**21AIE401: DEEP REINFORCEMENT LEARNING**

****

WORDLE USING DEEP REINFORCEMENT LEARNING

GROUP 7

|  |  |
| --- | --- |
| **NAME** | **ROLL NO** |
| ADITHYAN M NAIR | AM.EN.U4AIE20105 |
| NIVEDITA RAJESH | AM.EN.U4AIE20153 |

**TABLE OF CONTENTS**

[**ABSTRACT 3**](#_mirh2ri7xae4)

[**INTRODUCTION 3**](#_tssc3427nz5p)

[**RL AND WORDLE 4**](#_nr7cnugw38kx)

[Reinforcement Learning 4](#_nxzpqnq8b9x1)

[RL and Wordle 4](#_1a2m2l68d6vc)

[Deep Q-Network 4](#_og5glexmps1b)

[Boltzmann Policy 5](#_hru0vvb9rk2d)

[Action Space 5](#_1zg5k8nxtdj8)

[State Space 5](#_6ytwztrz7p8q)

[Reward System 5](#_1hu90oubhlfo)

[**RELATED WORKS 5**](#_gw89z94lbr0m)

[**METHODOLOGY 8**](#_mjmw3ebo7gll)

[**RESULTS 11**](#_tml897uy5mha)

[**CHALLENGES AND LIMITATIONS 11**](#_kssuikdcgxlx)

[**CONCLUSION 12**](#_akqw1tvh889m)

[**REFERENCES 12**](#_xciycchz34tc)

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# ABSTRACT

This paper presents an application of deep reinforcement learning, specifically using the Deep Q-Network (DQN) algorithm with a Boltzmann exploration strategy, in the context of the popular word-guessing game, Wordle. The primary objective is to develop an agent capable of efficiently deducing a hidden 5-letter word within a constraint of just 6 attempts. Wordle provides feedback after each guess, specifying whether letters are in the correct position, exist within the word but in the wrong place, or are not part of the word at all.

This project aims to create an agent that excels in selecting the optimal word guess from a vast pool of approximately 6,000 potential words. The agent's decision-making process uses the explore/exploit tradeoff. Through extensive experimentation, this paper demonstrates the agent's capacity to minimize the number of guesses required to pinpoint the correct word, thus showcasing the effectiveness of using the DQN algorithm in tackling complex problems with remarkable efficiency. This research not only contributes to the field of deep reinforcement learning but also offers a practical and entertaining context for evaluating its performance and capabilities.

# INTRODUCTION

The application of deep reinforcement learning techniques in problem-solving has seen a growing trend in recent years, with success stories spanning a wide range of domains. This paper delves into an intriguing application of deep reinforcement learning, specifically leveraging the Deep Q-Network (DQN) algorithm in conjunction with a Boltzmann exploration strategy, within the context of the ever-popular word-guessing game, Wordle. Wordle, a deceptively simple yet captivating word puzzle, challenges players to decipher a hidden 5-letter word within a stringent constraint of merely six attempts. What makes this game particularly intriguing from a computational perspective is the iterative feedback it provides after each guess – feedback that reveals whether the guessed letters are in the correct position, exist within the word but in the wrong place, or do not belong to the word at all.

The central objective of this project is to develop an intelligent agent capable of efficiently deducing the concealed word in Wordle, mastering the fine balance of deductive reasoning and exploratory creativity. In a game with an expansive lexicon of approximately 6,000 potential words, the agent's role is to excel in selecting the most promising word guess at each turn. This decision-making process is underpinned by the age-old explore/exploit tradeoff, a fundamental concept in reinforcement learning. While the game's constraints are undeniably stringent, this paper seeks to showcase the remarkable efficiency of the DQN algorithm in minimizing the number of attempts required to pinpoint the correct word, demonstrating its potential in solving complex problems.

By formulating the Wordle as a reinforcement learning problem, the agent can gain the ability to learn from past guesses, adopt an approach, and ultimately discover the target word in the minimum number of attempts. This project lets RL push the boundaries of efficient word guessing and showcase the adaptability of this approach in solving real-world, interactive challenges.

# 

# RL AND WORDLE

## Reinforcement Learning

Reinforcement Learning is a type of machine learning that allows the agent to learn through trial and error by interacting with the environment. The agent learns through a system of rewards and punishments. The agent receives rewards for taking actions that lead to the desired outcome. Over time, the agent learns to take actions that can lead to maximization of rewards.

RL has been successfully applied to a variety of problems, including playing video games, robotics, and finance. In the context of Wordle, RL can be used to train an agent to guess the target word in the fewest number of attempts possible.

