SJSU SAN JOSÉ STATE UNIVERSITY

2048 Game with Reinforcement Learning

CMPE 252 - AI & Data Engineering
12.3.2025

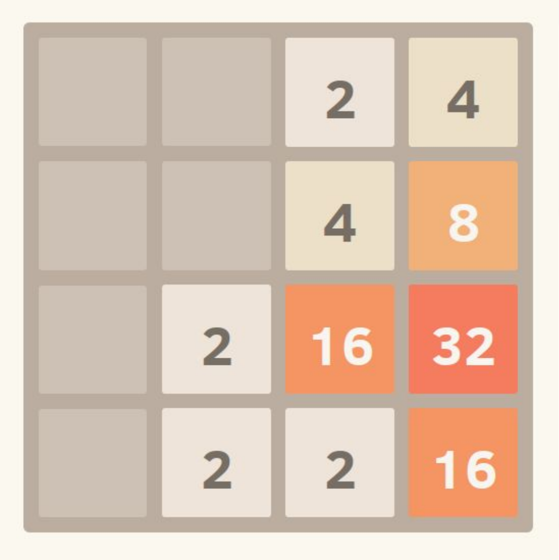
- Saurav Kashyap (018205655)
- Nidhi Sharma (018181527)
- Hiten Bharatkumar Galiya (018216861)

INTRODUCTION

This project applies **Deep Reinforcement Learning (DQN)** to teach an AI agent how to play the puzzle game **2048**. Instead of using fixed rules or heuristics, the agent learns gameplay strategies by interacting with the environment, receiving rewards, and improving through experience.

Why 2048?

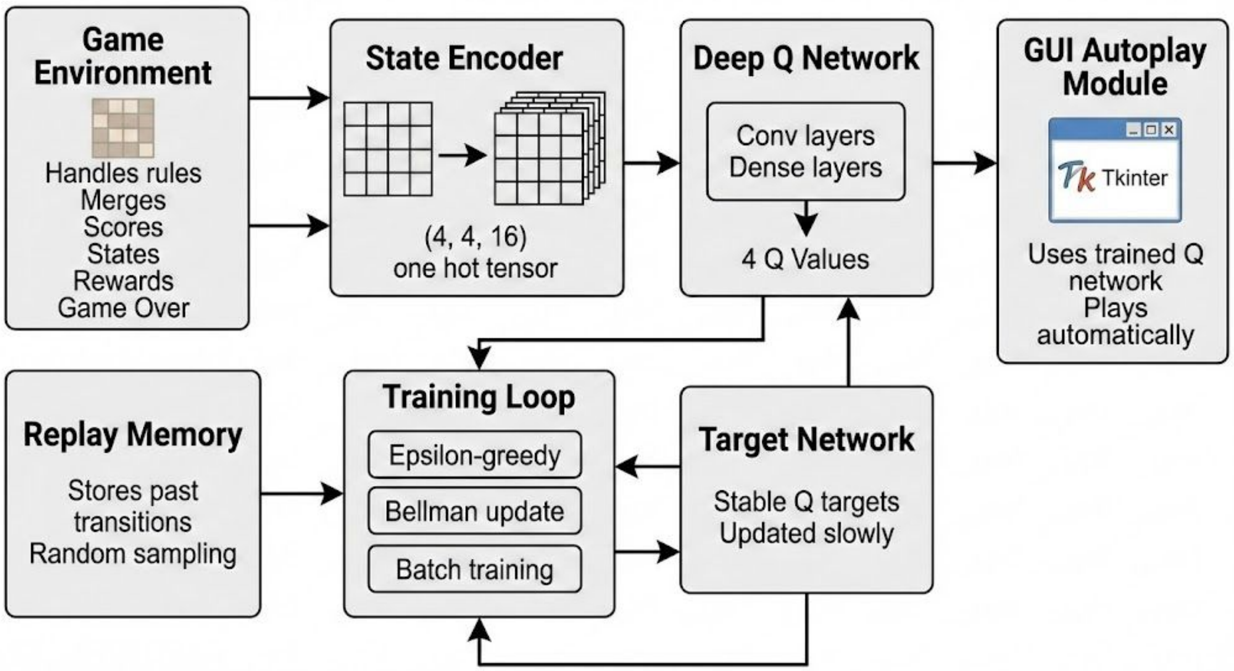
- Simple to understand, challenging to master
- Large state space due to random tile spawns
- Good benchmark for evaluating RL algorithms
- Requires reasoning, planning, and long-term strategy



