COMPUTOLOGY: Journal of Applied Computer Science and Intelligent Technologies
Volume 3, Issue 2, Jul-Dec 2023, pp. 24-42
DOI: 10.17492/computology.v3i2.2302
https://www.journalpressindia.com/cjacsit
© 2023 Journal Press India

EnigmaScan: Streamlining Matrix Puzzle Solving through Image Recognition

Abhyuday Rai[1]*, Vanshagra Rai*[1]*, Aditya Chaudhary*[1]*, Aditya Mishra*[1]*, Vaibhav Saini*[1] *and Shiv Naresh Shivhare*[1]***

ABSTRACT

Abstract. In an era increasingly shaped by technology, the enduring allure of word puzzles, such as word searches, crosswords, and Sudoku, remains steadfast. Yet, manually tackling these puzzles can prove to be a time-intensive and frequently exasperating endeavor. To meet this challenge, we suggest employing image recognition-based algorithms designed for puzzle-solving, specifically targeting Sudoku, word searches, and crossword puzzles. EnigmaScan is publicly accessible at https://github.com/theNewtonCode/EnigmaScan, allowing users to promptly solve their puzzles in real-time by uploading snapshots of unsolved Sudoku, word searches, or crosswords. This ground-breaking tool is designed to streamline and hasten the puzzle-solving process for users of all proficiency levels, thereby promoting a more enjoyable and accessible interaction with these timeless pastimes. The proposed algorithms exhibit remarkable efficiency, solving these puzzles within seconds and achieving an accuracy of 100%.

Keywords: *Sudoku; Word Search; Crossword; Puzzle Solving; Dancing Links Algorithm; Optical Character Recognition.*

1.0 Introduction

In a world where technology has become an integral part of our daily life, the joy of solving word puzzles, such as word searches, crosswords, and Sudoku, remains ever-present. However, the process of solving these puzzles manually can sometime prove challenging and time-consuming. To provide puzzle enthusiasts with an innovative and efficient solution, we propose the development of an automated puzzle solver that

[1]*School of Computer Science Engineering and Technology, Bennett University, Noida, Uttar Pradesh, India*
**Corresponding author e-mail: shiv827@gmail.com*

harnesses the capabilities of image recognition technology to instantly solve and display results for uploaded puzzles. The primary objective of this puzzle solver is to simplify and expedite the puzzle-solving experience for users by leveraging advanced image recognition algorithms. This proposed puzzle solver, named EnigmaScan, is publicly available and has the option to upload images of unsolved puzzles. It will swiftly process these images to showcase the solved puzzle grids. The significance of our image recognition-powered puzzle solver lies in several key areas:

1. *Enhanced User Experience:* Our puzzle solver simplifies and expedites the process of solving classic puzzles such as word searches, crosswords, and Sudoku. It provides users with immediate and accurate solutions, reducing frustration and enhancing the overall enjoyment of puzzle-solving.
2. *Accessibility:* The EnigmaScan has user-friendly interface and cross-device compatibility makes it accessible to a wide range of users, including those who may not be tech-savvy. This accessibility can introduce a broader audience to the world of word puzzles.
3. *Educational Value:* Word puzzles are known for their cognitive benefits, including improved vocabulary, pattern recognition, and problem-solving skills. EnigmaScan contributes to the educational domain by providing immediate access to accurate solutions for various word puzzles. In educational institutions, students can quickly verify their answers and gain exposure to correct solutions, facilitating a more efficient learning process. While EnigmaScan currently focuses on delivering solved puzzles, the instant availability of accurate solutions can aid students in understanding the correct answers, verifying their own solutions, and improving their puzzle-solving skills.
4. *Time Savings:* Users can save valuable time that would otherwise be spent manually solving puzzles. This is especially significant for individuals who enjoy puzzles but have limited leisure time.
5. *Versatility:* EnigmaScan stands out in terms of versatility, offering a comprehensive solution for a diverse range of word puzzles, including word searches, crosswords, and Sudoku. The criteria used to define the versatility of EnigmaScan include puzzle type support, difficulty level, adaptability, and its unique feature of downloadable results. EnigmaScan excels in solving various word puzzle types, setting it apart from tools that may focus on a single puzzle category. The tool is designed to handle puzzles of different difficulty levels, catering to both novice and experienced puzzle enthusiasts. EnigmaScan’s adaptability to different puzzle structures ensures its effectiveness across a broad spectrum of challenges.

