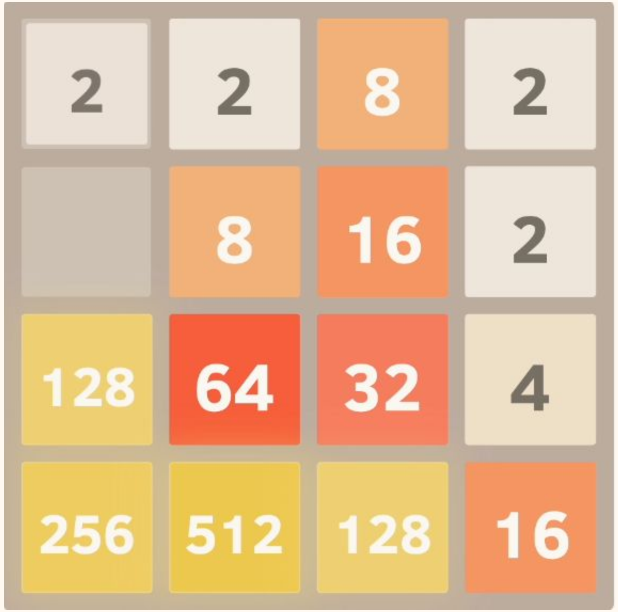
Comparing AI Agents Performance in 2048



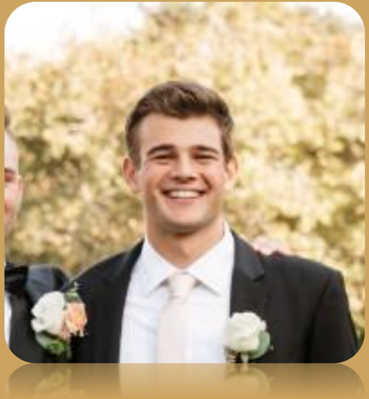
By Lucas Summers, Braeden Alonge, Nathan Lim, and Megan Fung

The Team: Contributions & Collaboration

Megan Fung

Computer Science ‘26

- Heuristics, Expectimax
- Presentation
- Report



Braeden Alonge

Computer Science ‘26

- Expectimax algorithm
- Report
- Presentation

Lucas Summers

Computer Science ‘26

- MCTS + Hybrid Models
- Heuristic design
- Flask server + applet
- Report + Presentation

Nathan Lim

Computer Science ‘26

- Reinforcement Learning agent / training
- MCTS + RL Hybrid Agent
- Report
- Presentation

Abstract

GOAL	Build high performing AI agents to play 2048.
AI Methods	Expectimax, MCTS, Reinforcement Learning (DQN), Hybrid
Outcome	Evaluate agent performance through simulations and heuristics.
Purpose	Benchmark planning algorithms in a stochastic environment.

Motivation

1

2048 is simple to play, hard to master, and ideal for AI experimentation.

2

Combines planning, randomness, and strategy.

3

Demonstrates real world decision-making under uncertainty.

4

Helps evaluate how AI agents can adapt and optimize gameplay over time.

Background + Prior Work

Expectimax:

- Used in many 2048 bots for modeling random tile placement
- Effective but struggles with large branching factors

MCTS:

- Balances exploration and exploitation.
- Uses a statistical framework to guide decision-making.

Reinforcement Learning:

- Requires significant training.
- Learn optimal strategies from scratch through self play.

Hybrid:

- Combines strengths of MCTS with predictive evaluation
- Better quality rollouts leads to better decisions overall

Toolchain & Technical Stack

Category	Tool/Framework	Purpose
Programming Language	Python 3.12	Common lang for AI projects
Game Environment	Flask + JavaScript	Interactive applet based on original game look
AI Algorithms	Expectimax, RL, MCTS, Hybrid	Custom AI agents to test
Deep Learning Framework	PyTorch	For RL (DQN Agent)
Math/Numerical Library	NumPY	Fast array operations + math utilities
Progress Monitoring	tqdm	Progress bars for simulations
Version Control	Git + Github	Team collaboration and code management
Results Visualization	Matplotlib + Seaborn	Easily generate plots from collected data

Overview of AI Agents

- **Random**: Baseline, no planning
- **Greedy**: Chooses moves based on immediate rewards
- **Expectimax**: Explores deterministic moves and stochastic tile spawns
- **MCTS**: Simulates outcomes using random rollouts and UCB1
- **Reinforcement Learning**: Learns state action values (Q values) from gameplay
- **Hybrid (w/ Expectimax)**: MCTS + Expectimax rollouts for smarter simulations
- **Hybrid (w/ RL):** MCTS + RL rollouts

Game Environment

Heuristics

We use a weighted composite of heuristics to evaluate how promising a board is for achieving high scores. These heuristics guide agents like Expectimax and MCTS, whose game tree rarely reaches a terminal state.

