Exploring Reinforcement Learning and Information Theory for Alphalock

Dinan Elsyad
Harvard Statistics Department
Harvard College
dinanelsyad@college.harvard.edu

Berke Altiparmak
Harvard Faculty of Arts and Sciences
Harvard University
berkealtiparmak@fas.harvard.edu

Abstract

Wordle has inspired research into optimizing gameplay strategies using information theory and reinforcement learning (RL). This study extends these methods to a more challenging variant, Alphalock, where feedback provides counts of correct, misplaced, and incorrect tiles without positional information. We propose two distinct solvers: an Information Theory-based (IT) solver, which selects guesses by maximizing entropy to reduce uncertainty, and a human-like Reinforcement Learning (RL) solver, which dynamically balances exploration and exploitation through a learned policy. Our findings highlight the trade-offs between deterministic heuristics and learned policies, providing insights into computational approaches for solving constrained feedback-based games.

1 Introduction

Wordle has gained widespread popularity as a word-guessing game where players attempt to deduce a 5-letter word in six tries using feedback on guessed letters. The game requires a balance between *exploration*, which involves testing new possibilities to gather information, and *exploitation*, which leverages known information to narrow down solutions. Inspired by Wordle, we became curious about whether reinforcement learning (RL) could be used to mimic human decision-making and reduce the average number of guesses required to win.

While research shows that RL has been successfully applied to Wordle, a related game called Alphalock introduces a more challenging problem. In Alphalock, players only receive counts of correct (green), misplaced (yellow), and incorrect (gray) tiles, without knowing which letters correspond to these colors. Players have 10 attempts to guess a 4-letter word. Since no prior research has explored RL for Alphalock, our study aims to fill this gap by comparing two models: a gold standard information theory-based approach and a human-like reinforcement learning model.

2 Literature Review

Alphalock, a lesser-known counterpart to Wordle, lacks existing reinforcement learning (RL) research, prompting us to draw insights from Wordle studies. Wordle optimization research has explored various computational methods, revealing their strengths and limitations. A video on Wordle highlighted entropy as a powerful tool for decision-making, using it to evaluate guesses based on expected information gain. [1] Early entropy-based models achieved 4.12 guesses per game, later refined to 3.43 guesses by incorporating word frequency and two-step forward searches. [1] As the information theory approach remains a gold-standard in Wordle, this has has motivated us to use an information theory model as a baseline for Alphalock strategies.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Another approach used Deep Reinforcement Learning with the Advantage Actor-Critic (A2C) algorithm. [3] This method achieved a remarkable 99.5% win rate and an average of 3.9 guesses per game. The model leveraged a staged training process, starting with small vocabularies (e.g., 100 words) before scaling to the full Wordle dictionary of approximately 13,000 words. To optimize efficiency, the researchers reduced the action space by encoding words as compact representations. Despite its success, the computational costs were immense, requiring millions of games to train effectively. Furthermore, its strategy was impractical as a model for human behavior [3].

Other studies focused on RL approaches reflecting human strategies but with lower success rates. A Q-learning model integrating letter probabilities achieved a 64.8% win rate, balancing information gain and error minimization. [2] Another Q-learning approach used strategies like maximizing green/yellow letters and dynamically adapting guesses to game states, achieving an 87.8% win rate. [6] However, these models struggled with rare scenarios, such as states with multiple green letters but no yellows.

Despite their limitations, RL methods offer valuable insights into balancing exploration and exploitation, guiding human decision-making. By leveraging these findings, we aim to develop interpretable strategies that improve gameplay for both Wordle and Alphalock. Inspired by these approaches, we developed a method of our own.

Our approach diverges from these methods by introducing a continuous action space defined by two weights, α and β, which dynamically balance exploration and exploitation. Unlike traditional RL algorithms such as Q-learning or actor-critic models, we utilize Vanilla Policy Gradient (REINFORCE) with episodic updates tailored for Alphalock's unique challenges. Although the application of these methods are fairly simple, we make our project more challenging by manually implementing the REINFORCE algorithm by scratch. The use of REINFORCE enables us to prioritize adaptability and robustness, allowing the agent to optimize its strategy for solving constrained feedback-based tasks. Our model focuses on learning a policy that dynamically adjusts the weights α and β, which are used to optimize the scoring function $\text{Score} = \alpha \cdot \text{IT} + \beta \cdot \text{RWF}$, guiding the agent toward successful guessing strategies. This approach maintains simplicity and interpretability while achieving competitive performance, setting it apart from computationally intensive heuristics or models traditionally used for Wordle.

