Reinforcement Learning for Constrained 2048: A Comparative Study of Deep Q-Networks and AlphaZero

Ayman Madani
Department of Computer Science
Carleton University
aymanmadani@cmail.carleton.ca

Jason Xu
Department of Computer Science
Carleton University
jasonxu@cmail.carleton.ca

Sachin Bansal
Department of Computer Science
Carleton University
sachinbansal@cmail.carleton.ca

Meetkumar Dudhat
Department of Computer Science
Carleton University
meetkumardudhat@cmail.carleton.ca

Arya Shah
Department of Computer Science
Carleton University
aryashah@cmail.carleton.ca

Abstract

This project investigates the effectiveness of Model-Free versus Model-Based Reinforcement Learning in solving the stochastic puzzle game 2048, with a specific focus on a "Constrained" variant containing an immovable obstacle. We implement a baseline Deep Q-Network (DQN) using Stable Baselines3 and compare it against a custom AlphaZero implementation featuring a Monte Carlo Tree Search (MCTS) engine and a Hybrid-Kernel Convolutional Neural Network. Empirical results demonstrate that while the reactive DQN struggles to adapt to the constrained environment (Average Score: 1,226), the deliberative AlphaZero agent successfully plans around the obstacle (Average Score: 2,231), achieving a 1.8x performance improvement. The study confirms that lookahead search is essential for navigating constrained stochastic domains.

1 Introduction

The game 2048 is often used to study reinforcement learning algorithms because it contains a mix of randomness and requires long-term planning. Although it looks simple at first, it actually has a large number of possible board states (on the order of 10^{16}). This game requires an agent to think about immediate merges for points together with long term board structure so it does not get stuck later. Most existing RL agents rely heavily on deterministic human strategies like the "corner strategy", where the highest tile is kept in one corner to maintain stability throughout the game.

In many real life decision-making tasks, including variants of 2048, the agent does not always get a perfectly open environment. There may be some blocked areas, fixed obstacles, or parts of the space that cannot be used, which means that common heuristics like the corner strategy do not always hold. To explore this idea further, we examined a tougher version of the 2048 game. In our setup, we placed an immovable block at the coordinate $(3, 0)$ on a standard 4×4 board. This block has a value of -1 and permanently occupies a grid cell. It never shifts or merges with any other tile. Although it

Preprint. Under review.

is only a single blocked space, it changes how the entire board evolves because the agent now must work around this obstacle for the entire duration of the game.

This added obstacle immediately disrupts the usual corner-based approach and reduces the safe spaces the agent can use. As a result, the agent must discover different ways to structure the board and can no longer rely on keeping high valued tiles locked in a corner anymore. The decision-making problem becomes significantly harder because the agent needs to reconsider how it forms merges over time, avoid blocked cell, and still manage the randomness of new tiles appearing. Studying this constrained setup helps reveal how both model-free and model-based RL methods behave when the environment is not fully open and imposes permanent spatial limits like the immovable block restricting movement.

1.1 MDP Specifications

- **State Space (S):** A 4×4 grid where each cell contains a tile value $t \in \{2^1, \ldots, 2^{16}\}$ or is empty. In Constrained environment, one fixed coordinate contains an Immovable block with value of -1, reducing reachable grid configurations and altering legal transitions.
- **Action Space (A):** A Discrete set of four actions $\{$Up, Down, Left, Right$\}$. Each action slides all tiles in the chosen direction and merges matching tiles.
- **Reward Function (R):** The standard 2048 reward is the merge value gained after combining tiles. Sparse binary reward ($+1$ for Win (achieving 2048 tile), -1 for Loss (no moves available)).
- **Transition Function (P):** After each valid move, a new tile appears in a random location: a 2 with 0.9 probability and a 4 with 0.1 probability. In the Constrained environment, the Immovable Block is excluded from movement and from tile spawning.

  This MDP formulation allows us to compare how different RL approaches, specifically a Model-Free DQN and a Model-Based AlphaZero-style agent, adapt to the spatial obstruction, handle restricted transitions, and develop strategies that remain robust under constrained dynamics.

2 Related Work

Szepesvári's *Algorithms for Reinforcement Learning* (8) gives us the theory behind value-based methods and was the first to establish PAC bounds on tabular Q-learning and explore the difficulties of function approximation. Our DQN baseline implementation is a DQN version of these classical principles, but operates in a high-dimensional setting where neural networks approximate $Q(s, a)$. This is exactly the case where Szepesvári's exploration-exploitation paradox is particularly important in the given context and where extra dimensionality complicates things.