Empty Tile
- Rewards boards with more empty spaces
- Empty tiles → greater flexibility for future moves

Monotonicity
- Encourages tile ordering in increasing/ decreasing sequences along rows + columns

Smoothness
- Measures how similar adjacent tiles are:
- Differ in large amounts → penalized
- Gradual value transitions → rewarded

Quality/Stability
- How resistant the board is to quality loss
- Rewards the max tiles being protected by other high tiles

Corner Max
- Encourages strategic placement of the highest tile in a corner, a common expert strategy

Merge Potential
- Evaluates how many adjacent tiles are able to be merged (either vertically or horizontally)

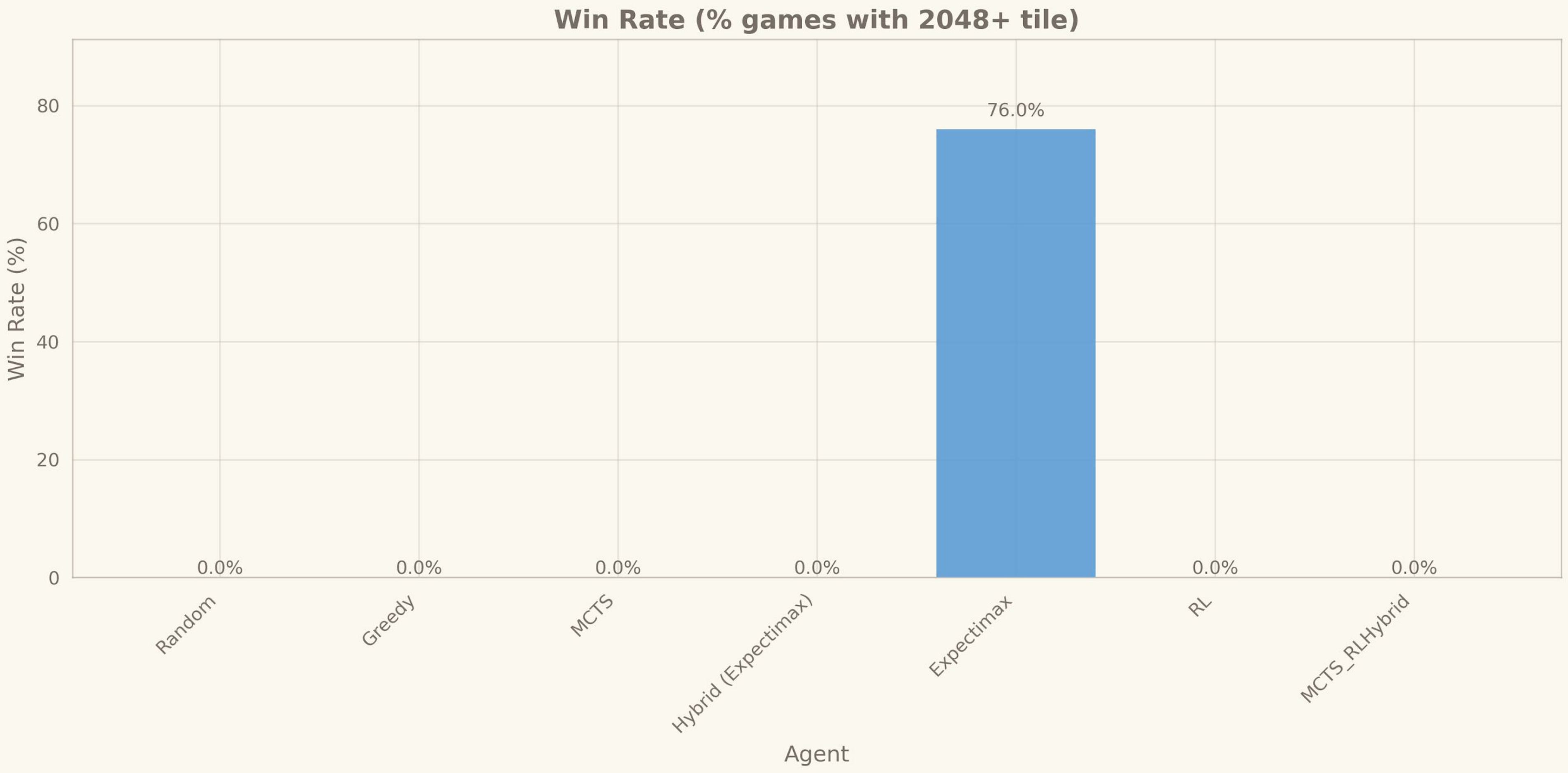
Experiment Design

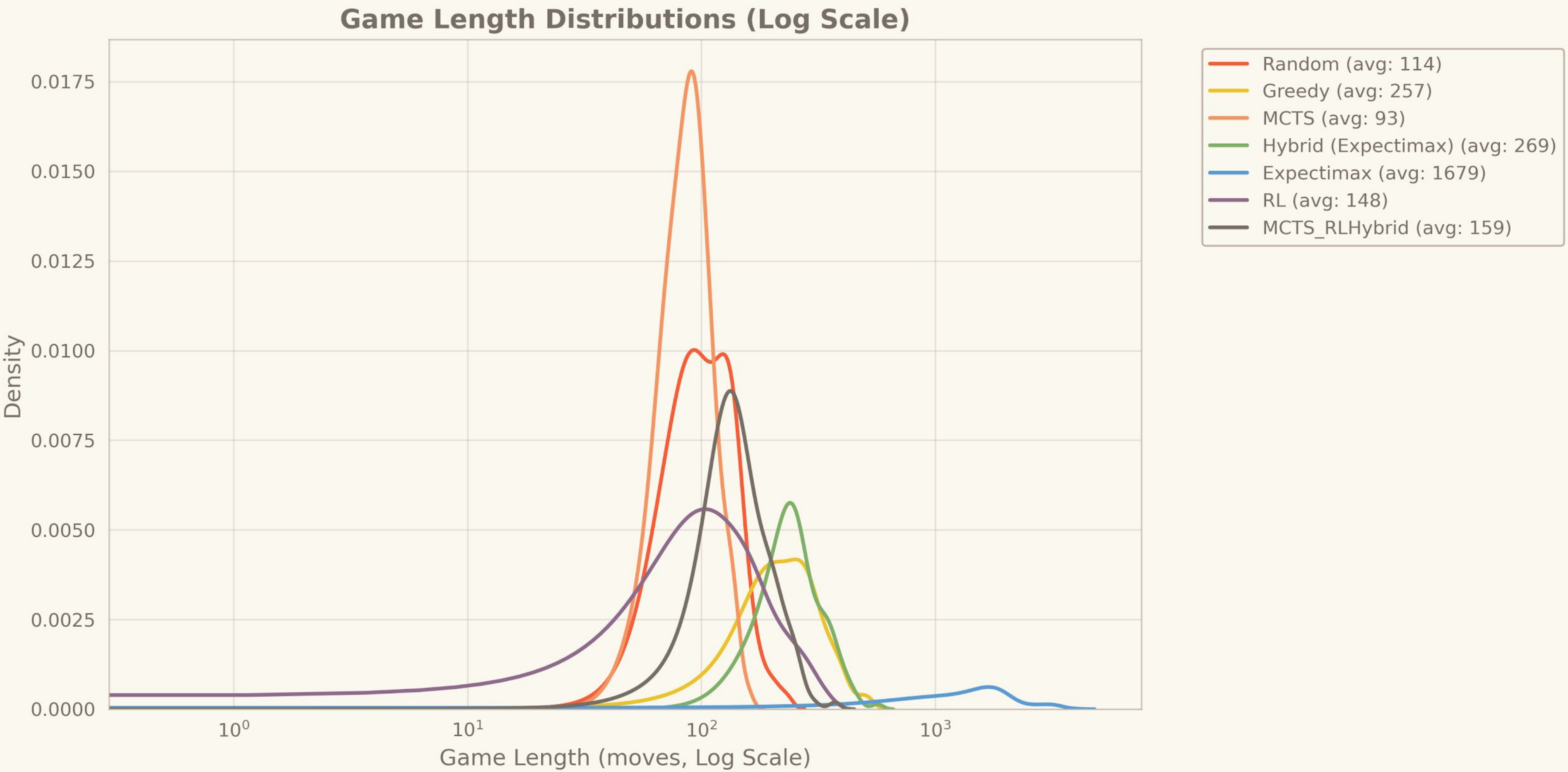
Agents Tested:
- Random
- Greedy
- Expectimax
- MCTS
- Reinforcement Learning
- MCTS/Expectimax Hybrid
- MCTS/RL Hybrid

Each agent plays 100+ games with a fixed random seed for reproducibility.

