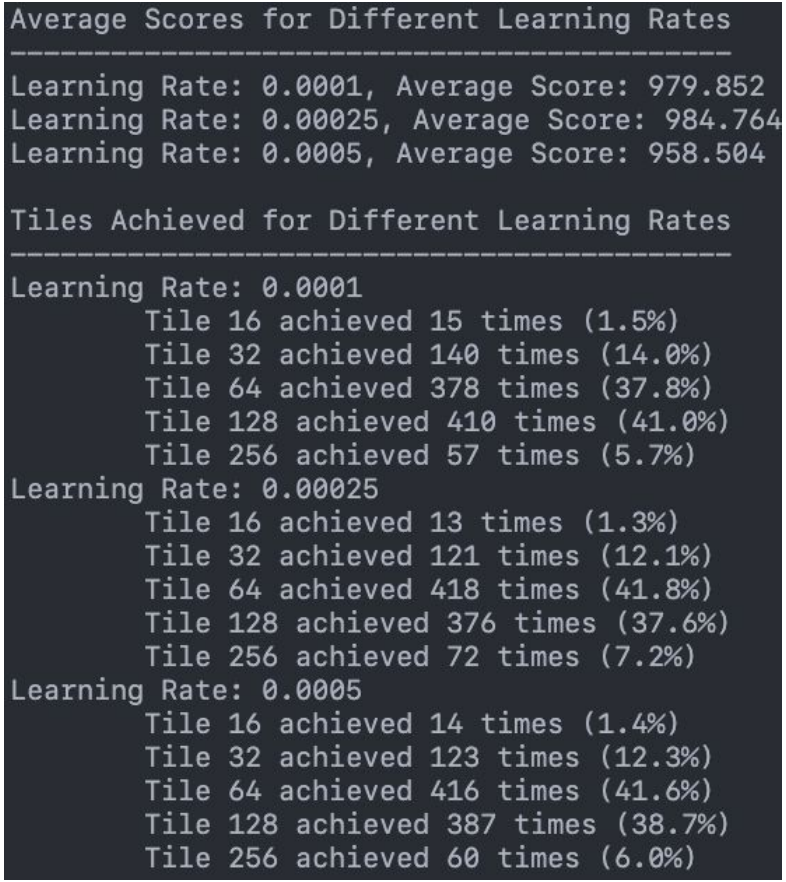


Figure 20. DDQN Summary Table (with different learning rates)

DDQN - Gamma Value Experiments (Max Tiles)

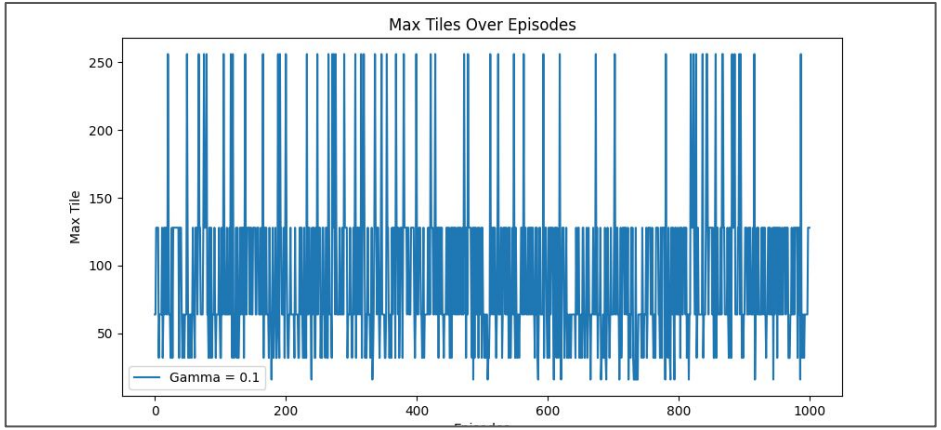


Figure 21. Max Tile vs. Episode (gamma value=0.1)