3 Data

The datasets used in this project include the *Google Books NGram Dataset* (`ngram_freq.csv`) and the *English Word Dataset* (`words_alpha.txt`), each serving distinct purposes.

The *Google Books NGram Dataset*, derived from the Google Books Ngram Viewer [4], provides word usage frequencies from books published between the 1800s and 2019. With over 9 million unique entries, it includes 1-gram counts but also contains occasional errors from digitization. This dataset emphasizes real-world usage and historical context.

The *English Word Dataset* is a curated list of over 466,000 valid English words [5], focusing on alphabetical entries without providing frequency data. It serves as a comprehensive lexical reference and is widely used in applications like autocomplete and spell checkers.

Together, these datasets complement each other by providing both frequency-based context and a robust dictionary of valid English words, enabling effective linguistic analysis for the project.

We implemented a script to combine these datasets to create two dictionaries: `4letter_word_freqs.json`, which maps 4-letter words to their absolute usage frequencies, and `4letter_word_freqs_relative.json`, which maps these words to their relative frequencies.

4 Reinforcement Learning Outline

4.1 Problem Formulation

The goal of reinforcement learning (RL) in this context is to dynamically balance two key components in a word-guessing game:

1. **Information Theory (IT)**: Measures potential information gain through entropy, prioritizing guesses that most reduce the pool of possible solutions.
2. **Relative Word Frequency (RWF)**: Encourages selecting common words based on their likelihood of being the solution within the current pool.

The challenge lies in determining optimal weights for IT (exploration) and RWF (exploitation) as the game progresses, transitioning from exploration in the early stages to exploitation as the pool narrows. This balance of exploration and exploitation mimics the human approach to these games. By prioritizing frequent words in the later stage of the game, the solver increases its chances of faster success, leveraging linguistic biases inherent in word-based games.

4.2 RL Framework

4.2.1 State Representation

The environment provides the following state features to guide the reinforcement learning (RL) agent:

- **pool_entropy:** Represents the size of the remaining solution pool, serving as a measure of uncertainty. Smaller values indicate a more narrowed-down set of possible solutions.
- **attempts_remaining:** The number of guesses left, which provides a temporal constraint on the agent's decision-making process.
- **feedback_history:** Encodes patterns of feedback from previous guesses, summarized as counts, to track progress and adapt strategies accordingly.

4.2.2 Action Space

The RL agent predicts two continuous values that determine its strategy:

- α**:** A weight assigned to entropy-based scoring, prioritizing exploration by selecting guesses that maximize information gain.
- β**:** A weight assigned to relative word frequency scoring, emphasizing exploitation by favoring more likely solutions based on word frequency data.

These weights combine to form the scoring function used by the agent:

$$\text{Score} = \alpha \cdot \text{IT} + \beta \cdot \text{RWF},$$

where IT represents information-theoretic measures, and RWF reflects relative word frequency.

4.2.3 Reward Function

The reward function incentivizes the agent to guess the solution efficiently while balancing exploration and exploitation:

- **Success Reward:** When the correct solution is guessed, the agent receives a reward proportional to the number of guesses remaining, encouraging faster solutions:

$$R_{\text{success}} = \text{success\_reward} \cdot \frac{\text{attempts\_remaining}}{\text{max\_attempts}}.$$

- **Failure Penalty:** If the agent exhausts all guesses without finding the solution, a fixed negative reward is applied:

$$R_{\text{failure}} = \text{failure\_penalty}.$$

- **Intermediate Rewards:** For each guess, the agent receives a partial reward proportional to the reduction in the size of the solution pool:

$$R_{\text{intermediate}} = \text{intermediate\_scaling} \cdot \frac{\text{pool\_size\_before} - \text{pool\_size\_after}}{\text{pool\_size\_before}},$$

where *pool_size_before* and *pool_size_after* denote the solution pool size before and after the guess, respectively.

This reward structure balances short-term progress (via intermediate rewards) with the long-term goal of guessing the solution efficiently. We set the parameters as success_reward = 100, intermediate_scaling = 10, and failure_penalty = -100, yet our approach makes the reward function customizable.

