2021A7PS2627G

Kislay Ranjan Nee Tandon

Brief:

Tic-Tac-Toe is a classic two-player game played on a 3x3 grid. The marker, usually a "X" or a "O," is placed on an empty square by each player in turn. The goal is to line up three markers in a diagonal, vertical, or horizontal pattern while also keeping the opposition from doing the same. Despite its simplicity, Tic-Tac-Toe is a solved game, which means that no matter what the opponent does, an ideal strategy will guarantee a win or a draw. Because it enables quantifiable performance gains over time, this feature makes it a great testbed for investigating artificial intelligence techniques, especially reinforcement learning.

In this assignment, the focus shifts to a Shallow Q-Network (SQN), a simplified version of a Deep Q-Network (DQN). In order to reduce complexity while maintaining the fundamental ideas of reinforcement learning, the SQN employs a neural network with fewer layers than deep neural networks, which have numerous layers. A Q-function, which calculates the expected rewards for performing particular actions in predetermined game states, is approximated by the SQN. Through repeated training, the SQN gradually enhances its decision-making skills by interacting with the game environment. This entails using important strategies such as epsilon-greedy exploration, experience replay, and the Bellman equation for Q-value updates.

Disclaimer: The program utilizes random.seed('x') to ensure the results' reproducibility during training and evaluation. However, this approach results in deterministic gameplay by the opponent, causing it to make the same moves irrespective of the agent's strategies. Consequently, the program does not evaluate the adaptability of the opponent's decision-making and may not reflect the dynamic challenges posed by real-time smart opponents.

Approach:

The foundational approach to building and training the Shallow Q-Network (SQN) involved creating a reinforcement learning agent that learns optimal strategies for Tic-Tac-Toe by interacting with the game environment. The agent utilized a neural network to approximate Q-values for all possible actions in a given state. A key focus was on implementing the core components of Q-learning: action selection, reward assignment, and iterative Q-value updates using the Bellman equation. The training pipeline included collecting experiences through gameplay, storing them in a replay buffer, and periodically training the neural network on mini-batches sampled from this buffer to ensure stability and efficiency.

The training process began with high exploration through an epsilon-greedy policy (epsilon = 1.0), which allowed the agent to try diverse actions. Over time, epsilon decayed to favor exploitation of learned strategies. The opponent's difficulty level was progressively increased by adjusting the smartMovePlayer1 parameter, ensuring the agent faced increasingly complex scenarios as it improved. This stepwise progression allowed the agent to adapt gradually while maintaining a robust learning process.

To compare the models, we generated 1000 random seed values ranging from 1000 to 10000. Each model was then evaluated using these seed values to ensure a comprehensive and unbiased comparison. By running each model on the same set of seed values, we could accurately assess their performance across various scenarios. This method allowed us to determine which model performed best in terms of wins, losses, and draws, providing a clear benchmark for their effectiveness.

SQN Implementation and Training Process:

The Shallow Q-Network (SQN) was implemented as a reinforcement learning agent designed to play Tic-Tac-Toe. The core design principles revolved around approximating Q-values using a neural network, storing gameplay experiences for replay, and iteratively improving the agent's decision-making through training. Below is a detailed breakdown of the implementation:

1. **Neural Network Architecture :** The SQN used a neural network implemented with TensorFlow/Keras. It had:
    - **Input Layer**: A vector of size 9 representing the Tic-Tac-Toe board state. Each position could be 0 (empty), 1 (player 1), or 2 (player 2). This was preprocessed into values of -1, 0, and 1 to improve learning.
    - **Hidden Layers**: Two fully connected layers with 64 neurons each, using the ReLU activation function to capture non-linear patterns.
    - **Output Layer**: A layer with 9 linear outputs, each representing the Q-value of a possible move.

    The network was compiled with the Mean Squared Error (MSE) loss function and the Adam optimizer with a learning rate of 0.001 to balance efficiency and stability.

2. **Reinforcement Learning Process :** ng.
    - **Action Selection**:
      Actions were chosen using an epsilon-greedy policy.
        - With probability epsilon, the agent explored by selecting a random valid action.
        - Otherwise, it exploited by selecting the action with the highest predicted Q-value among valid actions.

    - **Experience Replay**:
      Gameplay experiences were stored in a replay buffer as tuples of (state, action, reward, next_state, done). This allowed the agent to train on past experiences, breaking correlations in sequential data and improving stability.
      Actions were chosen using an epsilon-greedy policy.

    - **Q-Value Updates**:
      During training, mini-batches of experiences were sampled from the replay buffer. For each experience, the target Q-value was computed using the Bellman equation:

$$Q_{target}(s,a) = r + \gamma a' \max Q(s',a')$$

Here, rrr is the immediate reward, γ\gammaγ is the discount factor (0.95), and Q(s′,a′)Q(s', a')Q(s′,a′) is the predicted Q-value for the next state.

- **Training Loop**:
  The training process iterated through episodes, where the agent played games against an opponent with progressively increasing smartness (smartMovePlayer1). The agent's epsilon was gradually decayed (from 1.0 to 0.01) to shift from exploration to exploitation. Training was performed after every episode if enough samples existed in the replay buffer.

- **Default Values**:

SQNAgent Class

- **state_size**: 9
- **action_size**: 9
- **gamma**: 0.95
- **learning_rate**: 0.001
- **epsilon**: 1.0
- **epsilon_min**: 0.01
- **epsilon_decay**: 0.9995
- **batch_size**: 32
- **replay_buffer**: deque(maxlen=10000)

Model Architecture

- **First Dense Layer**: 64 units, ReLU activation
- **Second Dense Layer**: 64 units, ReLU activation
- **Output Layer**: 9 units, Linear activation
- **Loss Function**: Mean Squared Error (MSE)
- **Optimizer**: Adam with learning_rate=0.001
-

Model 1: First Implementation Specifics:

Model 1 was the initial version of the SQN, designed with the above principles. It acted as a baseline for further experiments. Below are the specifics:

1. **Training Initialization:**
   - epsilon started at 1.0 to allow extensive exploration.

- A replay buffer size of 10,000 was chosen to store diverse experiences.
- The batch size for training was set to 32.

2. **Progressive Smartness**:

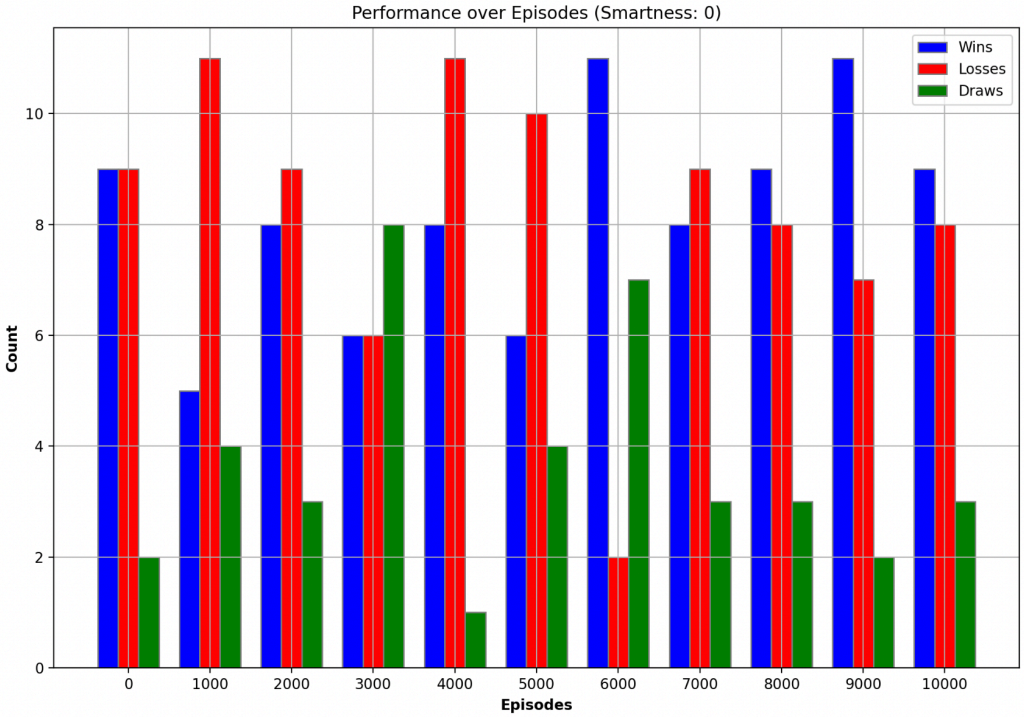
   The opponent's smartness, controlled by the smartMovePlayer1 parameter, increased as training progressed. It started as a random player (smartness = 0) and reached a fully intelligent player (smartness = 1) by 60% of the total episodes.

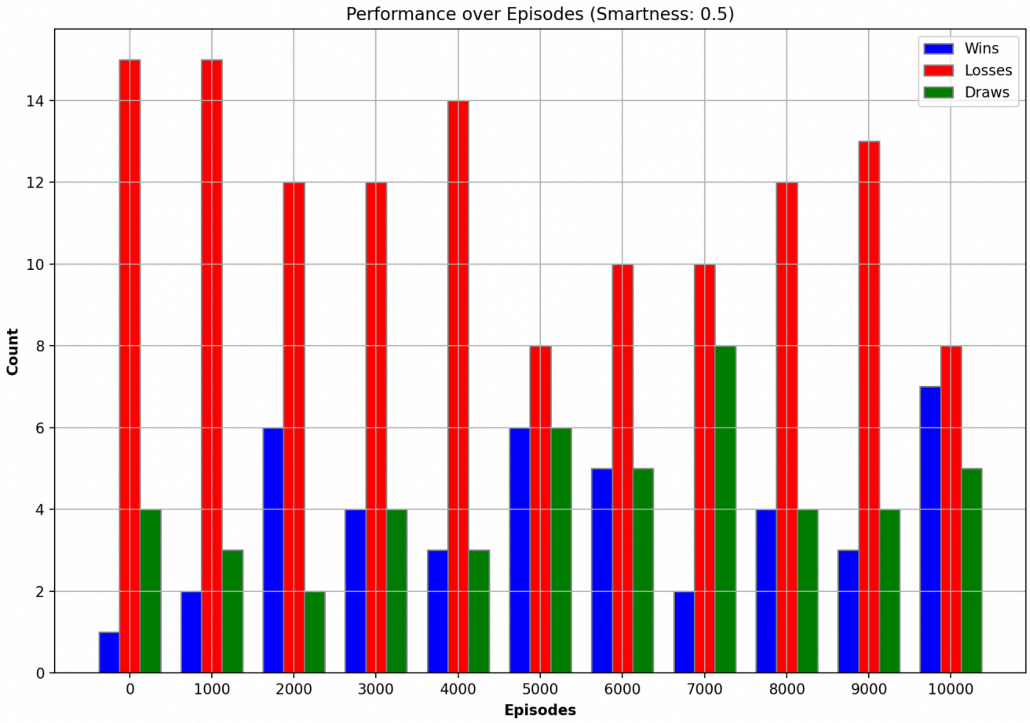
3. **Reward Structure**:

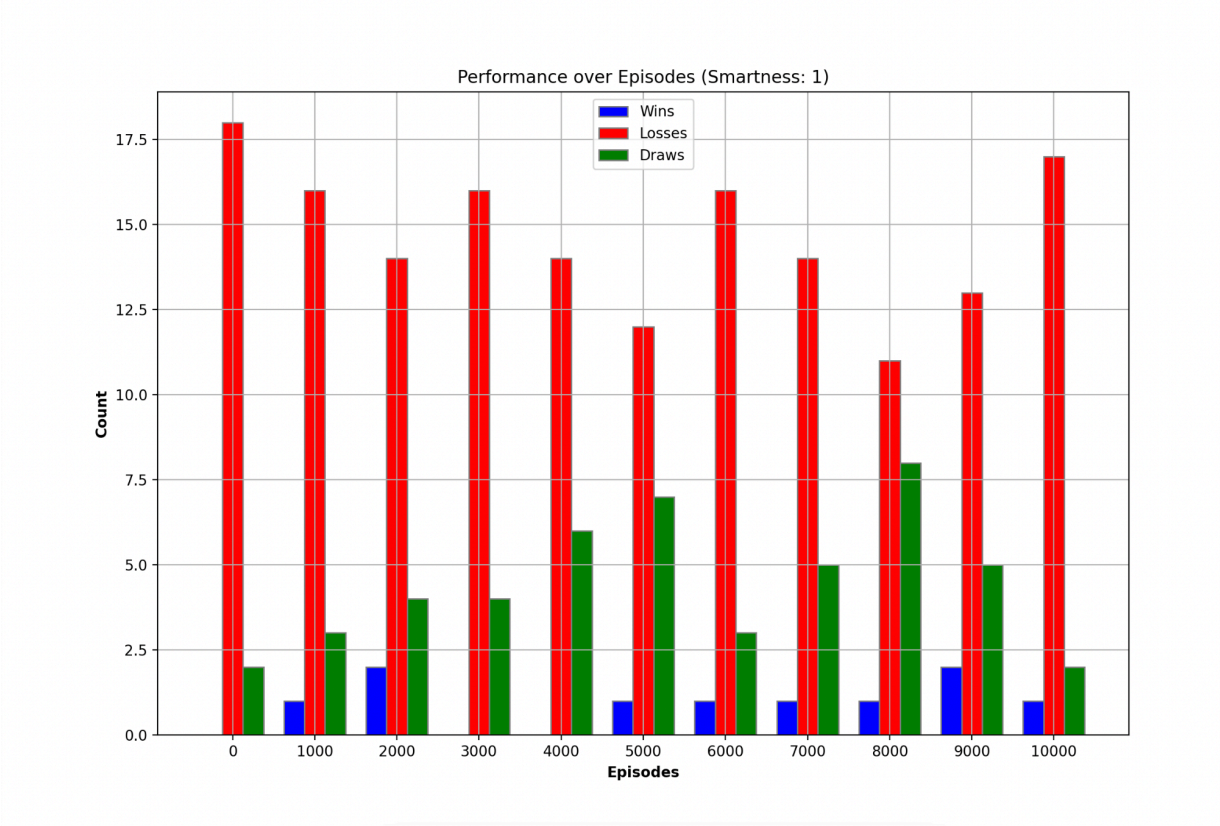
   Rewards were set as:

   - +1 for a win,
   - -1 for a loss,
   - 0 for a draw.

4. **Performance Checkpoints**:
   - Model weights were saved every 1,000 episodes for comparison and analysis (model1_episode_x.h5).
   - The final model was saved as model1.h5.
   - Performance metrics, including win rate, loss rate, and draw rate, were logged every 100 episodes for evaluation.
5. **Results over number of episode compared over 20 seed values:**:


Performance over Episodes (Smartness: 0)
Wins
Losses
Draws
Count
Episodes
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
0
2
4
6
8
10


Performance over Episodes (Smartness: 0.5)
Wins
Losses
Draws
Count
Episodes
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
0
2
4
6
8
10
12
14

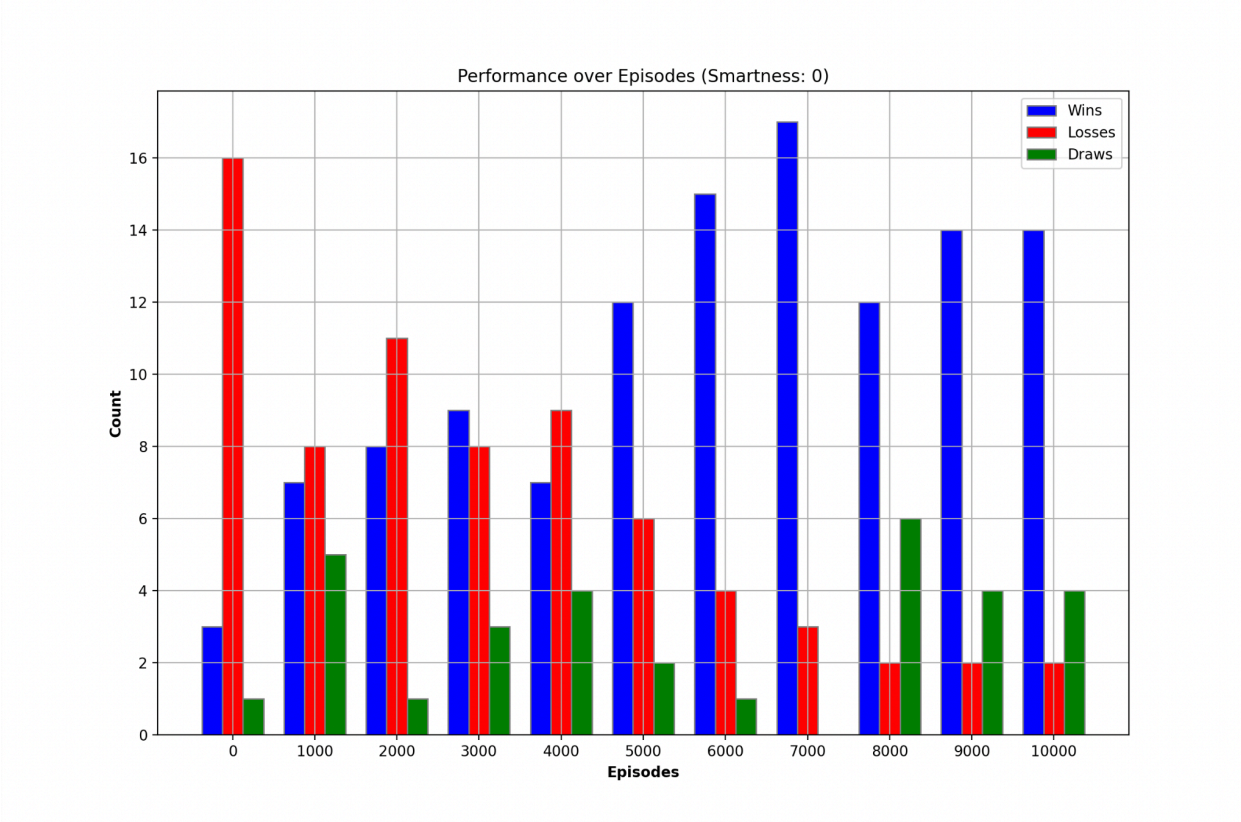


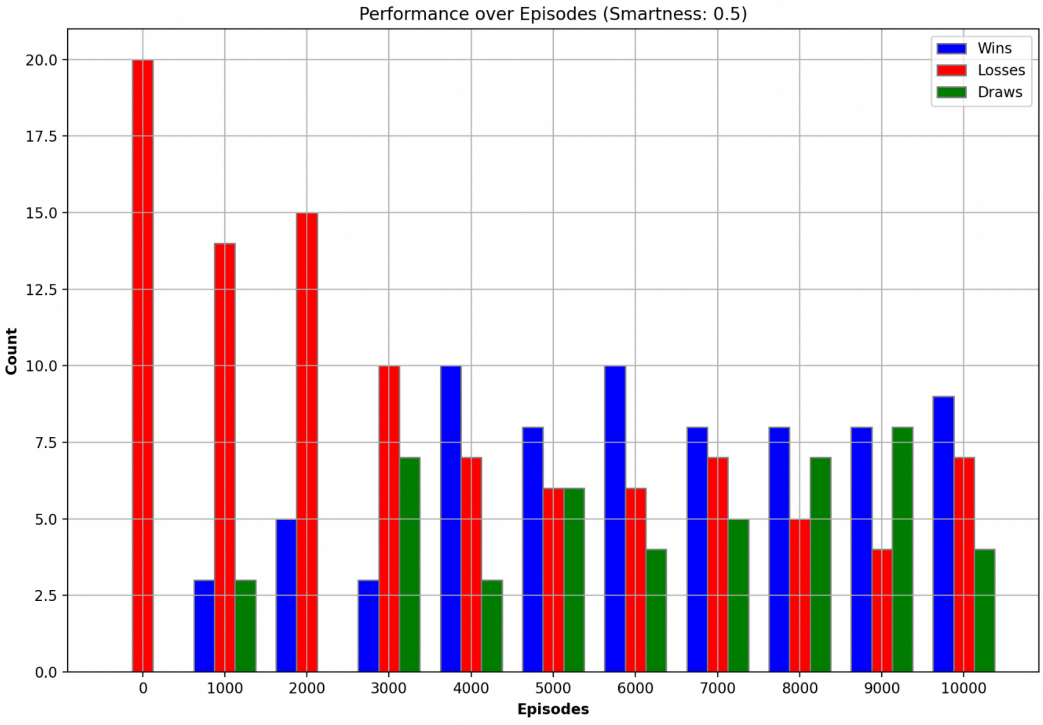
6. **The Results (1000 random seed values)**:
    - Smartness - 0 :
        - Wins : 476
        - Loss : 304
        - Draws : 220
    - Samrtness - 0.5:
        - Wins : 303
        - Loss : 477
        - Draws : 220
    - Smartness - 1:
        - Wins : 80
        - Loss : 677
        - Draws : 249