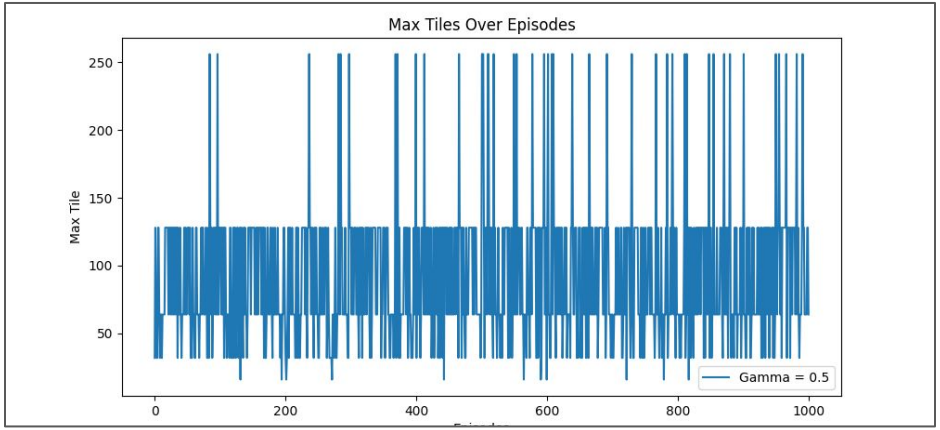


Figure 22. Max Tile vs. Episode (gamma value=0.5)

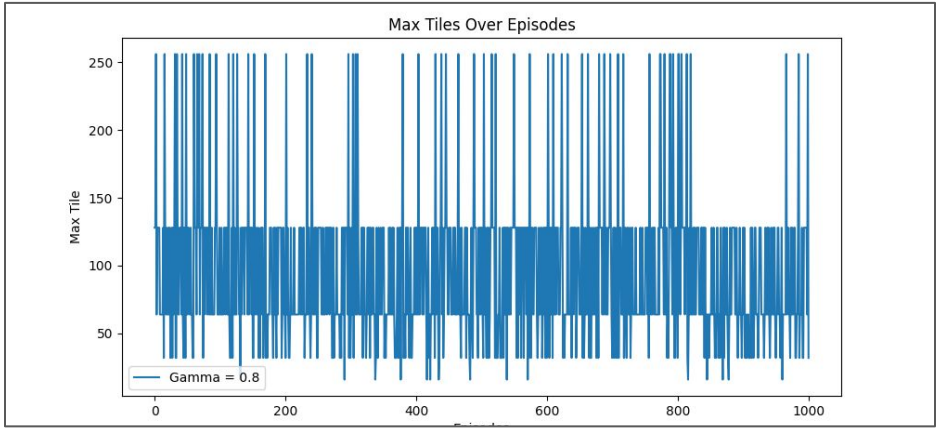


Figure 23. Max Tile vs. Episode (gamma value=0.8)

DDQN - Gamma Value Experiments (Scores)

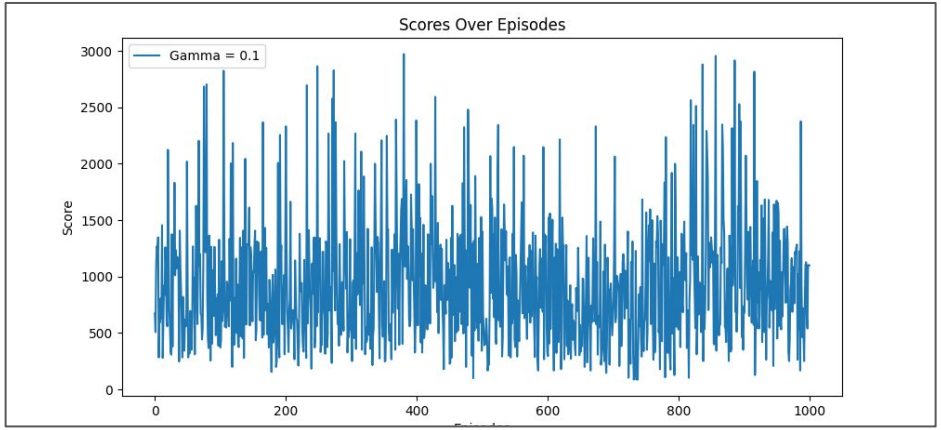


Figure 24. Score vs. Episode (gamma value=0.1)

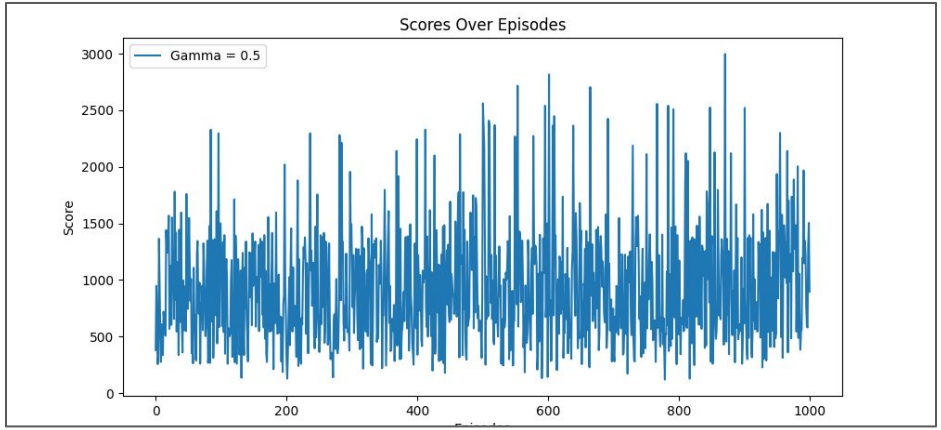


Figure 25. Score vs. Episode (gamma value=0.5)

