Learning how to play 2048 using TD(λ) learning over n-tuple networks

Maadhav Gupta

Ashoka University

March 2025

Actions and States

- Actions = {up, down, left, right}
- States = $\{s_0, s_1, s_2, \dots\}$
  $\#\{\text{States}\} \approx 10^{16}$

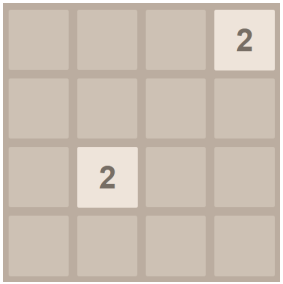


Figure: s_0

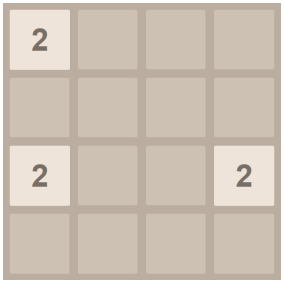


Figure: $s_1 = s_0$ after left

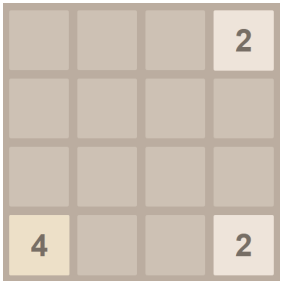


Figure: $s_2 = s_1$ after down

2048 Heuristics

- Monotonicity
- High Corner Value
- High number of blank tiles

n-Tuple Networks

- Uses fixed groups of tile positions (tuples).
- Each tuple maps observed tile patterns to a value.
- Multiple overlapping tuples are used.
- Captures local board structures and generalizes across similar configurations.
- Fast to evaluate and effective for learning non-linear patterns.

n-Tuple Networks

- Some states

- Network of tuples

Value Function Approximation

- The value function $V(s)$ estimates expected future reward from state s.
- Approximated as:

$$V(s) = \sum_{i=1}^{n} w_i[f_i(s)]$$

- $f_i(s)$: index in the lookup table for tuple i based on observed pattern.
- w_i: weight table storing values for tuple i.
- Enables generalization by sharing weights across states with similar features.

TD(λ) with Eligibility Traces

- Temporal Difference learning with eligibility traces updates value estimates across sequences.
- TD error: $\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$
- Eligibility trace update:
  - $e_t(f) \leftarrow \gamma\lambda e_{t-1}(f) + 1$ (if feature f is active)
- Weight update:
  - $V(f) \leftarrow V(f) + \alpha \cdot \delta_t \cdot e_t(f)$
- Assigns credit to both recent and earlier features, improving long-term learning.
- Suits 2048 as reward is delayed and survival is crucial.

ϵ-Greedy Policy with Decay

- Balances exploration and exploitation during training.
- At each decision point:

$$a = \begin{cases} \text{random action} & \text{with probability } \epsilon \\ \arg\max_a Q(s,a) & \text{with probability } 1-\epsilon \end{cases}$$

- ϵ decays from 1.0 to 0.01 over episodes.
- Encourages exploration early and stable policies later.

