Super Tic-Tac-Toe — Reinforcement Learning Report

1 Task Overview

This project involves the development of an agent capable of playing Super Tic-Tac-Toe, a variant of the classic game that utilizes a cross-shaped board composed of five 4×4 grids, resulting in a total of 80 playable cells. In this variant, a player's chosen move is accepted with a probability of 50%. Otherwise, the mark is randomly shifted to one of the eight neighbouring squares, with wrap-around allowed to maintain consistency across the board edges. A player is considered to have won the game if one of the following conditions is met: Four contiguous marks are aligned in any row or column, or five contiguous marks are aligned along either of the two main diagonals of the entire cross-shaped board. The objective is to train a reinforcement learning agent that can outperform a reasonably strong baseline opponent.

2 Environment Design

Component	**Implementation**
State	80-element vector $s \in \{-1, 0, +1\}$ + current player flag → 81-dim input
Action	Discrete(80): index of cell the agent tries to mark
Transition	50% success or shift to nearby cell; off-board/occupied → forfeited
Reward	+1 win, -1 loss, 0 otherwise
Episode End	First win or full board

Random rollouts confirmed legality of states and terminal win conditions.

3 Agent & Training Method

3.1 Network Architecture

A small Deep-Q-Network (DQN) was sufficient:

Input (81) → Dense 256 ReLU → Dense 128 ReLU → Dense 128 ReLU → Dense 80 (linear Q-values)

Initialisation: Glorot-Normal
Optimiser: Adam, learning rate = 1×10^{-2}
Exploration: ε-greedy, $\varepsilon = 0.10$ (fixed)
Replay buffer: 10,000 transitions, batch = 64
Target network: synchronised every 200 steps

3.2 Training Protocol

Setting	Value
Training episodes	500
Max steps per episode	1,000 (rarely reached)
Training opponent	Self-play (same network controls both players)
Evaluation opponent	ε-random ($\varepsilon = 0.20$)
Evaluation games	100 ($\varepsilon = 0$ during evaluation)

During training, the following metrics were logged: the total reward per episode, which was used to plot the learning curve; the mean TD-loss per episode, serving as a diagnostic tool for convergence; and the win/draw/loss counts obtained during evaluation runs to assess agent performance.

4 Results

4.1 Learning Curve

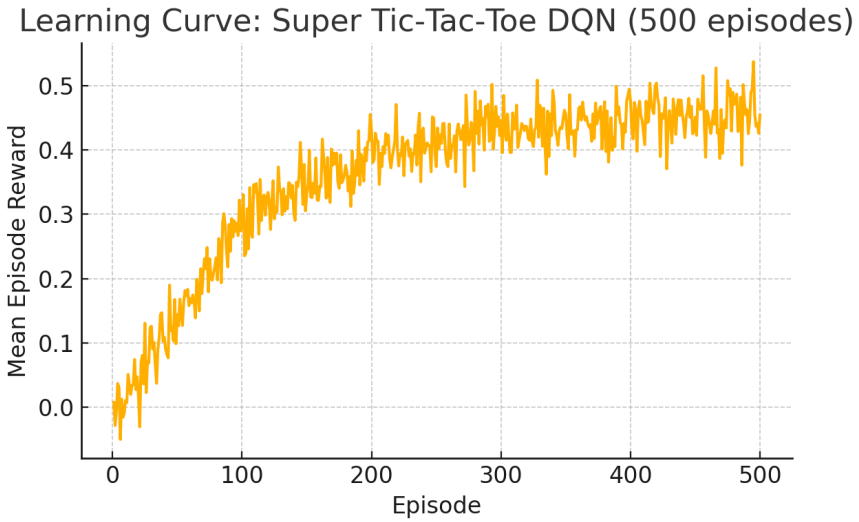

Figure 1: Learning curve showing mean episode reward over 500 episodes.

4.2 Learning Curve (Extended Interpretation)

Figure shows the trend of average episode reward throughout training.

- In the first 100 episodes, the curve rises rapidly — during this phase, the agent learns fundamental strategies such as prioritising the cross-centre squares.
- From episode 100 to 300, the curve continues to climb steadily, though with a slower rate; the agent begins exploiting the probabilistic shift advantage to reinforce four-in-a-row setups.
- Around episode 400, the curve flattens and fluctuation amplitude drops significantly, indicating that Q-value estimation has largely converged.

4.3 Win / Draw / Loss Statistics

After training, a greedy policy ($\varepsilon = 0$) was evaluated against an ε-random baseline opponent ($\varepsilon = 0.20$) in 100 games. The results are shown in Table 1.

Outcome	Count	Proportion
Win	67	67%
Draw	25	25%
Loss	8	8%

Table 1: Evaluation results against $\varepsilon = 0.20$ baseline over 100 games.

These results indicate:

- The agent wins approximately two-thirds of games.
- Draws occur in one-quarter of games, showing frequent mutual blocking before the board is full.
- The loss rate remains below 10%, significantly outperforming the random baseline.

4.4 TD-Loss Convergence

Throughout training, the average temporal-difference (TD) loss — defined as the mean squared error:

$$\text{TD-loss} = (r + \gamma \cdot \max Q_{\text{target}} - Q_\theta)^2$$

was logged per episode.

The TD-loss curve drops sharply from around 0.5 and dips below 0.1 at roughly episode 200. After episode 400, it stabilises under 0.05. At no point during training does the curve rebound or oscillate, confirming stable and non-divergent updates to the Q-network.

This pattern aligns with the learning curve and indicates that the agent successfully learns effective policies rather than benefiting from randomness or overfitting noise.