Bulletin of the *Transilvania* University of Braşov • Vol. 17 (66), No. 2 – 2024
Series I: Engineering Sciences
https://doi.org/10.31926/but.ens.2024.17.66.2.1

# DEEP REINFORCEMENT LEARNING: INSIGHTS FROM THE RUBIK'S CUBE

**MOUTAMAN M. ABBAS** [1]

***Abstract:*** *This study explores the Rubik's Cube as a medium for investigating both human cognition and artificial intelligence. Through two experiments—one involving novice human participants and the other employing a Deep Q-Network (DQN) reinforcement learning agent—the research examines how different systems learn to solve complex problems. Human participants demonstrated improvement over time, highlighting adaptability, individual strategy development, and learning without formal guidance. In contrast, the DQN agent learned to solve the cube through trial-and-error interactions within a simulated environment, guided by reward feedback and policy refinement. While the AI model achieved high solving accuracy, it required extensive computational resources and lacked generalization beyond the specific task. The findings underscore key differences and potential complementarities between human and machine intelligence. This comparison offers insights into the strengths and limitations of both approaches, reinforcing the value of hybrid systems and continued cross-disciplinary research in understanding intelligent behaviour.*

***Key words:*** *games, algorithm, problem solving, reinforcement learning, cognitive experiment, deep Q-network.*

## 1. Introduction

Every person in the world has played games. This popularity can be attributed to games being intuitive and enjoyable. These unique aspects of games also ensure they are perfect for mind studies by being intuitive, meaning games offer a unique platform for observing inductive biases supporting behaviour in more ecological, naturalistic contexts than are possible within classical lab experiments. Being fun, games enable scientists to investigate new cognition questions like the nature of 'play' and intrinsic motivation, while enabling more extensive and more varied data collection through drawing in large numbers of participants.

Since games are often constructed to challenge our skills and engage our passions, games were historically employed to analyse the mind [17], [12]. Game playing remains a favourite form of leisure entertainment among children and adults alike, spanning

[1] *Faculty of Civil Engineering*, *Transilvania* University of Braşov.

cultures [22], [3] and throughout history [6]. Our capability to analyse cognition with games was vastly broadened in recent years by twin revolutions in massive online games (which generate huge quantities of data and can frequently be played over a phone) and powerful statistical modelling methods.

Rubik's cube is a classic combinatorial puzzle with a vast state space having a unique goal state. In a sequence of randomly generated moves, it is unlikely to be reached [1]. A general intelligent agent would need to be able to learn to solve problems in large domains with little or no supervisory intervention from a human. Deep reinforcement learning with self-play has recently reached superhuman performance, without access to human data or domain expertise [16].

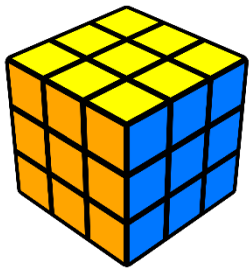

Fig. 1. *Illustration of Rubik's Cube.*

Rubik's Cube is a 3D combination puzzle invented by Hungarian professor of architecture and sculptor Ernő Rubik in 1974 [11] and was originally known as the Magic Cube [5], [7]. The invention generated global interest due to its unique nature, whose impact on mankind was immense. Rubik's Cube was named one among the 100 most pivotal inventions in the 20th century [23]. It is also world's best-selling toy by general acclaim [13]. It secured a special German Game of the Year award [4] and achieved awards for best toy in the UK, France, and US [8]. The Rubik's Cube although became most popular in mains-steam from its peak in the 1980s, it remains widely known worldwide even today. It not just welcomes Rubik's Cube enthusiasts who are researching on Rubik's Cube reduction algorithms [19], [15], [21] but also welcomes scientists and technical professionals from different backgrounds with its advanced design and concepts [20].

Rubik’s Cube structure possesses multiple properties like rotation, permutation and combinations, cycle and symmetry, which were considered physical models or instruments to examine certain scientific problems or were examined by applying scientific theory or methods in certain fields. Overall, Rubik’s Cube principles are embedded in multiple scientific systems related to permutations and combinations, symmetries, and cyclicality. Scholars, by contrast, have explored the inner motion principles of Rubik’s Cube structure. The uses of Rubik’s Cube are described in terms of its rotation properties [24].

Numerous people are now attempting to solve a cube using machine learning. OpenAI trained a single human-type robot hand, called Dactyl, to solve a Rubik's Cube. The learning was done entirely in simulation, and the policies learned were mapped directly to the physical robot without more training in the real world. The feat was possible by

using a new approach called Automatic Domain Randomization (ADR) that gradually increases simulation environment complexity and variability to improve the policies learned against real-world uncertainty [18].

While the system demonstrated impressive capabilities, it was not without limitations:

- May Success Rate: the robot had a 60% rate of success in utilizing simpler scrambles involving 15 rotations and circa 20% for more complex ones involving 26 rotations [18].
- Training Time: the training procedure was computationally demanding, comparable to a simulation equivalent to about 10,000 years [18].
- Generalization: the model was trained specifically for the Rubik's Cube problem, and its generalizability to other manipulation tasks is limited [18].

While the Rubik's Cube-solving robot demonstrates the impressive capabilities of AI in specific tasks, it remains a tool designed for narrow applications. The human brain's versatility, creativity, emotional depth, and efficiency continue to set it apart, underscoring the unique and unparalleled nature of human intelligence. Understanding the mind requires a paradigm shift away from only using highly controlled and simplified experiments and towards the rich landscape of studying cognition.

## 2. The Rubic's Cube

Erno Rubik invented Rubik's Cube in 1974. Within a month, he had devised the original algorithm to solve the cube. The Rubik's Cube became popular all over the world after that, and numerous human-oriented methods for solving it have been found [9]. The methods are easy to memorize and instruct humans how to solve the cube in a step-by-step, structured format.

