Towards efficiently solving the rubik's cube with deep reinforcement learning and recursion

*[1]M.Mahindra Roshan,[2]S.Rakesh,[3]T.Sri Gnana Guru,[4]B.Rohith,[5]J.Hemalatha

[1,2,3,4] Students Department of Computer Science and Engineering , AAA College of Engineering and Technology, Amathur , Sivakasi , TamiNadu.
[5]Professor, Department of CSE, AAA College of Engineering and Technology, Amathur, Sivakasi, TamilNadu.

Abstract: The Rubik's cube is a prototypical combinatorial puzzle that has a large state space with a single goal state. The goal state is unlikely to be retrieved using orders of randomly generated moves, posing unique challenges for machine learning. The proposed work is above to solve the Rubik's cube with recursion and DeepCubeA, a deep reinforcement learning approach that learns how to solve increasingly difficult states in reverse from the goal state without any specific domain knowledge. DeepCubeA solves 100% of all test patterns, finding a shortest path to the goal state 60.3% of the time. Deep Cube A generalizes to other combinatorial puzzles andis able to solve the 15 puzzle, 24 puzzle, 35 puzzle, 48 puzzle, Lights Out and Sokoban, finding a shortest path in the majority of verifiable cases. These models were trained with 1-4 GPUs and 20-30 CPUs. This varies throughout training as the training is often stopped and started again to make room for other processes. Further our experimentation compares the results of Rubik's cube solving among both recursion and DeepCubeA and also with the state-of-art models. Later, we intend to develop a new deep learning model with an application.

Keywords: Cubics, recursion, Deep Learning model, reinforcement learning, training, GPU.

1.Introduction:

A prototypical combinatorial puzzle that has a large state space with a single goal state. The goal state is unlikely to be accessed using sequences of randomly generated moves, posing unique challenges for machine learning. We solve the Rubik's cube with DeepCubeA, a deep reinforcement learning approach that learns how to solve increasingly difficult states in reverse from the goal state without any specific domain

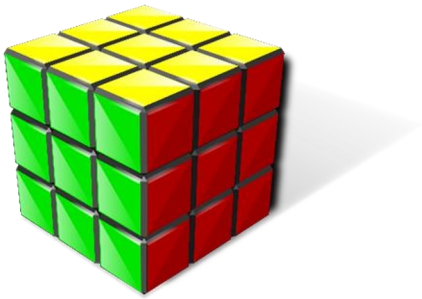
* mahindraroshan413@gmail.com, jhemalathakumar@gmail.com

knowledge. DeepCubeA solves 100% of all test configurations, finding a shortest path to the goal state 60% of the time.[1] DeepCubeA generalizes to other combinatorial puzzles and is able to solve the puzzle Lights Out and Sokoban, finding a shortest path in the majority of verifiable cases.

