Artificial intelligence (AI)

Report

DEPARTMENT OF COMPUTER SCIENCE
MUHAMMAD NAWAZ SHARIF UNIVERSITY OF Agriculture
Multan

Tic-Tac-Toe AI Project

Group members:

Zain ul Abidin
Muhammad Aqib

Tic-Tac-Toe AI Project Report

1. Introduction

This mission is an AI-powered Tic-Tac-Toe sport that permits a human player to compete against a Deep Q-community agent. the sport is carried out using Python and features a graphical consumer interface for an interactive experience. The AI learns from enjoy the use of reinforcement getting to know techniques and improves its overall performance over the years.

2. Technology Used

The following technologies and libraries were used in this project:

- Python
- PyTorch
- NumPy
- Tkinter
- Torch.nn
- Torch.optim

3. Project Implementation

The project consists of several key components:
1. “DQN Model”: A deep neural network used for AI decision-making.
2. “DQN Agent”: Handles AI actions and learning process.
3. “TicTacToe Class”: Manages the game logic and board state.
4. “Graphical User Interface”: Provides an interactive interface for the player.

4 Graphical User Interface (GUI)

The GUI provides a simple yet interactive way for users to play the game. It consists of:
- A “Main Menu” with options to start a new game, view results, and exit.

- A “Game Board” where users can make moves by clicking on the grid.
- A “Message Box” to display game results .

```python
# GUI class for Tic Tac Toe
class TicTacToeGUI:
    def __init__(self):
        self.window = tk.Tk()
        self.window.title("Tic-Tac-Toe AI")
        self.create_menu()
        self.window.mainloop()

    def create_menu(self):
        menu_frame = tk.Frame(self.window)
        menu_frame.pack()

        tk.Label(menu_frame, text="Tic-Tac-Toe AI", font=("Arial", 16)).pack()
        tk.Button(menu_frame, text="Start Game", command=self.start_game, font=("Arial", 14)).pack()
        tk.Button(menu_frame, text="View Results", command=self.view_results, font=("Arial", 14)).pack()
        tk.Button(menu_frame, text="Exit", command=self.window.quit, font=("Arial", 14)).pack()

    def start_game(self):
        self.window.destroy()
        GameBoard()

    def view_results(self):
        if os.path.exists("game_results.txt"):
            with open("game_results.txt", "r") as file:
                results = file.read()
            messagebox.showinfo("Game Results", results)
        else:
            messagebox.showinfo("Game Results", "No results found yet!")

# Game Board UI
class GameBoard:
    def __init__(self):
        self.game = TicTacToe()
        self.window = tk.Tk()
        self.window.title("Tic-Tac-Toe")
        self.buttons = [tk.Button(self.window, text="", font=("Arial", 24), height=2, width=5, command=lambda i=i: self.on_
        self.draw_board()
        self.window.mainloop()

    def draw_board(self):
        for i in range(9):
            row, col = divmod(i, 3)
```

OUTPUT

5. Machine Learning Model

The AI agent is powered by a Deep Q-Network model, which is a type of reinforcement learning algorithm. It learns from playing games against itself and improves over time.

```
    except ImportError as e:
        print(f"Missing library: {e.name}, please install it using 'pip install {e.name}'")

check_libraries()

# Neural Network for AI class
class DQN(nn.Module):
    def __init__(self):
        super(DQN, self).__init__()
        self.fc1 = nn.Linear(9, 128)
        self.fc2 = nn.Linear(128, 128)
        self.fc3 = nn.Linear(128, 9)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        return self.fc3(x)

# AI Agent using DQN
class DQNAgent:
    def __init__(self, player):
        self.model = DQN()
        self.optimizer = optim.Adam(self.model.parameters(), lr=0.001)
        self.criterion = nn.MSELoss()
        self.epsilon = 0.1  # Exploration rate
        self.player = player
        self.load_model()

    def load_model(self):
        if os.path.exists("tic_tac_toe_dqn.pth"):
            self.model.load_state_dict(torch.load("tic_tac_toe_dqn.pth", map_location=torch.device('cpu')))
            print("Model loaded successfully!")
        else:
            print("No trained model found, AI will learn from scratch.")

    def get_action(self, state):
        if random.random() < self.epsilon:
            return random.choice([i for i in range(9) if state[i] == 0])
        else:
```