Evariste Galois came up with a new mathematics branch: group theory was born out of efforts towards finding the solvability of polynomial equations—the roots of the quadratic equation $ax^2 + bx + c = 0$. There are analogous but more involved formulas for the cubic and the quartic polynomials found in the Middle Ages. Group theory was inspired by what was quite possibly the most important open mathematical question of the time: Does there exist a similar algebraic formula with radicals only in the coefficients for an equation of degree five or more? It was a problem that had been unsolved for centuries, despite attacks by the brightest and the best math brains.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

However, since one sticker's position in a cubelet defines where the rest of the stickers in that cubelet are, we can potentially reduce the dimensionality of our representation by simply looking at where one sticker per cubelet is. We eliminate the redundant central cubelets and store only the 24 possible edge and corner cubelet locations. This gives us a 20x24 state representation depicted in figure xx. The moves are represented by notation originally invented by David Singmaster: a move consists in a letter specifying in which face to perform a turn. F, B, L, R, U, D refer to turning front, back, left, right, up, down

faces, respectively. A single letter describes a clockwise rotation, while a letter preceded by an apostrophe describes a counterclockwise rotation. For instance: R and R' would be turning the right face 90° clockwise and counterclockwise, respectively [16].

$$v_{x_i}(a) \tag{1}$$

$$a \in \{U, U', L, F, F', \cdots\} \tag{2}$$

$$A(xi, a) \tag{3}$$

$$R(A(xi, a)) \tag{4}$$

$$v_{x_i}(a) + R(A(xi, a)) \tag{5}$$

$$y_i = \max_a \ (v_{x_i}(a) + R(A(xi, a))) \text{ for } a \in \{U, U', \cdots, F, F'\} \tag{6}$$

$$y_{Pi} = arg \max_a \ (v_{x_i}(a) + R(A(xi, a))) \text{ for } a \in \{U, U', \cdots, F, F'\} \tag{7}$$

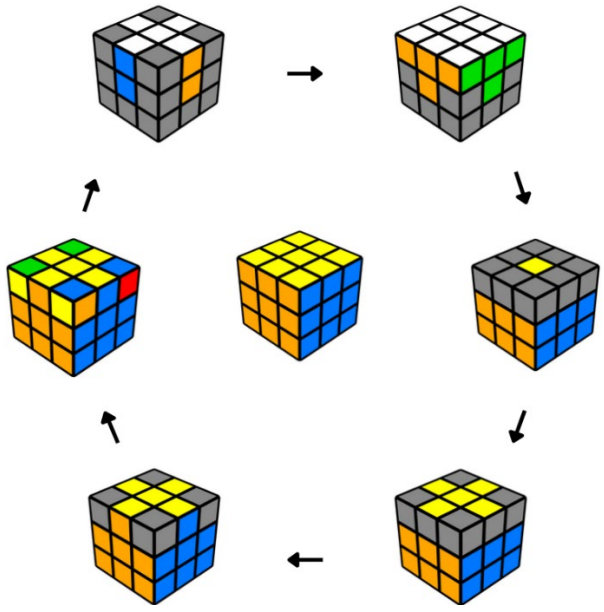

Fig. 1. *Rubik's Cube from scramble to solution*.

The quantity of moves to solve any Rubik's cube state was a subject of long-standing speculation during more than 25 years — since Rubik's cube emerged. This value is also known as "God's number". An upper bound of 29 (using the face turn metric) was found in early 1990's, after which we got another bound of 27 in 2006 [14].

Shorter solutions to Rubik's cube have been a fascination — for researchers in search techniques and enumeration methodologies, as well as by enthusiasts — across decades.

Rubik's cube is a famous challenge problem against which otherwise disparate methods can be compared for researchers. Singmaster and Frey [10] concluded their book Cubik Math in 1982 by conjecturing in it that "God's number" is in the low 20's.

OpenAI taught a one-handed robotic setup, called Dactyl, to solve a Rubik's Cube in a study. The training took place entirely within simulation, and policies learned were efficiently mapped to the physical robot without further real-world training. This was achieved by using a new approach called ADR, where simulated complexity and variability are ratcheted up in a controlled fashion, making policies learned more resilient to real-world uncertainties.

The work [1] introduces DeepCubeA, a new algorithm that integrates search with deep reinforcement learning to solve combinatorial puzzles like the Rubik's Cube without any domain knowledge. DeepCubeA solves the Rubik's Cube at a rate of 100% with the shortest solution path found 60.3% of the time, while it achieves more than 96% optimal solution for puzzles like those involving the 15- and 24-puzzles. Unlike database-to-table methods based upon Pattern Databases (PDBs) requiring extensive memory and domain engineering, DeepCubeA acquires a cost-to-go function through approximate value iteration by learning from scrambled goal states. This learned function serves as a heuristic within a batch-weighted A* search, enabling puzzles to be solved by DeepCubeA more quickly, with fewer nodes expanded, and much less memory. It generalizes effectively to other realms like Lights Out and Sokoban and shows intelligent behaviour by learning about symmetric solutions and shared move structures like those found in group theory. These results show DeepCubeA to be a general-purpose, efficient solver for large state space planning problems.

## 3. Experiment one

### 3.1. Participants

Three people were chosen for this experiment so that there would be diversity in terms of age group and gender. They were not experienced in solving the Rubik's Cube. They learned by themselves in their own method without any guidance on how to solve it. They learned in different ways and different methods.

### 3.2. Procedure

Each subject was given a regular 3x3 Rubik's Cube. They were asked to try to solve the cube by themselves, and to determine how to seek out tutorials or directions that suit them. All subjects had three distinct solving trials after learning for 1 week. The time it took to solve the cube and how many moves it took them were taken for each trial.