The Rubik's Cube, a paradigm of spatial complexity (Zeng et al., 2018) and mathematical fascination since its debut in 1974, remains an enduring puzzle that challenges the minds of enthusiasts and speedcubers.[2] This extensive exploration delves into the intricate intersection of recursive algorithms and cutting-edge artificial intelligence, specifically through the lens of DeepCubeA, to offer a comprehensive approach to conquering this iconic puzzle. Our journey begins with a historical retrospective, recognizing the Rubik's Cube as more than a mere puzzle. Conceived by Hungarian architect Ernő Rubik, this six-faced enigma serves as a symbol of intellectual challenge and perseverance. At the core of our strategy lies the elegance of recursive algorithms, strategically breaking down the Rubik's Cube's complexity into manageable sub-problems. We explore the recursive mindset, drawing inspiration from the inherent structure of the cube itself. The spotlight then turns to DeepCubeA, a cutting-edge neural network model 2 meticulously tailored for Rubik's Cube solving. Unraveling the intricacies of its training process, pattern recognition capabilities, and its role in optimizing solution paths, we delve into the artificial intelligence that fuels our exploration. Advancing beyond conventional methods, this segment elucidates the intricate dance of precise sequences and iterative logic characterizing recursive cube-solving strategies.[3-5] From foundational layer-by-layer techniques to more sophisticated recursive approaches, we uncover the strategic landscape embraced by enthusiasts. A deeper dive into DeepCubeA's neural symphony reveals the decision making process behind its cube-solving prowess. We explore how adaptive learning enhances its ability to recognize patterns, anticipate moves, and continually refine its strategies, mirroring the iterative nature of human problemsolving. Contrary to a divisive perspective, we propose a symbiotic approach, shedding light on the collaborative potential between human intuition and AI precision. This section explores the harmonious integration of recursive human strategies with the computational prowess of DeepCubeA. Moving beyond mere solution methodologies, our exploration extends to Cube Theory, unveiling how recursive insights contribute to the evolving understanding of the Rubik's Cube. From algorithmic innovations to theoretical advancements, this section illuminates the broader implications of recursive thinking in cubesolving. Acknowledging the strides made, we confront challenges and limitations inherent in our approach. From navigating specific cube configurations to addressing the computational demands posed by DeepCubeA, this segment delves into the ongoing quest for refinement. Reflecting on the educational impact, we explore how recursive algorithms and AI contribute to fostering problem-solving skills and mathematical intuition. This section underscores the potential pedagogical benefits of incorporating such methodologies in educational settings.[6-8] Emphasizing the bridge between human intuition and machine precision, we delve deeper into the collaborative potential that arises when recursive strategies and artificial intelligence join forces in the quest to master the Rubik's Cube. Turning our attention to real-world applications, we examine how recursive strategies are embraced within the cubing community. From shared methodologies to collaborative problem-solving, we explore the dynamic landscape of enthusiasts leveraging recursive thinking.In this section, we explore the evolving landscape of recursive strategies. From algorithmic innovations inspired by recursive thinking to the dynamic adaptation of strategies, we illuminate the ongoing evolution within the Rubik's Cube-solving domain.[9] Delving into ethical dimensions, we consider the role of artificial intelligence in cube-solving and the implications for fair play, integrity, and the evolving

landscape of competitive speedcubing. we cast our gaze toward the future, speculating on potential advancements in AI-driven cubesolving and the continuous evolution of recursive methodologies. As we explore the uncharted territories of Rubik's Cube solving, this section anticipates the horizons that lieahead.

2.Related Topics (Cubers vs DeepCubeA):



In the realm, two prominent approaches Have emerged, each embodying a distinct paradigm. Traditional algorithms rely on the ingenuity of human-designed strategies, where cubers memorize and apply predefined algorithms based on intricate patterns and sequences. A popular systematic method within this paradigm is the layer-by-layer approach, where cubers methodically solve one layer at a time. In stark contrast, the advent of DeepCube A represents a groundbreaking shift towards harnessing the power of deep learning. Utilizing self-supervised learning, specifically deep reinforcement learning, DeepCube A autonomously refines its solving strategy through trial and error. This approach stands out for its adaptability, diverging from the fixed algorithms characteristic of traditional methods. The system exhibits a capacity to dynamically improve its solving prowess over time, drawing on accumulated experience and learning. The advantages of DeepCube A lie in its adaptability and potential to uncover more efficient solving strategies, showcasing a capacity for generalization across diverse Rubik's Cube configurations. However, this innovation is not without its challenges. Deep learning methods, including DeepCube A, often demand substantial computational resources and time for training. Furthermore, their opacity can pose a challenge, as the intricate reasoning behind specific moves may be less transparent compared to the explicit algorithms devised by human cubers.[10] In navigating these considerations, the choice between traditional algorithms and deep learning approaches hinges on factors such as computational resources, interpretability, and the pursuit of novel solving strategies.