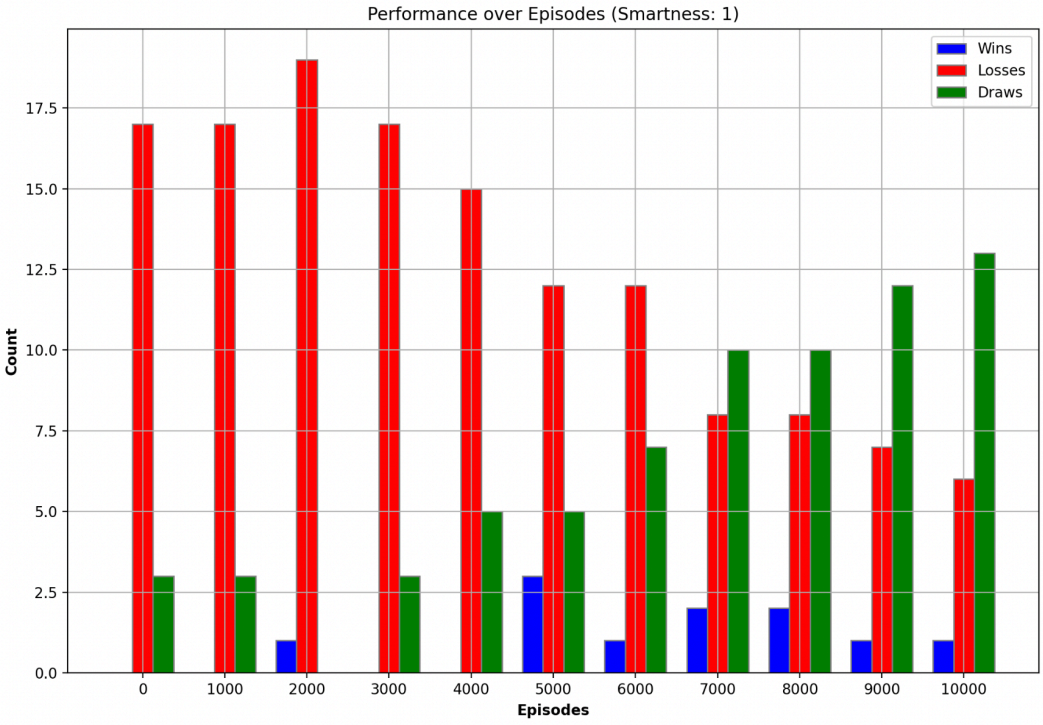
Model 2: Enhanced Exploration Strategy:

Model 2 was an improved implementation of the SQN framework, building upon the foundational principles of Model 1. The goal of this version was to enhance exploration during training while addressing potential limitations in learning against smarter opponents. Below are the specifics:

1. **Epsilon Strategy:**
    - The exploration rate started at 1.0 to facilitate extensive exploration, but it was adjusted more dynamically:
    - Epsilon decays linearly within blocks of 2000 episodes, but after each 2000-episode block, epsilon is increased by +0.4 to encourage further exploration because of continuous increasing value of smarntess
2. **Replay Buffer**: A buffer size of 10,000 was maintained to store diverse experiences for efficient training.
3. **Results over number of episode compared over 20 seed values:**:




Performance over Episodes (Smartness: 0.5)
Wins
Losses
Draws
Count
Episodes
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


Performance over Episodes (Smartness: 1)
Wins
Losses
Draws
Count
Episodes
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

4. **The Results (1000 random seed values)**:
    - Smartness - 0 :
        - **Wins**: 634
        - **Losses**: 88
        - **Draws**: 278
    - Samrtness - 0.5:
        - **Wins**: 376
        - **Losses**: 165
        - **Draws**: 459
    - Smartness - 1:
        - **Wins**: 67
        - **Losses**: 250
        - **Draws**: 683
5. **Key Observationsa and Explaination:**The change in performance between Model 1 and Model 2 can be attributed to the dynamic epsilon adjustment strategy introduced in Model 2. In Model 2, epsilon increases periodically after every 2000 episodes, encouraging exploration when the agent as smartness increases over time might otherwise over-exploit its learned strategies. This approach helps the agent discover better moves and avoid premature convergence to suboptimal behavior. Against random and moderately intelligent opponents, this increased exploration allows the agent to perform better, as it is more likely to avoid local optima and find optimal strategies. However, against the fully intelligent opponent (smartness = 1), the agent struggles because the increased exploration may prevent it from sufficiently honing its strategy, leading to more exploratory (and less optimal) actions that are less effective against a highly strategic adversary. This trade-off between exploration and exploitation explains the shift in performance across varying opponent smartness levels.

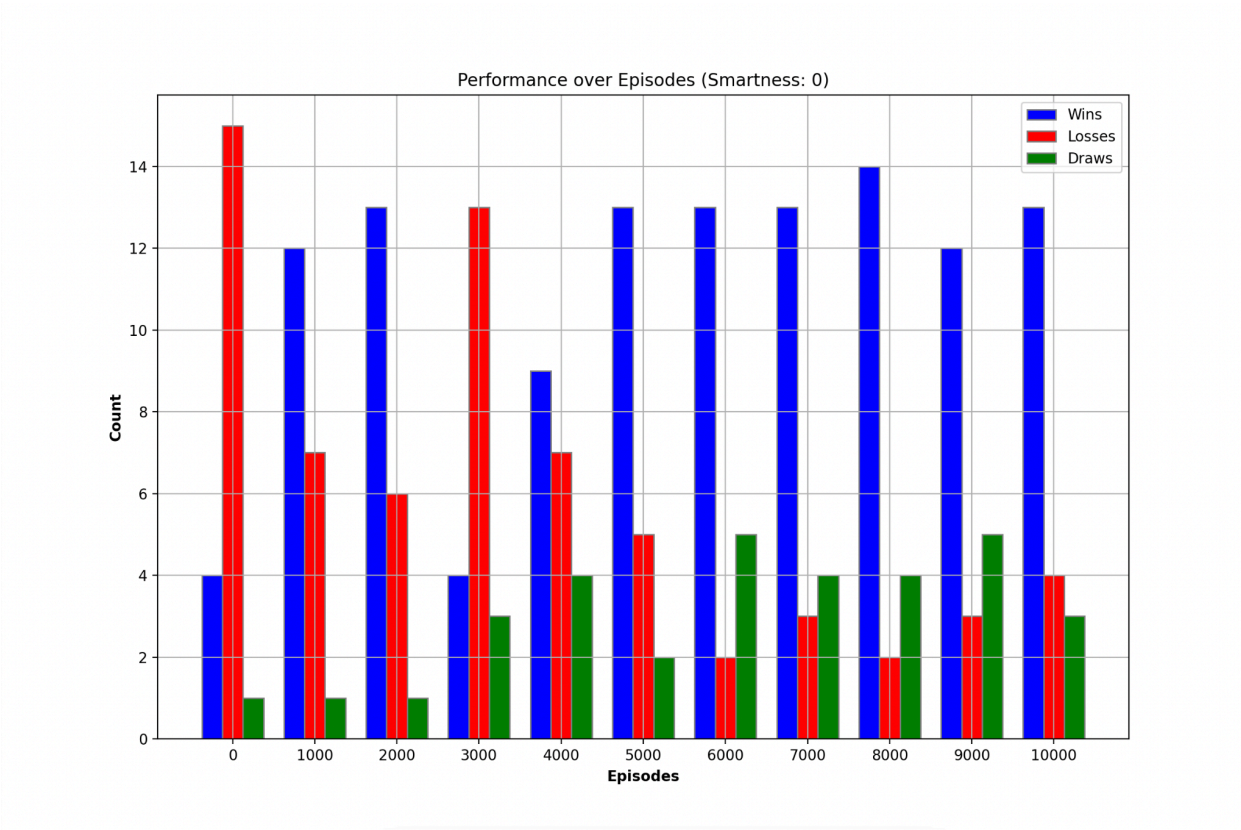
Model 2: Enhanced Exploration Strategy:

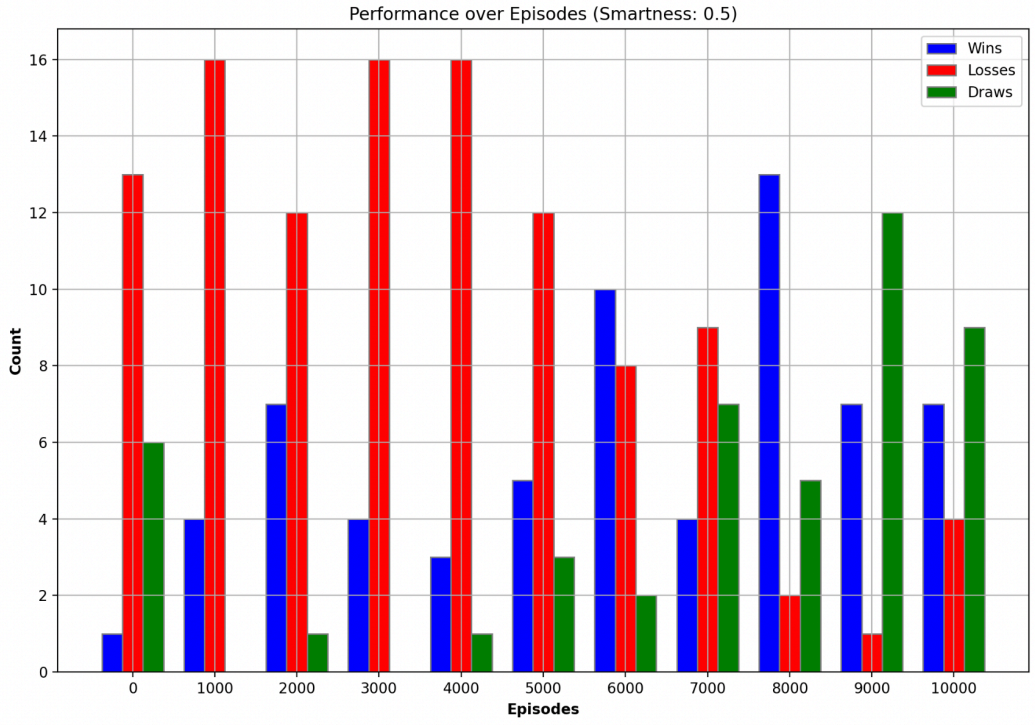
Model 2 was an improved implementation of the SQN framework, building upon the foundational principles of Model 1. The goal of this version was to enhance exploration during training while addressing potential limitations in learning against smarter opponents. Below are the specifics:

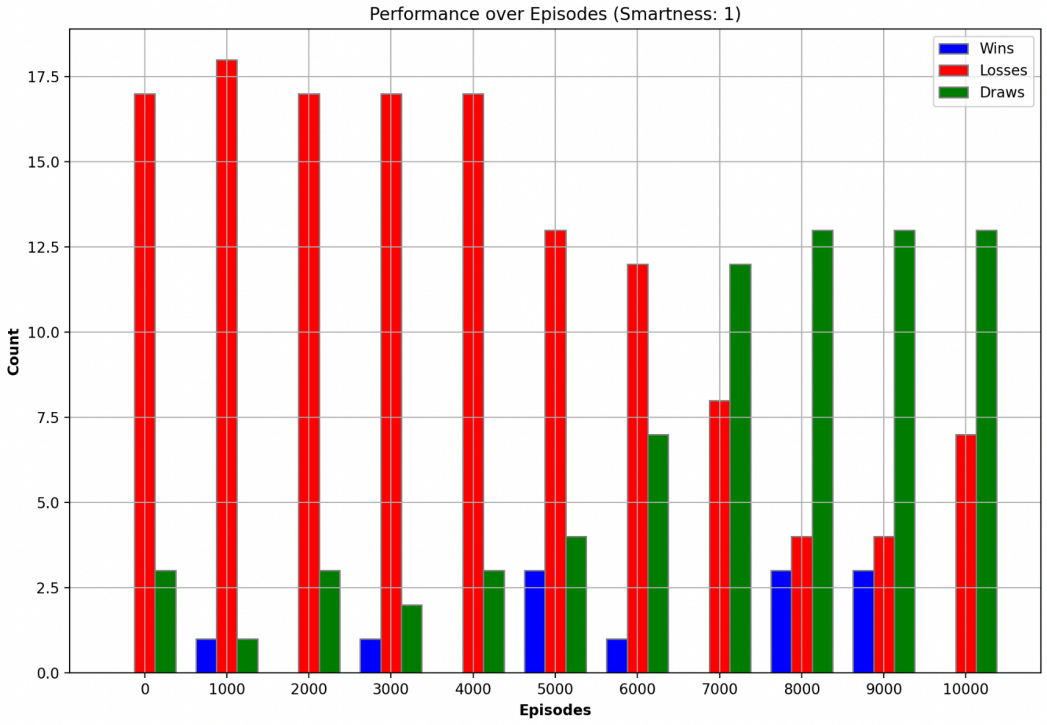
1. **Epsilon Strategy:**

- The exploration rate started at 1.0 to facilitate extensive exploration, but it was adjusted more dynamically:
- Epsilon decays linearly within blocks of 2000 episodes, but after each 2000-episode block, epsilon is increased by +0.4 to encourage further exploration because of continuous increasing value of smarntess

2. **Replay Buffer**: A buffer size of 10,000 was maintained to store diverse experiences for efficient training.
3. **Results over number of episode compared over 20 seed values:**:




Performance over Episodes (Smartness: 0.5)
Wins
Losses
Draws
Count
Episodes
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
0
2
4
6
8
10
12
14
16


Performance over Episodes (Smartness: 1)
Wins
Losses
Draws
Count
Episodes
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

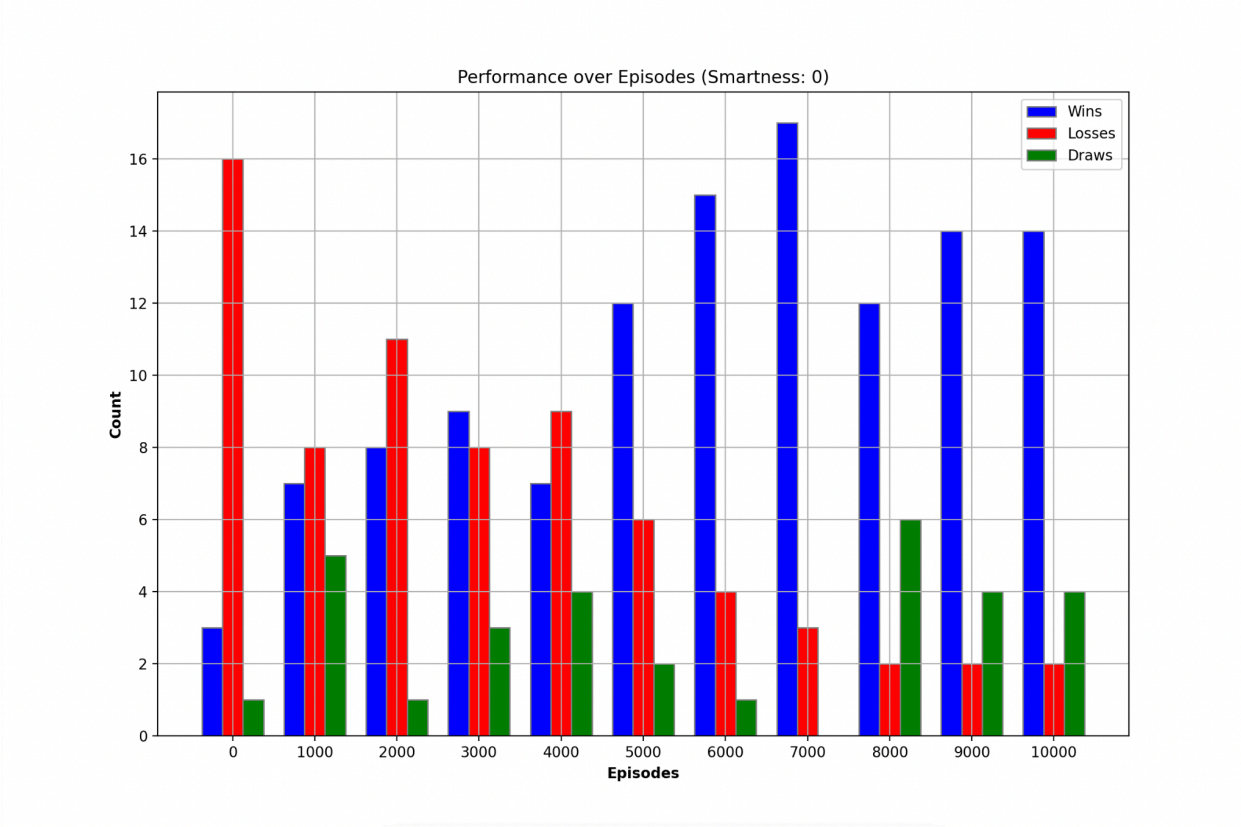
4. **The Results (1000 random seed values)**:
    - Smartness - 0 :
        - **Wins**: 634
        - **Losses**: 88
        - **Draws**: 278
    - Samrtness - 0.5:
        - **Wins**: 376
        - **Losses**: 165
        - **Draws**: 459
    - Smartness - 1:
        - **Wins**: 67
        - **Losses**: 250
        - **Draws**: 683
5. **Key Observationsa and Explaination:**The change in performance between Model 1 and Model 2 can be attributed to the dynamic epsilon adjustment strategy introduced in Model 2. In Model 2, epsilon increases periodically after every 2000 episodes, encouraging exploration when the agent as smartness increases over time might otherwise over-exploit its learned strategies. This approach helps the agent discover better moves and avoid premature convergence to suboptimal behavior. Against random and moderately intelligent opponents, this increased exploration allows the agent to perform better, as it is more likely to avoid local optima and find optimal strategies. However, against the fully intelligent opponent (smartness = 1), the agent struggles because the increased exploration may prevent it from sufficiently honing its strategy, leading to more exploratory (and less optimal) actions that are less effective against a highly strategic adversary. This trade-off between exploration and exploitation explains the shift in performance across varying opponent smartness levels.

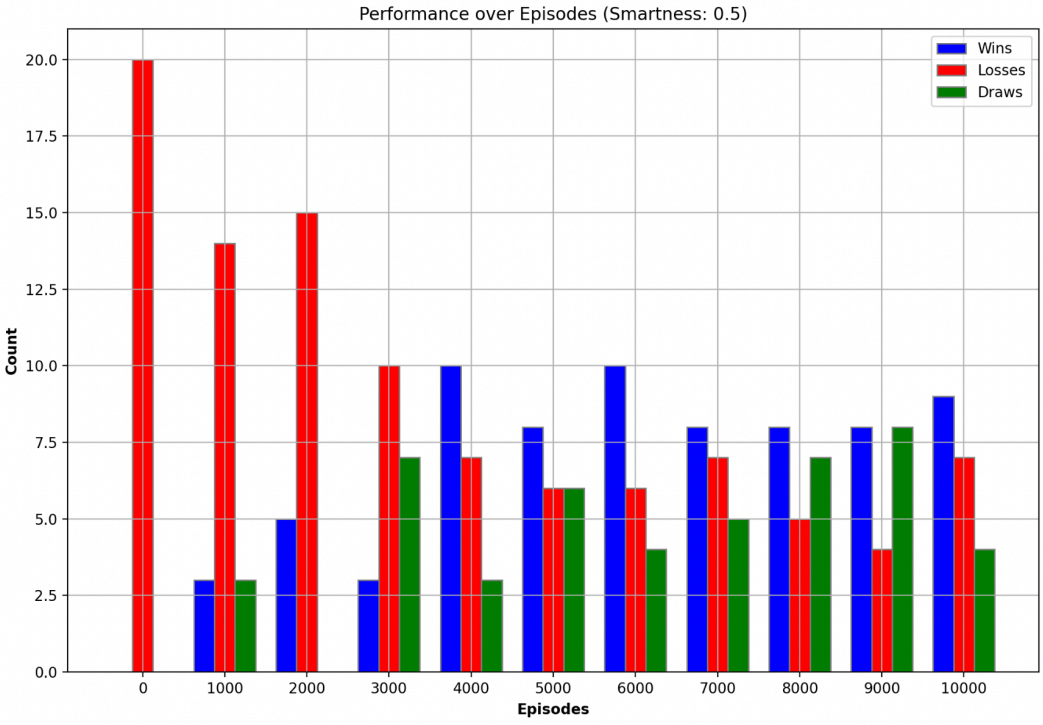
Model 3: Further Refinement of Exploration Strategy:

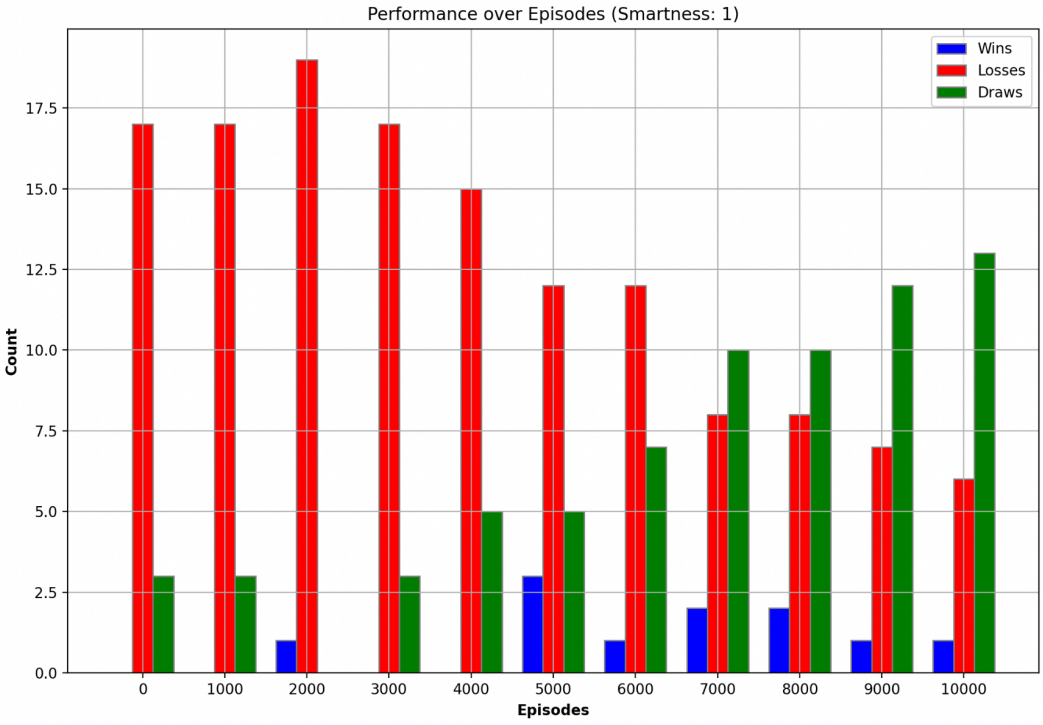
Model 3 represents an enhanced version of Model 2, where further refinements were made to the exploration strategy and training dynamics. The goal of Model 3 was to address the limitations of Model 2 in handling highly intelligent opponents (smartness = 1) and to introduce a

more sustainable balance between exploration and exploitation over time. Below are the key aspects of this model:

1. **Epsilon Strategy:** Similar to Model 2, Model 3 starts with an exploration rate of 1.0 (epsilon), but with a key change: the epsilon value is reset to 1.0 every 2000 episodes. This strategy was introduced to ensure the model adapts to increasingly difficult opponents as the smartness parameter is progressively increased over time. By resetting epsilon, the model maintains a balance between exploration and exploitation, allowing it to refine its strategies against more challenging scenarios
2. **Results over number of episode compared over 20 seed values:**:




Performance over Episodes (Smartness: 0.5)
Wins
Losses
Draws
Count
Episodes
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


Performance over Episodes (Smartness: 1)
Wins
Losses
Draws
Count
Episodes
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

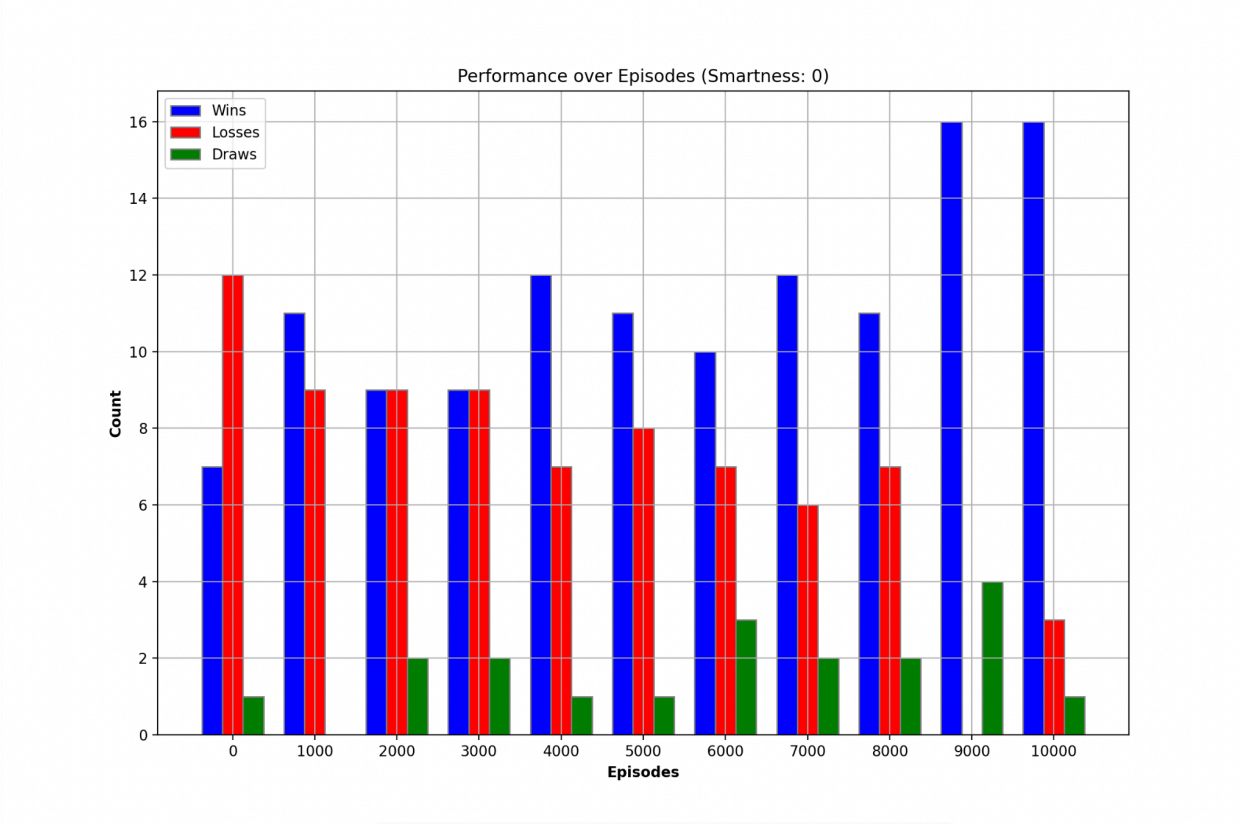
3. **The Results (1000 random seed values)**:
    - Smartness - 0 :
        - **Wins**: 646
        - **Losses**: 122
        - **Draws**: 232
    - Samrtness - 0.5:
        - **Wins**: 419
        - **Losses**: 190
        - **Draws**: 391
    - Smartness - 1:
        - **Wins**: 58
        - **Losses**: 264
        - **Draws**: 678
4. **Key Observationsa and Explaination:**The key difference between Model 2 and Model 3 lies in the handling of exploration and exploitation. Model 2's strategy of periodic epsilon increases encourages broader exploration as training progresses, which benefits the agent against moderately intelligent opponents but harms its ability to exploit learned strategies against highly intelligent opponents. In contrast, Model 3 resets epsilon every 2000 episodes, encouraging better controlled exploration and more efficient exploitation. While this approach improves performance against random and moderately intelligent opponents, it slightly reduces effectiveness against the most intelligent opponents, where the agent's exploration is too constrained.This demonstrates a trade-off: Model 3 strikes a better balance between exploration and exploitation, but in highly strategic scenarios, it may not be as effective as Model 2, which offers broader exploration opportunities for the agent to discover new strategies.

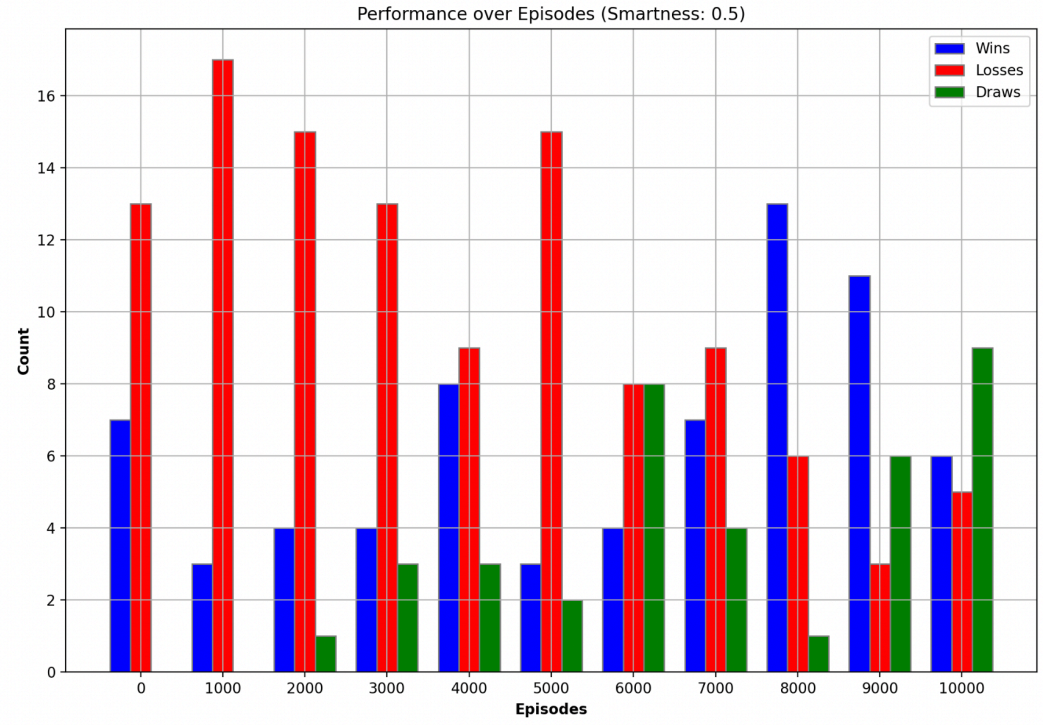
Model 4: Further Refinement of Exploration Strategy:

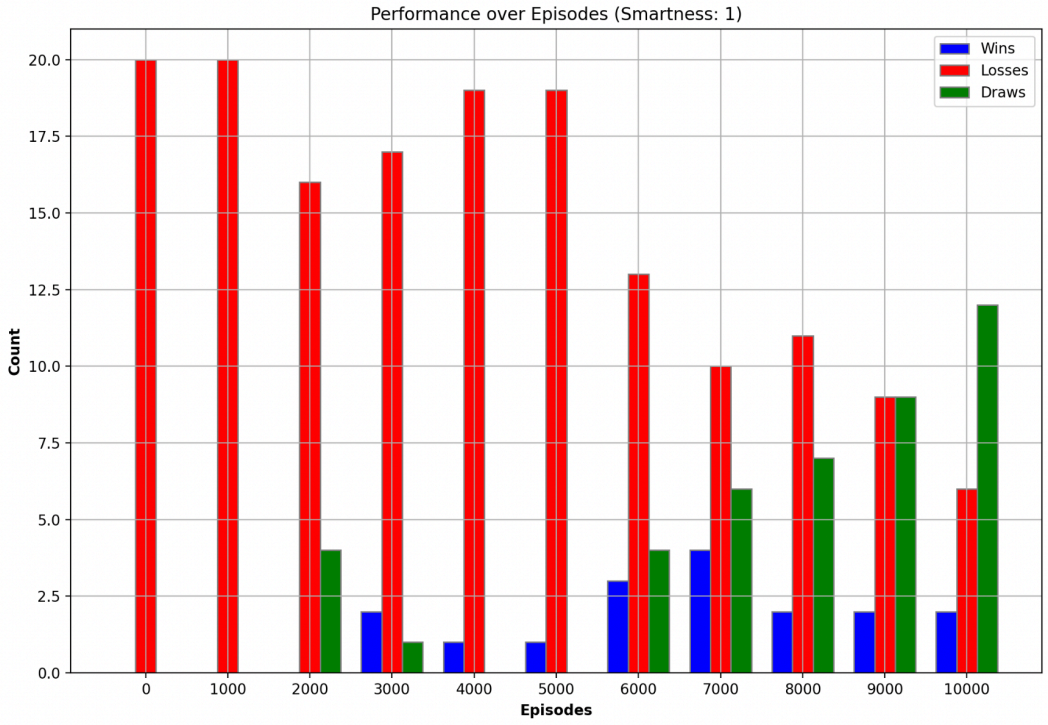
Model 4 represents a significant step forward compared to Model 3, particularly in terms of the reward strategy and exploration dynamics. The goal of Model 4 was to further refine the agent's learning approach, focusing on increasing its adaptability to intelligent opponents, while

introducing more nuanced reward values to reinforce better strategies during gameplay. Below are the key features of Model 4:

1. **Adjusted Reward Scheme:** Model 4 introduced a more pronounced reward system to differentiate between win, loss, and draw scenarios. Unlike Model 3, where the reward was +1 for a win and -1 for a loss, Model 4 assigns a higher positive reward for winning (+10) and a larger negative penalty for losing (-10). Draws remain neutral (0.0). This adjustment was made to emphasize the importance of winning and to penalize losses more severely, helping the agent to focus on learning winning strategies more aggressively.
2. **Results over number of episode compared over 20 seed values:**




Performance over Episodes (Smartness: 0.5)
Wins
Losses
Draws
Count
Episodes
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
0
2
4
6
8
10
12
14
16


Performance over Episodes (Smartness: 1)
Wins
Losses
Draws
Count
Episodes
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

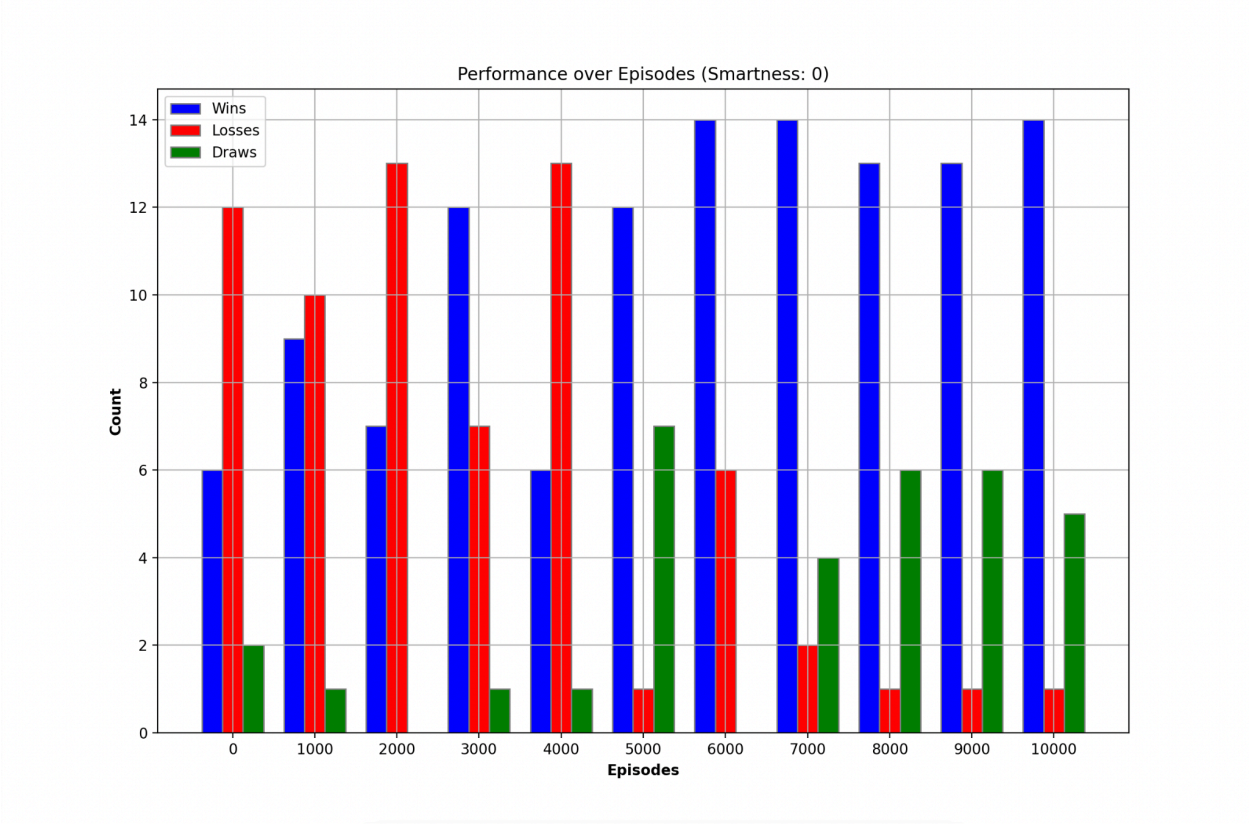
3. **The Results (1000 random seed values)**:
    - Smartness - 0 :
        - **Wins**: 700
        - **Losses**: 121
        - **Draws**: 220
    - Samrtness - 0.5:
        - **Wins**: 408
        - **Losses**: 236
        - **Draws**: 356
    - Smartness - 1:
        - **Wins**: 75
        - **Losses**: 362
        - **Draws**: 566
4. **Key Observationsa and Explaination:**Model 4 represents a refined approach that balances aggressive pursuit of wins with careful adaptation to smart opponents. However, like Model 3, it struggles in highly competitive scenarios due to a more restrained exploration strategy. The agent's performance against moderately intelligent and highly intelligent opponents reflects a strategic trade-off between exploration and exploitation, where the agent's caution under high-stakes conditions leads to fewer victories but a higher number of draws.

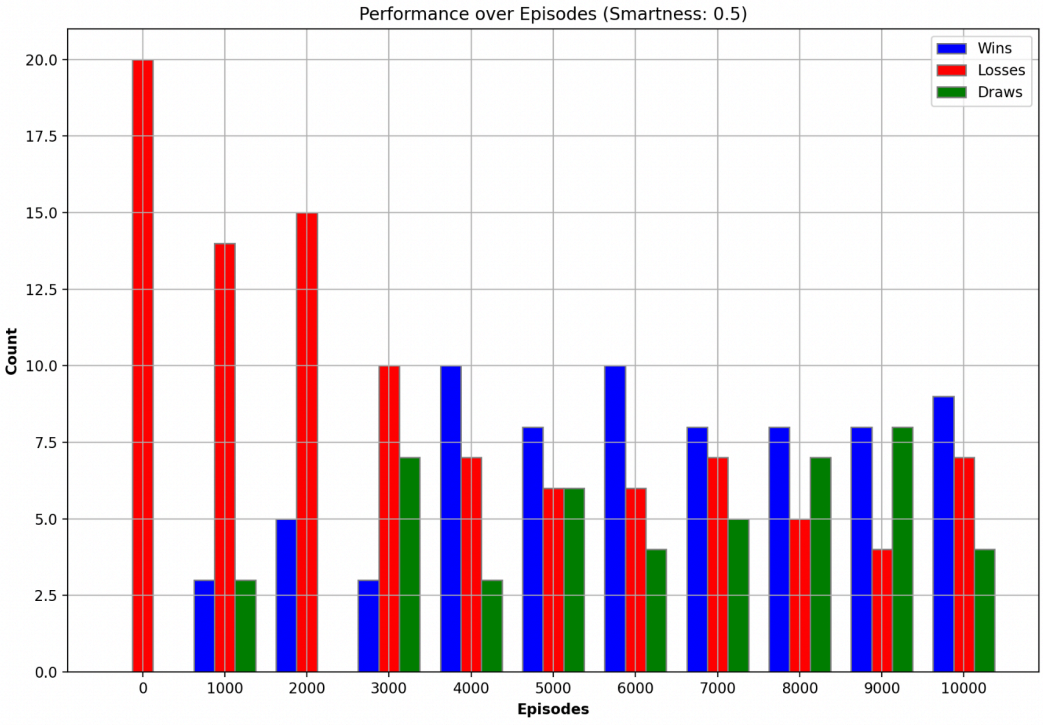
    In summary, Model 4 strikes a more aggressive stance in pursuit of victories while maintaining a structured approach to exploration, but faces challenges in adapting to highly intelligent opponents. The model's adjusted reward system is crucial for ensuring that the agent prioritizes winning strategies, even if it comes at the cost of reduced success against the smartest opponents.

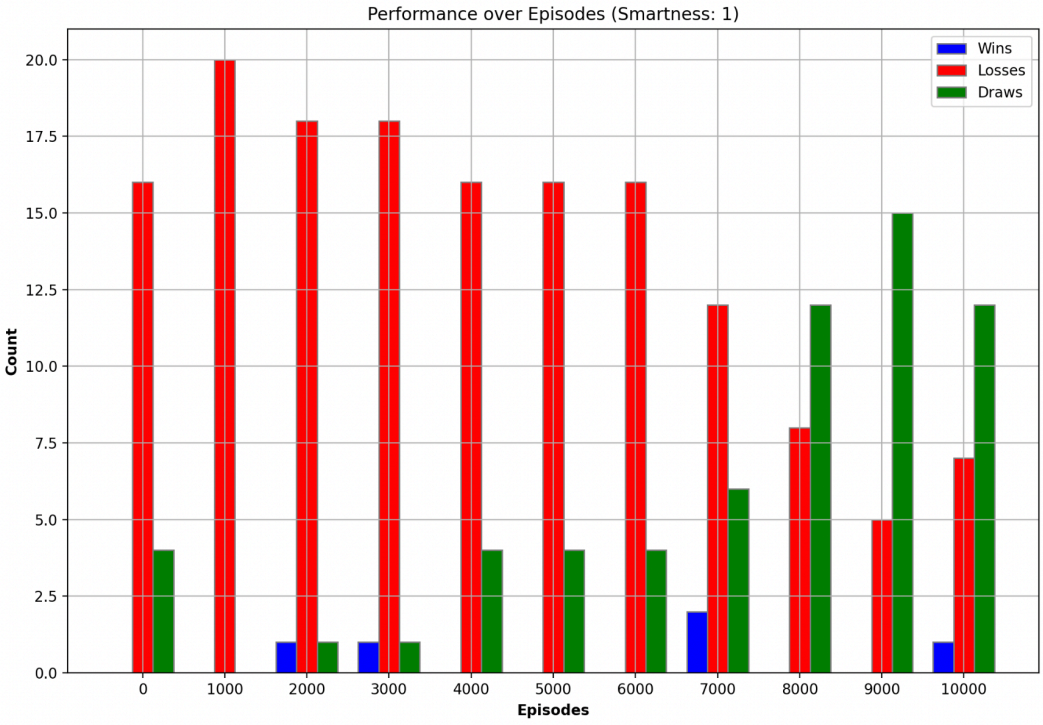
Model 5: Refinement of Exploration and Reward Strategies:

Model 5 represents another step forward in refining the agent's learning strategy compared to Model 4. While the core structure of the agent and its deep learning model remains largely the same, there are notable adjustments in both the reward scheme and exploration strategies that lead to different results when compared to Model 4. Below are the key features and observations for Model 5:

1. **Adjusted Reward Scheme:** In Model 5, the reward for a win has been reduced from +10 (in Model 4) to +1. The penalty for a loss is also softened from -10 to -1. A slight negative reward (-0.5) is applied in the case of a draw, unlike Model 4, where the draw was neutral (0.0). These adjustments aim to make the agent more sensitive to losses and encourage more nuanced exploration
2. **Results over number of episode compared over 20 seed values:**:




Performance over Episodes (Smartness: 0.5)
Wins
Losses
Draws
Count
Episodes
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


Performance over Episodes (Smartness: 1)
Wins
Losses
Draws
Count
Episodes
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

3. **The Results (1000 random seed values)**:
    - **Smartness = 0 (Beginner-level opponent)**:
        - Wins: 647
        - Losses: 147
        - Draws: 206
    - **Smartness = 0.5 (Moderate-level opponent)**:
        - Wins: 353
        - Losses: 235
        - Draws: 412
    - **Smartness = 1 (Advanced-level opponent)**:
        - Wins: 9
        - Losses: 354
        - Draws: 637

4. **Key Observationsa and Explaination:**Model 4 outperforms Model 5 when facing beginner-level and moderate-level opponents, primarily due to its more aggressive reward system that emphasizes quick victories. Against smartness level 0, Model 4 achieves 700 wins, while Model 5 has 647, showing its stronger offensive approach. Similarly, against moderate-level opponents (smartness = 0.5), Model 4 wins 408 games compared to Model 5's 353. This difference is due to Model 4's higher reward for wins and harsher penalties for losses, which drives the agent to adopt more aggressive strategies. Model 5, however, tends to result in more draws as it focuses on cautious exploration and avoiding losses, especially in scenarios where it might risk a loss for a win.However, when both models face highly intelligent opponents (smartness = 1), the performance diverges significantly. Model 4 still manages 75 wins, though it faces 362 losses and 566 draws, while Model 5 only wins 9 times with 354 losses and 637 draws. This indicates that while Model 4’s more aggressive tactics give it an edge in attempting to win, Model 5's defensive approach leads to many more draws but almost no wins. Model 5’s approach seems better at minimizing losses, but it sacrifices the offensive drive needed to secure victories against tougher opponents. In summary, Model 4 is more successful in securing wins, while Model 5 focuses on stability and avoiding losses, resulting in more draws and fewer victories, particularly when facing smarter opponents.

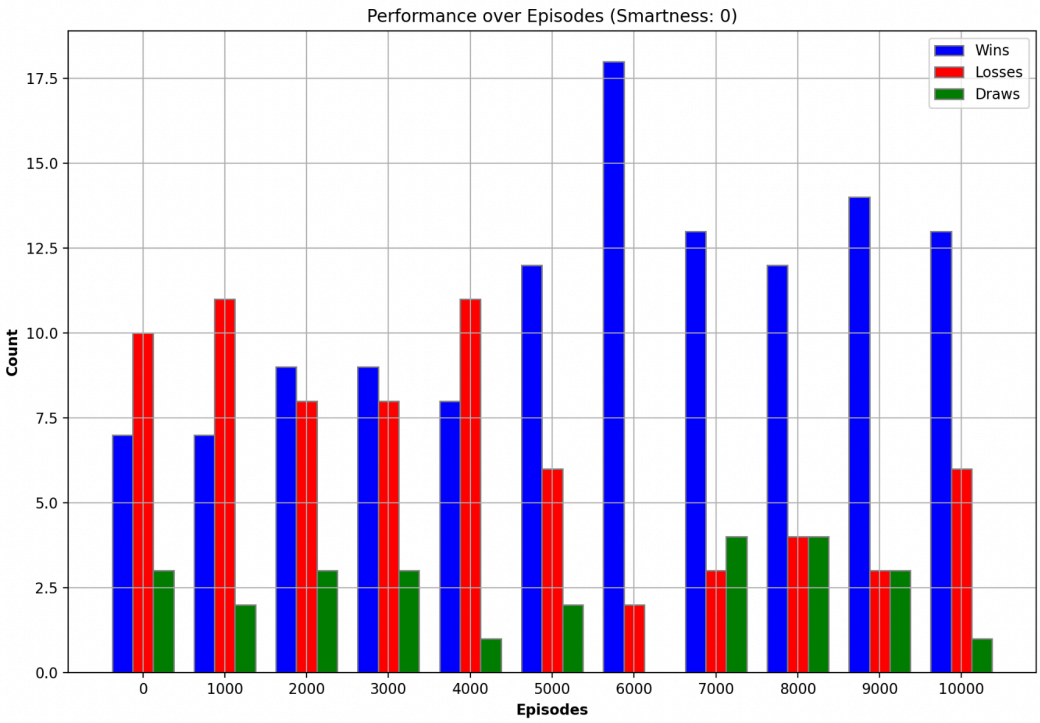
Model 6: Enhanced Exploration with Variable Draw Rewards:

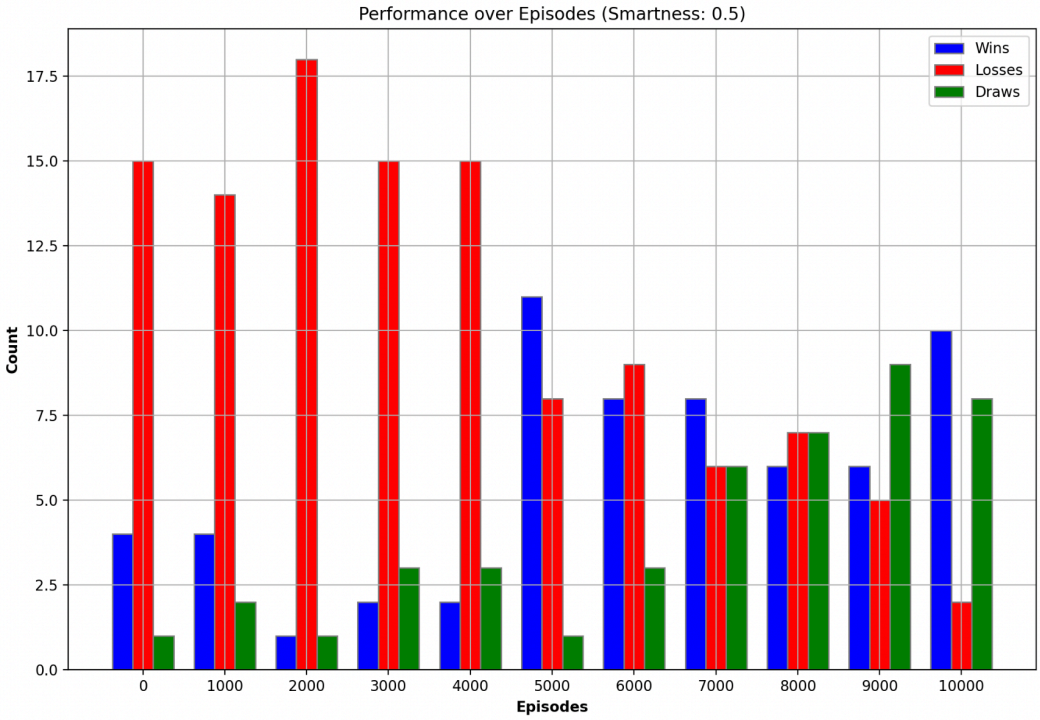
Model 6 introduces subtle modifications to the exploration and reward strategies compared to Model 5. The most notable change lies in the reward for draws, where Model 6 introduces a dynamic reward system based on the opponent's "smartness" level. Specifically, when playing against a less intelligent opponent (smartness = 0), the draw penalty is softer, which may encourage more attempts to win even in uncertain situations. This contrasts with Model 5's fixed penalty for draws (-0.5).

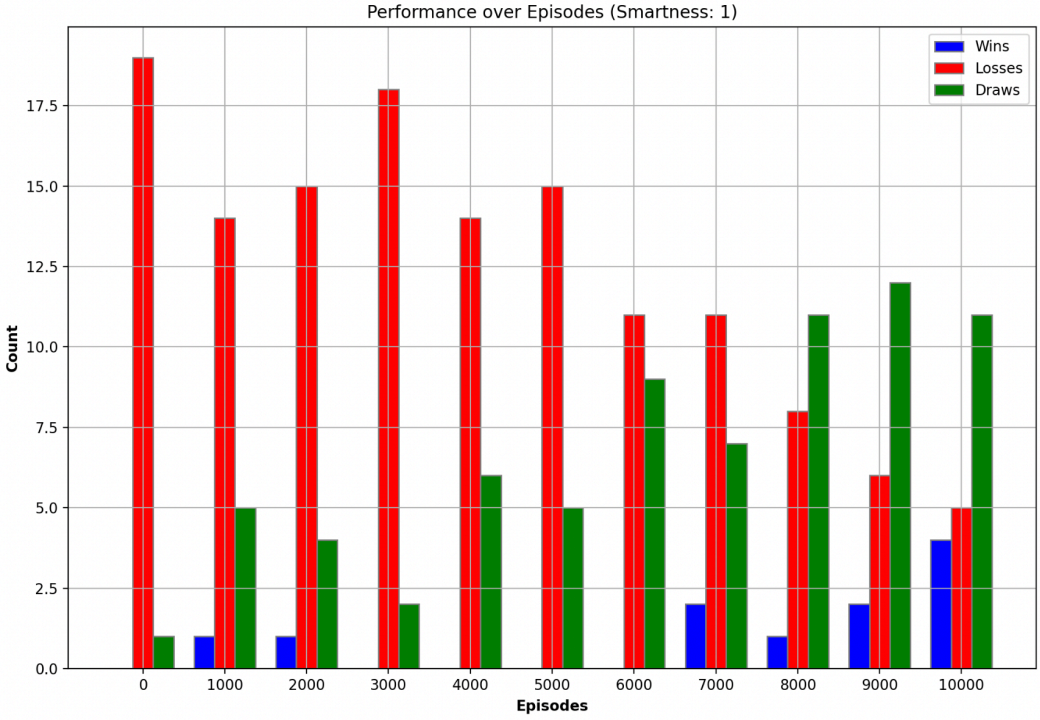
In addition to this, Model 6 maintains the key structural elements of Model 5: a moderate reward for wins (+1) and a mild penalty for losses (-1). However, the penalty for a draw becomes more adaptive, with a stronger penalty as the smartness of the opponent increases. This encourages Model 6 to aim for wins against more intelligent players, rather than settling for draws.:

1. **Draw Penalty Strategy:** Model 5 applies a fixed penalty of -0.5 for a draw, regardless of the opponent's smartness level. In contrast, Model 6 introduces a dynamic draw penalty that varies based on the opponent's smartness. Specifically, when facing a beginner-level opponent (smartness = 0), the penalty for a draw is -0.1. For a moderate-level opponent (smartness = 0.5), the penalty remains at -0.5. Similarly, for an advanced-level opponent (smartness = 1), the penalty is also -0.5. This dynamic approach in Model 6 aims to adapt the agent's strategy according to the difficulty of the opponent, encouraging more aggressive play against less intelligent opponents while maintaining caution against more challenging ones.
2. **Draw Penalty Strategy:** In Model 5, the epsilon (exploration rate) decays in a fixed manner over episodes, influencing the agent's decision-making process as it nears the end of training. Conversely, Model 6 employs a dynamic epsilon decay strategy that adjusts based on the opponent's smartness level. This approach allows the agent to explore more in easier games while balancing exploration and exploitation more effectively in harder games. By tailoring the exploration rate to the difficulty of the opponent, Model 6 aims to optimize learning and performance across varying levels of challenge

3. Results over number of episode compared over 20 seed values::




Performance over Episodes (Smartness: 0.5)
Wins
Losses
Draws
Count
Episodes
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


Performance over Episodes (Smartness: 1)
Wins
Losses
Draws
Count
Episodes
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

4. **The Results (1000 random seed values)**:
    - **Smartness = 0 (Beginner-level opponent)**:
        - Wins: 719
        - Losses: 124
        - Draws: 157
    - **Smartness = 0.5 (Moderate-level opponent)**:
        - Wins: 385
        - Losses: 227
        - Draws: 388
    - **Smartness = 1 (Advanced-level opponent)**:
        - Wins: 78
        - Losses: 314
        - Draws: 608

5. **Key Observationsa and Explaination :** Model 6 introduces several key changes that improve its performance compared to Model 5, including a dynamic draw penalty based on the opponent's smartness and a more adaptive exploration strategy. These adjustments allow Model 6 to balance exploration and exploitation more effectively, leading to better results. Specifically, against beginner-level opponents, Model 6 wins 72 more games and draws 49 fewer, while against moderate-level opponents, it wins 32 more games and loses 24 fewer. Against advanced-level opponents, Model 6 outperforms Model 5 significantly with 69 more wins and 40 fewer losses. Overall, these changes result in Model 6 showing improved stability, more wins, and fewer losses across all difficulty levels, especially when facing smarter opponents.

Other Methods and Difficulty Faced:

Model 6 established the baseline performance with a simple yet effective approach using a basic reward structure (+1 for wins, -1 for losses) and 1000-episode epsilon decay cycles. It demonstrated strong performance with 719 wins and only 124 losses against a basic opponent (smartness=0), and maintained reasonable effectiveness with 385 wins against a moderate

opponent (smartness=0.5). This model's straightforward approach proved most effective for learning fundamental Tic-tac-toe strategies.

Model 7 attempted to improve upon Model 6 by extending the epsilon decay cycle to 2000 episodes and implementing more aggressive negative rewards for draws. However, this resulted in decreased performance, achieving only 564 wins against a basic opponent and struggling more against moderate opponents with 346 wins. The increased penalty for draws may have made the agent overly cautious, leading to less effective gameplay strategies.

Model 8 introduced significant architectural changes with prioritized experience replay buffer and modified network structure. These changes led to a dramatic decline in performance, managing only 182 wins while suffering 750 losses against a basic opponent. This substantial drop suggests that the added complexity severely hindered the agent's ability to learn effective strategies, making it the poorest performing model in the series.

Model 9 attempted to recover performance by implementing dynamic reward scaling based on opponent difficulty. While it showed improvement over Model 8, achieving 573 wins against basic opponents, it still fell short of Model 6's performance. The dynamic reward system, while theoretically sound, didn't translate into better practical performance, particularly struggling against stronger opponents with only 303 wins at smartness=0.5.

Model 10 introduced several sophisticated features including target networks, smart move detection, and an increased batch size. Its performance was similar to Model 9, with 567 wins against basic opponents and 325 wins against moderate opponents. Despite the advanced features, it couldn't match Model 6's effectiveness, suggesting that the added complexity might have made the learning process less stable.

Model 11 emerged as the second-best performer in the series, incorporating progressive difficulty increases and reward scaling with opponent smartness. It achieved 616 wins against

basic opponents and 352 wins against moderate opponents. While this showed improvement over most other iterations, it still couldn't surpass Model 6's performance, further reinforcing that simpler approaches might be more effective for learning Tic-tac-toe strategies. The results across all models suggest that increasing complexity in the learning process may actually hinder performance in relatively simple game environments like Tic-tac-toe.

Comparing Result for all models:

Smartness: 0

Model	Wins	Losses	Draws
model1.h5	476	304	220
model2.h5	634	88	278
model3.h5	646	122	232
model4.h5	708	121	171
model5.h5	647	147	206
model6.h5	719	124	157
model7.h5	564	298	138
model8.h5	182	750	68
model9.h5	573	296	131
model10.h5	567	237	196
model11.h5	616	187	197

Smartness: 0.5

Model	Wins	Losses	Draws
model1.h5	303	477	220
model2.h5	376	165	459
model3.h5	419	190	391
model4.h5	408	236	356
model5.h5	353	235	412
model6.h5	385	227	388
model7.h5	346	412	242
model8.h5	107	836	57
model9.h5	303	549	148
model10.h5	325	433	242
model11.h5	352	379	269

Smartness: 1

Model	Wins	Losses	Draws
model1.h5	80	671	249
model2.h5	67	250	683
model3.h5	58	264	678
model4.h5	72	362	566
model5.h5	9	354	637
model6.h5	78	314	608
model7.h5	42	590	368
model8.h5	54	902	44
model9.h5	13	791	196
model10.h5	96	552	352
model11.h5	100	568	332

Retraining Model 6:

Model 6 emerged as the most robust implementation in the series, thanks to its thoughtful balance of exploration and exploitation. The dynamic epsilon decay and variable draw penalties were pivotal in adapting to opponents of varying difficulty levels. Against random opponents (smartness = 0), it achieved an impressive 719 wins, showcasing its ability to learn fundamental strategies effectively. Against moderately intelligent opponents (smartness = 0.5), the model secured 385 wins, reflecting its adaptability and strategic decision-making. When facing fully intelligent opponents (smartness = 1), Model 6 achieved 78 wins, the highest among all models, highlighting its ability to retain effective strategies even under challenging conditions. The use of dynamic rewards for draws, tailored to opponent difficulty, allowed Model 6 to optimize its gameplay by being aggressive against weaker opponents and cautious against smarter ones, striking a perfect balance between offense and defense.

Model 6’s performance highlighted several key improvements over other models. Against random opponents, it achieved 719 wins, demonstrating superior learning of basic strategies compared to Model 5's 647 wins. Its periodic epsilon resets enabled continued exploration, avoiding early convergence to suboptimal strategies. Against moderately intelligent opponents, Model 6 achieved 385 wins compared to Model 5's 353, showcasing its adaptability to intermediate gameplay challenges. Against fully intelligent opponents, Model 6’s 78 wins were the highest among all models, surpassing Model 4’s 75. Its dynamic draw penalties and balanced exploration allowed it to retain optimal strategies even in challenging scenarios.

Model 6 addressed potential training instabilities effectively through several mechanisms. Random sampling experiences from the replay buffer broke the correlation between sequential

data points, ensuring stable and unbiased Q-value updates. The use of a large replay buffer (size 10,000) allowed the model to store varied game scenarios, enhancing its ability to generalize and reducing the risk of overfitting. Additionally, the gradual increase in opponent smartness during training prevented the model from being overwhelmed early, fostering stable learning progress. These measures collectively minimized divergence risks and enhanced the training process's robustness

The reward structure in Model 6 was pivotal in achieving consistent performance. The balanced reward system (+1 for wins, -1 for losses) encouraged the model to aim for victories while avoiding overly aggressive strategies that might lead to failures. Dynamic draw penalties, which varied with opponent smartness (-0.1 for random, -0.5 for moderately/fully intelligent opponents), motivated the agent to aim for wins against weaker opponents while promoting cautious gameplay against stronger ones. Unlike Model 4’s aggressive reward scaling (+10/-10), Model 6’s moderate rewards reduced overfitting to overly aggressive tactics and improved overall adaptability. This balanced reward scheme allowed Model 6 to achieve consistent results across all difficulty levels.

Model 6 was chosen for retraining due to its consistent performance and adaptability across all opponent smartness levels, achieving the highest win rates among all models.

After further training, Model 6 showed improved performance, refining its ability to achieve higher accuracy and better adaptability in various scenarios. With the additional training, its win rate against a player with a smartness of 0 increased further, while its win rate against a player with a smartness of 0.5 also showed improvement, demonstrating the model’s enhanced strategic capabilities.

Comparison for Model6 after Multiple Training:

Smartness: 0

Episodes	Wins	Losses	Draws
10k episodes	719	124	157
30k episodes	811	41	148
50k episodes	779	49	172

Smartness: 0.5

Episodes	Wins	Losses	Draws
10k episodes	385	227	388
30k episodes	497	116	387
50k episodes	501	152	347

Smartness: 1

Episodes	Wins	Losses	Draws
10k episodes	78	314	608
30k episodes	77	235	688
50k episodes	70	335	595

Summary and Future Improvements:

This research demonstrated the successful implementation of Shallow Q-Networks (SQN) for Tic-Tac-Toe, with Model 6 emerging as the most effective implementation through its dynamic reward structure and balanced exploration strategy. The study revealed that simpler architectural approaches often outperformed more complex implementations, achieving up to 719 wins against beginner opponents while maintaining competitive performance against

advanced players. However, several areas for potential improvement remain. Future work could explore implementing prioritized experience replay to focus learning on more significant state-action pairs, and introducing curriculum learning by gradually increasing opponent difficulty during training. The integration of Monte Carlo Tree Search (MCTS) with SQN could enhance decision-making capabilities, particularly against skilled opponents. Technical enhancements could include optimizing computational efficiency, implementing early stopping mechanisms based on performance plateaus, and developing more sophisticated reward shaping techniques based on board position analysis. Additionally, investigating multi-task learning approaches could enable simultaneous learning from different opponent skill levels, potentially improving the agent's adaptability. The study also suggests that exploring self-play training could reduce dependency on predetermined opponent strategies and potentially lead to the discovery of more robust game-playing strategies. These improvements could address the current limitations in handling skilled opponents while maintaining the successful aspects of the current implementation.