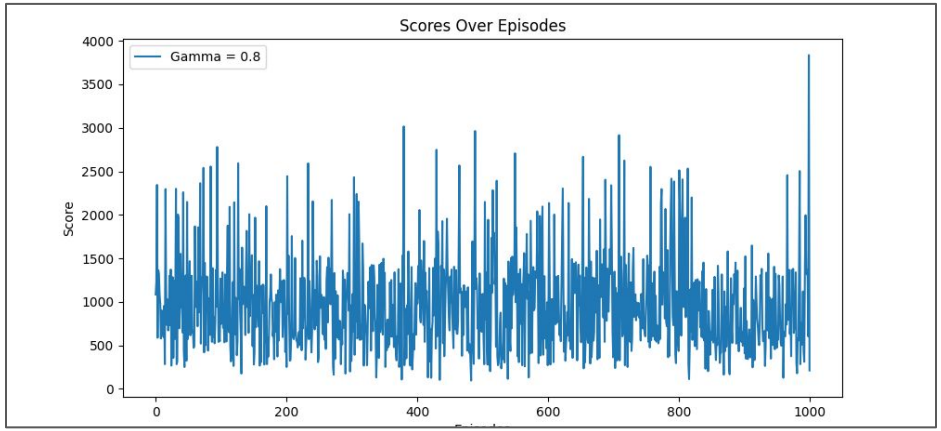


Figure 26. Score vs. Episode (gamma value=0.8)

DDQN - Gamma Value Experiments (Scores)