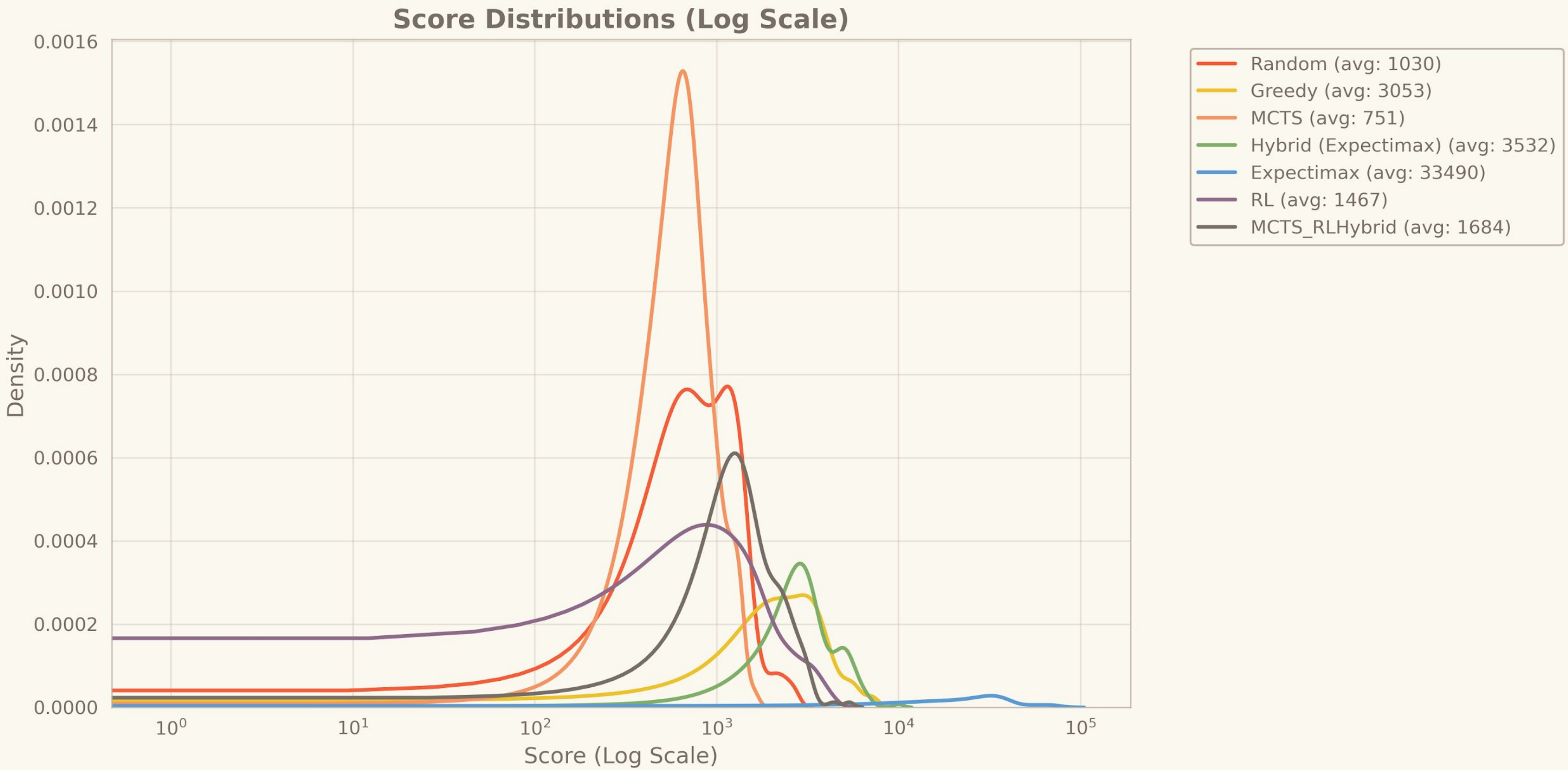
- Avg, Min, and Max Final Score
- Win Rate
- Average Moves Per Game
- Efficiency (score/moves)
- Median Max Tile
- Absolute Max Tile