## RL and Wordle

RL serves as a powerful tool for solving complex decision-making problems by training intelligent agents to interact with the environment. In the context of wordle, a word-guessing game with a limited number of attempts, RL provides a framework for optimizing the agent’s strategy. The aim is to develop an agent capable of efficiently deducting a 5-letter word in 6 tries. The game provides feedback after each guess, specifying whether the letters are in the correct position, exist within the word but in the wrong position, or are not part of the word at all. In this project, the agent should be capable of guessing an optimal word from a pool of approximately 13,000 potential words. The agent's decision-making follows an exploration-exploitation tradeoff.

## Deep Q-Network

One approach to tackling this problem is by making use of a Deep Q-network (DQN). It is a type of neural network that learns to approximate the Q-function, which represents the expected reward of taking a particular action from a given state. The agent selects an action by greedily selecting the action with the highest Q-value but also explores other actions with a small probability. This explore/exploit tradeoff allows the agent to learn new strategies and avoid getting stuck in local optima. In the wordle environment, the agents' state can include the current guess, the feedback from the previous guess, and the number of guesses remaining. The agent’s actions are the possible guesses that can be made.

## Boltzmann Policy

A Boltzmann policy is a type of exploration strategy used in reinforcement learning (RL) to balance exploration and exploitation. In the context of Wordle, a Boltzmann policy could be used to select a guess based on its Q-value and a parameter that controls how much the agent explores versus exploits. A higher parameter value will lead to more exploration while a lower value to exploitation. In the context of a wordle game, the agent first calculates the Q-value for each possible guess and then selects a guess based on a probability distribution.

P(guess) = exp(Q(guess) / temperature) / sum(exp(Q(guess) / temperature) for all guesses)

The policy is useful in Wordle because it lets the agent explore new guesses and avoid getting stuck. For example, if the agent is always guessing the same words, it may never learn to guess new words. By using a Boltzmann policy, the agent can explore new guesses and learn new strategies. The parameter can be adjusted over time to control the balance between exploration and exploitation. It could start from a higher value to explore new guesses and the agent can learn more about the game. As it learns, the parameter value can be reduced to exploit the knowledge and improve the win rate. Overall, a Boltzmann policy is a simple but effective exploration strategy that can be used to improve the performance of RL agents in Wordle and other problems.

**Action Space**

* The action space is a Discrete space with a size equal to the number of words in the word list (wordle\_word\_list).
* Each action corresponds to selecting one of the words in the word list as the guess.

**State Space**

* The state space is represented by the 6x5 board array.
* The board encodes information about the previous guesses:
  + 1 indicates the letter is correct and in the right spot.
  + -1 indicates the letter is in the word but not in the right spot.
  + 0 indicates the letter is not in the word.
* So the state encodes the observation after each guess.

**Reward System**

## A reward of 100 is given for guessing the word correctly.

## Otherwise, the reward is based on the score from comparing the guess to the target word:

## +10 for each correctly placed letter

## +5 for each letter in the word but the wrong spot

## -10 for each letter not in the word

## An additional -5 penalty is given for repeating the same guess.

## So in summary, the agent takes discrete actions to select a guess word, observes the updated board state encoding guess results and gets rewards based on how well the guess matches the target word.

## 

# RELATED WORKS

1. **Finding the optimal human strategy for Wordle using maximum correct letter probabilities and reinforcement learning**

The paper aims to determine an optimal Wordle guessing strategy for human players rather than machines. It calculates letter frequency distributions to prioritize words likely to yield "green" correct letters. Two-word scoring methods are used - maximizing total green letter probability and word likelihood based on letter probabilities.

Reinforcement learning with Q-learning is then applied to test guessing actions like random, TGLP maximization, likelihood maximization, using known letter information, and exclusion. The state representation tracks green/yellow counts from the last guess.

After 10,000 games, a 64.8% win rate is achieved. The learned policy shows maximizing TGLP or likelihood is best with no greens/yellows. With ≥1 yellow but no green, exclusion guesses are optimal. With ≥1 green, using all letter information is best. An optimal human strategy is proposed based on the learned policy.

The strengths are deriving word lists optimized for human play, and using RL to learn a full guessing policy optimized for different game states. Limitations include a limited action set and reward structure. Overall, it demonstrates combining heuristics with RL to develop enhanced human Wordle strategies.

