Adaptive Strategy Optimization for 2048 Using Reinforcement Learning

J Yim*, Sidharth Srinivasan†, and Nikita Bhardwaj‡
Stanford University, Stanford, CA, 94305

The game 2048 is a single-player puzzle that combines randomness and strategic decision-making, presenting a compelling case for testing reinforcement learning (RL) methodologies. This project explores the development and evaluation of a reinforcement learning agent for solving 2048 using Q-learning with a neural network function approximator. Key contributions include the design of a dynamic reward function, the integration of double Q-learning for stability, and the creation of an interactive dashboard for visualization and experimentation. Results demonstrate the agent's ability to improve policy performance over time, though challenges related to reward sensitivity and state space complexity remain. Future work aims to refine reward structures, explore advanced exploration strategies, and scale training methodologies for broader applications.

I. Introduction

The game 2048 is an engaging and widely studied puzzle that challenges players to combine tiles of equal value on a 4x4 grid until achieving a target value of 2048 or higher. With its combination of stochastic tile generation and deterministic player actions, 2048 provides a unique testbed for reinforcement learning algorithms.

This project aims to develop a reinforcement learning agent capable of achieving high scores consistently in 2048. Using Q-learning as the foundation, we employ a neural network to approximate the Q-value function and integrate techniques such as double Q-learning to enhance stability. The project is guided by the following objectives:

- Design an effective reward structure that incentivizes strategic gameplay.
- Implement an interactive dashboard for real-time experimentation.
- Analyze the agent's performance under various training configurations.

II. Methodology

A. Reinforcement Learning Framework

The agent employs Q-learning, a model-free reinforcement learning algorithm, to optimize its action-selection policy. The Q-value function is approximated using a neural network, which takes the board state as input and outputs the Q-values for each of the four possible actions (up, down, left, right). Key algorithmic enhancements include:

- **Double Q-learning**: Reduces overestimation bias by maintaining two sets of Q-value estimations and alternating updates.
- **Epsilon-Greedy Exploration**: Balances exploration and exploitation by selecting random actions with probability ϵ.
- **Dynamic Reward Scaling**: Adapts rewards based on game progression to encourage long-term strategic planning.

*Undergraduate Student, Department of Computer Science, Stanford University.
†Undergraduate Student, Department of Computer Science, Stanford University.
‡Undergraduate Student, Department of Computer Science, Stanford University.

B. Neural Network Architecture

The neural network consists of:

- **Input Layer**: 16 nodes representing the flattened 4x4 grid.
- **Hidden Layers**: Two fully connected layers with 128 neurons each, using ReLU activation.
- **Output Layer**: Four nodes corresponding to the Q-values of the four possible actions.

C. Appropriateness of Approach

Q-learning with a neural network function approximator is a suitable approach to this problem space for the following reasons:

- **State-Action Space**: The 2048 grid is a high-dimensional state space; neural networks allow for efficient Q-value approximation within such spaces, while exact methods would be infeasible.
- **Deterministic Actions with Stochastic Outcomes**: Within 2048 gameplay, the player's actions are deterministic while tile generation is random – this aligns well with model-free RL methods, since they can learn optimal strategies without an explicit model.
- **Scalability**: The chosen architecture balances computational efficiency and performance, which makes it feasible for training the agent on large numbers of episodes.

III. Prior Work

The game 2048 has emerged as a challenging benchmark for testing reinforcement learning (RL) methodologies due to its unique combination of stochastic tile generation and deterministic player actions. Numerous prior studies have explored RL strategies for optimizing performance in 2048, with significant contributions addressing Q-learning, deep Q-networks (DQNs), reward engineering, and policy optimization techniques.

One of the foundational works in this space is by Szubert and Jaśkowski[1], who applied temporal difference learning with manually engineered features to optimize tile movements. Their agent achieved high scores but relied heavily on domain-specific heuristics, limiting its adaptability to other combinatorial optimization problems. Our work builds on this foundation by eliminating the need for feature engineering through the use of a neural network as a function approximator.

The introduction of deep reinforcement learning by Mnih et al.[2] was a watershed moment for RL research, particularly with the development of deep Q-networks (DQNs). DQNs demonstrated the ability to achieve superhuman performance in Atari games by using neural networks to approximate Q-values, paired with experience replay and target networks to stabilize training. These techniques have since been applied to 2048[3], highlighting their potential for handling large state-action spaces. However, overestimation bias inherent to Q-learning often led to suboptimal policies. To address this, Hasselt et al.[4] introduced double Q-learning, which decouples the action selection and evaluation processes. This approach reduces overestimation bias and has been incorporated into our work to ensure stable learning.