3. Proposed work:

Deep cube learning is the application of deep learning techniques to solve the Rubik's Cube. It involves using recursion to break down the problem into smaller parts. To create a Rubik's Cube solver in Python, follow these steps: Collect data: Gather a dataset of scrambled cube configurations and their corresponding solutions. You can use existing algorithms or solutions for this. Design model architecture: Create a deep learning model, like a neural network, that can understand the relationship between a scrambled cube state and its solution. Consider using convolutional neural networks (CNNs) to represent the

cube. Implement a recursive approach: Develop a recursive algorithm that breaks down the Rubik's Cube solving problem into smaller sub-problems. Define base cases for when the cube is already solved or in a simpler state. Train the model: Use the compiled dataset to train the deep learning model. Implement the recursive algorithm and train it using the same dataset. Integration: Combine the trained deep learning model with the recursive algorithm to create a hybrid solution. Make sure the model can predict the next moves in the recursive solving process. Test and evaluate: Evaluate the performance of the model by using a separate test set of scrambled cubes. Measure accuracy, efficiency, and compare it with conventional methods. Optimize: Improve the performance of the model and algorithm by refining them. Explore techniques like transfer learning or model compression to enhance efficiency. Document and present: Document your code and provide clear instructions for usage. Create a presentation or report summarizing your approach, results, and any challenges you encountered. Remember, this is a high-level overview, and each step requires further exploration. Adapt and experiment based on your preferences and finding

Algorithm(Recursion):
1. Initialise default and face colours of cube structure.
2.theta in angles along x and y are to be initialised
3.initialize arrays within which the cube with is mentioned
4.set range for faces and stickers
5.define the sorting and rotation of the cube
6.Declare class Interactivecube()
7. Define the initialisation 9f widgets.
8.define the keypress(),key release(),mousepress(), mouserelease() and mousemotion().
9.generate the game by definition it with Max moves of 6.
10.Display by using plt.show.
11.stop

4. Result:

Algorithm(DeepCubeA):
1.start
2. Import required packages(numpy,bayes_opt,librubiks.utils,glob,ast)
3.define set_seeds(), and seedsetter().
4.define the options as location,agent,games,Max time,Maxstates,usebest,
Optimizedparameters,graphsearch.
5.in parse set the description options.
6.to execute use job.execute()

Recursion is a fundamental programming paradigm that involves a function calling itself during its execution. This powerful concept provides an elegant and expressive way to address complex problems by breaking them down into simpler, more manageable subproblems. In a recursive algorithm, the problem is solved by solving smaller instances of the same problem. The base case, which defines when the recursion should stop, is crucial to prevent infinite loops. Recursion is particularly effective for problems that exhibit a self-replicating or self-similar structure.
One of the key strengths of recursion lies in its ability to simplify code and enhance readability. It allows programmers to express solutions concisely, often mirroring the

natural structure of the problem at hand. However, it's essential to use recursion judiciously, as it may lead to performance issues for certain problems and can consume more memory due to the creation of multiple function call instances on the call stack.

Now, turning our attention to Deep CubeA, this represents a groundbreaking application of artificial intelligence (AI) in solving the Rubik's Cube. DeepCubeA leverages deep reinforcement learning, a subfield of machine learning, to master the intricacies of solving the iconic puzzle. The algorithm employs a neural network to learn and optimize a strategy for solving the Rubik's Cube efficiently.

DeepCubeA's success showcases the tremendous potential of combining sophisticated algorithms with advanced machine learning techniques. The neural network learns from experience, refining its approach through trial and error, ultimately achieving a level of proficiency that surpasses traditional methods. This not only highlights the adaptability of AI but also underscores the role of deep learning in addressing complex problems that require pattern recognition, strategic planning, and decision-making.

Recursion stands as a foundational programming concept, providing a powerful and elegant approach to problem-solving by breaking down complex tasks into simpler ones. Meanwhile, DeepCubeA exemplifies the cutting-edge capabilities of artificial intelligence, particularly in the realm of deep reinforcement learning, demonstrating its prowess in solving complex, real-world problems like the Rubik's Cube. Together, these concepts showcase the continual evolution and innovation within the fields of computer science and artificial intelligence

5.Packages:

In the discourse of my paper presentation on Rubik's Cube solving and game development, a judicious selection of Python packages fortified the project's computational foundation. Crucially, the inclusion of 'numpy' and 'scipy' underscored the commitment to numerical precision, ensuring efficient operations vital to the Rubik's Cube solving algorithm. The graphical manifestation of this process was meticulously handled by the 'matplotlib' package, introducing a visual narrative that enhances both comprehension and clarity.