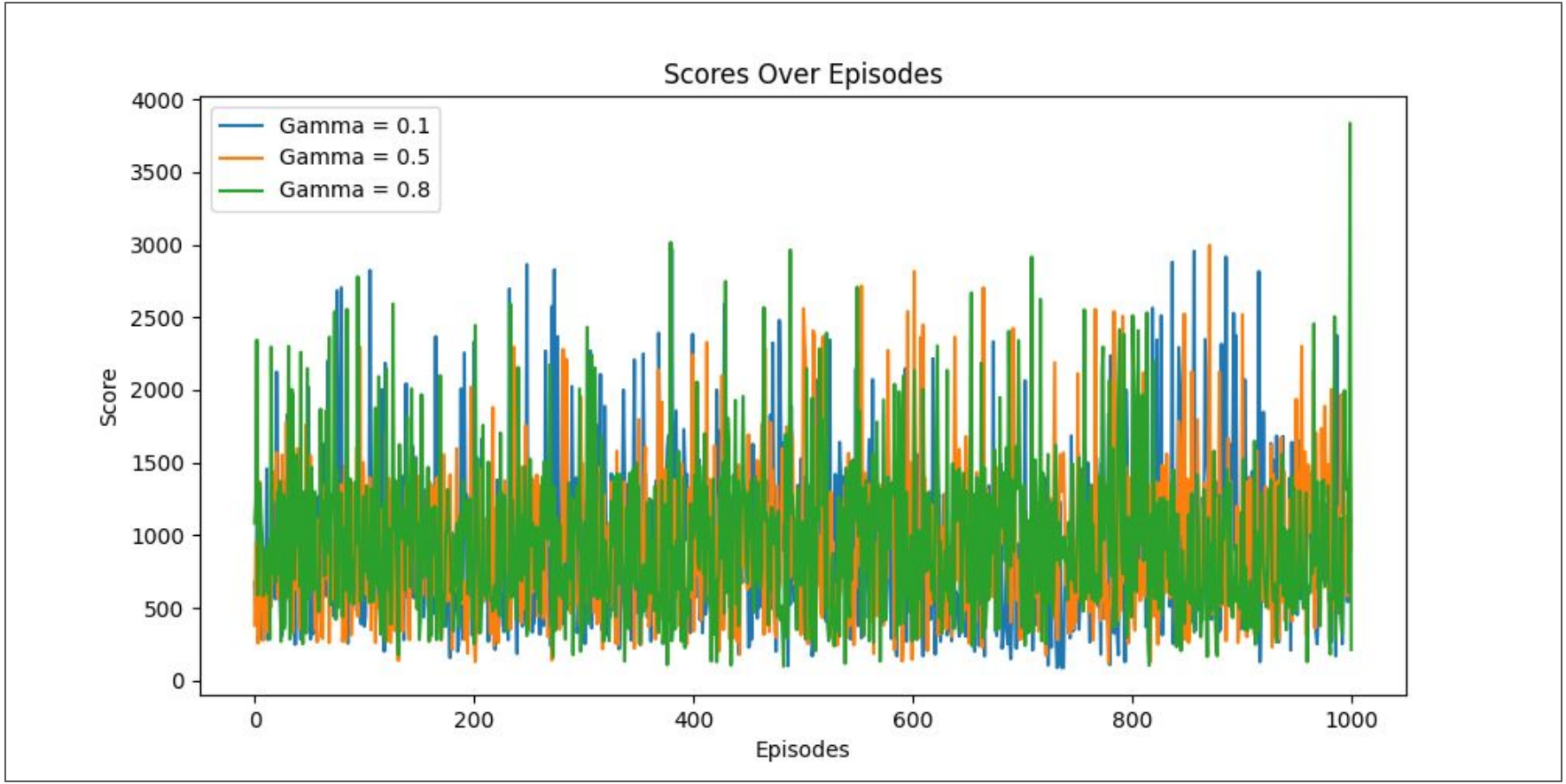


Figure 27. DDQN Score vs. Episode (with different gamma values)

DDQN - Gamma Value Experiments

```
Average Scores for Different Gamma Values
-----------------------------------------
Gamma: 0.1, Average Score: 928.34
Gamma: 0.5, Average Score: 924.256
Gamma: 0.8, Average Score: 963.048

Tiles Achieved for Different Gamma Values
-----------------------------------------
Gamma: 0.1
        Tile 16 achieved 26 times (2.6%)
        Tile 32 achieved 146 times (14.6%)
        Tile 64 achieved 408 times (40.8%)
        Tile 128 achieved 363 times (36.3%)
        Tile 256 achieved 57 times (5.7%)
Gamma: 0.5
        Tile 16 achieved 11 times (1.1%)
        Tile 32 achieved 157 times (15.7%)
        Tile 64 achieved 386 times (38.6%)
        Tile 128 achieved 403 times (40.3%)
        Tile 256 achieved 43 times (4.3%)
Gamma: 0.8
        Tile 16 achieved 15 times (1.5%)
        Tile 32 achieved 146 times (14.6%)
        Tile 64 achieved 389 times (38.9%)
        Tile 128 achieved 387 times (38.7%)
        Tile 256 achieved 63 times (6.3%)
```

Figure 28. DDQN Summary Table (with different gamma values)

DDQN - Buffer Size Experiments (Max Tiles)

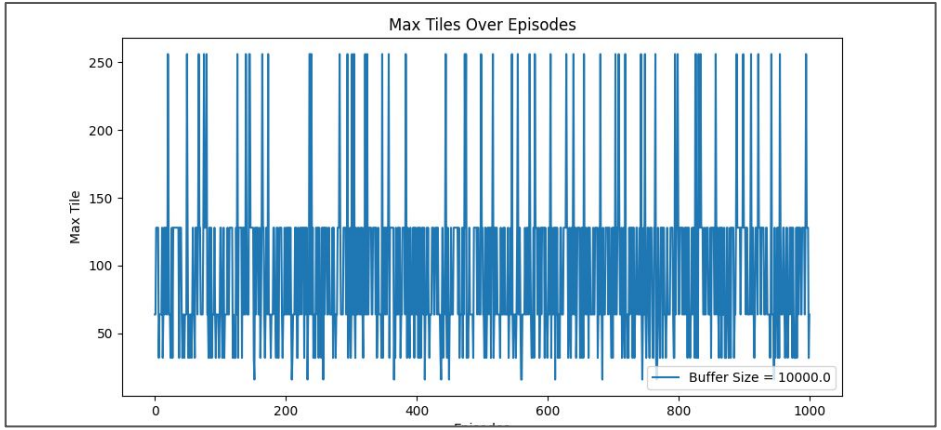


Figure 29. Max Tile vs. Episode (batch size=1E4)