Model-free value-based reinforcement learning was popularized by the Deep Q-Network (DQN) algorithm (5) by Mnih et al., which improved Q-learning on Atari with the help of replay buffers and target networks. Our baseline approach uses essentially the same recipe (replay and target network) on a reduced 4×4 grid with tile placement noise and a static obstacle such that generalization is more a result of exploration than image feature generalization.

In Prioritized Experience Replay from Schaul et al. (6), sample efficiency is increased by replaying transitions with high TDerror. For this project, we chose to only implement uniform replay; however, with the potential benefits of replay priorities, fewer episodes may be necessary for obstacle-aware policies.

Rainbow DQN (4) combines six extensions (Double Q-learning, PER, multi-step return, dueling networks with multi-output heads, Noisy Nets, distributional RL) and achieves significant Atari improvements. Our baseline excludes these for being simpler; the multi-step return and Noisy Nets from Rainbow might assist with sparse rewards from the game's merge action and continued exploration after the replay buffer is full.

AlphaZero by Silver et al. (7) integrates policy-value networks with Monte Carlo Tree Search (MCTS) for perfect information game play. Our model-based agent adopts this planning approach and modifies the application scope of MCTS to a stochastic single-player environment with chance

nodes for tile appearances and hybrid kernels for examining full rows and columns, thus explaining its strength within the limited environment that requires significant planning.

Gayas (Expectimax for 2048) (3) implement depth-limited searches with carefully designed features involving the ability to merge/monotonicity. In comparison to the above-mentioned traditional depth-limited searches as baselines, while the DQN learns to estimate values from experience instead and lacks any form of lookahead, which degrades when the immovable block breaks the standard cornering heuristics.

Antonoglou et al. (1) investigate planning with learnable models in stochastic tasks. Their observations confirm our finding: AlphaZero-style model-based rollouts perform better than reactive value-based approaches if one has to plan for uncertainty about the future (tile appearance) and structural constraints (cell occupied).

3 Approaches

3.1 Baseline: Deep Q-Network (DQN)

We apply a standard DQN from Stable Baselines3 with an MLP policy (two layers with 256-dimensional hidden representations) on a `Log2`-normalized 4×4 game board. Our agent is purely reactive with no forward thinking or search. Important hyperparameters included, with some variations between training, a replay buffer size of 100K transitions, batch size of 128, target network update frequency at every 1K steps, and learning rate at 10^{-4} with ε-greedy exploration schedule decaying to 0.02 over training time. We optimize with unprocessed game merge rewards (or log-merge for ablated experiments). It is a lightweight method, unable to play well when the obstacle obstructs the corner showing the difficulty with model-free methods within spatial constraints and for uncertain game start locations.

3.2 Improved: AlphaZero (MCTS + Hybrid CNN)

To address the limitations of reactive policies in the Constrained environment, we implemented a variation of the AlphaZero algorithm adapted for single-player stochastic domains. The architecture consists of two primary components: a Monte Carlo Tree Search (MCTS) engine for planning and a Deep Neural Network for intuition.

3.2.1 Stochastic MCTS Engine

Unlike the standard AlphaZero algorithm which assumes an adversarial opponent (Minimax), our MCTS models the environment as a stochastic actor. The search tree alternates between two node types:

- **Decision Nodes:** Represent states where the agent selects an action $a \in A$. Selection follows the Predictor + Upper Confidence Bound (PUCT) formula to balance exploration and exploitation.
- **Chance Nodes:** Represent states where the environment spawns a tile. These nodes do not use UCB but instead expand children based on the transition probabilities (0.9 for a 2, 0.1 for a 4).

This distinction allows the agent to calculate the *expected value* of a move over 150 future simulations, effectively "imagining" the board filling up around the obstacle.

3.2.2 Hybrid-Kernel Neural Network

Standard Convolutional Neural Networks (CNNs) utilize 3×3 filters, which are optimized for local spatial features in images. However, 2048 mechanics rely on sliding entire rows and columns. To capture this, we designed a Hybrid-Kernel architecture:

- 1×4 **Kernels:** Scan entire horizontal rows to detect merge potential and obstructions.
- 4×1 **Kernels:** Scan entire vertical columns.