4.3 Key Properties

Our RL framework exhibits the following unique properties:

- By using continuous weights (α and β), the agent dynamically adjusts its priorities between exploration and exploitation at each step based on the guesses remaining, the current pool size, and the history of the feedbacks on the previous guesses.
- Early in the game (t small), the model lets α_t to dominate, encouraging exploration. As t increases, β_t dominates, focusing on exploitation. This method mimics the human approach, as the game progresses words that are expected to be the solution are preferred over the words that would narrow the pool down more to get the answer faster.
- The model does not employ a value function, simplifying the architecture. Instead, it directly optimizes the agent's policy using the REINFORCE algorithm.
- The agent's scoring function incorporates entropy, enabling it to intelligently explore high-uncertainty guesses.
- The use of intermediate rewards ensures granular feedback for pool reduction, accelerating learning and improving robustness.

4.4 Policy and Algorithm

The RL agent leverages the Vanilla Policy Gradient (REINFORCE) algorithm with episodic updates to optimize its policy:

- **Policy Network:** The policy network is a fully connected neural network with two hidden layers. It takes the state features (*pool_entropy, attempts_remaining, feedback_history*) as input and outputs two continuous values: α and β.
- **Policy Optimization:** The policy network is updated to maximize the expected cumulative reward:
$$J(\theta) = \mathbb{E}_\pi[R_t],$$
where R_t is the discounted cumulative reward for episode t.
- **Gradient Computation:** The REINFORCE algorithm computes gradients as:
$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[\nabla_\theta \log \pi_\theta(a_t|s_t) R_t\right].$$
- **Episodic Updates:** After each batch of episodes, the policy network is updated using the collected rewards and log probabilities.

By dynamically adjusting exploration and exploitation weights based on state features, the policy enables the agent to solve the Alphalock game efficiently while maintaining simplicity and interpretability.

5 Project Scope and Implementation

The scope of our project involves developing a reinforcement learning-based solution to optimize gameplay strategies for Alphalock, complemented by an information theory-based baseline. The project encompasses data preprocessing, environment and agent design, and training workflows, as detailed below:

- **Data Preparation:** We utilized scripts such as `create_word_freq.py` to preprocess and normalize word frequency data, generating the dictionaries `4letter_word_freqs.json` and `4letter_word_freqs_relative.json`.

Table 1: Performance Metrics Comparison

Solver	Average Guesses	Average Time (seconds)
Information Theory-based	4.93	14.60
RL-based	4.46	14.24

- **Environment Design:** The Alphalock game logic was implemented in `rl_environment.py`, providing state features like `pool_entropy`, `attempts_remaining`, and `feedback_history`.
- **Agent Development:** The RL agent was built in `rl_agent.py`, incorporating a policy network to predict α and β weights for dynamic scoring. The training loop, including reward calculations and episodic updates, was handled in `rl_trainer.py`.
- **Baseline Solver:** An information theory-based solver was developed in `infotheory_solver.py`, leveraging entropy to optimize guesses.
- **Visualization and Analysis:** Post-training analysis was supported by scripts like `visualize_training.py` and `visualize_alpha.py`, which generated insights into the agent's learning dynamics and performance.
- **Code Integration:** All components were integrated through `main.py`, allowing direct comparisons between RL and IT solvers.

The complete implementation is available in our GitHub repository: `https://github.com/BerkeAltiparmak/AlphaLock-RL`

6 Results

The results of our experiments highlight the effectiveness of the reinforcement learning (RL) model over the information theory (IT)-based baseline. Below, we present key observations and performance metrics derived from models' performances on the 1000 most common 4-letter words in the English literature:

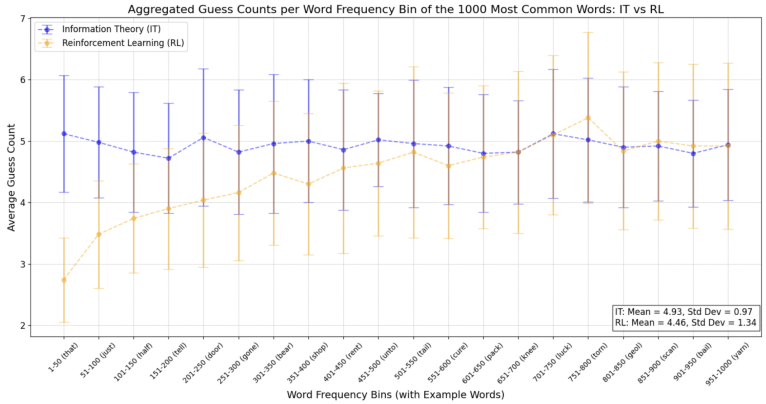

Figure 1: Performance of RL model and IT model on the 1000 most common 4-letter words.