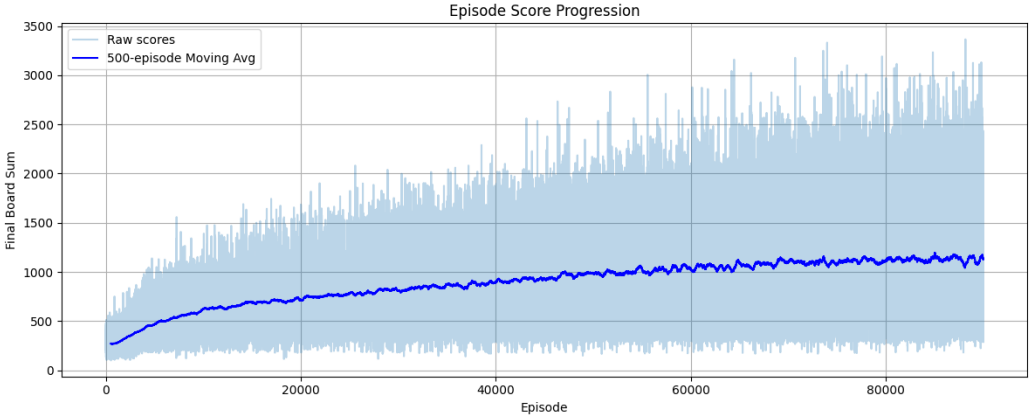
Experimentation

- TD(0) attempt: $\alpha = 0.01$, $\epsilon = 0.01$ fixed, $\gamma = 0.99$, *rows* only in n-tuple network
  - Episodes observed: 50000
  - Average score: below 600
  - Runtime: 6 hours
- TD(λ) attempt 1: $\alpha = 0.1$, $\epsilon = 0.1$ fixed, $\gamma = 1.0$, $\lambda = 0.9$, set of 30 n-tuples considered
  - Episodes observed: 50000
  - Average score: below 400, (as much as the random agent would score (or worse))
  - Runtime: 18 hours

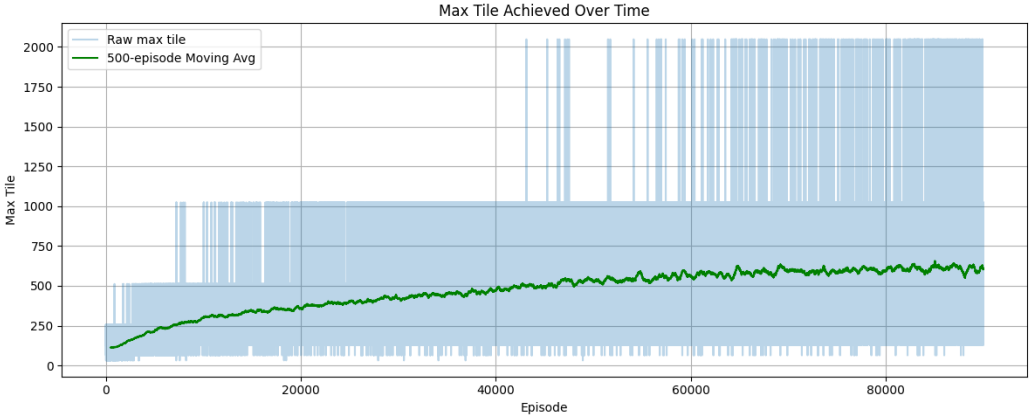
Experimentation

- TD(λ) attempt 2: $\alpha = 0.01$, $\epsilon = 0.01$ decaying to 0.1, $\gamma = 0.99$, $\lambda = 0.5$, n-tuple network = $\{rows, columns, diagonals\}$
  - Episodes observed: 100000
  - Average scores: reached 1200
  - 2048 achieved 5% of the times
  - Runtime 10 hours
  - ϵ decays to 0.01 in 11,500 episodes