Reward engineering has been another critical area of focus in 2048 research. Babaeian et al.[5] emphasized the importance of aligning reward functions with intermediate gameplay objectives, such as maintaining empty tiles and prioritizing high-value tile placement. Their findings informed the milestone-based reward structure used in our intermediary implementation, which incentivizes strategic gameplay and long-term planning. Expanding on this idea, dynamic reward scaling has been proposed in hierarchical RL systems[6], where different reward functions are used to guide agents through various stages of a task. Inspired by this, we implemented a dynamic reward system tailored to the early, mid, and late stages of 2048 gameplay, focusing on board mobility, tile organization, and high-value tile placement, respectively.

Another notable approach is the use of convolutional neural networks (CNNs) to capture spatial relationships between tiles, as explored by Sahu et al.[7]. While CNN-based methods effectively exploit spatial dependencies, they introduce significant computational overhead. Our work opts for fully connected layers to maintain computational efficiency while preserving the ability to capture meaningful state representations. Similarly, Liang et al.[8] investigated the state-space reduction strategies for 2048 by leveraging tile symmetries and compressing input dimensions. Although effective, these methods impose additional preprocessing steps, which we avoided by training the neural network directly on the raw game board.

Policy gradient methods, such as those described by Silver et al.[9], have also been explored in the context of 2048. Policy optimization techniques, while theoretically promising, often struggle with the large action spaces and sparse rewards characteristic of 2048. In contrast, value-based methods like Q-learning have proven more effective, particularly when paired with strategies like double Q-learning and carefully crafted reward functions.

This project also draws inspiration from research on other grid-based strategy games and decision-making problems. For example, hierarchical approaches in robotics[10] and adaptive reward mechanisms in resource allocation tasks[11] provided a conceptual framework for designing dynamic rewards. These studies highlight the versatility of RL in solving diverse optimization problems and informed our efforts to generalize the methodology for 2048.

In summary, our work builds upon the advancements made in deep RL for games like 2048 by integrating double Q-learning, dynamic reward scaling, and efficient neural network architectures. By addressing gaps in prior research, such as reliance on handcrafted features and static reward structures, this project aims to advance the state-of-the-art in RL for complex decision-making tasks.

IV. Results and Analysis

A. Intermediary Step Results

Before implementing dynamic rewards, we evaluated the agent's performance with a refined static reward structure. This approach yielded improved results compared to the initial implementation. Key observations include:

- Enhanced convergence of training loss, reflecting a stable and consistent learning process.
- Significant improvement in average rewards over 400 episodes, demonstrating the effectiveness of milestone-based rewards.
- Achieved higher tile values, with the agent frequently reaching 1024.

B. Impact of Dynamic Rewards

Dynamic reward scaling was introduced to adapt the agent's decision-making strategy to different stages of gameplay, ensuring the reward system aligned with the unique challenges of early, mid, and late game phases. This approach aimed to enhance the agent's ability to prioritize actions that maximize long-term performance by adapting its focus as the game progressed.

- **Early Game**: The primary focus in the early game was on board mobility and maintaining as many empty tiles as possible. This encouraged the agent to create opportunities for future merges while minimizing the risk of reaching a terminal state prematurely. The reward structure heavily penalized actions that limited mobility or resulted in suboptimal tile placements.
- **Mid Game**: During the mid game, the emphasis shifted toward organizing tiles into patterns conducive to large merges. The agent was incentivized to create clusters of similar tiles and align high-value tiles in predictable formations. This phase required the agent to balance short-term merges with the need to set up future opportunities.

