Project Title: AI-Based Solver for the 2048 Game

Submitted By: [Hassan Ali (22K-4637), Abdul Sami (22K-4354), Hussain (22K-4244)]
Course: AI
Instructor: Sir Khalid
Submission Date: 11th May' 2025

1. Executive Summary

Project Overview:

This project aims to develop an intelligent AI agent that can autonomously play and solve the 2048 puzzle game. The main objective was to implement an Expectimax-based solver capable of analyzing game states and making optimal decisions. The game uses a custom scoring heuristic that considers tile positioning, adjacency, and board space availability. The solver plays the game independently without any human input, continuously predicting and executing the best moves.

2. Introduction

Background:

2048 is a single-player puzzle game where the goal is to combine numbered tiles to create a tile with the value of 2048. It is typically played on a 4x4 grid where players slide tiles in one of four directions. When two tiles with the same number touch, they merge into one. The game ends when no legal moves are left.
This project was selected due to its simple rules but deep strategic complexity, making it an ideal candidate for implementing and testing AI decision-making strategies.

Objectives of the Project:

- Implement a fully automated AI agent using the Expectimax algorithm.
- Design an effective heuristic function for evaluating game states.

- Visually simulate the AI playing the game using a GUI built with Pygame.
- Evaluate the performance of the AI through gameplay observation.

3. Game Description

Original Game Rules:

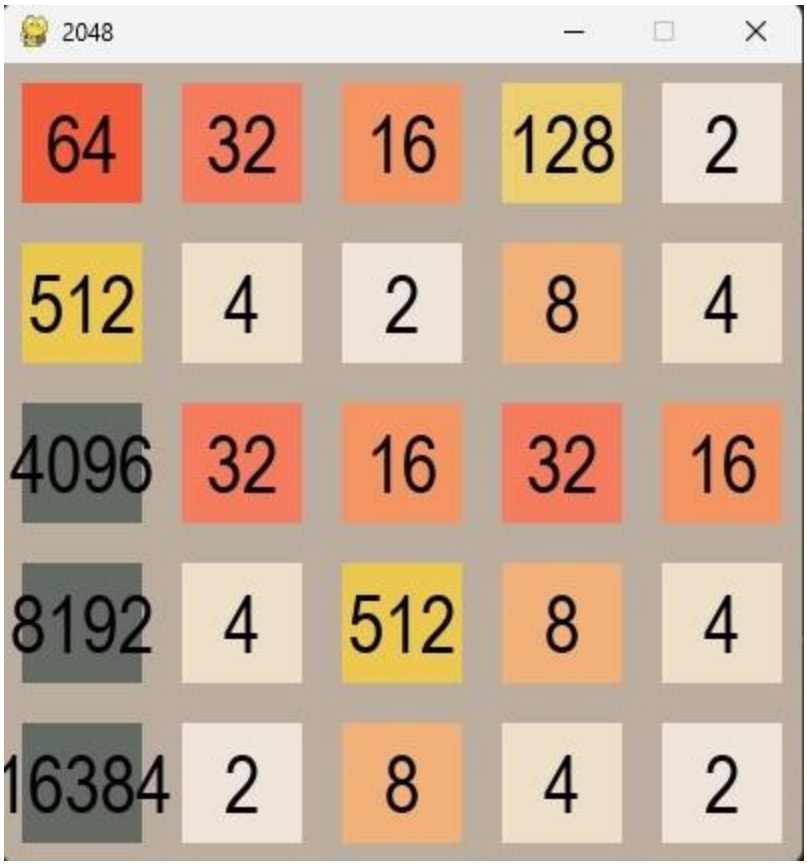
2048 is played on a 4x4 board. The player moves all tiles in one of four directions: up, down, left, or right. After each move, a new tile (2 or 4) appears randomly on the board. If two tiles with the same number collide, they merge into a single tile with double the value. The game continues until no valid moves are left.

Innovations and Modifications:

While the core rules of 2048 were maintained, an Expectimax-based AI was introduced to automatically play the game. The game now includes:

- Automatic decision-making based on a scoring heuristic.
- Visualization of moves using Pygame.
- Modified scoring function to favor long-term strategic play.

Also typical 2048 solver has a grid size of 4*4 we introduced that user can enter grid size as per his requirements. For example output for grid size 5*5 is shown below:



4. AI Approach and Methodology

AI Techniques Used:

The Expectimax algorithm was used for decision-making. Unlike Minimax, Expectimax accounts for stochastic outcomes, making it ideal for 2048 where tile generation is random.

Algorithm and Heuristic Design:

The AI evaluates all four possible directions by simulating moves and scoring the resulting board. The `get_score()` function combines:

- **Tile adjacency score**: Encourages combining similar tiles.
- **Snake pattern score**: Encourages placing higher-value tiles in corners.

- **Empty tile count**: Encourages keeping the board open.
  The snake score is given 3x weight due to its strategic importance, and the total score is normalized by dividing by 6.

AI Performance Evaluation:

The AI's performance was evaluated based on how long it could continue playing before no moves remained, and the maximum tile achieved. In most runs, it consistently reached 1024+ tiles with intelligent decision-making.

5. Game Mechanics and Rules

Modified Game Rules:

- The game is fully automated; the player does not provide input.
- After each move, the AI decides the next move based on score prediction.
- A new tile is generated only if the board changes.

Turn-based Mechanics:

Each turn follows this sequence:

1. Expectimax predicts the best move.
2. The move is applied.
3. A new tile is generated.
4. The board is re-evaluated.

Winning Conditions:

Although the original winning condition is reaching a 2048 tile, the AI continues playing until no more moves are possible to maximize performance.

6. Implementation and Development

Development Process:

The project was implemented in Python using object-oriented design. The `Grid` class handles the board logic, rendering, and tile generation. The `Solver` class uses Expectimax to evaluate moves. The Pygame library was used to visualize the board and render tile values in real-time.

Programming Languages and Tools:

- **Programming Language**: Python
- **Libraries**: Pygame, NumPy
- **Tools**: Visual Studio Code

Challenges Encountered:

- Designing an effective scoring function that balances tile merging and board space.
- Ensuring that Expectimax simulations didn't alter the real game state (solved using deep copies).
- Handling game-over conditions correctly without AI crashing.

7. Team Contributions

Team Members and Responsibilities:

- **Hassan:**
  - Implemented board logic, rendering, and game loop in Pygame (grid.py)
  - Did all the documentation.
- **Hussain**:
  - Designed and implemented Expectimax algorithm (game.py)
  - Helped in handling grid.py errors and integrating with game.py
- **Sami:**

- Developed custom heuristic scoring function in get_score() function along with snake scoring technique.
- Handled testing and performance validation.

8. Results and Discussion

● AI Performance:

The Expectimax AI consistently made intelligent decisions, reaching tile values up to 1024+ in most sessions. It efficiently managed space and merged tiles using the snake pattern strategy. Although it does not guarantee reaching 2048 every time (due to randomness), its decision quality was visibly strong.

- **Average Max Tile**: 1024
- **Average Survival Time**: ~100+ moves per game
- **Average Decision Time**: < 1 second per move (in real-time)

9. References

- 2048 Game Mechanics: https://play2048.co/
- Expectimax Algorithm: https://en.wikipedia.org/wiki/Expectimax
- Heuristic Strategies for 2048: https://arxiv.org/abs/1509.02237
- Pygame Documentation: https://www.pygame.org/docs/
- NumPy Documentation: https://numpy.org/doc/