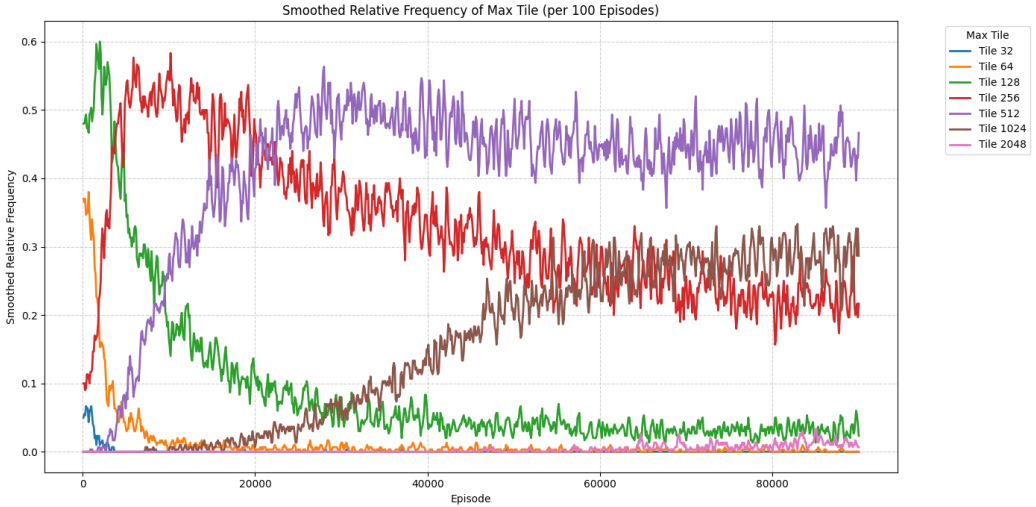
Results


Episode Score Progression
Raw scores
500-episode Moving Avg
Final Board Sum
Episode
0
500
1000
1500
2000
2500
3000
3500
0
20000
40000
60000
80000

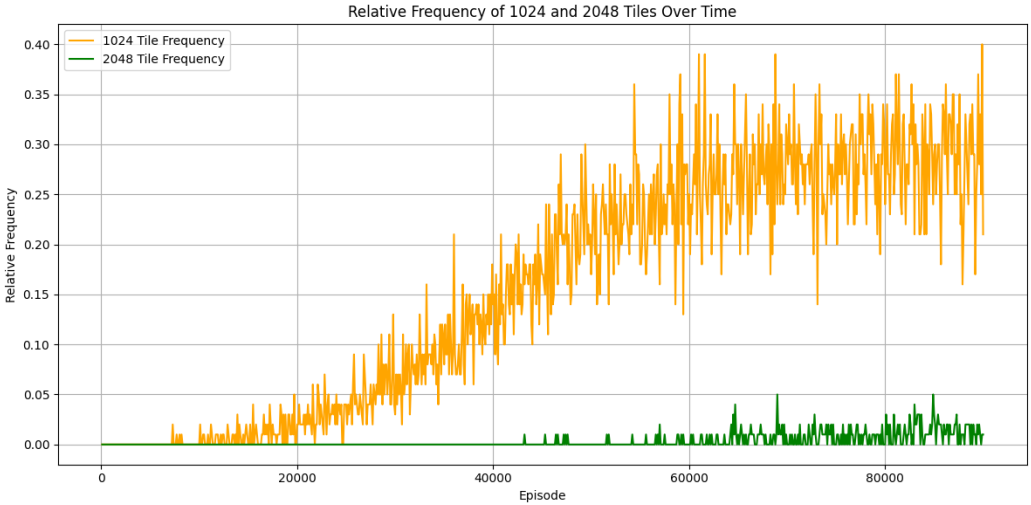
Results


Max Tile Achieved Over Time
Raw max tile
500-episode Moving Avg
Max Tile
Episode
0
250
500
750
1000
1250
1500
1750
2000
0
20000
40000
60000
80000