6.1 Performance Metrics

We evaluated both models on average guesses and average time required to find the solution, where the solution is picked to be one of the 1000 most common 4-letter words in the English literature. We observed that the RL-based solver consistently outperforms the IT-based model, as shown in Table 1.

While IT prioritizes reducing uncertainty in larger pool sizes by systematically eliminating unlikely words, the RL model adapts dynamically, balancing exploration and exploitation based on its learned

policy. This enables the RL model to consistently require fewer guesses while maintaining comparable computation times.

Figure 1 provides complementary insights into how the two solvers perform.

- **Superior Accuracy:** The RL solver consistently requires fewer guesses compared to the IT solver, with an average guess of 4.46 compared to IT solver's 4.93. This highlights the RL agent's ability to adapt dynamically and balance exploration and exploitation effectively.
- **Behavioral Variability:** While the RL solver achieves better average performance, its higher variability in certain cases indicates room for improvement, particularly in edge scenarios or less frequent words.
- **Efficiency Across Runs:** Both graphs confirm that the RL solver is robust across different test cases, offering consistent reductions in guess count while maintaining computational efficiency especially if the word is relatively common.
- **Strategic Adaptation:** These visualizations reaffirm the RL model's strength in adapting its exploration-exploitation strategy dynamically, outperforming the IT solver across diverse game scenarios.

6.2 Case Studies

The Most Common Case: The RL-based model frequently diverges from selecting the word with the highest information-theoretic (IT) score and instead opts for words that balance high IT scores with high real-world frequency. This approach allows the RL model to solve puzzles more quickly by effectively exploiting the linguistic bias of Alphalock's word pool, which favors commonly used words.

Best-Case Scenario: In rare instances, the RL model's early prioritization of frequent words proves highly advantageous, enabling it to solve puzzles significantly faster (sometimes on the second guess) than the IT-based solver. This highlights the RL model's ability to exploit real-world frequency patterns in an optimal manner.

Challenging Scenario: Occasionally, the RL model encounters difficulty when faced with multiple valid options that share similar patterns. For instance, if the word being solved has the pattern *sXar*, where *X* is an unknown letter, the true word could be *star*, *spar*, *scar*, etc. In such cases, the RL model often guesses sequentially through the options until it finds the correct answer. An exploration-focused approach, such as the IT solver's entropy-maximizing strategy, would perform better here by targeting a guess that tests the potential letters for *X* more systematically. Despite these challenges, even in such suboptimal scenarios, the RL model frequently performs comparably to the IT solver, as its early exploration phase helps mitigate these situations.

6.3 Visual Analysis

6.3.1 Guess Number vs Pool Size

The figure provides an insightful visualization of how the RL agent narrows the pool size dynamically while balancing exploration and exploitation. The key observations from the graph include:

- **Early Exploration:** During the initial guesses, the pool size is significantly large, and higher α values (red tones) dominate, indicating a focus on exploring potential word candidates.
- **Pool Reduction:** As the game progresses, the pool size decreases drastically, particularly between guesses 2 and 4, showcasing the effectiveness of the RL agent in narrowing the solution space.
- **Shift to Exploitation:** By guess 5, the α values (lighter tones) reflect a transition to exploitation, where the model uses accumulated knowledge to refine its guesses.
- **Consistency Across Runs:** The clustering of points within each guess number demonstrates the model's reliability in reducing the pool size systematically.