This architectural choice provides the agent with the global context necessary to detect if a "snake" chain is broken by the immovable block in the constrained environment.

3.2.3 State Representation & Optimization

We addressed the exponential scale of tile values (2 vs 65,536) by implementing a `Log2` normalization wrapper, converting values to a linear scale $(1 \dots 16)$. Furthermore, to support the computational demands of MCTS, the game physics engine was optimized using Numba JIT compilation, accelerating simulation speed by approximately $50\times$.

4 Empirical Studies

4.1 Experimental Setup

We evaluated both agents in two environments:

1. **Standard:** Classic 2048 rules.
2. **Constrained:** An immovable block (value -1) is placed at coordinate $(3, 0)$.

Evaluation metrics include Average Score, Max Tile Reached, and Win Rate over 100 test episodes.

4.2 Main Results

Table 1 summarizes the performance comparison.

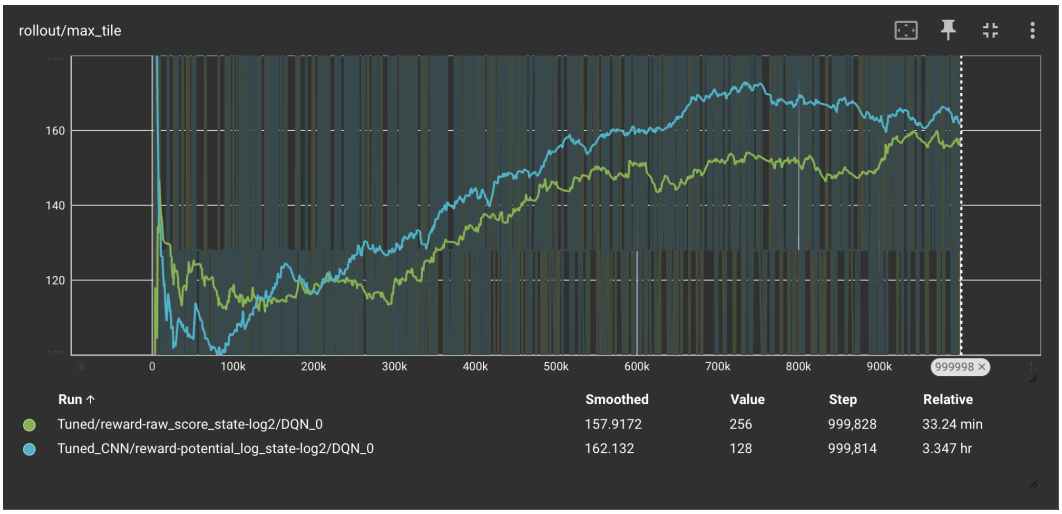
Table 1: Performance comparison over 100 episodes.

Environment	Agent	Avg Score	Max Tile (Avg)	Improvement
Standard	Baseline (DQN)	2,096	187	-
Standard	**AlphaZero (Ours)**	**7,371**	**560**	**+251%**
Constrained	Baseline (DQN)	1,226	129	-
Constrained	**AlphaZero (Ours)**	**2,231**	**202**	**+82%**

The results indicate a decisive advantage for the Model-Based approach. In the Standard environment, AlphaZero achieved an average score $3.5\times$ higher than the Baseline. More importantly, in the **Constrained** environment, the Baseline's performance degraded by 41% as it failed to adapt to the dead zone. The AlphaZero agent, utilizing lookahead search, successfully navigated the constraint, maintaining a strong average score of 2,231. Qualitative analysis of the final board states reveals that the AlphaZero agent learned to anchor its tile chain in the corner *opposite* the immovable block, a strategic adaptation that the reactive Baseline failed to discover.