EnigmaScan can serve as a helpful tool for individuals who occasionally get stuck on challenging puzzles or seek confirmation of their solutions. The pro-posed image recognition-powered puzzle solver is significant because it simplifies puzzle-solving, enhances accessibility, offers educational value, saves time, and modernizes classic puzzles. It has the potential to cater to a broad audience and fill a market gap, all while contributing to the advancement of image recognition technology in real-world applications. In a rapidly evolving technological landscape, where digital solutions have permeated every facet of daily life, the enduring appeal of word puzzles, including word searches, crosswords, and Sudoku, remains undiminished. Nevertheless, the process of manually solving these puzzles can often prove to be a daunting and time-consuming endeavour [8, 9, 16]. The primary objective of EnigmaScan is to expedite and streamline puzzle-solving, making it accessible to a broader audience. Users can effortlessly upload images of unsolved puzzles, and the application employs state-of-the-art image recognition algorithms to generate and display the solved puzzle grids [1, 4, 12].

The rest of the paper is organized as follows: Section 2 discusses various existing puzzle solving methods, including their pros and cons. Taking the limitations into consideration, Section 3 introduces the proposed solution for an efficient puzzle solver along with its significance. Moreover, Section 4 presents and discusses the experimental results obtained through the implementation of the proposed algorithms. Finally, the overall work is concluded by mentioning the outcomes and possible future directions in Section 5.

2.0 Related Work

The development of EnigmaScan draws inspiration from a range of notable research papers that have paved the way for innovative applications of image recognition and puzzle-solving methodologies. Breakthroughs in deep learning techniques, particularly ResNets, addressing challenges like the vanishing gradient problem, which have played a pivotal role in image recognition has been presented in [8]. The application of computer vision for Sudoku puzzle solving, laying the foundation for our image recognition-based approach is explored in detail in [16]. Moreover, [1] introduces parallel computing methods for Sudoku solving, inspiring our application’s focus on efficiency and speed. Furthermore, several researchers have contributed by introducing various innovative solutions easy and fast automated puzzle solving [2, 4, 6, 10, 12, 17, 15].

The crossword puzzle-solving system, Proverb, utilizes an open architecture featuring specialized modules designed for various clue types. It integrates concepts from information retrieval, database search, and machine learning. Each module produces lists of potential answers for clues, which are then consolidated and arranged in the puzzle grid by a centralized solver. Proverb demonstrates noteworthy performance, achieving an average accuracy of 95.3% for correct words and 98.1% for correct letters within a time frame of less than 15 minutes per puzzle. This success establishes it as a highly efficient crossword solver [13].

NP-complete nature of generalized Sudoku with symbols, equating it to the classical Hamiltonian cycle problem. A constructive algorithm transforms Sudoku into a sparse Hamiltonian cycle problem, with a subsequent conversion to an undirected version. Practical algorithms for deriving valid Sudoku solutions and reducing the resultant graph size are presented [7]. Constraint Programming (CP) to model and solve the Sudoku puzzle as a constraint satisfaction problem, assessing the performance of different Variable and value selection heuristics in the enumeration phase. Their study serves as a benchmark, offering insights into constraint modeling and solving for complex problems, potentially contributing to the development of new techniques in decision-making scenarios [5].