### 3.3. Procedure

Each trial had the following data collected for it:

- Solving time: Measured in minutes, seconds, and milliseconds.

- Number of moves: The total count of individual face rotations performed to reach the solved state.

### 3.4. Results

The performance of each participant across the three trials is summarized in Table 1 and Figure 3.

Table 1

*Individual Rubik's Cube Solving Times and Move Counts Across Three Trials for Participants of Different Ages*

| Participant | Trial | Age | Time (min:s:ms) | Moves |
|---|---|---|---|---|
| $P_1$ | | 27 | | |
| | $T_1$ | | 2:39:32 | 140 |
| | $T_2$ | | 3:09:20 | 167 |
| | $T_3$ | | 2:28:57 | 113 |
| $P_2$ | | 65 | | |
| | $T_1$ | | 2:46:22 | 136 |
| | $T_2$ | | 2:46:51 | 147 |
| | $T_3$ | | 3:12:34 | 165 |
| $P_3$ | | 22 | | |
| | $T_1$ | | 6:56:60 | 226 |
| | $T_2$ | | 7:32:22 | 288 |
| | $T_3$ | | 5:02:71 | 192 |

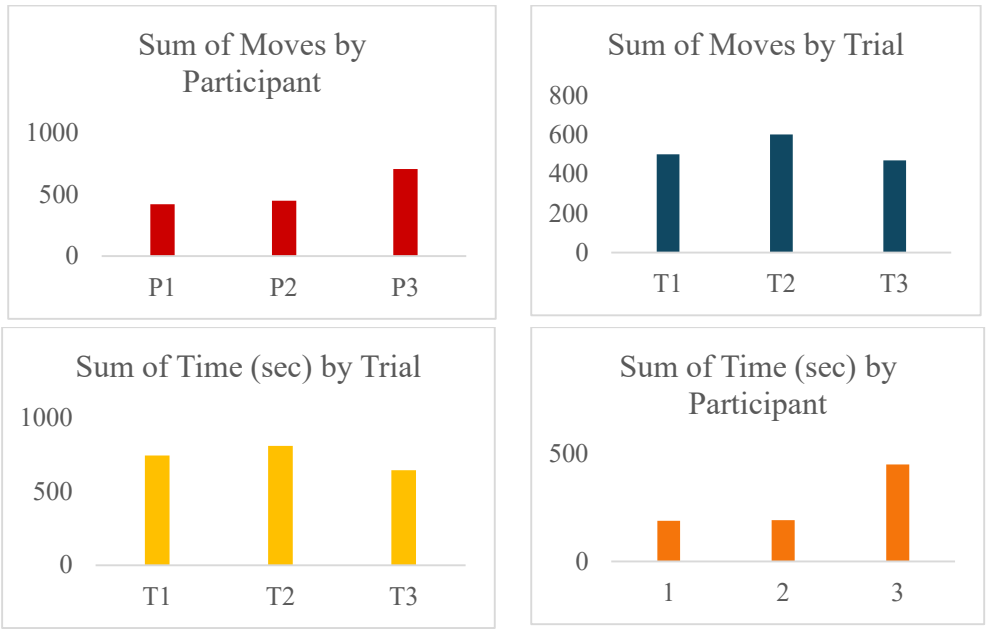


Fig. 3. *Participant performance graph*

## 4. Experiment two

Reinforcement Learning (RL) is a branch of machine learning in which an agent learns to make decisions by interacting with an environment in order to achieve a specific goal. In this process, the agent, which serves as the decision-maker, operates within an environment—such as a Rubik's Cube in your code—and makes decisions based on the current state, which represents the cube's configuration. The agent takes actions, such as rotating a face of the cube, and receives rewards from the environment depending on the outcome of these actions, with higher rewards typically given for configurations closer to the solved state. The agent follows a policy, which is the strategy it uses to select actions based on the state it observes. An episode in RL consists of a sequence of states, actions, and rewards, usually ending when the goal is achieved or after a set number of steps. The learning process involves the agent continuously observing the state, selecting actions (sometimes randomly, sometimes based on its learned strategy), and receiving feedback in the form of new states and rewards. Over time, the agent refines its policy to maximize the total reward. In summary, reinforcement learning is fundamentally about learning by doing—using rewards and penalties to discover the best path toward achieving a goal.

In our code developed to solve the cube, the agent attempts to solve the Rubik's Cube by learning which sequences of moves lead to more solved configurations. It improves its strategy through repeated trial and error, gradually learning which actions are most effective.

### 4.1. The DQN Algorithm

This is a Deep Q-Network (DQN) reinforcement learning agent that tries to solve a Rubik's Cube. The agent learns by interacting with the cube, receiving rewards for making progress, and updating its neural network to improve over time.

#### *4.1.1. Code Components*

a. Cube Environment
- The Cube class (from rubiks.py) represents the Rubik's Cube.
- The agent can scramble, rotate, and check the state of the cube.

b. DQN Agent
- The agent uses a neural network (DQN class) to estimate the value (Q-value) of each possible move in a given cube state.
- It chooses actions using an epsilon-greedy strategy: sometimes random (explore), sometimes the best known (exploit).

c. Replay Memory
- Stores past experiences (state, action, next state, reward) so the agent can learn from them later, not just immediately.

d. Training Loop
- The agent repeatedly scrambles a cube, tries to solve it, and learns from its actions.
- Progress and stats are printed and visualized.