This visualization underscores the RL agent's adaptability and efficiency in dynamically balancing exploration and exploitation to achieve optimal performance. Figure 1 serves as a testament to the

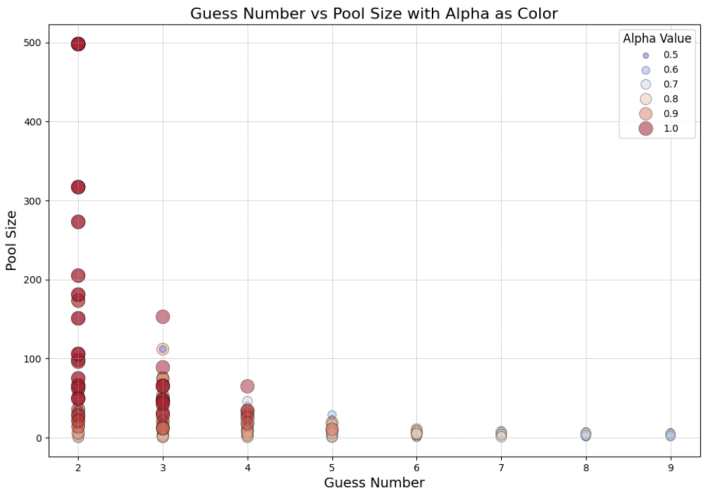

Figure 2: Guess Number vs Pool Size with Alpha as Color. The graph illustrates how the RL agent reduces the pool size over guesses while dynamically adjusting α values to balance exploration and exploitation.

model's robust decision-making process, which outperforms the IT-based baseline in both guesses and time efficiency.

6.3.2 Guess Number vs Alpha Trends and Confidence Interval (CI)

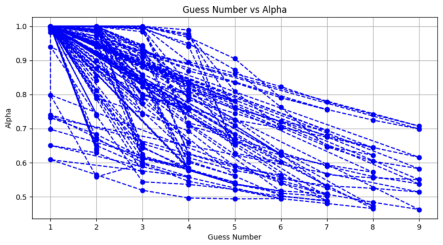

Figure 3: Guess Number vs Alpha across all episodes. Each line represents a single episode's trajectory, showing variability and adaptation.

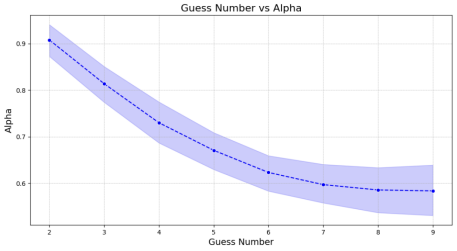

Figure 4: Guess Number vs Alpha (CI). Shows the average trend and confidence intervals for α.

Figure 3 and Figure 4 provide complementary perspectives on how the RL agent adapts its exploration-exploitation strategy:

- **Variability in Episodes (Figure 3):** The individual trajectories highlight the dynamic adjustments made by the RL agent. Early guesses show higher α values for exploration, while later guesses demonstrate a shift towards exploitation.
- **Strategic Adaptability(Figure 4):** The CI version shows a clear trend of decreasing α, reflecting the agent's systematic transition from exploration to exploitation. The narrow confidence intervals indicate consistency across episodes.

6.3.3 Pool Size vs Alpha Trends and Confidence Interval (CI)

Figure 5 and Figure 6 provide complementary perspectives on how the RL agent's α values adjust as the pool size changes:

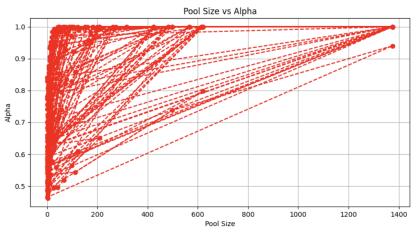

Figure 5: Pool Size vs Alpha across all episodes. Each line represents a single episode's trajectory, showcasing variability and adaptation.

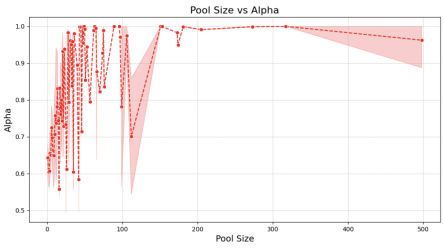

Figure 6: Pool Size vs Alpha (Cleaned). Depicts the average trend and confidence intervals for α.