Approaching crossword puzzle solving as a constraint satisfaction problem with a probabilistic aspect, this perspective recognizes the inherent imprecision in mapping clues to variable domains. By introducing a formal model of constraint satisfaction with probabilistic preferences assigned to variable values, the study outlines two distinct optimization objectives. The first objective involves maximizing the probability of obtaining a correct solution, while the second focuses on maximizing the number of correct words within the solution. An efficient iterative approximation, utilizing dynamic programming, is proposed for the latter, yielding promising results on both real and artificial crossword puzzles [19]. Developing a Deep-Text Recurrent Network (DTRN), this study treats scene text reading as a sequence labeling problem, bypassing character segmentation challenges by leveraging deep convolutional neural networks. The DTRN, incorporating a deep recurrent model with LSTM, excels at recognizing ordered CNN sequences, overcoming limitations of character-independent recognition. Noteworthy features include its capacity to decipher ambiguous words, resilience to image distortions, retention of explicit order information, and independence from predefined dictionaries, leading to substantial advancements in scene text recognition on multiple benchmarks [10]. Novel text detection algorithm focusing on natural images. The method employs edge-enhanced maximally stable extremal regions as the foundational candidates for letters. These regions are subsequently filtered by considering geometric characteristics and stroke width information, effectively eliminating non-text objects from the candidate pool. The algorithm further pairs letters to identify text lines and separates them into words. Evaluation on the ICDAR competition dataset and a mobile document database demonstrates promising experimental results, highlighting the algorithm's efficacy in text detection [3]. These research papers, among others, have been instrumental in shaping the EnigmaScan project, providing the theoretical underpinnings for our innovative solution. The summary of a few existing puzzle solving platforms is presented in Table 1 along with their pros and cons.

Table 1: The Summary of a Few Existing Puzzle Solving Methods

Author	Methodologies	Pros	Cons
ASolver Mobile Puzzle Solver [14]	Utilizes camera recognition for solving puzzles like Rubik's Cube and other mechanical puzzles.	Solving classic word puzzles: Our app is designed specifically for solving word puzzles (word searches, crosswords) using image recognition. Different puzzle genres: While a Solver focuses on mechanical puzzles, our app focuses on classic word puzzles.	Focused on mechanical puzzles.
Sudoku Solver by Py-ImageSearch [18]	Uses OpenCV, Deep Learning, and OCR for automatic Sudoku puzzle solving.	Supports multiple puzzle types: Unlike the mentioned solution that focuses on Sudoku, our app covers word searches, crosswords, and Sudoku. Immediate results for diverse puzzles: Our app provides solutions for various puzzles and displays results rapidly. User-friendly interface: Our app offers an intuitive interface for users to upload and access solutions easily.	Limited to Sudoku puzzles.
Abto Jigsaw Puzzle Solver [20]	Uses computer vision for automatic assembly of jigsaw puzzles.	Focus on word puzzles: Our app is dedicated to solving word puzzles like word searches and crosswords, offering a different puzzle-solving experience. Multiple puzzle types: Our app supports a broader range of puzzles, enhancing its versatility.	Limited to jigsaw puzzles.

Our approach excels by addressing key shortcomings found in existing puzzle-solving apps. Firstly, our method offers versatility by covering various word puzzle types, including word searches, crosswords, and Sudoku, which is not a common feature in most existing solutions that often focus on one specific puzzle type. Secondly, our user-centric approach places an emphasis on user verification before proceeding with puzzle-solving, ensuring accuracy and minimizing errors that might be present in automated processes. This step distinguishes our app from solutions that lack such a confirmation mechanism, thereby enhancing the reliability of results. Lastly, our commitment to user-friendliness, a modern interface, and quick results sets us apart from apps that might have complex or outdated interfaces and slower response times. In sum, our approach not only addresses these significant shortcomings but also offers a comprehensive, accurate, and user-friendly puzzle-solving experience.