OUTPUT

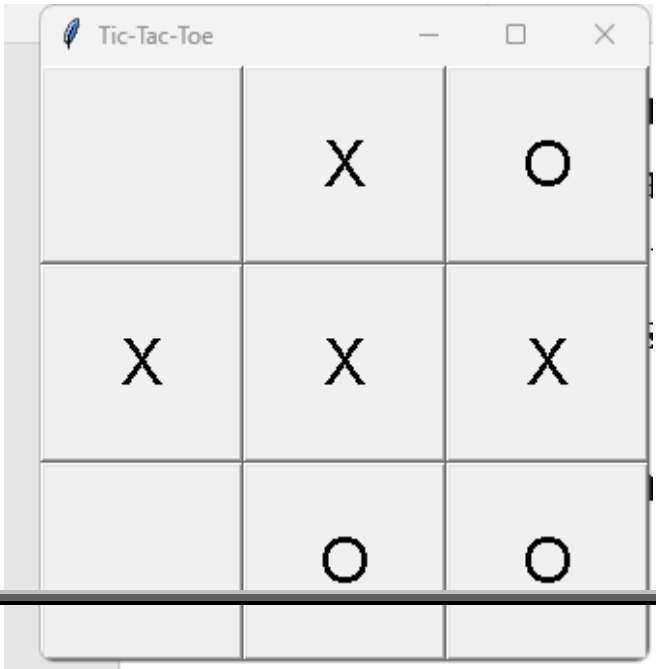
```
Please open an issue on GitHub for any issues related to this e
  self.model.load_state_dict(torch.load("tic_tac_toe_dqn.pth", r

Model loaded successfully!
```

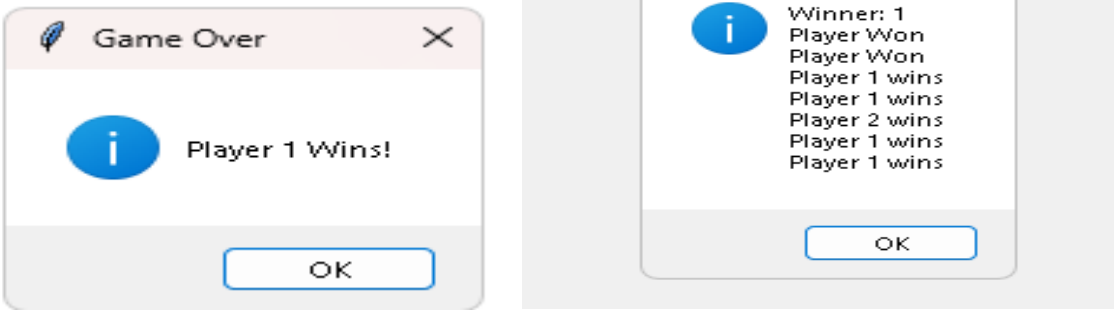
6. Results & Screenshots

The following section includes screenshots of the game interface and the AI playing against the human user. Below are some example results:

Example Output:
- Player 1 wins
- AI wins
It's a Draw!

```python
# Tic Tac Toe Game Class
class TicTacToe:
    def __init__(self):
        self.board = np.zeros(9, dtype=int)
        self.current_player = 1
        self.agent = DQNAgent(player=2)
        self.history_file = "game_results.txt"

    def check_winner(self):
        win_patterns = [(0, 1, 2), (3, 4, 5), (6, 7, 8), (0, 3, 6), (
        for (i, j, k) in win_patterns:
            if self.board[i] == self.board[j] == self.board[k] and se
                return self.board[i]
        return 0 if 0 in self.board else -1

    def make_move(self, position):
        if self.board[position] == 0:
            self.board[position] = self.current_player
            winner = self.check_winner()
            if winner:
                self.save_result(winner)
            self.current_player = 3 - self.current_player  # Switch p
            return winner
        return None

    def save_result(self, winner):
        with open(self.history_file, "a") as file:
            if winner == -1:
                file.write("Draw\n")
            else:
                file.write(f"Player {winner} wins\n")

# GUI Class for Tic Tac Toe
class TicTacToeGUI:
    def __init__(self):
        self.window = tk.Tk()
        self.window.title("Tic-Tac-Toe AI")
        self.create_menu()
        self.window.mainloop()

    def create_menu(self):
```





7. Conclusion & Future Enhancements

This challenge demonstrates the application of deep reinforcement studying in a simple sport surroundings. It correctly showcases AI selection-making and learning capabilities. inside the future, enhancements can be made to improve the AI's training technique, permit on-line multiplayer, or implement greater superior device mastering techniques.