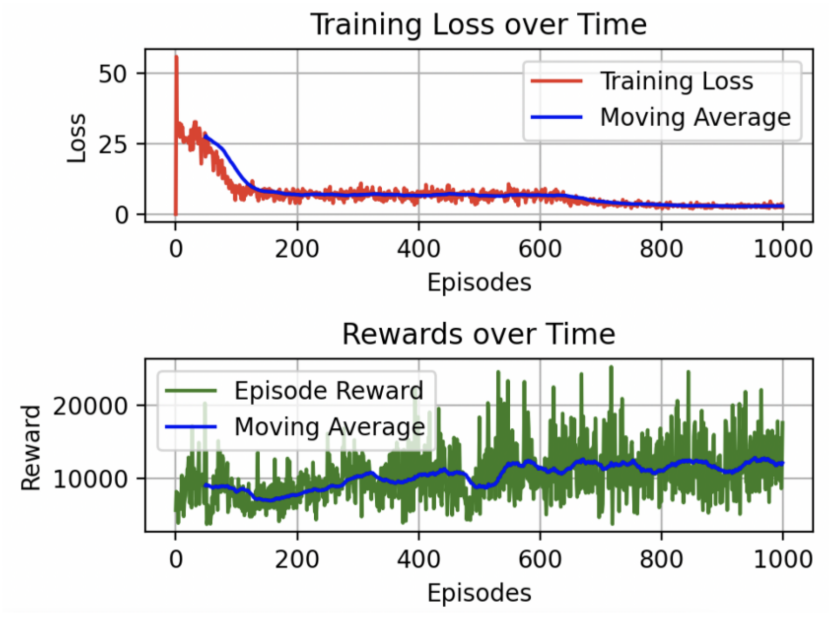


Fig. 1 Training Loss and Rewards During the Intermediary Phase (Static Rewards).

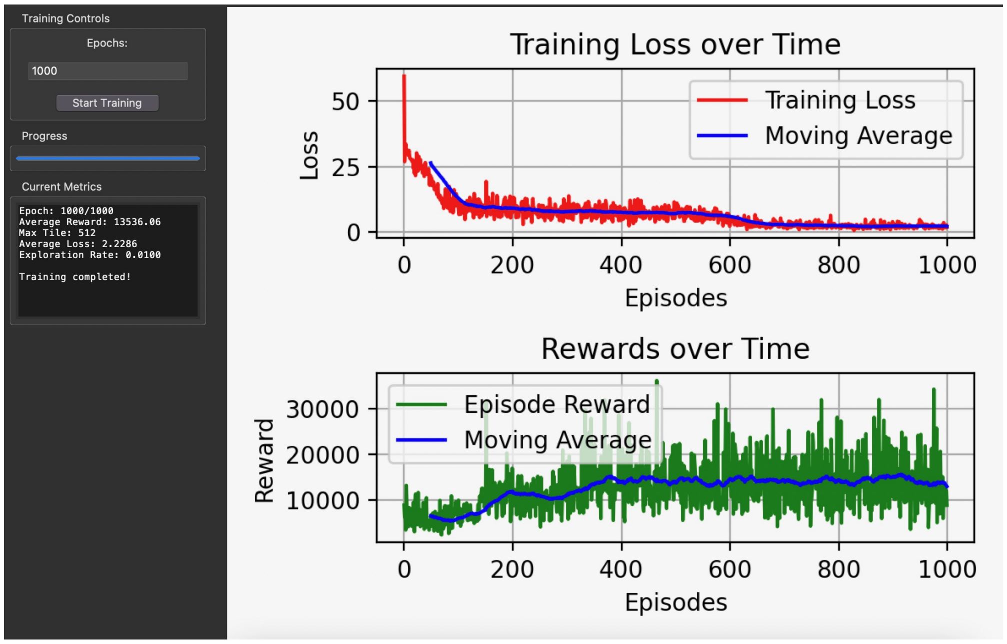
- **Late Game**: In the late game, the reward structure prioritized the consolidation of high-value tiles, particularly through the "snake pattern," where tiles are organized in a descending or ascending sequence around the grid's perimeter. This configuration maximized the potential for high-value merges while preserving board stability. Actions that disrupted this pattern or left critical tiles unaligned were penalized.

While this dynamic reward structure was conceptually sound and added nuance to the agent's strategy, experimentation revealed several trade-offs:

- **Performance Variability**: The dynamic reward system introduced additional complexity, resulting in greater variability in performance. The agent occasionally struggled to transition effectively between game stages, leading to suboptimal decisions in critical moments.
- **Over-Engineering Concerns**: The added complexity of the reward system sometimes caused the agent to over-prioritize specific patterns or objectives, sacrificing broader adaptability. For example, in some cases, the focus on late-game snake patterns led to neglect of simpler but effective mid-game opportunities.
- **Convergence Challenges**: The increased variability in rewards made policy convergence more challenging, with the agent requiring more training episodes to stabilize its performance compared to simpler, static reward structures.