Project Objective
Train a neural-network agent to:

- Select optimal actions (Up, Down, Left, Right)
- Maximize score and reach higher tiles
- Learn stable long-term strategies through exploration and rewards

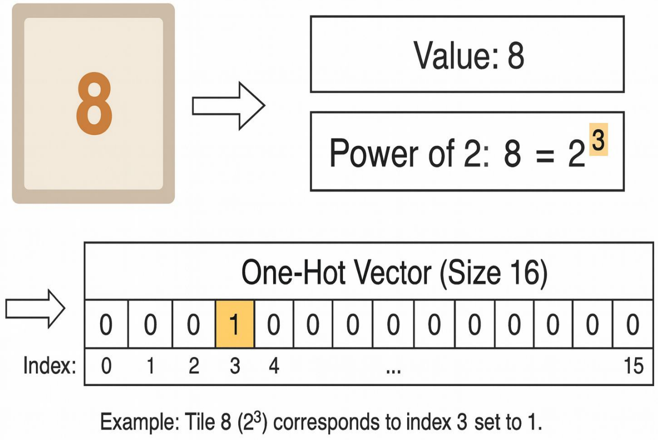
SYSTEM ARCHITECTURE



GAME ENVIRONMENT & STATE ENCODING

Game Environment

- Controls movement through up, down, left, right functions
- Performs merges, updates score, and shifts tiles
- Spawns new tiles after valid moves
- Checks win or lose conditions



State Encoding

- Each tile represented by its power of 2
- One hot vector of size 16 used for every cell
- Entire board encoded as a tensor of shape (1, 4, 4, 16)
- Structured format helps the CNN learn spatial tile patterns

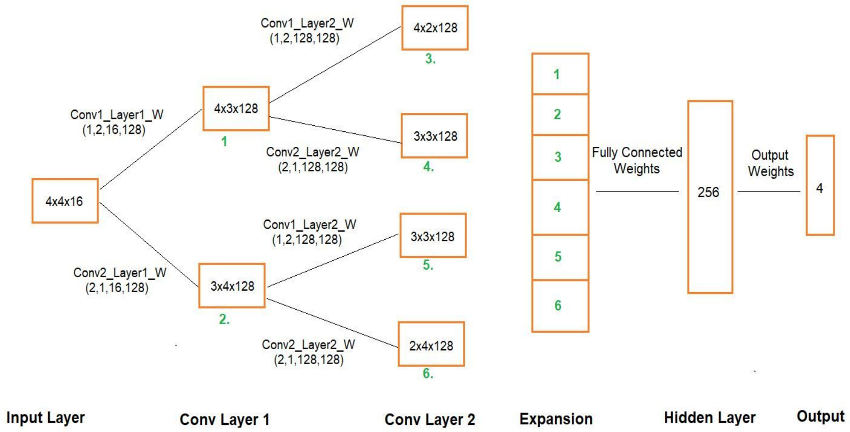
Q NETWORK MODEL

How the Model Works

- Input is a 4 by 4 by 16 encoded board
- Two parallel convolution paths extract local tile patterns
    - Kernels of size (1,2) and (2,1) capture horizontal and vertical relationships
- Second conv layer applies the same kernels again on each branch to deepen feature extraction
- Six feature maps are flattened and combined into a single vector
- Dense layer with 256 units learns high level strategy
- Output layer produces 4 Q values that correspond to possible actions
  (up, left, right, down)

Key Idea

The network evaluates the board and predicts which action leads to the highest expected future reward.



DEEP Q-LEARNING

Core Idea

The agent learns to choose the action that maximizes long term reward by estimating Q values for every possible move.

Key Concepts

- Uses Q values to measure how good each action is for the current board
- Epsilon greedy strategy balances exploration and exploitation
- Bellman update combines immediate reward with future expected reward

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

- Gradually improves decision making through repeated gameplay and training

Result

The agent starts to learn strategies like building larger tiles in one region and avoiding board lock situations.

REPLAY MEMORY AND TARGET NETWORK

Replay Memory

- Stores past transitions: state, action, reward, next state
- Random sampling breaks correlation between sequential moves
- Provides more stable and efficient learning
- Helps the agent learn from both good and bad experiences

$$(s, a, r, s') \sim \text{ReplayMemory}$$

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

Target Network

- A separate copy of the Q network
- Updated slowly to avoid unstable training
- Provides fixed Q targets during updates
- Reduces oscillations and improves convergence

Why This Matters

Replay Memory plus Target Network greatly improves the stability of Deep Q Learning and prevents model divergence during training.