### *4.1.2. Key Parameters*

The key parameters shown in Figure 4:

- edge_length = 2: The size of the cube (2x2 by default).
- num_layers = 3: Number of layers in the neural network.
- BATCH_SIZE = 4096: How many experiences are used in each training step.
- GAMMA = 0.95: Discount factor for future rewards (how much the agent cares about future vs. immediate reward).
- EPS_START = 0.975: Initial probability of choosing a random action (exploration).
- EPS_END = 0.03: Minimum probability of random action (after lots of training).
- EPS_DECAY = 1000000: How quickly the randomness decays.
- TRANSITION_MEMORY_SIZE = 15000: How many experiences to store in replay memory.
- term_iter = 600000: How many moves before forcibly resetting the cube.
- max_attempt_iterations = 100000: Max moves per attempt before giving up and starting over.

```
Correct:     (Current:   0, Running:   2.598, Max:   15)
Reward:      (Current:   1, Running:   5.654, Max:   23)
Randomness (eps_threshold): 27.30%
Total Iterations: 1358103
Batch Size: 4096
Gamma: 0.95
DQN Loss: 6.373172
Current Iter: 1000
Current Time: 119 seconds
```

Fig. 2. *Key parameters of the algorithm.*

### *4.1.3. How the Code Works*

a. Initialization

- Sets up the neural network, optimizer, and replay memory.
- Loads a saved model if available.

b. Main Loop

- For up to 1000 attempts:
  - Scramble a new cube (see Figure 5).
  - Try to solve it, up to max_attempt_iterations moves.
- For each move:
  - Choose an action (random or best known, depending on epsilon).
  - Rotate the cube.
  - Calculate reward (based on how close to solved).
  - Store the experience in memory.
  - Occasionally train the neural network using a batch of past experiences.
  - Print stats and update the visualization.
  - If solved, print "Solved!" and stats, then start a new attempt.
  - If too many moves, reset and start over.

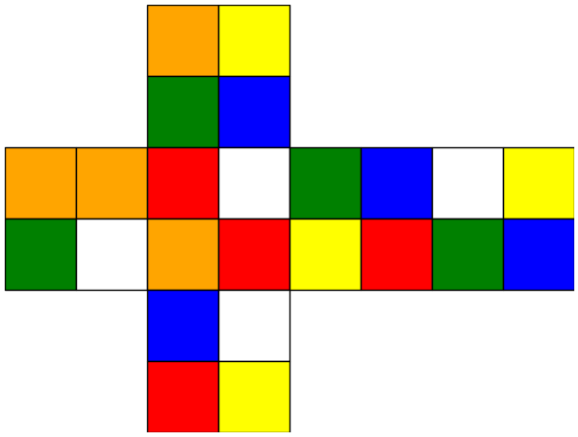

Fig. 3. *Unsolved cube state.*

c. Training

- The neural network is trained to predict the best move for any given cube state, using the experiences stored in replay memory (see Figure 6).

d. Visualization

- The cube's state is shown in a persistent matplotlib window, updating after every move.

e. Saving/Loading

- The model's weights are saved to model.pth after training, and loaded at the start if available.

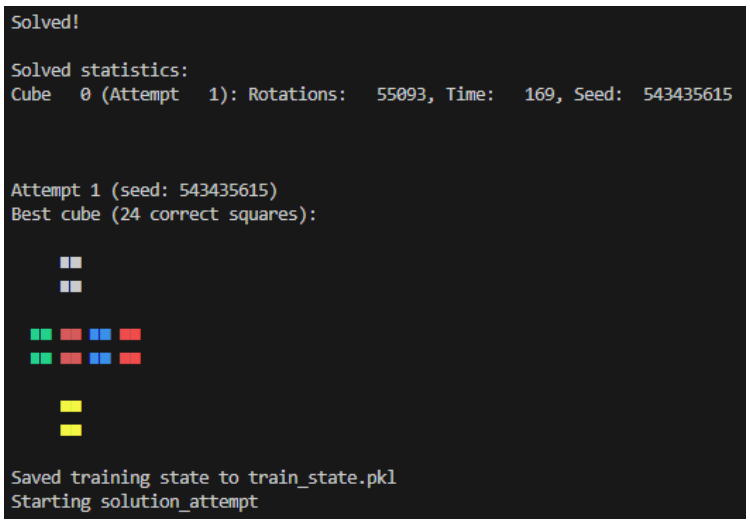


Fig. 4. *Training procedure.*

### *4.1.4. What Do the Stats Mean?*

- Correct: Number of correctly coloured squares (current, running average, max seen).
- Reward: The reward for the current state (current, running average, max seen).
- Randomness (eps_threshold): % chance the agent picks a random move (exploration).
- Total Iterations: Total moves made by the agent.
- Batch Size: Number of experiences used per training step.
- Gamma: Discount factor for future rewards.
- DQN Loss: How well the neural network is learning (lower is better).
- Current Iter: Moves in the current attempt.
- Current Time: Time spent on the current attempt.

### *4.1.5. DQN Learning*

The Big Picture: Reinforcement Learning (RL):

- The agent interacts with an environment (the cube).
- At each step, it observes the state, takes an action, receives a reward, and observes the new state.
- The goal: learn a policy (a way to choose actions) that maximizes total reward—i.e., solves the cube efficiently (see Figure 7).

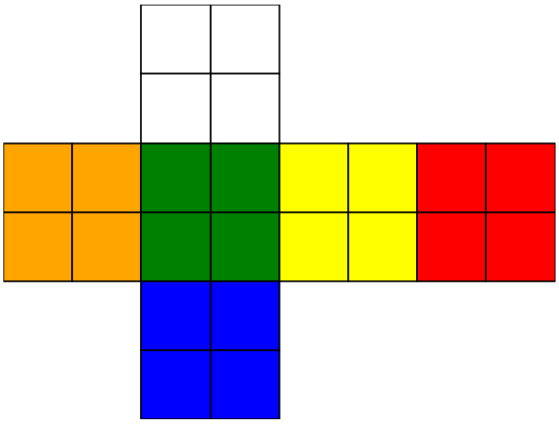

Fig. 5. *Solved cube state.*

The DQN (Deep Q-Network) Approach:

- The agent uses a neural network to estimate the "Q-value" for each possible action in a given state.
- The Q-value is an estimate of the total future reward the agent can expect if it takes that action from that state and acts optimally thereafter.