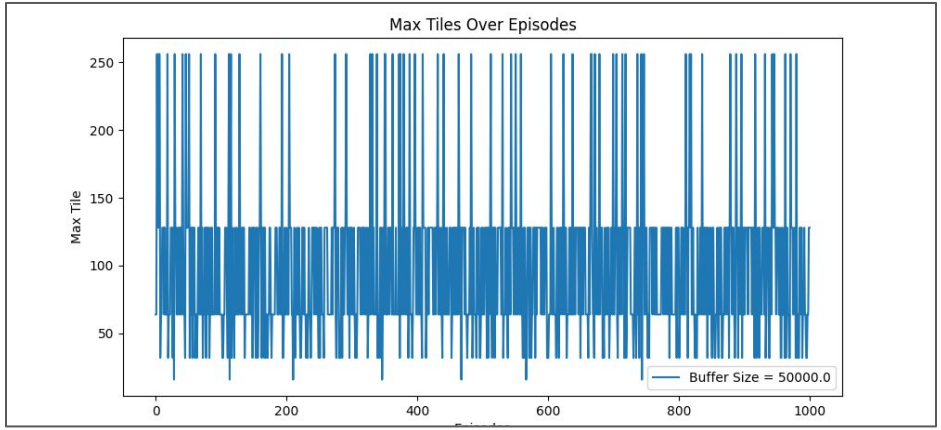


Figure 30. Max Tile vs. Episode (batch size=5E4)

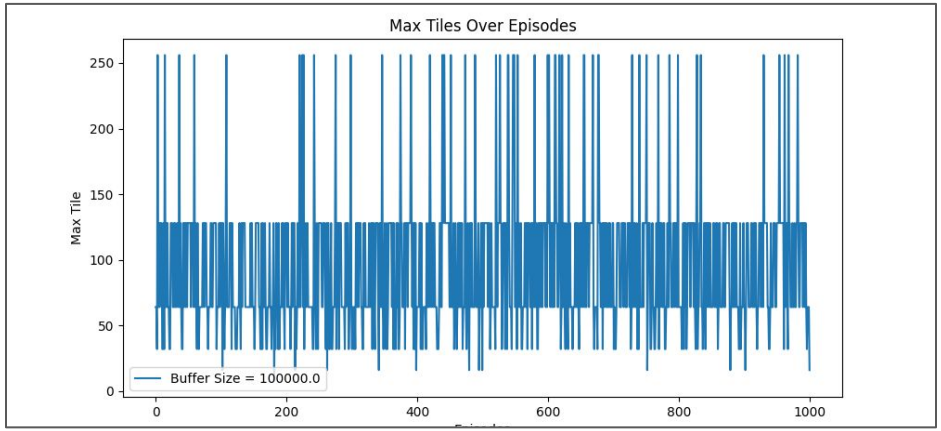


Figure 31. Max Tile vs. Episode (batch size=1E5)

DDQN - Buffer Size Experiments (Scores)

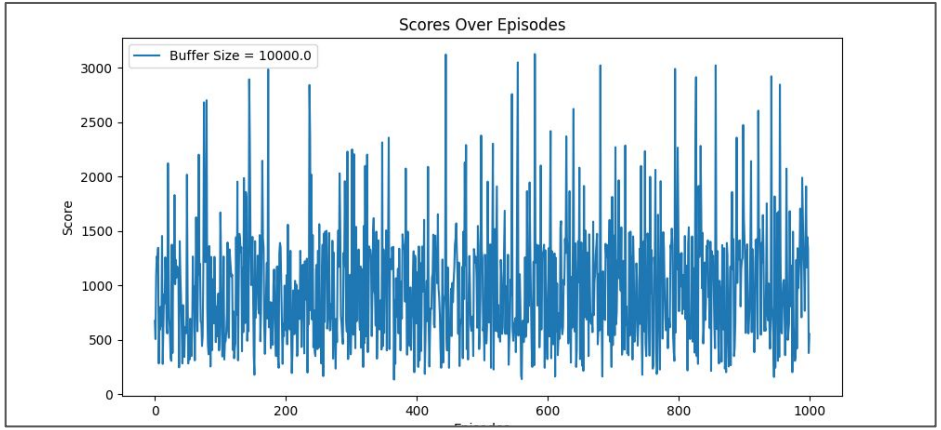


Figure 32. Score vs. Episode (batch size=1E4)