4.3 Ablation Studies

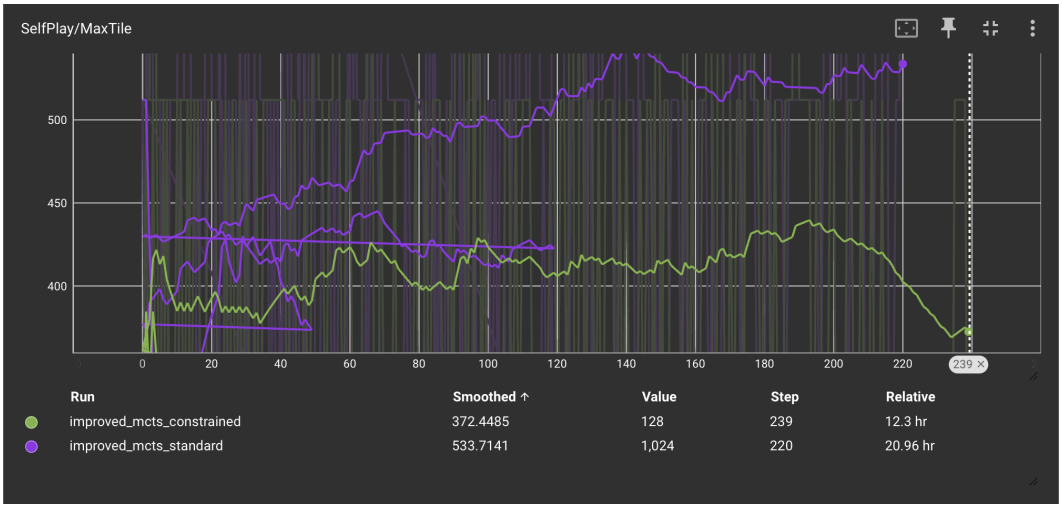
To understand the influence of state representation on learning stability, we conducted an ablation comparing two encodings of the 2048 board: (1) Raw tile values, where entries range exponentially from 2 to 65,536, and (2) a Log_2 encoding that compresses all tiles into the range 1–16. When trained with Raw values, the AlphaZero agent achieved an average score of only 1,025 and frequently produced erratic value estimates. The sharp numerical disparity between small and large tiles caused unstable gradients in the Hybrid-Kernel CNN, reducing the reliability of MCTS rollouts.



In contrast, the Log_2 representation yielded an average score of 2,096, nearly doubling performance. This improvement stems from transforming the exponentially scaled tile distribution into a linear space that is easier for the network to model. The normalized representation allowed the CNN to learn merge patterns more consistently and enabled MCTS to propagate value estimates more accurately across simulated trajectories. Overall, our ablation confirms that normalization is not merely beneficial but essential for robust policy learning in 2048-like domains.

4.4 Complexity Analysis

Although both agents operate in the same environment, their computational demands differ significantly. The Baseline DQN required approximately 30 minutes of training, driven largely by simple forward passes through a lightweight MLP and off-policy updates using batches from the replay buffer. In contrast, the AlphaZero pipeline required close to 10 hours to train under comparable evaluation conditions. The principal source of overhead is Monte Carlo Tree Search: each move requires 150 simulations, each simulation invoking multiple forward passes through the Hybrid-Kernel CNN.



Additional cost arises from processing Chance Nodes to model stochastic tile spawning, which further expands the number of states evaluated during planning. Despite this expense, the resulting policy is substantially stronger and far more adaptive, particularly in the Constrained environment where long-term spatial reasoning is required. Importantly, while training is costly, inference after training is efficient and requires only a single network evaluation per move. This reflects a classic trade-off in reinforcement learning: higher upfront computation yields significantly improved decision quality.

5 Conclusion

5.1 What we learned about the problem

We started our project with a seemingly simple puzzle 2048, but the introduction of an immovable, non-merging "-1" tile turned it into a much richer reinforcement learning and planning problem. In the standard game, many strong strategies are essentially short-sighted but geometry-aware: keeping the largest tile in a corner, preserving monotonic rows and columns, and greedily taking merges that maintain empty space. In our constrained variant, these rules of thumb are no longer reliably safe. A single push that funnels mass toward the blocked corner can irreversibly destroy board geometry, trapping large tiles behind the dead zone, and collapsing future movability.

This forced us to confront two important aspects of the problem:

1. Long-horizon credit assignment: moves that look harmless but can be catastrophic several steps later, making short-term reward maximization insufficient.
2. Structure awareness: the agent must understand the board's physics, that is, how the constraint interacts with tile flows and empty cells rather than just memorize local patterns.

What we learned is that the structural constraints transform 2048 from a greedy-friendly game into a planning heavy control problem.