The Learning Cycle

a. State Representation

- The cube's current configuration is encoded as a tensor (vector of numbers) that the neural network can process.

b. Action Selection (Epsilon-Greedy)
- With probability ε (epsilon, "Randomness"), the agent picks a random action (exploration).
- With probability 1-ε, it picks the action with the highest Q-value according to its neural network (exploitation).
- Epsilon starts high (lots of exploration) and decays over time (more exploitation as the agent learns).

c. Taking an Action
- The agent rotates a face of the cube (or makes a random move).
- The cube's state changes.

d. Reward Calculation
- The agent receives a reward based on how "solved" the cube is after the move.
- More correct squares = higher reward. Solving the cube gives the highest reward.

e. Storing the Experience
- The agent saves the experience (state, action, next state, reward) in a replay memory buffer.
- This allows the agent to learn from past experiences, not just the most recent one.

f. Training the Neural Network
- Periodically, the agent samples a batch of experiences from replay memory.
- For each experience, it computes:
  - The Q-value for the action it took in the original state (using the current network).
  - The "target" Q-value: reward + γ * max Q-value of the next state (using the current network).
- The network is trained to minimize the difference (loss) between its predicted Q-value and the target Q-value (using Huber loss).
- This is done via backpropagation and gradient descent.

g. Updating Epsilon
- After each step, epsilon decays a little, so the agent explores less and exploits more as it learns.

By repeatedly experiencing the environment, the agent learns which actions lead to higher rewards (i.e., getting closer to solving the cube), The neural network generalizes from past experiences to new, unseen cube states. Over time, the agent's policy improves, and it becomes more likely to solve the cube.

#### *4.1.6. Key Points*
- Exploration is crucial early on so the agent doesn't get stuck in bad habits.
- Replay memory breaks correlations between consecutive experiences, making learning more stable.
- Batch training allows the agent to learn from many experiences at once, improving efficiency.
- Discount factor (gamma) controls how much the agent values future rewards vs. immediate ones.

#### *4.1.7. "Learning" Process*

- At first, the agent acts randomly and gets low rewards.
- As it trains, the neural network's Q-value predictions improve.
- The agent starts to make better moves, gets higher rewards, and eventually solves the cube (see Figure 8).
- The randomness (epsilon) decreases, and the agent relies more on its learned policy.

```
Solved statistics:
Cube   0 (Attempt   1): Rotations:   55093, Time:   169, Seed:  543435615
Cube   1 (Attempt   6): Rotations:   94659, Time:   338, Seed:  763707007
Cube   2 (Attempt   7): Rotations:   12387, Time:    50, Seed:   39989727
Cube   3 (Attempt   8): Rotations:   20861, Time:    86, Seed:  196132319
Cube   4 (Attempt   9): Rotations:   47123, Time:   176, Seed:  571067644
Cube   5 (Attempt  10): Rotations:   30791, Time:    73, Seed:   44890473
Cube   6 (Attempt  11): Rotations:   23423, Time:    69, Seed:  908294955
Cube   7 (Attempt  13): Rotations:   54965, Time:   160, Seed: 1028448317
Cube   8 (Attempt  14): Rotations:   22395, Time:    52, Seed:  657358287
Cube   9 (Attempt   1): Rotations:   68487, Time:   189, Seed:   90371604
Cube  10 (Attempt   0): Rotations:   53913, Time:   131, Seed:  148666609
Cube  11 (Attempt   3): Rotations:   53297, Time:   133, Seed:  373352073
Cube  12 (Attempt   5): Rotations:   59165, Time:   260, Seed:  848189226
Cube  13 (Attempt   9): Rotations:   60219, Time:   266, Seed:  385366406
Cube  14 (Attempt  10): Rotations:   37817, Time:   100, Seed:  738830690
Cube  15 (Attempt  11): Rotations:   13485, Time:    34, Seed:  899696241
Cube  16 (Attempt  12): Rotations:   64923, Time:   247, Seed:  563446843
Cube  17 (Attempt  14): Rotations:   83715, Time:   238, Seed:  903773564
Cube  18 (Attempt  19): Rotations:   13029, Time:    49, Seed:  434269215
Cube  19 (Attempt  20): Rotations:   78291, Time:   198, Seed:  833155769
Cube  20 (Attempt  22): Rotations:   13851, Time:    48, Seed:  534123389
Cube  21 (Attempt  24): Rotations:   16941, Time:    50, Seed:  240014222
Cube  22 (Attempt   3): Rotations:   14661, Time:    61, Seed:   54415166
Cube  23 (Attempt   5): Rotations:    6055, Time:    18, Seed:  557010861
Cube  24 (Attempt   0): Rotations:   33885, Time:    82, Seed:  724196996
Cube  25 (Attempt   1): Rotations:   44083, Time:   120, Seed:  444547410
```

Fig. 6. *Successful algorithm attempt statistics*

#### *4.1.8 Outcome*

- At the beginning, the agent will not solve the cube and will make mostly random moves.
- As training progresses, "Max Correct" and "Max Reward" should increase.
- Eventually, it will solve the cube as the agent learns and repeats.

## 5. Discussion

### 5.1. Experiment one

These findings reveal substantial variability in both solving time and moves made by participants and across trials. The younger participants didn't tend to solve the cube more quickly and in fewer moves, hinting at a possible age factor that is non-influencing problem-solving speed and efficiency. All participants did improve from one trial to another, however, indicating learning by practice and development of customized strategies.