Demo

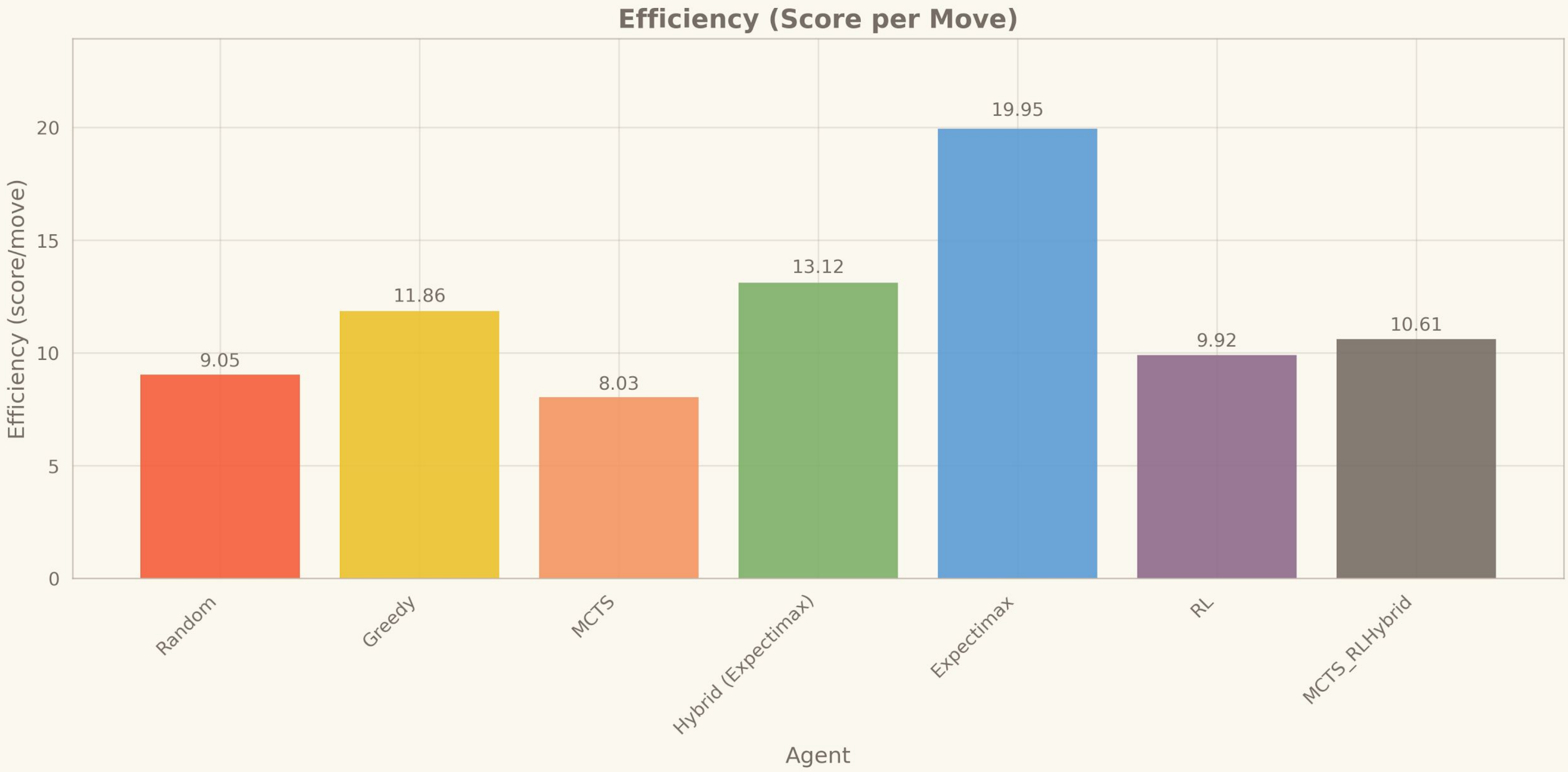

Win Rate (% games with 2048+ tile)
Win Rate (%)
0
20
40
60
80
76.0%
0.0%
0.0%
0.0%
0.0%
0.0%
0.0%
Random
Greedy
MCTS
Hybrid (Expectimax)
Expectimax
RL
MCTS_RLHybrid
Agent