5.2 What we learned about the approaches

We implemented and compared deep RL baselines algorithms like DQN-style methods in our custom constrained environment with a model-based AlphaZero style agent using MCTS. This comparison provided us with several insights:

- Model-free deep RL conflict under constraints, even with sensible state encodings and reward shaping, the DQN baselines tended to overvalue short-term merges and often drifted tiles toward the blocked corner. They could achieve reasonable scores in easier settings, but in the truly constrained scenario they frequently converged to locally consistent, failing to systematically avoid the dead zone.
- Model-based RL handled the constraint more gracefully. The AlphaZero agent explicitly simulates future roll-outs before committing to an action, allowing it to see that certain moves will squeeze tiles into unsalvageable positions. In qualitative implementations, it was observed that the planning agent tended to maintain a clean region away from the -1 block, keeping high-value tiles in safer lanes and maintaining more empty cells over time.

We realized that the constrained 2048 problem is not only "difficult 2048", but is fundamentally a testbed for the value of look-ahead and for the limits of model-free deep RL.

5.3 Limitations and Failure Modes Observed

Despite the advantages of search, our system can still fail when stochastic spawns align unfavorably, or when the search budget is too small to notice rare but catastrophic futures (e.g., a merge that looks good but collapses the safe stack). On the model-free side, we observed sensitivity to reward shaping and replay content; agents could learn locally stable but globally brittle strategies if evaluation emphasized short horizons.

5.4 How the method can be improved with more time and resources

1. Train for longer and extend the scale search Run more self-play training for the AlphaZero agent and allocate more MCTS simulations per move, especially in late game positions where the constraints are more dangerous. This might sharpen the policy's understanding of catastrophic futures and improve robustness.
2. Introduce dynamic constraints So far the constraint is static. A natural next step can be dynamic constraints, where we progressively increase constraint difficulty during training. This would test generalization and encourage the agent to learn more abstract principles like avoiding high-value tiles isolation rather than over-lifting to one specific layout.

3. Use a deeper ResNet Backbone A deeper ResNet could better capture the global board structure such as empty-cell patterns around the -1 block and provide policy estimates for MCTS to build on.

   Beyond these primary directions, hybrid approaches can also be considered, with distributional RL for risk aware planning and systematic studies on reward shaping.

5.5 Summary

In this project, we designed a constrained 2048 environment, implemented model-free deep RL baselines and an AlphaZero-style MCTS agent, and systematically compared how they behave under structural constraints.

We learned that:

- The constraint fundamentally changes the nature of the task, making long-term planning and geometry preservation essential.
- Standard deep RL, which can perform reasonably in unconstrained 2048, often fails to internalize the consequences of the dead zone.
- Model-based RL with MCTS is far better suited to this setting, precisely because it can simulate the environment's physics and choose actions that remain viable many steps into the future.

The most promising directions are to train for longer, upgrade to a deeper ResNet backbone, and explore dynamic constraints that further stress-test the agent's ability to generalize. More broadly, our findings highlight a transferable lesson for real-world RL: when the environment contains hard constraints and irreversible failures, planning and model-based reasoning are often necessary for robust performance.

References

[1] Antonoglou, I., et al. (2022). Planning in Stochastic Environments with a Learned Model. *International Conference on Learning Representations (ICLR)*. Available at: `https://openreview.net/pdf?id=X6D9bAHhBQ1`

[2] Fuchs, S. (2020). *2048 AI | Develop & Fix*. Available at: `https://www.sam-fuchs.com/s/projects/2048/`

[3] Gayas, V. (2014). *2048 Using Expectimax*. Miner School of Computer and Information Sciences, UMass Lowell. Available at: `https://www.cs.uml.edu/ecg/uploads/AIfall14/vignesh_gayas_2048_project.pdf`

[4] Hessel, M., Modayil, J., Van Hasselt, H., et al. (2018). *Rainbow: Combining improvements in deep reinforcement learning*. In AAAI Conference on Artificial Intelligence (AAAI). https://arxiv.org/abs/1710.02298

[5] Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). *Human-level control through deep reinforcement learning*. Nature. https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf

[6] Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). *Prioritized experience replay*. In International Conference on Learning Representations (ICLR). https://arxiv.org/abs/1511.05952

[7] Silver, D., Hubert, T., Schrittwieser, J., et al. (2018). *A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play*. Science. https://storage.googleapis.com/deepmind-media/DeepMind.com/Blog/alphazero-shedding-new-light-on-chess-shogi-and-go/alphazero_preprint.pdf

[8] Szepesvári, C. (2010). *Algorithms for reinforcement learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers. https://doi.org/10.2200/S00268ED1V01Y201005AIM009