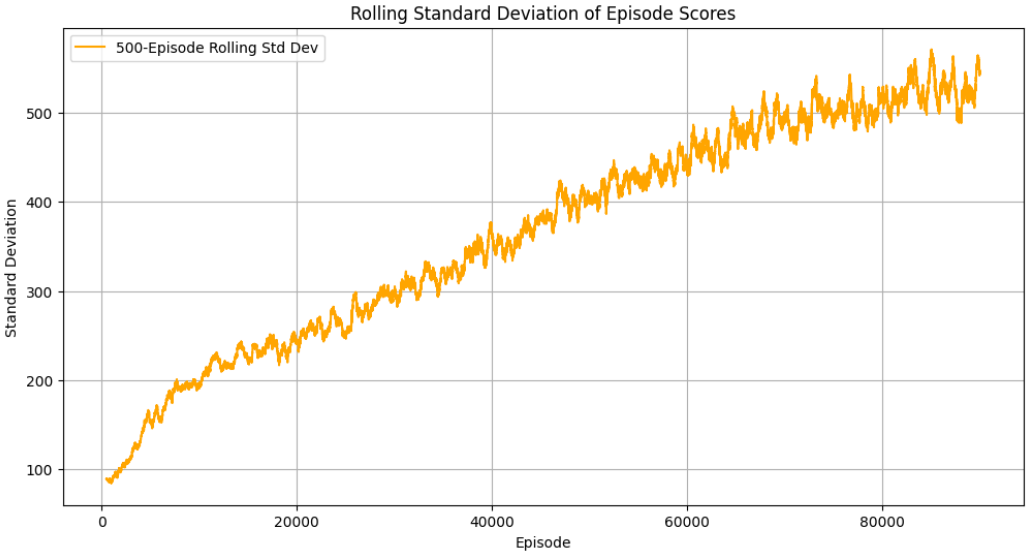
Results


Smoothed Relative Frequency of Max Tile (per 100 Episodes)
Smoothed Relative Frequency
Episode
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
20000
40000
60000
80000
Max Tile
Tile 32
Tile 64
Tile 128
Tile 256
Tile 512
Tile 1024
Tile 2048

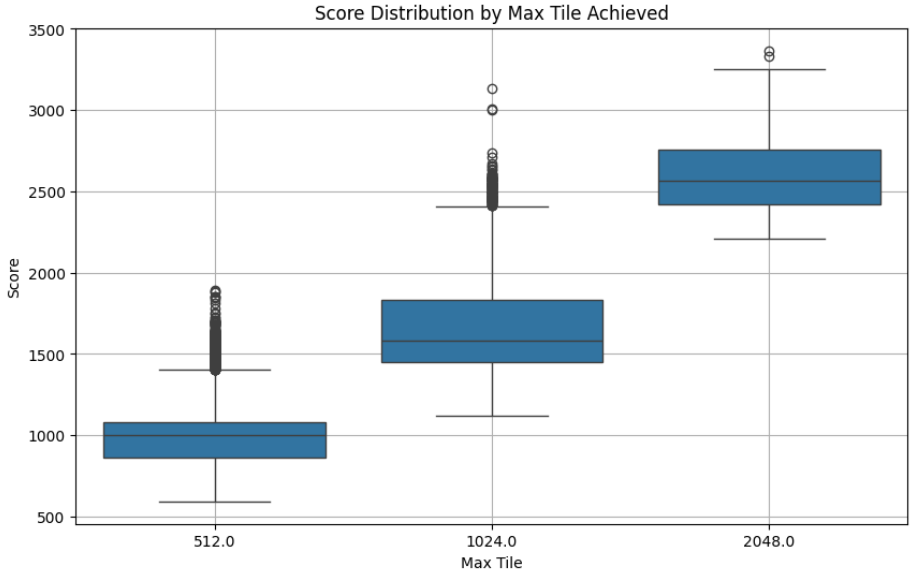
Results


Relative Frequency of 1024 and 2048 Tiles Over Time
1024 Tile Frequency
2048 Tile Frequency
Relative Frequency
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0
20000
40000
60000
80000
Episode

Results


Rolling Standard Deviation of Episode Scores
500-Episode Rolling Std Dev
Standard Deviation
100
200
300
400
500
0
20000
40000
60000
80000
Episode

Results


Score Distribution by Max Tile Achieved
Score
3500
3000
2500
2000
1500
1000
500
512.0
1024.0
2048.0
Max Tile

Improvements

- Can use more n-tuples and try different parameters or for more episodes
- Introduce exploration again after a few, say 10000, episodes
- Need to encourage corner points, present value function focusing on sum of grid