Reinforcement Learning for Super Tic-Tac-Toe: DQN Implementation and Evaluation

I. Introduction

This report presents the development and training of a Deep Q-Network (DQN) agent to play a variant of tic-tac-toe called Super Tic-Tac-Toe. The game is played on a cross-shaped board consisting of five 4x4 squares arranged in a 12x12 grid. The objective is to align four in a row or column, or five diagonally. The gameplay introduces stochasticity: a player's chosen move is accepted with 50% probability; otherwise, the move is randomly assigned to one of the eight adjacent squares with a 1/16 probability each or forfeited if no valid adjacent square exists.

II. Environment and Model Design

The environment is implemented using OpenAI Gym's interface, where the board state is represented as a 12x12 numpy array. The five 4x4 playable regions are embedded within this grid, with invalid positions marked as -1. Actions are discretized into 80 possible moves corresponding to the 5 regions with 16 positions each.

The DQN model consists of a fully connected neural network with two hidden layers of 256 and 128 units respectively, using ReLU activations. The input is the flattened board state, and the output layer predicts Q-values for all 80 possible actions.

III. Training Procedure

The agent is trained over 10000 episodes using an epsilon-greedy policy for exploration. Initially, epsilon is set to 1.0 and decays multiplicatively by a factor of 0.995 per episode until it reaches a minimum threshold of 0.01. At each decision

point, the agent either selects a random action (exploration) or chooses the action that maximizes the predicted Q-value (exploitation).

To promote sample efficiency and stability, experience replay is employed with a fixed buffer size of 1000 transitions. Once the buffer contains at least 64 samples, the agent begins training by randomly sampling mini-batches to perform gradient descent. The loss is computed as the mean squared error between the predicted Q-values for the taken actions and the target Q-values, which are estimated using a separate target network. The target network is synchronized with the main Q-network every 10 episodes to stabilize learning.

In addition, a **restart strategy inspired by Gagliolo and Schmidhuber (2007)** is employed to mitigate the risk of prolonged unproductive exploration phases caused by the stochastic nature of the learning process. Specifically, the training process is **restarted every 1000 episodes**, resetting the agent's weights and experience buffer. This restart schedule acts as a fixed cutoff strategy, which can help avoid heavy-tailed run-time behaviors and accelerate convergence. This approach is motivated by findings in restart strategy literature, which show that even coarse knowledge of run-time distributions can lead to near-optimal performance when restarts are applied appropriately.

IV. Key Implementation Details

- Action selection: The Q-values for legal actions are filtered before choosing the maximum, ensuring invalid moves are excluded.
- Move stochasticity: The environment's `step` method incorporates the 50% acceptance chance and random adjacent moves with forfeiture.
- State normalization: After each move, the board state is normalized to the current player's perspective by swapping player markers, facilitating consistent learning.
- Loss calculation: The loss uses `targets.detach()` to prevent gradients flowing into the target network.
- Training stability: Target network updates and epsilon decay contribute to stable convergence.

V. Evaluation

The trained DQN agent was rigorously evaluated against a random player over 1000 games to assess its performance. Under the current training configuration, the agent achieved a remarkable win rate of 86.2%, with the random player winning only 13.8% of the games and no draws observed. This result clearly demonstrates the agent's ability to learn effective strategies and outperform a baseline random policy despite the inherent stochasticity of the environment.

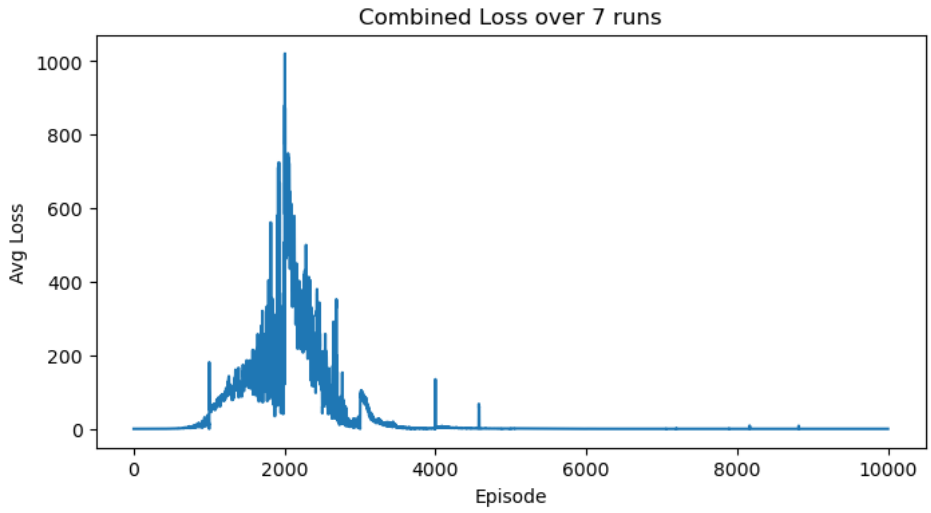
To further analyze the training dynamics and the impact of the exploration decay rate, we present two key plots:

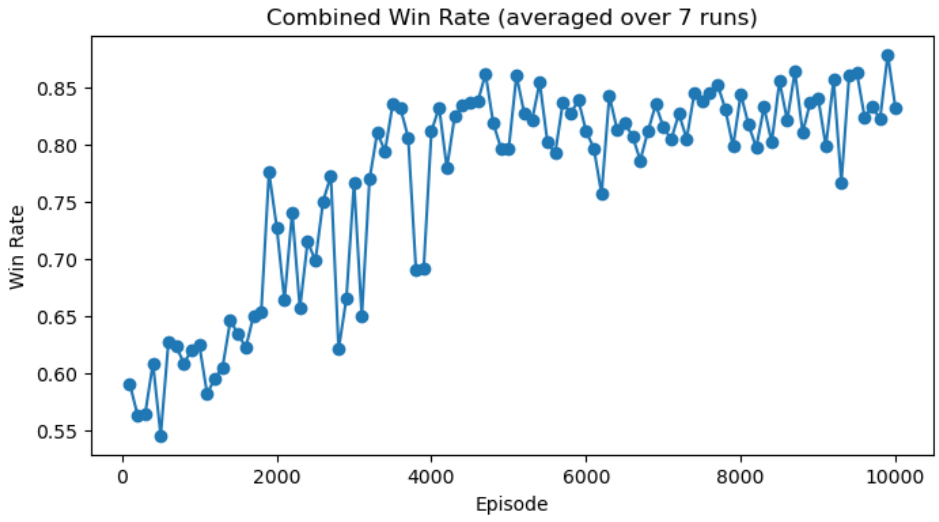
- **Epoch vs. Loss:** This plot illustrates the evolution of the training loss over episodes for different epsilon decay rates, offering insights into the convergence dynamics of the learning process.

  At the beginning of training, the agent explores a wide range of states due to high epsilon values, leading to updates of the value function based on sparse or suboptimal experiences. This imbalance often causes an initial rise in loss. As training progresses and the agent gathers more meaningful experience, the value function begins to stabilize. Consequently, the loss decreases rapidly, reflecting more accurate predictions and better state-action evaluations, eventually reaching convergence.

- **Epoch vs. Win Rate:**
  This plot presents the progression of the agent's win rate against a random baseline opponent across epochs, evaluated under different epsilon decay schedules. It demonstrates how the trade-off between exploration and exploitation influences both learning speed and overall performance.

  As training continues, the agent becomes increasingly proficient at making optimal decisions, resulting in a steady improvement in win rate. Despite some fluctuations due to the inherent randomness in gameplay, the win rate eventually plateaus, indicating the agent has reached a near-optimal strategy under the given conditions.





VI. Conclusion

This project successfully demonstrated the application of Deep Q-Networks to a challenging, stochastic variant of tic-tac-toe. The agent not only learned to navigate the probabilistic move acceptance mechanism but also developed robust strategies to consistently defeat a random opponent. The evaluation metrics and training curves confirm the stability and efficacy of the approach.

Future work could involve experimenting with more advanced neural architectures, incorporating prioritized experience replay to accelerate learning, or leveraging TensorFlow Agents for enhanced scalability and performance. Such improvements may further boost the agent's capabilities and offer additional academic credit opportunities.

`

Citation:
Gagliolo, M., & Schmidhuber, J. (2007). *Learning Restart Strategies*. In Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07), pp. 792–797.