TRAINING LOOP

How It Works

- Each episode starts with a new board
- Current state is encoded and passed to the Q network
- Epsilon greedy strategy selects an action
- Environment returns next state and reward
- Transition (s, a, r, s') is stored in replay memory

Model Update

- A random batch is sampled from replay memory
- Target Q values are computed using the Bellman update

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

- Loss is computed between predicted Q values and target Q values
- Network weights are updated using gradient descent

Why It Works

Consistent sampling and updates allow the agent to refine its strategy over thousands of episodes and steadily increase its performance.

TRAINING BEHAVIOR

What We Observe

- Early episodes show low scores and many invalid moves
- The agent explores frequently because epsilon is high
- Over time, Q values become more accurate as the model learns patterns

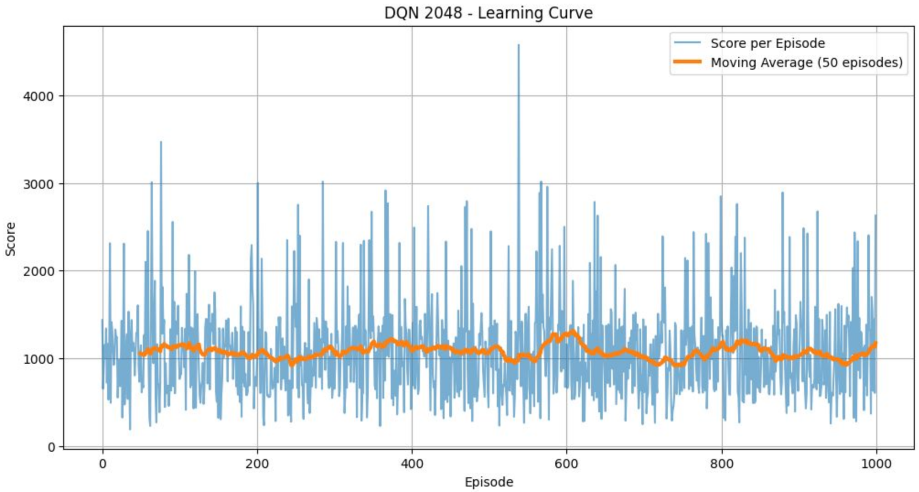
Performance Trends

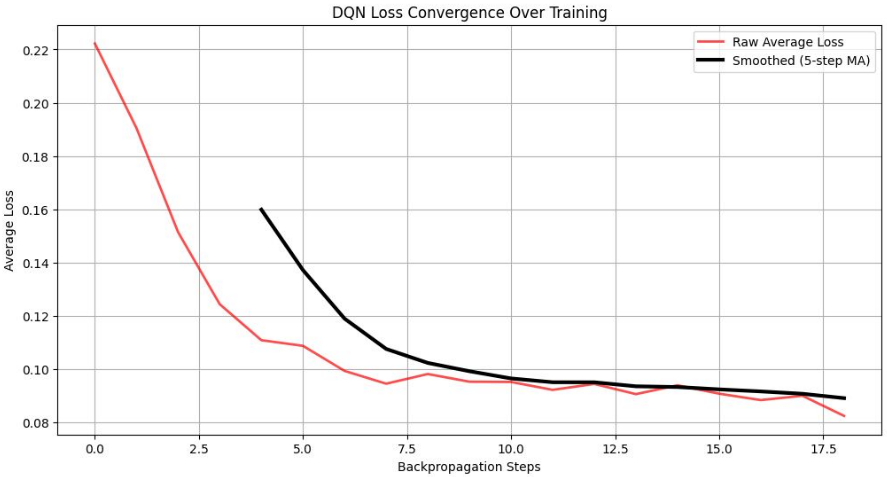
- Scores gradually increase across episodes
- Model often forms tiles up to 128-256, occasionally higher, and plays clearly better than random
- Invalid moves reduce as the agent understands board structure

Why It Matters

These trends confirm that the Q network is learning effective strategies and improving long term reward across training.

RESULTS





SJSU SAN JOSÉ STATE UNIVERSITY

RESULTS

CONCLUSION

- Complete RL pipeline built from scratch
- Combined game design and deep learning
- Demonstrated how Q learning can handle board games
- Future upgrades:
    - Double DQN
    - Prioritized replay
    - Better reward shaping

THANK YOU!