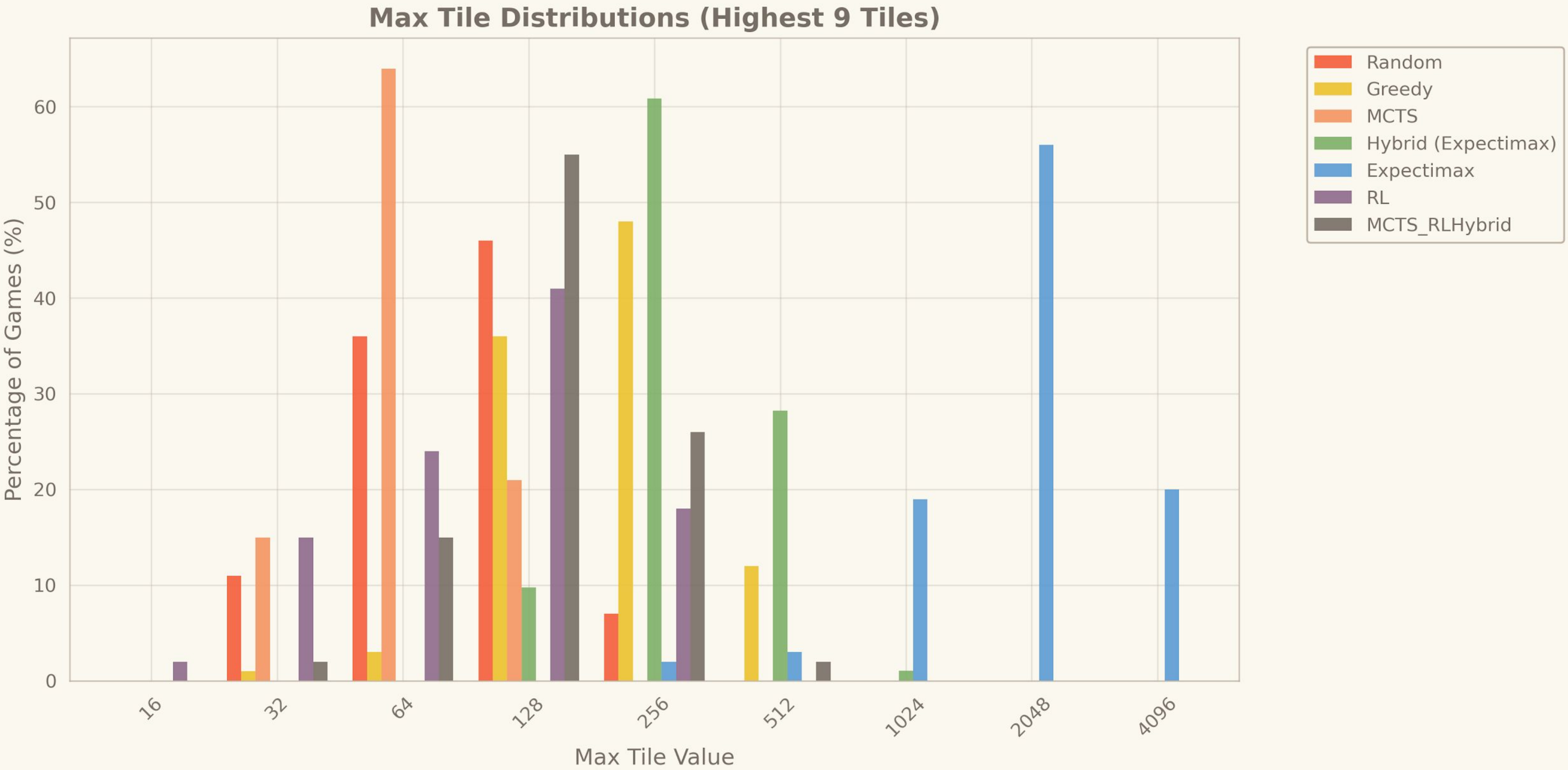


Game Length Distributions (Log Scale)
Density
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
10^0
10^1
10^2
10^3
Game Length (moves, Log Scale)
Random (avg: 114)
Greedy (avg: 257)
MCTS (avg: 93)
Hybrid (Expectimax) (avg: 269)
Expectimax (avg: 1679)
RL (avg: 148)
MCTS_RLHybrid (avg: 159)