Figure 9 illustrates the correlation between solving time and moves in all trials. In general, longer solving times were associated with more moves, particularly in subsequent trials. The initial trials had average times and moves, likely due to participants' initial probing. Some participants had increases in both time and moves, potentially due to difficulty or less effective strategies, while others improved by minimizing one or both. Strikingly, more moves did not necessarily imply longer times, indicating participants moved efficiently despite less effective strategies, yet others took longer yet made fewer moves. The diversity of problem-solving strategies and learning's dynamic nature in solving the Rubik's Cube are stressed by the scatter plot, making it crucial to use quantitative and qualitative analysis to grasp these processes.

This experiment demonstrates problem-solving diversity in humans and learning even without training. Future research can include a broader participant population or compare human strategies to strategies by AI-based solvers.

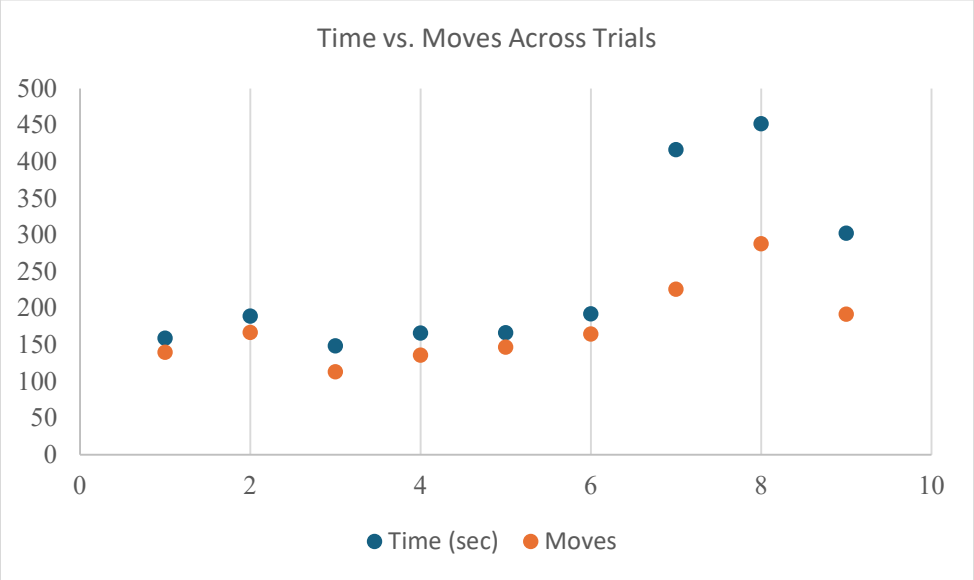


Fig. 7. *Scatter plot showing Time vs. Moves Across Trials.*

### 5.2. Experiment two

- The RL agent operates within a simulated Rubik's Cube environment, where the state represents the current configuration of the cube.
- The agent selects actions (rotations of cube faces) based on its current policy, which is refined over time through learning.
- Rewards are assigned based on progress toward the solved state: moves that bring the cube closer to completion yield higher rewards, while moves that do not contribute to solving the cube result in lower or negative rewards.
- The agent's policy is updated using experiences stored in replay memory, allowing it to learn from both recent and past actions

### *5.2.1. Implementation Details*

- The code is built around a DQN agent, which uses a neural network to estimate the value (Q-value) of possible moves (see Figure 10) in each state.
- The agent employs an epsilon-greedy strategy (see Figure 11) to balance exploration (random moves) and exploitation (best-known moves).

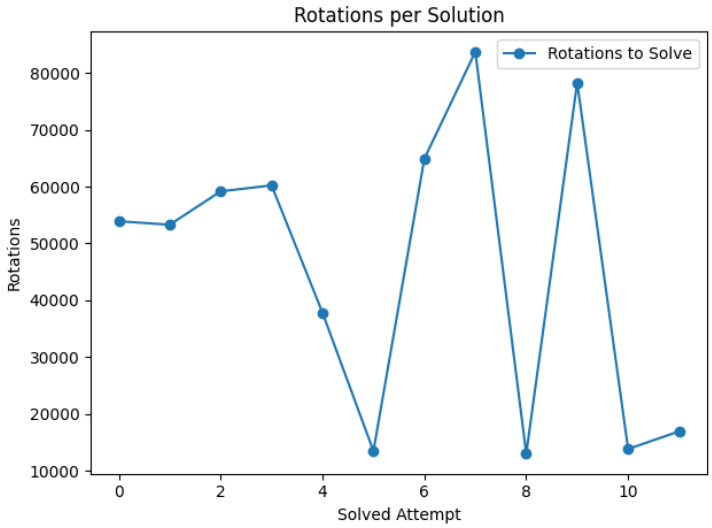


Fig. 8. *Rotation per solution.*

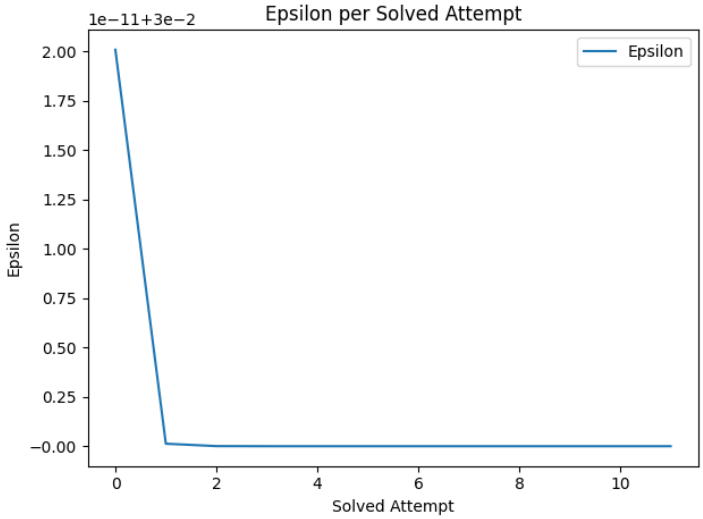


Fig. 9. *Epsilon per solution.*

### *5.2.2. Procedure*

- Each training episode begins with a scrambled cube.
- The agent attempts to solve the cube within a set number of moves.
- After each move, the agent:
  - Observes the new state

  - Receives a reward
  - Stores the experience in replay memory
  - Periodically samples batches from memory to train the neural network
- The agent's performance is tracked via metrics such as the number of correctly colored squares, rewards, randomness (exploration rate), and DQN loss (see Figures 12 and 13).