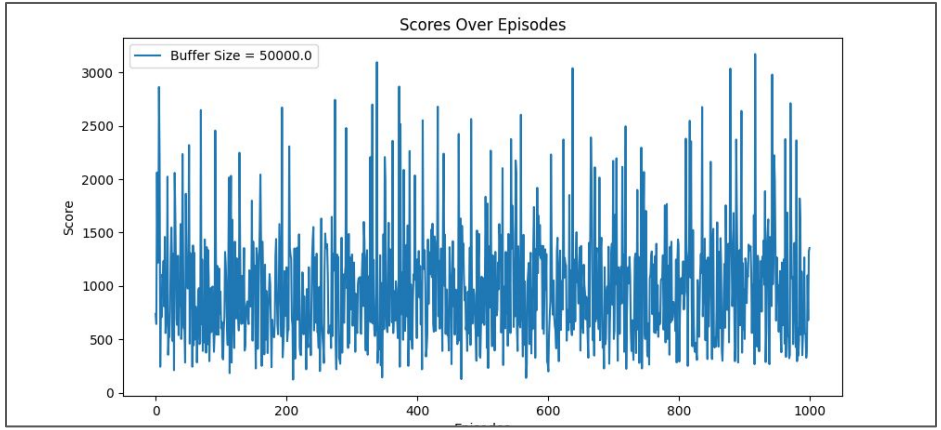


Figure 33. Score vs. Episode (batch size=5E4)

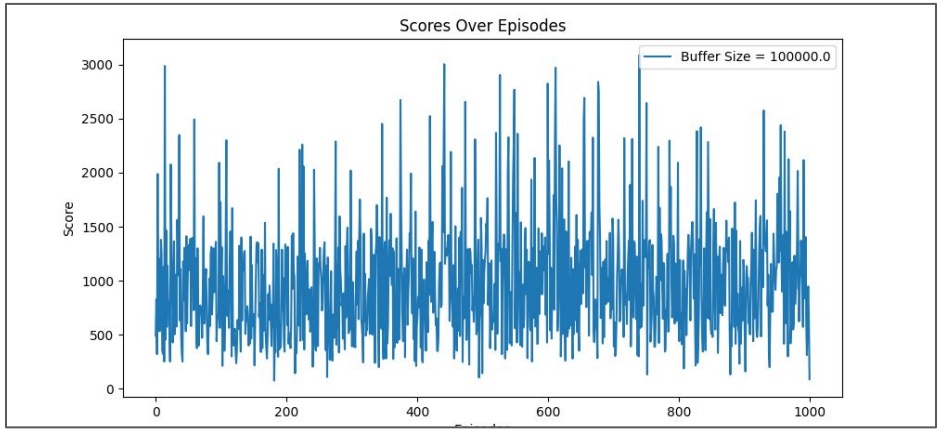


Figure 34. Score vs. Episode (batch size=1E5)

DDQN - Buffer Size Experiments (Scores)