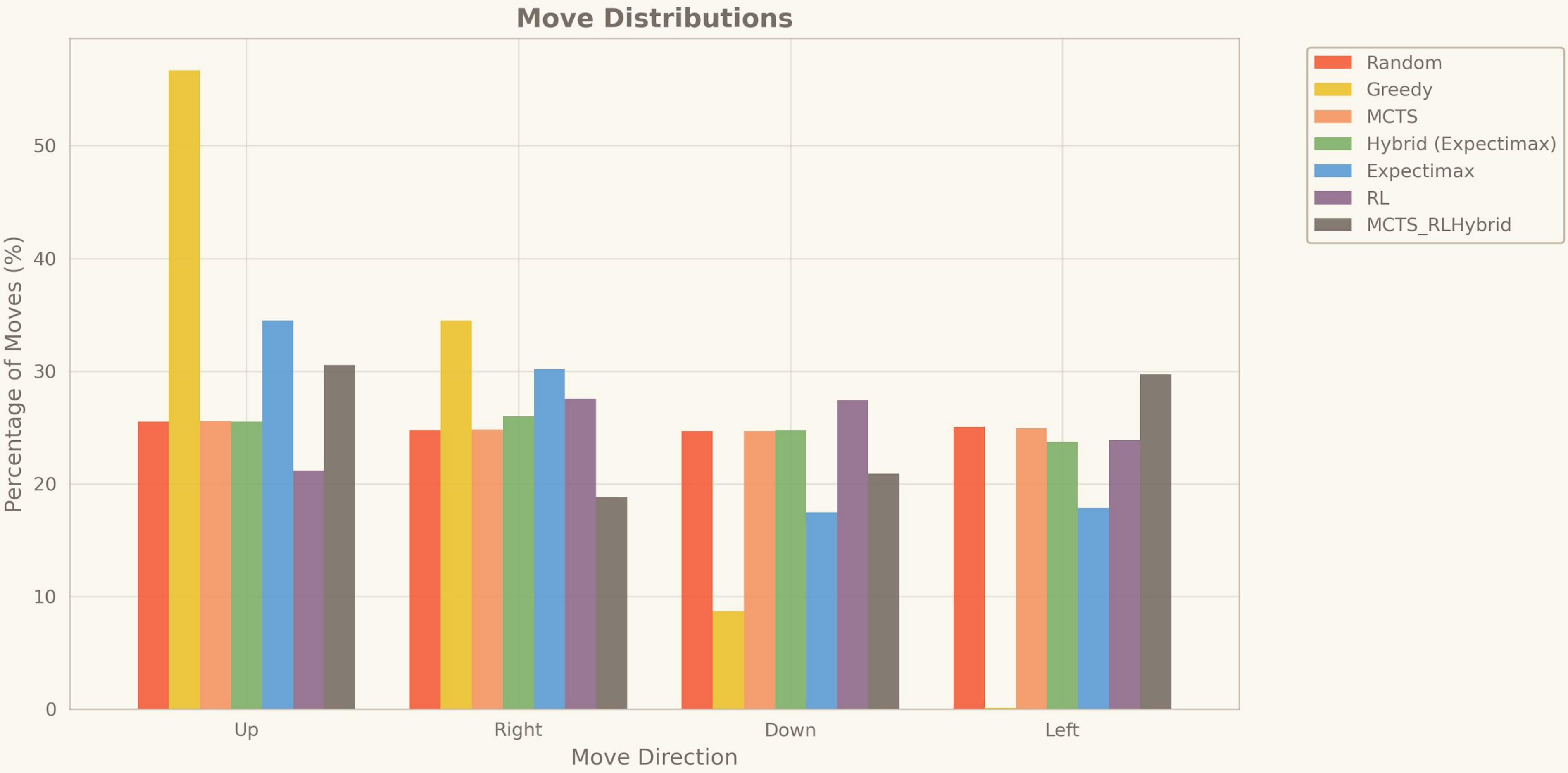


Score Distributions (Log Scale)
Density
0.0016
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
10^0
10^1
10^2
10^3
10^4
10^5
Score (Log Scale)
Random (avg: 1030)
Greedy (avg: 3053)
MCTS (avg: 751)
Hybrid (Expectimax) (avg: 3532)
Expectimax (avg: 33490)
RL (avg: 1467)
MCTS_RLHybrid (avg: 1684)


Efficiency (Score per Move)
Efficiency (score/move)
0
5
10
15
20
9.05
11.86
8.03
13.12
19.95
9.92
10.61
Random
Greedy
MCTS
Hybrid (Expectimax)
Expectimax
RL
MCTS_RLHybrid
Agent



Max Tile Distributions (Highest 9 Tiles)
Percentage of Games (%)
0
10
20
30
40
50
60
16
32
64
128
256
512
1024
2048
4096
Max Tile Value
Random
Greedy
MCTS
Hybrid (Expectimax)
Expectimax
RL
MCTS_RLHybrid


Move Distributions
Percentage of Moves (%)
0
10
20
30
40
50
Up
Right
Down
Left
Move Direction
Random
Greedy
MCTS
Hybrid (Expectimax)
Expectimax
RL
MCTS_RLHybrid

Results Conclusion

1. **Expectimax** achieved the highest absolute max tile (4096) and average score (33490.32), outperforming all other agents
2. **MCTS + Expectimax Hybrid** agent follows with 5081.01 average score and decent tile depth
3. **Greedy** agent has surprisingly good performance, coming in third
4. **MCTS** (Random), **MCTS** (RL), and **RL** showed balanced performance, with RL slightly outperforming MCTS in score
5. **Random** agent, as expected, had the lowest score and tile depth

These trends validate the strength of structured planning and evaluation in Expectimax and show Greedy agents can perform well with minimal computation.