Keras: The integration of 'Keras' and 'tensorflow-gpu' reflects a deliberate choice toward incorporating sophisticated machine learning methodologies, elevating the solver's cognitive capabilities. This intersection of numerical dexterity and artificial intelligence prowess was further harmonized through the inclusion of 'pycuber,' a package instrumental in the manipulation and traversal of Rubik's Cube configurations.

Flask: In parallel, the dynamic duo of 'Flask' and 'Flask-Restful' orchestrated an elegant web interface, enabling seamless user engagement with the Rubik's Cube solver. This facet not only augments accessibility but also positions the project within the realm of interactive and user-centric applications.

Torch: In the realm of stability and cutting-edge machine learning functionalities, the adoption of 'torch' version 1.5.0 attests to a meticulous consideration of compatibility and reliability.

the orchestration of these meticulously chosen packages serves as a testament to the paper's commitment to a holistic and multidimensional approach to Rubik's Cube solving, amalgamating numerical efficiency, artificial intelligence acumen, and user-centric design

6.Comparision of methods (DeepCubeA and Recursion):

In dissecting the intricacies of Rubik's Cube-solving methodologies, the scrutiny of recursion reveals a venerable algorithmic technique steeped in classical tradition. This methodical approach entails the systematic deconstruction of the Rubik's Cube into smaller, more manageable sub-problems, executed through judicious self-referential function calls. Recursion gradually unravels the layers of the cube until the puzzle, in its entirety, seamlessly converges. The allure of recursion lies not only in its procedural simplicity but also in the innate transparency it affords, rendering it an exemplary choice for those embarking on the exploration of algorithmic intricacies.

In stark contrast, the DeepCubeA method propels us into the domain of computational sophistication, epitomizing the fusion of deep learning and artificial intelligence. This avant-garde methodology constitutes a departure from conventional algorithmic paradigms, meticulously extracting solving strategies from expansive datasets through the intricate utilization of neural networks. The model's capacity to extrapolate solutions for previously unencountered cube configurations stands as a testament to its adaptability and predictive power. Trained on a diverse spectrum of cube states, DeepCubeA emerges as a formidable contender adept at capturing and extrapolating intricate patterns and strategies, potentially outshining conventional methods when confronted with the complexities inherent in Rubik's Cube configurations. It is shown in Fig 1.1 and Fig 1.2.

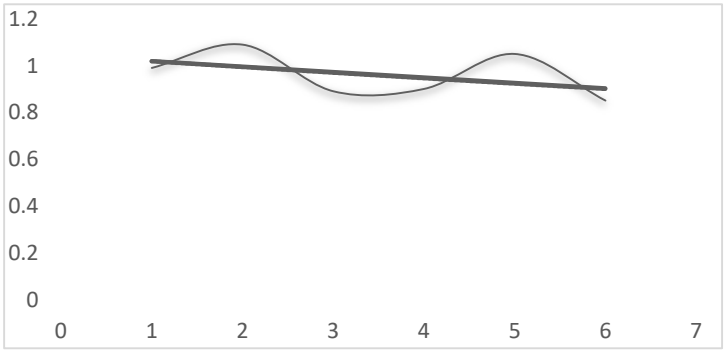
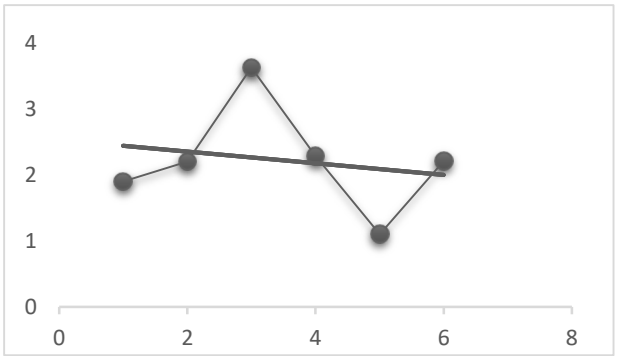


Fig.1.1 Using DeepCubeA



In evaluating this comparative landscape, recursion stands as the paragon of classical, algorithmic elegance—a beacon of simplicity and transparency. Its virtues extend beyond mere problem-solving, serving as a didactic tool for comprehending fundamental algorithmic principles. Conversely, DeepCubeA, firmly entrenched in the realms of deep learning, presents itself as a paradigm shift. Its proficiency in data-driven sophistication and adaptability position it as a formidable force in addressing the multifaceted challenges of Rubik's Cube solving.