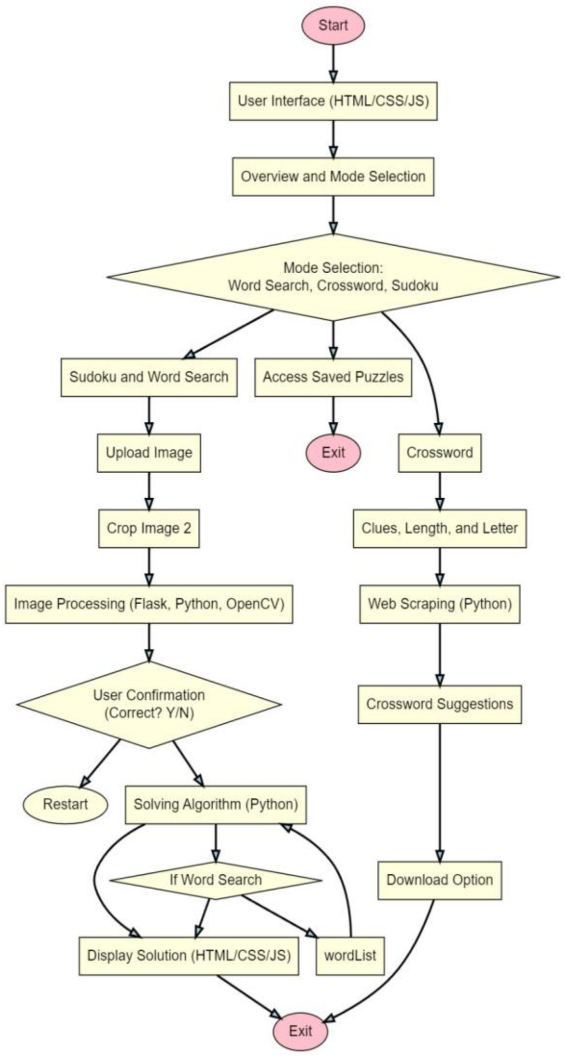
3.0 Proposed Method

The methodology for our image recognition-powered puzzle solver involves a streamlined and user-centric approach. Initially, users will upload puzzle images, which they can select and crop based on their preferences. These images will then undergo a conversion process where they are transformed into matrix representations suitable for puzzle-solving algorithms. Importantly, before proceeding with the solving process, the app will prompt users to confirm the recognized puzzle to ensure accuracy. This confirmation step serves as a crucial quality control mechanism, allowing users to verify that the app correctly identified the puzzle type and layout from their image. Once confirmed, the app will swiftly apply its puzzle-solving algorithms and display the solved puzzle grid as the output, providing users with immediate and accurate results. This user-driven approach prioritizes accuracy, usability, and an efficient puzzle-solving experience.

The process begins with the extraction of puzzle data from user-uploaded images. Our custom extraction algorithm is applied to identify and convert characters, numbers, or words from the images into a matrix representation. Easy OCR plays a pivotal role in accurate character recognition. Our web app incorporates specialized solving algorithms for different puzzle types, namely Sudoku, word puzzles, and crosswords. For Sudoku, we implement best-in-class solving algorithms, ensuring efficiency and accuracy. For word puzzles, we employ algorithms that consider word patterns and recognition results to generate solutions. In the case of crosswords, we plan to utilize web scraping techniques to extract clues and answer lengths, offering suggestions to users. The web app is designed with a user-friendly interface, allowing smooth puzzle uploading and real-time interaction. Users can submit their images via a web-based form or directly from their device's camera. The interface provides an immediate display of solutions and offers options for downloading the results for future reference or sharing. The flowchart of the proposed approach is depicted in Figure 1.

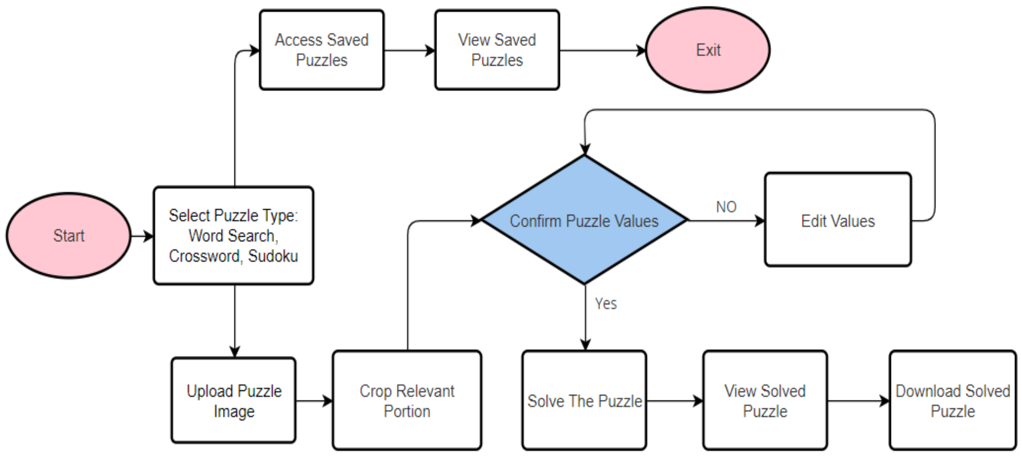
Quality control mechanisms are integrated into the system to verify the accuracy of character recognition and solution generation. Error detection algorithms highlight discrepancies in real-time, allowing users to make corrections promptly. Ensuring optimal performance is crucial to provide users with rapid solutions without delays, even during periods of high usage. Additionally, scalability measures are implemented to accommodate a growing user base. Our development process includes rigorous testing phases to identify and address any bugs or inaccuracies in character recognition and solving algorithms. We actively seek user feedback and incorporate suggestions to enhance the app's usability and reliability.