Discussion

Observations

- Structured planning outperforms reactive or random strategies
- Expectimax's lookahead pays off despite longer computation time
- RL showed promise with generalization, though training time limited its performance

Challenges

- Runtime efficiency for deeper search agents
- RL tuning and training resource constraints
- Difficulty comparing agents with different decision speeds

Future Work

- Train deeper and more complex RL models
  - Double DQN or Dueling Networks)
- Explore more rollout strategies for MCTS
- Integrate AlphaZero-style RL
- Expand the analysis to larger board sizes (ex. 5×5 or 6×6) and compare scalability
- Evaluate computational efficiency more rigorously
  - Measure time per move under different hardware configurations
- Test adding, removing, or modifying existing heuristics
- Implement a grid search for ideal hyperparameters

Limitations

Thinking time:
- Less predictions and evaluation allowed for each algorithm
- Less thinking time means less effectiveness

Computational Power:
- No access to NVIDIA GPUs for CUDA
- Reduced efficiency and tasks completed with the given limited thinking time

Time Constraint:
- **Scope of the project was too large to produce desired results**
- **More time would allow for more research and better development of possible algorithms and heuristics**

Stochastic Game:
- **High variance led to poor algorithm effectiveness**
- **RL and MCTS struggled the most**

Retrospective

Successes

- Built a modular system with swappable agents and visual feedback
- Designed and implemented competitive AI agents using diverse strategies
- Created reusable evaluation metrics and tools

Lessons

- Heuristics matter immensely in search based agents
- Agent design is a balance between performance and practicality
- RL requires large amounts of training time and computational power

Reflection

- Developed a deeper understanding of reinforcement learning's real-world limitations
- Recognized the trade-offs between model accuracy and computational cost/thinking time
- Learned the importance of modular design for scalability and future improvements

Conclusion

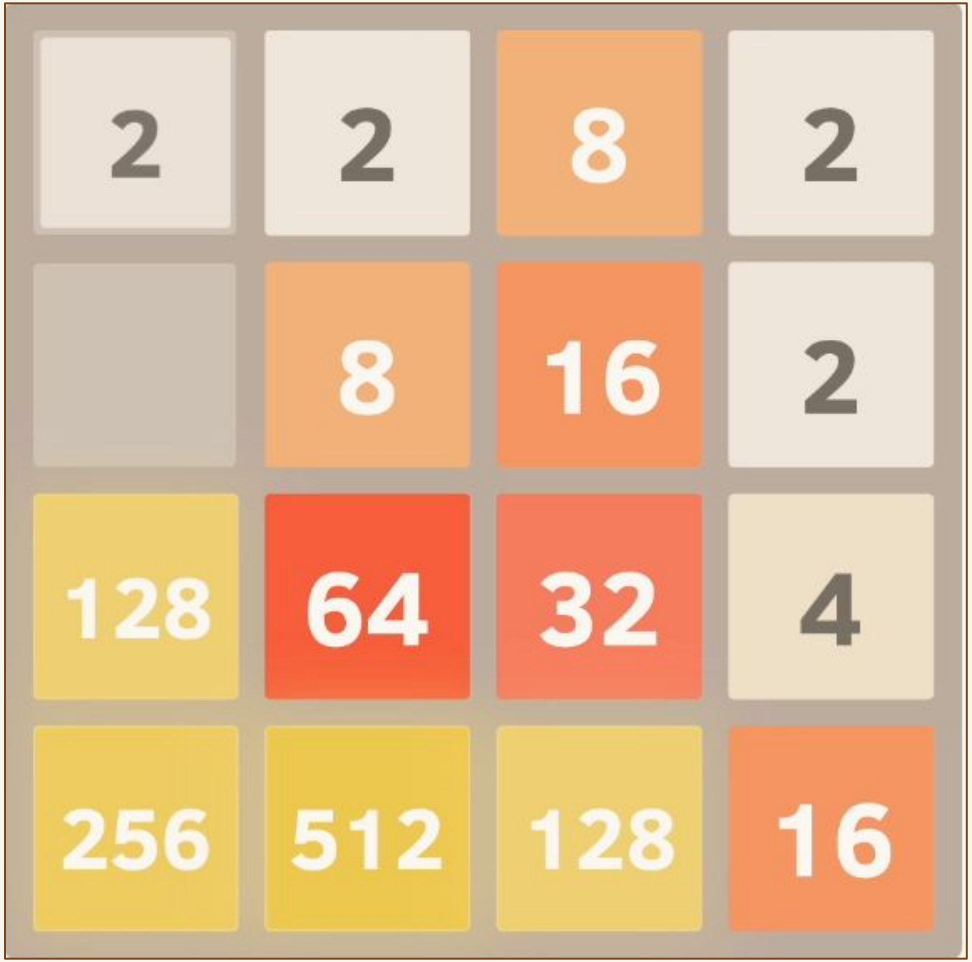
→ AI planning must balance depth, adaptability, and efficiency

Expectimax: High accuracy, low speed

MCTS: Balanced performance

RL: Potential for learning beyond heuristics

Hybrid: Promising results with smart rollouts



Thank You!

Q & A...