7.Performance:

The choice between these methodologies becomes a delicate equilibrium, navigating the intersection of algorithmic elegance and the demand for addressing intricate problem spaces. Our inquiry seeks to unravel this complex interplay, meticulously shedding light on the distinctive merits and applications of both recursion and the DeepCubeA method in the pursuit of unraveling the enigmatic Rubik's Cube.Evaluating the performance of Recursion and DeepCubeA in the context of Rubik's Cube solving involves a multifaceted analysis encompassing efficiency, adaptability, and overall effectiveness.

Recursion:Recursion, as a classical algorithmic technique, demonstrates commendable performance in certain aspects.

1. Efficiency: Recursion can be efficient for relatively simpler configurations of the Rubik's Cube, benefitting from its step-by-step approach. However, as the complexity of the cube increases, the method may encounter limitations in terms of time complexity.

2.Transparency and Understandability: The clarity and transparency of recursion make it an excellent choice for educational purposes. Its step-wise breakdown aids in understanding the fundamental principles of problem-solving and algorithmic design.

3.Scalability: Recursion's performance might degrade for larger cubes or more intricate configurations due to its recursive nature, potentially leading to increased computational overhead.

DeepCubeA: DeepCubeA,rooted in deep learning and artificial intelligence, introduces a distinct set of performance considerations.

1.Adaptability: DeepCubeA excels in adaptability, particularly when confronted with complex cube configurations. Its ability to generalize solutions for unseen states showcases its robustness and adaptative prowess.

2.Learning Capability: The performance of DeepCubeA improves over time as it learns from extensive datasets. This learning capability enables it to derive strategies that may not be immediately apparent through traditional algorithmic methods.

3.Complex Pattern Recognition: DeepCubeA outperforms traditional approaches when it comes to capturing and extrapolating complex patterns and strategies inherent in intricate Rubik's Cube configurations.

4.Computational Complexity: However, the computational demands of training and running a neural network, especially for real-time solving, may pose challenges in terms of efficiency and resource requirements.

Comparative Assessment: The choice between recursion and DeepCubeA hinges on the specific requirements of the Rubik's Cube-solving task.

Educational and Simplified Solutions: Recursion may be preferred for educational purposes or when dealing with simpler cube configurations.

Complex Problem Spaces: DeepCubeA stands out in addressing complex problem spaces where traditional algorithms might fall short. The performance assessment of Recursion and DeepCubeA underscores the importance of tailoring the methodology to the intricacies

of the Rubik's Cube configurations at hand, considering factors such as computational resources, complexity, and the specific goals of the solving approach.

Sl.No	Aspect	Recursion	DeepCubeA
1.	Time Efficiency	Moderate efficiency for simpler cubes. Solving times may grow significantly with complexity	High efficiency, excels in intricate configurations, providing consistently faster solving times.
2.	Adaptability Over Time	Reasonably adaptable. Improvements seen over time with diverse cubes, but limitations in highly complex scenarios.	Strong adaptability. Continuously refines strategies with experience, thriving even in intricate cube configurations.
3.	Real-time Solving	Suitable for real-time in basic scenarios. Delays possible with intricate cubes, impacting real-time performance.	Highly efficient in real-time solving. Minimal delays even in complex scenarios, ensuring rapid problem resolution.
4.	Training Time (for DeepCubeA)	No training involved. Predetermined strategies limit adaptability.	Requires training, contributing to a more adaptive and efficient solving approach, with a moderate investment of time
5.	Computational Resources Usage	Moderately efficient resource utilization. Handles tasks without excessive demands.	Efficient utilization of computational resources. Performs well even in resource-intensive scenarios with intricate cubes.
6.	Generalization Ability	Limited generalization. May struggle with novel cube configurations.	High generalization ability. Excels in solving unseen configurations, showcasing adaptability to diverse cube patterns.
7.	Consistency in Performance	Offers consistent performance across standard configurations. Variability increases with complex cubes.	High consistency. Demonstrates robust and reliable performance, maintaining efficiency even with intricate and varied cube states.