- **Variability in Episodes (Figure 5):** The individual trajectories highlight the dynamic adjustments of α. Early on, α tends to be higher, prioritizing exploration, while later, as the pool size decreases, α values stabilize to favor exploitation.
- **Average Trend (Figure 6):** The cleaned version demonstrates a clear and systematic decrease in α as the pool size reduces. This trend reflects the RL agent's strategic adaptation, narrowing its focus as uncertainty reduces.
- **Strategic Adaptability:** Both graphs together demonstrate the RL model's capability to dynamically shift between exploration and exploitation based on the size of the remaining pool.
- **Stabilization in Alpha:** As the pool size becomes smaller (below 100), both figures show that α values stabilize closer to 0.6-0.7, demonstrating strong exploitation compared to exploration.

7 Conclusion

This research highlights the effectiveness of reinforcement learning (RL) in solving complex decision-making problems such as the Alphalock game. By leveraging a dynamic balance between exploration and exploitation, the RL agent demonstrates consistent and, in many cases, superior performance compared to the information theory (IT)-based baseline in terms of adaptability, efficiency, and guess accuracy. The introduction of a continuous action space with adjustable α and β values allows the model to seamlessly transition from early exploratory phases, where information gain is prioritized, to later exploitative phases, where common word frequencies are leveraged. This mechanism optimizes the decision-making process across all episodes while maintaining a high level of interpretability.

From a technical standpoint, the RL framework uses a manually implemented Vanilla Policy Gradient (REINFORCE) algorithm with episodic updates to learn an optimal policy tailored to Alphalock's unique constraints. The policy network dynamically predicts α and β, which guide the scoring function:

$$\text{Score} = \alpha \cdot \text{IT} + \beta \cdot \text{RWF},$$

where information-theoretic (IT) measures drive exploration and relative word frequency (RWF) encourages exploitation. This setup eschews traditional value functions in favor of directly optimizing cumulative rewards, resulting in a simpler yet highly effective model, as its superiority is shown in the first graph.

The visual analyses further underscored the RL agent's strategic behavior. The second graph illustrated how the agent dynamically reduces the solution pool across episodes by adjusting its exploration and exploitation parameters in response to the game state. Subsequent visualizations provided insights into the RL agent's learning process, highlighting its ability to adapt to varying levels of uncertainty as the solution pool narrowed. Additionally, these analyses revealed the agent's alignment with human-like strategies, leveraging linguistic biases in the word pool to solve puzzles efficiently.

By incorporating RL methods with interpretable mechanisms such as entropy-based measures and relative word frequencies, this study bridges the gap between theoretical models and practical, scalable

applications. These findings not only validate the RL agent's capabilities in solving Alphalock but also set the foundation for extending this framework to a broader range of decision-making tasks.

8 Broader Impact and Future Plans

The promising results from this research open avenues for several extensions and applications of the RL agent for word-guessing games and similar decision-making problems:

1. **Generalization:** Extend the RL agent to handle games with varying lengths and constraints, including longer or shorter word lengths and diverse feedback mechanisms. This step aims to make the agent adaptable to a broader class of games.
2. **Enhanced Reward Shaping:** Refine the reward function to include more sophisticated intermediate objectives and more hyperparameter exploration. This could guide the RL agent more effectively and further improve its convergence and decision-making capabilities.
3. **Real-Time Application:** Integrate the RL and IT solvers into an interactive application for real-world testing and gameplay. This would allow users to experience the model's strategies firsthand and provide insights into practical usability and further refinements.

These future directions aim to improve the robustness and applicability of the RL agent, expanding its capabilities and testing its performance in diverse and realistic scenarios.

References

[1] 3Blue1Brown. Using information theory to solve wordle, 2023. Accessed: December 5, 2023.

[2] Benton J. Anderson and Jesse G. Meyer. Finding the optimal human strategy for wordle using maximum correct letter probabilities and reinforcement learning. 2022. Accessed: December 5, 2023.

[3] Siddhant Bhambri, Amrita Bhattacharjee, and Dimitri Bertsekas. Reinforcement learning methods for wordle: A pomdp/adaptive control approach. *Proceedings of AI Journal*, 2023. Accessed: December 5, 2023.

[4] Google Books. Google books ngram viewer dataset, 2019. Accessed: December 5, 2023.

[5] Infochimps. English word dataset, 2011. Accessed: December 5, 2023.

[6] Hongyi Zeng. Finding wordle strategies that can be mastered by humans. *Highlights in Science, Engineering and Technology: CMLAI 2023*, 39:714–719, 2023. Accessed: December 5, 2023.