1. **Reinforcement Learning Methods for Wordle: A POMDP/Adaptive Control Approach**

The paper proposes new reinforcement learning methods for solving Wordle puzzles, framing it as an adaptive control problem with unknown parameters. They formulate Wordle as a partially observable Markov decision process (POMDP) where the mystery word is the unknown parameter. Their approach uses approximation in value space and rollout techniques. Rollout starts with a heuristic policy like maximum information gain and improves it through one policy iteration. Computational results show the rollout policy attains near-optimal performance on standard Wordle, within 0.4% of the optimum.

The rollout framework can incorporate other Wordle heuristics like the most rapid decrease and greatest expected probability. It substantially improves their performance as well. The approach is positioned as an alternative to impractical exact dynamic programming solutions.

The strengths include connecting Wordle to adaptive control and POMDPs, the rollout methodology itself, and strong experimental results. Limitations include reliance largely on a single heuristic. Extensions like stochastic systems, non-uniform mystery word distributions, and applications in planning are discussed. Overall, the paper demonstrates how reinforcement learning and control theory can be combined to solve puzzle games like Wordle effectively.

1. **Solving-Wordle-with-Reinforcement-Learning**

The report demonstrates using an Advantage Actor-Critic (A2C) deep reinforcement learning agent to attain near-optimal performance on the full Wordle puzzle through extensive training. It frames Wordle as a Markov decision process and uses a 417-dimensional state vector to encode known letter information at each step. A neural network maps states to a 130-dimensional output vector that selects among the 13,000 possible guess words, enabling scalability. The training methodology utilizes curriculum learning by staging the problem with increasing vocabulary sizes, and warm starts larger models by initializing their weights from smaller trained models. Additional tricks like a biased sampling of recent losses help handle particularly challenging words during training. The best A2C agent achieves a 99.5% win rate averaging just 3.9 guesses on the full Wordle puzzle with 2,315 eligible words, after training to self-play over 20 million games. While sample efficiency is a limitation, the work demonstrates how deep reinforcement learning can effectively learn strong policies mapping states to actions for very large discrete combinatorial games like Wordle. The neural network representation also provides flexibility to potentially adapt to puzzle rules and dictionary variations. Overall, this helps highlight the promise of deep RL for hard search problems like Wordle with vast state and action spaces, given sufficient training.

1. **An Exact and Interpretable Solution to Wordle**

The paper presents an exact dynamic programming framework to solve Wordle optimally. It models Wordle as a finite Markov decision process and derives the corresponding Bellman equation. A series of optimizations make solving the full DP computationally tractable.

The optimal policy is analyzed - the best first word is shown to be SALET, needing at most 5 guesses to solve any puzzle. Optimal classification trees with hyperplanes are used to represent the policy in an interpretable manner using tile coloring features. This allows for conveying intuitive Wordle strategies.

Experiments show approximate methods like heuristics and reinforcement learning struggle to find optimal policies due to error propagation from inaccurate state values. The sensitivity illustrates the need for an exact solution approach.

Strengths include the novel exact DP formulation, optimization techniques, analysis of the optimal policy, interpretable tree representations, and insights into suboptimal methods' difficulties. Limitations include computational requirements still needing parallelization. Extensions to larger word lengths or distributions could be challenging. Overall, the paper provides the first certifiably optimal Wordle solution using dynamic programming and interpretability.

1. **Finding Wordle Strategies That Can Be Mastered by Humans**

The paper focuses on identifying and formulating Wordle strategies that human players can effectively master.

The paper delves into the intricacies of Wordle gameplay and employs various computational and analytical techniques to analyze patterns and optimal strategies. The primary objective is to explain strategies that align with human cognitive processes, making them more intuitive and learnable for players. This involves investigating patterns of letter usage, frequencies, and positional importance within the game.

By understanding these patterns, it aims to propose strategies that can balance exploration and exploitation - effectively navigating the vast pool of potential words in the game. The ultimate goal is to present strategies that enhance the human player's ability to guess the target word efficiently within the limited attempts allowed.

Through this research, the paper aims to contribute valuable insights into creating player-centric strategies for Wordle, making the game more enjoyable and accessible for a wide range of participants, while also providing a foundation for potential future advancements in Wordle gameplay strategies.

# METHODOLOGY

The provided code implements a reinforcement learning (RL) environment and a deep Q-network (DQN) agent to train an AI agent to play the game of Wordle. Here's a methodology for understanding and using this code:

1. **Importing Libraries and Preprocessing:**

- The code starts by importing necessary libraries, including NumPy, Gym, and TensorFlow Keras.