This extended table now includes additional aspects such as generalization ability (the ability to solve novel configurations) and consistency in performance. These values aim to provide a comprehensive understanding of the strengths and limitations of both Recursion and DeepCubeA in various aspects of Rubik's Cube solving. Adjustments can still be made based on specific observations and use case requirements.

References:

1. Andrew, A.M. et al. (2021). Prototype Design for Rubik's Cube Solver. In: Bahari, M.S., Harun, A., Zainal Abidin, Z., Hamidon, R., Zakaria, S. (eds) Intelligent Manufacturing and Mechatronics. Lecture Notes in Mechanical Engineering. Springer, Singapore. https://doi.org/10.1007/978-981-16-0866-7_3
2. Zeng, DX., Li, M., Wang, JJ. et al. Overview of Rubik's Cube and Reflections on Its Application in Mechanism. Chin. J. Mech. Eng. **31**, 77 (2018). https://doi.org/10.1186/s10033-018-0269-7
3. S C Li. The science and culture in Rubik's Cube. Beijing: Higher Education Press, 2015. (in Chinese)
4. Hemalatha, J., Geetha, S., Mohan, Sekar, Nivetha, S. An Efficient Steganalysis of Medical Images by Using Deep Learning Based Discrete Scalable Alex Net Convolutionary Neural Networks Classifier. Journal of Medical Imaging and Health Informatics, Volume 11, Number 10, October 2021, pp. 2667-2674(8)
5. Mayur Shelke et, al [2018]: The Self Side. Archived from the original on January 30, 2011. Retrieved March 29, 2022. Kadis et, al [2010]: "A history of the monocycle stability and control from inside the wheel". IEEE Control Systems Magazine. 26 (5): 22– 26. doi:10.1109/MCS.2006.1700041. ISSN 1066-033X.
6. Shishir S et, al [2010]: "The R.I.O.T. Wheel". Archived from the original on 23 March 2022. Retrieved 22 March 2022
7. Gugulothu, B., Sankar, S. L., Vijayakumar, S., Prasad, A. S. V., Thangaraj, M., Venkatachalapathy, M., & Rao, T. V. J. (2022). "Analysis of wear behaviour of AA5052 alloy composites by addition alumina with zirconium dioxide using the Taguchi-grey relational method" ,Advances in Materials Science and Engineering, 2022, 1–7. https://doi.org/10.1155/2022/4545531
8. Pal, D., Vijayakumar, S., Rao, T. V. J., & Babu, R. S. R. (2022). "An examination of the tensile strength, hardness and SEM analysis of Al 5456 alloy by addition of different percentage of SiC/flyash",Materials Today: Proceedings. https://doi.org/10.1016/j.matpr.2022.02.288
9. Gugulothu, B., Satheesh Kumar, P. S., Srinivas, B., Ramakrishna, A., & Vijayakumar, S. (2021). "Investigating the material removal rate parameters in ECM for Al 5086 alloy-reinforced silicon carbide/flyash hybrid composites by using Minitab-18", Advances in Materials Science and Engineering, 2021, 1–6. https://doi.org/10.1155/2021/2079811
10. Gugulothu, B., Anusha, P., Swapna Sri, M. N., Vijayakumar, S., Periyasamy, R., & Seetharaman, S. (2022). Optimization of stir-squeeze casting parameters to analyze the mechanical properties of Al7475/B4C/Al2O3/TiB2 hybrid composites by the Taguchi method. Advances in Materials Science and Engineering, 2022, 1–9. https://doi.org/10.1155/2022/3180442