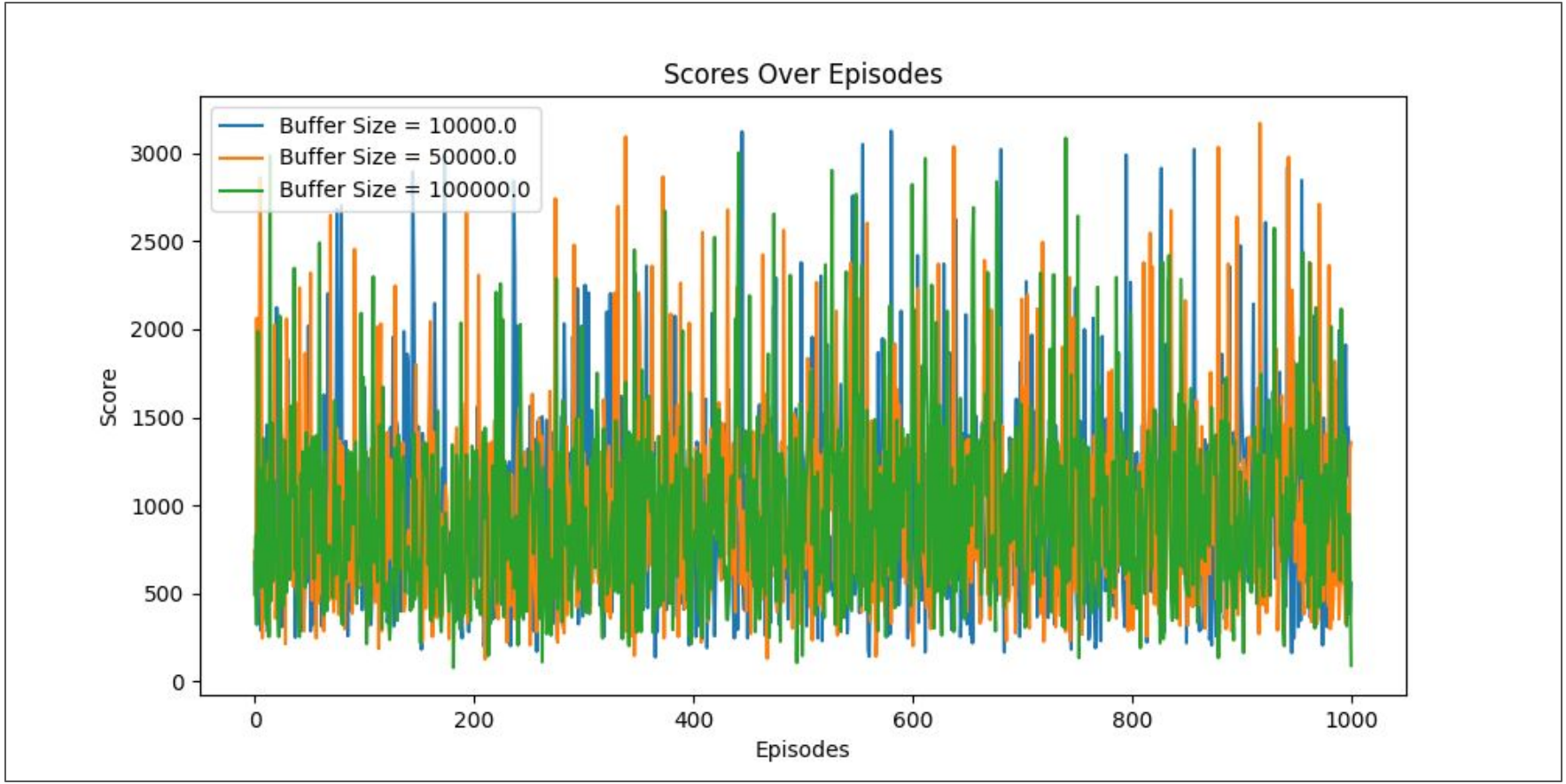


Figure 35. DDQN Score vs. Episode (with different batch sizes)

DDQN - Buffer Size Experiments

```
Average Scores for Different Buffer Sizes
-----------------------------------------
Buffer Size: 10000, Average Score: 979.852
Buffer Size: 50000, Average Score: 989.64
Buffer Size: 100000, Average Score: 976.432

Tiles Achieved for Different Buffer Sizes
-----------------------------------------
Buffer Size: 10000
        Tile 16 achieved 15 times (1.5%)
        Tile 32 achieved 140 times (14.0%)
        Tile 64 achieved 378 times (37.8%)
        Tile 128 achieved 410 times (41.0%)
        Tile 256 achieved 57 times (5.7%)
Buffer Size: 50000
        Tile 16 achieved 7 times (0.7%)
        Tile 32 achieved 129 times (12.9%)
        Tile 64 achieved 390 times (39.0%)
        Tile 128 achieved 407 times (40.7%)
        Tile 256 achieved 67 times (6.7%)
Buffer Size: 100000
        Tile 8 achieved 1 times (0.1%)
        Tile 16 achieved 12 times (1.2%)
        Tile 32 achieved 125 times (12.5%)
        Tile 64 achieved 406 times (40.6%)
        Tile 128 achieved 401 times (40.1%)
        Tile 256 achieved 55 times (5.5%)
```

Figure 36. DDQN Summary Table (with different gamma values)

DDQN - Discussion

- DDQN performs worse than both Min-Max Tree Search and MCTS. There might be several underlying reasons for that:
  - Complexity of the Game:
    - 2048 involves a significant amount of strategic depth.
    - MCTS and Min-Max Tree Search may be performing due to their lookahead capabilities.
    - DDQN, being a value-based method, might fail to plan strategically. It is much easier for DDQN to learn to play control/instant action games (e.g. Pong)

DDQN - Discussion

- Other threats for the DDQN's performance can include:
  - Neural Network Architecture
  - Hyperparameters
  - Training Duration
  - Rewarding System

Conclusion

- We explored the ways to “solve” the game 2048 with AI algorithms.
- We used Min-Max Tree Search, MCTS, and DDQN algorithms
- Performances: MCTS > Min-Max Tree Search > DDQN
  - Due to the previously discussed points, plus the fact that:
  - Tree-based algorithms handle the future planning better, and perform well on strategical games. Hence, 2048 is not an exception.

Workload Distribution

- Ceren Akyar: MCTS experiments, plotting & inferring the results
- Deniz Mert Dilaverler: Implementation of Min-Max Tree Search and its experiments
- Berk Çakar: Implementation of MCTS, DDQN algorithms
- Elifsena Öz: DDQN experiments, plotting & inferring the results
- İpek Öztaş: DDQN experiments, plotting & inferring the results