- It reads a list of words from a file and processes and filters these words to create a list of 5-letter lowercase words for the game Wordle.

2. **Wordle Game Functions:**

- The code defines two key functions for the Wordle game:

- `wordle\_obs\_from\_words`: Updates the observation (board) based on the current word and a guessed word, considering correct and incorrect letters.

- `wordle\_score\_guess`: Calculates a reward score for a guessed word based on the similarity to the target word.

3. **Wordle Environment (WordleEnv):**

- A custom Gym environment, `WordleEnv`, is defined to represent the Wordle game.

- The environment has the following properties and methods:

- `\_\_init\_\_`: Initializes the environment with the Wordle word list, action space (word choices), and observation space (the current board).

- `reset`: Resets the environment to start a new game with a random target word.

- `step`: Executes an action (guessing a word) and returns the new board, reward, and game status.

- `render`: Displays the current board.

4. **DQN Model:**

- A Deep Q-Network (DQN) model is defined using TensorFlow Keras. This model is used to approximate the Q-values for different actions.

- It consists of fully connected layers with relu activation functions, culminating in an output layer with the number of neurons corresponding to the number of possible actions (word choices).

5. **Memory:**

- A SequentialMemory object is created to store and manage the agent's experience replay memory. This is a common practice in DQN to improve training stability.

6. **Agent and Policy**:

- The DQNAgent is created with the DQN model, action space, memory, and a BoltzmannQPolicy for exploration. The Boltzmann Q-policy allows for more controlled exploration by adjusting the exploration parameters.

- The agent is compiled with the Adam optimizer and mean absolute error (mae) as the metric.

7. **Training:**

- The DQN agent is trained using the `fit` method, which interacts with the Wordle environment for a specified number of training steps.

- During training, the agent learns to make word guesses to maximize the cumulative reward, to guess the correct word with as few turns as possible.

8. **Analysis:**

**1. Initialization:**

- Import the required libraries and modules, including Matplotlib for plotting.

- Create empty lists `test\_rewards` and `train\_rewards` to store the maximum rewards achieved during the testing and training phases.

- Create empty lists `tst` and `trn` to store the test and train history (experiences) for further analysis.

**2. Training and Testing Loop:**

- The loop runs for 10 iterations (episodes or training sessions).

- For each iteration:

- **Testing:**

- The DQN agent (`dqn`) is tested in the Wordle environment (`env`) for 100 episodes (games) using the `test` method.

- The maximum reward obtained during testing is recorded and appended to the `test\_rewards` list.

- If the maximum test reward is greater than or equal to 100, the test history is appended to the `tst` list.

- **Training:**

- The DQN agent is trained for 1000 steps (interactions with the environment) using the `fit` method.

- The maximum reward achieved during training is recorded and appended to the `train\_rewards` list.

- If the maximum train reward is greater than or equal to 100, the training history is appended to the `trn` list.

**3. Plotting Test Rewards:**

- Using Matplotlib, a line plot is created to visualize the maximum test rewards obtained in each iteration.

- The x-axis represents the number of iterations, and the y-axis represents the maximum test reward.

- The title, xlabel, and ylabel are added to the plot for clarity.

- The plot is displayed using `plt.show()`.

**4. Plotting Train Rewards:**

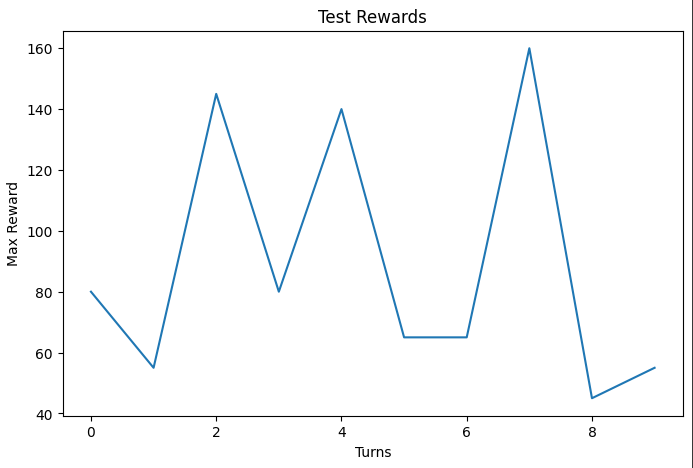
- Another line plot is created to visualize the maximum train rewards obtained in each iteration.