Insights and Conclusion on Dynamic Rewards

Dynamic reward scaling provided valuable insights into the potential of tailored reward systems for guiding reinforcement learning agents through complex, multi-phase tasks. However, the results suggest that over-engineering reward structures may lead to diminishing returns, with simpler, static rewards often outperforming more complex designs in terms of robustness and convergence efficiency. Future work will aim to refine the balance between simplicity and adaptability in reward engineering.



Epochs: 1000

```
Epoch: 1000/1000
Average Reward: 13536.06
Max Tile: 512
Average Loss: 2.2286
Exploration Rate: 0.0100

Training completed!
```

Fig. 2 Training Loss and Rewards Over Time (Dynamic Rewards). The variability in rewards reflects the agent's exploration of dynamic strategies, with convergence taking longer than during static reward experimentation.

V. Discussion

A. Challenges and Insights

Key challenges encountered during the project include:

- **Reward Sensitivity**: Fine-tuning the reward structure had significant, often unpredictable impacts on performance.
- **State Space Complexity**: The exponential growth of possible game states required efficient network optimization techniques.
- **Exploration-Exploitation Trade-off**: Epsilon-greedy parameters required careful tuning to avoid premature convergence to suboptimal policies.

An important insight is that over-engineering the reward function can lead to diminishing returns, as seen with dynamic scaling. A simpler, static reward structure often provided more robust performance.

VI. Contributions

The project contributions were divided as follows:

- J Yim: Coordinated the project, aided in literature review, and led the documentation effort.
- Sidharth Srinivasan: Focused on algorithm implementation and dynamic reward design.
- Nikita Bhardwaj: Lead the literature review and handled results analysis.

VII. Future Work

Future directions for this project include:

- **Advanced Exploration Strategies**: Explore techniques like Upper Confidence Bound (UCB) for improved exploration.
- **Transfer Learning**: Apply the trained agent to variants of 2048 or other grid-based games.
- **Scalable Training**: Implement distributed training to handle the large state-action space more efficiently.

VIII. Conclusion

This project demonstrates the efficacy of reinforcement learning in solving complex decision-making tasks like 2048. While dynamic rewards introduced interesting new directions, the results suggest that simpler, static reward structures may often outperform more complex designs. These findings provide a foundation for further exploration and optimization of RL techniques for stochastic, strategic games.

Acknowledgments

The authors extend their gratitude to the AA228/CS238 teaching team for their guidance and the open-source community for tools such as PyTorch and Matplotlib.

References

[1] Szubert, M., and Jaśkowski, W., "Temporal difference learning of n-tuple networks for the game 2048," *2014 IEEE Conference on Computational Intelligence and Games*, 2014, pp. 1–8.

[2] Mnih, V., et al., "Human-level control through deep reinforcement learning," *Nature*, Vol. 518, No. 7540, 2015, pp. 529–533.

[3] Van der Ouderaa, T., et al., "Deep reinforcement learning in 2048," *Proceedings of the Computational Intelligence in Games Conference*, IEEE, 2016.

[4] van Hasselt, H., et al., "Deep reinforcement learning with double Q-learning," *AAAI Conference on Artificial Intelligence*, 2016, pp. 2094–2100.

[5] Babaeian, M., et al., "Reward shaping for 2048 using intermediate goals," *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[6] Kulkarni, T., et al., "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," *Advances in Neural Information Processing Systems*, 2016, pp. 3675–3683.

[7] Sahu, R., et al., "Deep reinforcement learning for 2048 with convolutional networks," *Workshop on Deep Learning for Games*, 2018.

[8] Liang, J., et al., "State-space reduction for 2048 using tile symmetries," *IEEE Symposium on Foundations of Computational Intelligence*, 2018, pp. 182–190.

[9] Silver, D., et al., "Mastering the game of Go with deep neural networks and tree search," *Nature*, Vol. 529, 2016, pp. 484–489.

[10] Levine, S., et al., "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, Vol. 17, No. 39, 2016, pp. 1–40.

[11] Shao, J., et al., "Adaptive reward mechanisms in reinforcement learning for resource allocation," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 32, No. 5, 2021, pp. 1803–1815.