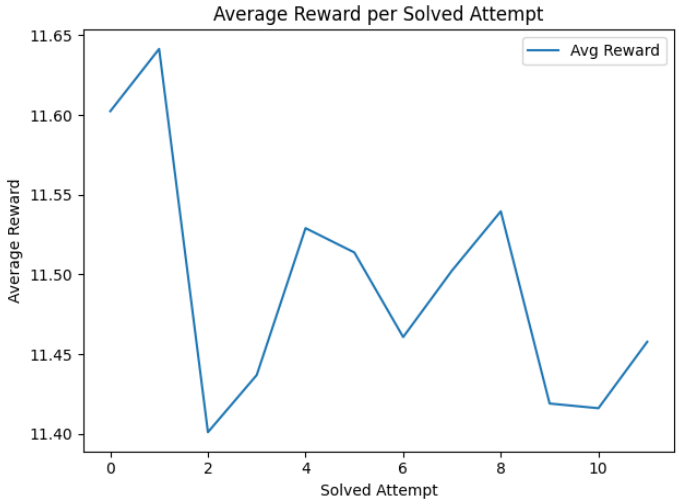


Fig. 10. *Rewards per solving attempt.*

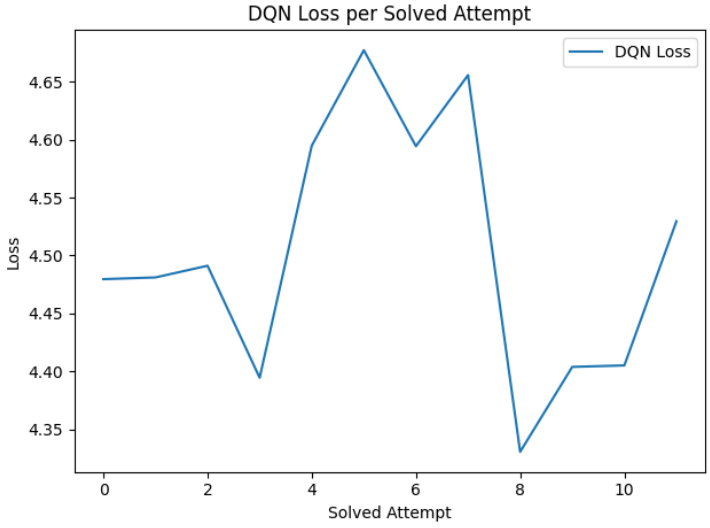


Fig. 11. *DQN loss status per attempt.*

### *5.2.3. Results and Visualization*

- The agent's progress is visualized in real-time, showing the cube's state and learning metrics.

- Over time, the agent improves its ability to solve the cube, requiring fewer moves and achieving higher rewards as its policy becomes more effective.
- The model's weights are saved for future use, allowing continued training or evaluation

This experiment demonstrates the effectiveness of reinforcement learning, specifically DQN, in solving complex combinatorial puzzles like the Rubik's Cube. The agent learns optimal strategies through trial and error, refining its approach to maximize rewards and efficiently reach the solved state.

## 5. Conclusion

The Rubik's Cube, aside from being a best-selling puzzle, itself acts as a viable vehicle for investigating artificial intelligence and human thinking. With its staggering quantity of possible arrangements, the cube provides a perfect environment for problem-solving techniques, learning mechanisms, and search heuristics to be explored. Since its discovery by Ernő Rubik in 1974, the cube has continued to challenge and motivate experts from every walk of life.

Here, we observed how novice participants solved the Rubik's Cube without prior experience. The findings showed varying strategies and learning curvatures across participants. All three became better with each passing day, yet their distinct journey to arriving at the solution indicates flexibility and creativity in problem-solving by humans. This concurs with the view that puzzles like the Rubik's Cube provide insightful information about processes like pattern detection, memory, and decision-making—processes in which humans are superior even without any training.

At the same time, new developments in artificial intelligence exhibit impressive performance in solving the Rubik's Cube through means like DeepCubeA, where deep reinforcement learning in collaboration with search methods are applied to identify optimal moves without any knowledge from humans. Robotic platforms like Dactyl are able to exhibit how complex manipulation problems can be trained entirely with simulation and translated to real-world setup through means like ADR. These systems are nonetheless limited when it comes to generalization, success rate, and computational efficiency, showing the disparity between machine and human intelligence.

This study explored the Rubik's Cube as a platform for investigating both human and artificial problem-solving strategies. Through two distinct experiments, we examined the learning processes of human participants and a reinforcement learning agent. In the first experiment, human participants with no prior experience were able to learn and improve their Rubik's Cube solving abilities within a short period, demonstrating the adaptability and creativity inherent in human cognition. Their progress highlighted the role of self-guided exploration and diverse learning strategies in mastering complex tasks.