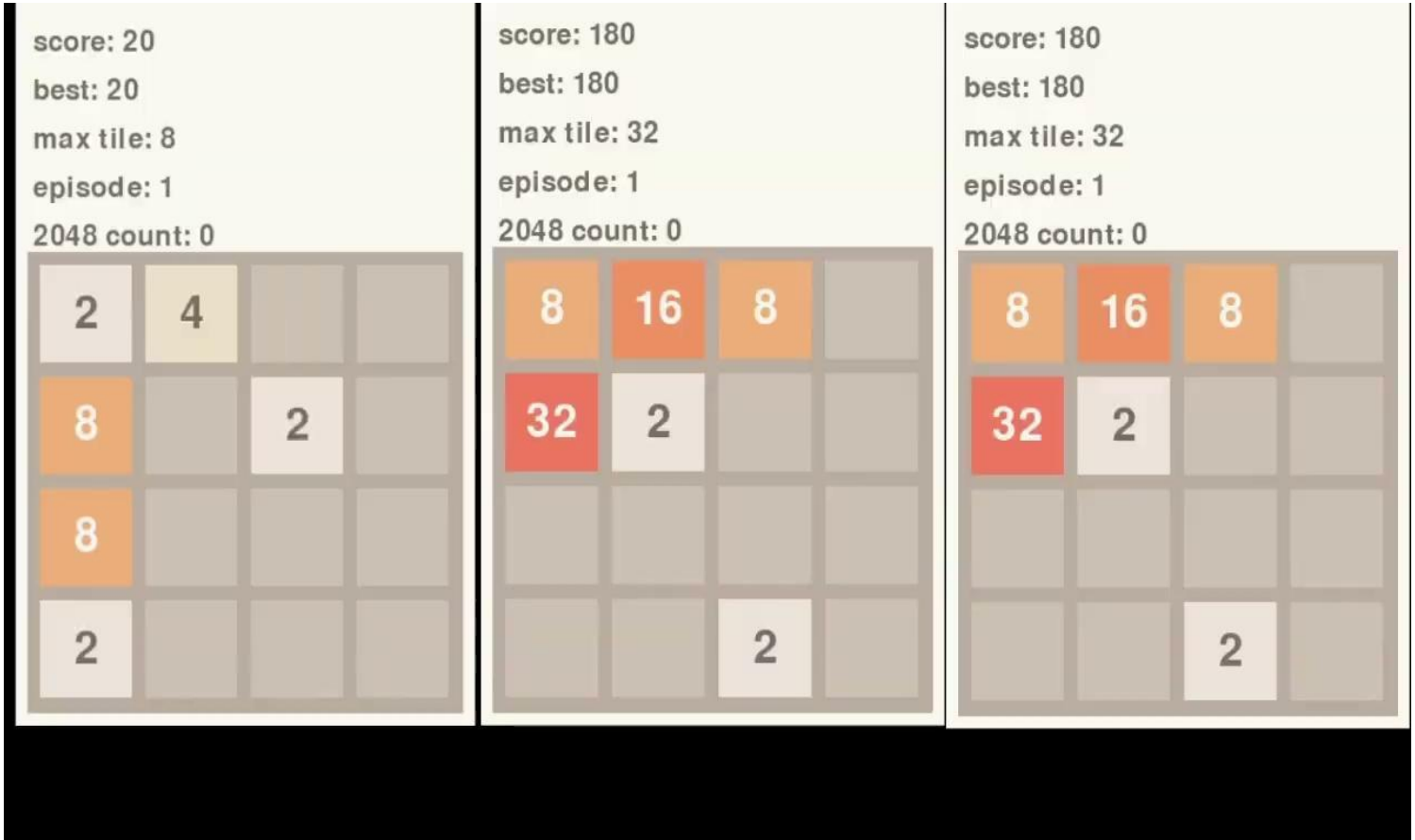
References

[1] "2048 (video game)," Wikipedia. Oct. 20, 2023. Accessed: Nov. 26, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=2048_(video_game)&oldid=1185818281

[2] L. H. Chan, "Playing 2048 with Deep Q-Learning (with pytorch implementation)," Medium, Oct. 20, 2023. [Online]. Available: https://medium.com/@qwert12500/playing-2048-with-deep-q-learning-with-pytorch-implementation-4313291efe61. [Accessed: Oct. 20, 2023].

[3] A. Goga, *Reinforcement learning in 2048 game*, Oct. 2017. Accessed: Oct. 20, 2023. [Online]. Available: http://cogsci.fmph.uniba.sk/~farkas/theses/adrian.goga.bak18.pdf

Demo Video


score: 20
best: 20
max tile: 8
episode: 1
2048 count: 0
2 4
8 2
8
2
score: 180
best: 180
max tile: 32
episode: 1
2048 count: 0
8 16 8
32 2
2
score: 180
best: 180
max tile: 32
episode: 1
2048 count: 0
8 16 8
32 2
2