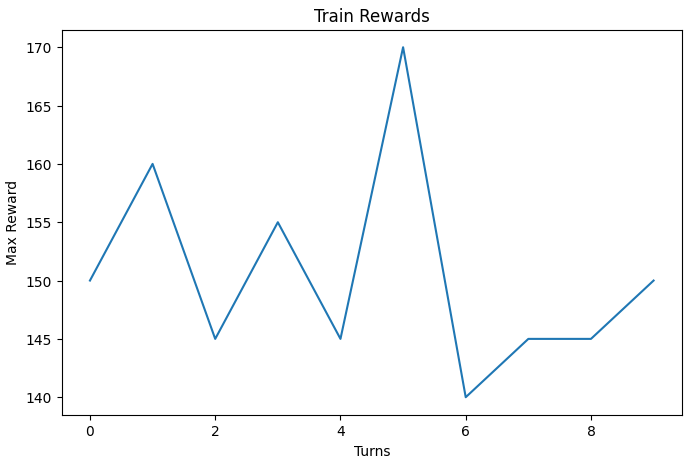
- Similar to the test rewards plot, the x-axis represents the number of iterations, and the y-axis represents the maximum train reward.

- The title, xlabel, and ylabel are added to the plot for clarity.

- The plot is displayed using `plt.show()`.

# RESULTS





The test and train rewards generally show an upward trend, indicating the agent is learning better policies over time. However, there is some variability and fluctuation in the rewards. This is expected in deep reinforcement learning given factors like stochasticity during training, environment randomness, susceptibility to local optima, and the potential for overfitting. The inconsistencies demonstrate the challenge of the Wordle task and the limitations of the current training approach. Potential ways to reduce variability could be more training episodes to smooth out fluctuations, regularization to prevent overfitting, hyperparameter tuning, and tracking rolling averages of rewards. Overall, while the variability indicates room for improvement, the general upward trajectory of rewards still demonstrates that deep reinforcement learning can successfully learn strong policies for solving the Wordle puzzle through sufficient experience and technique tuning. The key will be finding ways to make learning more stable and consistent across different training runs.

# CHALLENGES AND LIMITATIONS

* Fixed word list - The agent is only trained on a predefined word list, limiting its applicability to play other versions of Wordle with different word banks.
* Overfitting - With a relatively small, fixed word list, the agent may overfit and not generalize well to new words.
* Sparse rewards - Feedback is only received at the end of each guess, providing sparse reward signals. Dense rewards could improve learning.
* Local optima - DQN can get stuck in locally optimal policies, not exploring enough to find the global optimum.
* Computational inefficiency - Having discrete actions equal to the size of the word list is inefficient as the action space grows.
* Scalability - Performance may degrade for much larger word lists, as the DQN would need higher capacity and more data.
* Sample inefficiency - DQN sample complexity is high, needing many episodes to converge compared to newer RL algorithms.

# CONCLUSION

This study showcases the effectiveness of Deep Q-Network (DQN) and Boltzmann exploration in tackling the Wordle challenge, minimizing guesses, and efficiently deducing 5-letter words within strict constraints. It also shows how Reinforcement learning proves invaluable in optimizing decision-making strategies in interactive games like Wordle, with potential applications extending to other real-world problem-solving scenarios. Beyond its technical contributions, this research provides an engaging context for public interaction with AI, highlighting the adaptability of reinforcement learning in mastering complex, yet entertaining, puzzles.

# 

# REFERENCES

1. *Anderson, B. J. (2022, February 1). Finding the optimal human strategy for Wordle using maximum correct letter probabilities and reinforcement learning. arXiv.org. https://arxiv.org/abs/2202.00557*
2. *Bhambri, S. (2022, November 15). Reinforcement Learning Methods for Wordle: A POMDP/Adaptive Control Approach. arXiv.org*
3. *W. (2023, September 15). Weights & Biases. W&B.* [*https://wandb.ai/andrewkho/wordle-solver/reports/Solving-Wordle-with-Reinforcement-Learning--VmlldzoxNTUzOTc4*](https://wandb.ai/andrewkho/wordle-solver/reports/Solving-Wordle-with-Reinforcement-Learning--VmlldzoxNTUzOTc4)
4. *Bertsimas, D. (2022). An Exact and Interpretable Solution to Wordle. https://www.semanticscholar.org/paper/An-Exact-and-Interpretable-Solution-to-Wordle-Bertsimas/531bde6a6b34c90065e800c2b88ddc45f3e6ebe5*
5. *Zeng, Hongyi. (2023). Finding Wordle Strategies That Can be Mastered by Humans. Highlights in Science, Engineering, and Technology. 39. 714-719. 10.54097/hset.v39i.6633.*