The second experiment implemented a Deep Q-Network (DQN) agent to autonomously learn to solve the Rubik's Cube. The agent's performance, as measured by metrics such as exploration rate, DQN loss, number of rotations, and average reward, showed substantial improvement over time. These results demonstrate the effectiveness of reinforcement learning in navigating large, combinatorial state spaces and developing

efficient solution strategies without explicit human instruction.

Together, these findings underscore the Rubik's Cube's value as a benchmark for both cognitive and computational research. While artificial agents can achieve high efficiency and consistency through iterative learning, human solvers continue to exhibit flexibility and ingenuity that remain challenging for current AI systems to fully replicate. Future work may extend these approaches to more complex puzzles or hybrid human-AI collaborative frameworks, further advancing our understanding of learning and problem-solving in both natural and artificial systems.

For further investigation:

1. Expand Participant Diversity and Sample Size: Conduct experiments with a larger and more diverse group of participants to better understand how factors such as age, gender, cognitive style, and prior puzzle experience influence Rubik's Cube problem-solving strategies and learning curves.

2. Integrate Behavioural and Neurocognitive Analysis: Incorporate tools like eye-tracking, EEG, or fMRI to study the cognitive processes underlying human problem-solving during Rubik's Cube tasks, providing deeper insights into attention, memory, and decision-making mechanisms.

| Abbreviation | Term |
|---|---|
| RL | Reinforcement Learning |
| DQN | Deep Q-Network |
| ADR | Automatic Domain Randomization |
| AI | Artificial Intelligence |
| PDB | Pattern Database |
| ADR | Automatic Domain Randomization |
| PDBs | Pattern Databases |
| $x_i$ | Current state of the Rubik's Cube |
| $a$: U, U', L, F, F', U, U', L, F, F' | move applied to the cube |
| $v_{x_i}$ | Estimated value of taking action aa in state $x_i$ |
| $A(xi, a)$ | The new cube state after applying action $a$ to state $x_i$. |
| $R(A(xi, a))$ | Reaching the new state |
| $y_i$ | Maximum value achievable from the state $x_i$ by choosing the best action |
| $y_{Pi}$ | The action x that achieves the maximum value $y_i$ (optimal action). |

## References

1. Agostinelli F., McAleer S., Shmakov A., Baldi P.: *Solving the Rubik's cube with deep reinforcement learning and search*, in Nat. Mach. Intell., Vol. 1, no. 8, 2019, pp. 356–363.

2. Allen, K. et al.: *Using games to understand the mind*. In: Nat. Hum. Behav., vol. 8, no. 6, Jun. 2024, pp. 1035–1043.
3. Brändle, F., Allen, K.R., Tenenbaum, J., Schulz, E.: *Using games to understand intelligence*. In: Proc. Ann. Meeting Cogn. Sci. Soc., 43, 2021.
4. Carlisle, R.P. *Encyclopedia of Play in Today's Society*. Rutgers University, USA. SAGE Publications, 2009.
5. De Castella, T.: *The people who are still addicted to The Rubik's Cube*. In: BBC News Magazine, Apr. 28, 2014.
6. DePaulis T.: *Board games before Ur?*, In: Board Game Stud. J., Vol. 14, 2020, pp. 127–144.
7. *'Driven Mad' Rubik's Nut Weeps on Solving Cube… after 26 Years of Trying*, Daily Mail Reporter, 2009.
8. *Europa. Interview with Ernő Rubik*, 2016. http://www.create2009 .europa.eu/ambassadors/profiles/erno_rubik.html.
9. Ferenc. D.: *How to solve the rubik's cube - beginners method*, https://ruwix.com/the-rubiks-cube/ how-to-solve-the-rubiks-cube-beginners-method/.
10. Frey, A.H., Singmaster D.: *Handbook of Cubik Math*. Enslow Publishers, 1982.
11. Gebhardt, D. et al.: *The cube*. New York: Black Dog & Leventhal Publishers, 2009.
12. Gobet, F., de Voogt, A., Retschitzki, J.: *Moves in Mind: The Psychology of Board Games.* Psychology Press, 2004.
13. Jerome T.: *Rubik's Cube 25 Years on: Crazy Toys, Crazy Times*. The Independent, 2007.
14. Kunkle D., Cooperman. G: *Twenty-six moves suffice for Rubik's cube*, in Proc. of the 2007 Int. Symposium on Symbolic and Algebraic Computation, New York, NY, USA: ACM, Jul. 2007, pp. 235–242.
15. Lee, J.: *Beginner Solution to the Rubik's Cube*, 2008. http://peter.stillhq.com/jasmine/rubikscubesolution.html.
16. McAleer, S., Agostinelli, F., Shmakov, A., Baldi P.: *Solving the Rubik's Cube Without Human Knowledge*, 2019.
17. Newell, A., Simon H.A.: *Human problem solving*. Prentice-Hall, 1972.
18. OpenAI. *Solving Rubik's Cube with a Robot Hand*.
19. Ori J.: *How Do You Beat the Rubik's Cube?*, 2017. https ://ourpa stime s.com/ do beat rubik s cube 65089 60.html.
20. *Rubik's Cube: A Question Waiting to be Answered*, 2014, https://www.youtube.com/watch?v=W1K2j djLhbo.
21. *Rubik's Cube Solver*, 2016, https://rubik s cube solver.com.
22. Suchow, J. W., Griffiths, T., Hartshorne, J. K.: *Workshop on scaling cognitive science*. In: sCognitive Science Society, 2020.
23. van Dulken, S. *Inventing the 20th Century: 100 Inventions That Shaped the World*. NYU Press, 2002.
24. Zeng, D.-X., Li, M., Wang, J.-J., Hou, Y.-L., Lu W.-J., Huang Z.: *Overview of Rubik's Cube and Reflections on Its Application in Mechanism*. In: Chinese Journal of Mechanical Engineering, Vol. 31, no. 1, Dec. 2018, p. 77.