Figure 1: Flowchart of the Proposed Puzzle Solving Approach



Given the upload of potentially sensitive content, the app prioritizes security and user privacy. Robust security measures are implemented to protect user data and ensure that puzzles remain private and inaccessible to unauthorized individuals. The methodology is a dynamic process that emphasizes continuous improvement. As new advancements in character recognition and puzzle-solving algorithms emerge, the app will be updated to maintain accuracy and relevance. Our web app utilizes web development tools for building the frontend interface and seamlessly integrates with the Flask framework for effective web application development [Figure 2]. This combination of tools ensures a robust, responsive, and interactive user experience.

Figure 2: The User Perspective and the Sequence of Events during the Puzzle Upload and Solving Process of the Proposed Puzzle Solving Approach



Algorithm 1: Algorithm for Word Search Solver

```
      num_rows = None
      num_cols = None
      directions = [[-1, 0], [1, 0], [1, 1], [1, -1], [-1, -1], [-1, 1], [0, 1], [0, -1]]
1.    **procedure** INIT
2.    Initialize num_rows, num_cols to None
3.    Initialize directions list
4.    **end procedure**
5.    **procedure** SEARCH_2D (grid, start_row, start_col, word)
6.    Check if the first character of the word matches the grid at the specified position
7.    Check for the word in all directions
8.    **Parameters:**
9.    grid: 2D grid to search
```

```
10. start_row: starting row index
11. start_col: starting column index
12. word: word to search
13. **Returns:** True if the word is found, False otherwise
14. **end procedure**
15. **procedure** ENIGMA_SEARCH(grid, word)
16. Search for a word in the 2D grid using search_2d
17. **Parameters:**
18. grid: 2D grid to search
19. word: word to search
20. **Returns:** Tuple (row, col) if the word is found, None otherwise
21. **end procedure**
```

Algorithm 2: Algorithm for Crossword Solver

```
    Hash Table: *hash_table*
1.  **procedure** INIT
2.  Initialize the crossword solver's hash table
3.  **end procedure**
4.  **procedure** LOAD_WORDLIST(filename)
5.  Load the wordlist from the specified file into the hash table
6.  **Parameters:** *filename* (name of the file containing the wordlist)
7.  **end procedure**
8.  **procedure** SEARCH_WORDS(length, known_letters)
9.  Search for words based on the specified length and known letters
10. **Parameters:**
11.  - *length* (desired length of the words)
12.  - *known_letters* (known letters in the word, represented by '$' for unknown)
13. **Returns:**
14.  - *highly_likely* (list of highly likely words with a probability of 50)
15.  - *likely* (list of likely words with a probability between 25 and 50)
16.  - *less_likely* (list of less likely words with a probability between 2 and 25)
17.  - *least_likely* (list of least likely words with probability less than or equal to 2)
18.  **end procedure**
```

3.1 User Interface Design Principles in EnigmaScan

This section delves into the intricacies of EnigmaScan's user interface (UI) design, which is deeply rooted in gaming UI concepts. The application leverages cyberpunk aesthetics to not only enhance visual appeal but also to optimize user engagement. The thoughtful integration of design elements contributes to an immersive and enjoyable puzzle-solving experience.

- *Cyberpunk Aesthetics and Visual Hierarchy:* The choice of neon colors and dynamic glow effects in EnigmaScan's UI serves a dual purpose. Beyond aesthetics, these elements play a crucial role in establishing visual hierarchy. Neon colors draw attention to key UI elements, guiding users seamlessly through the application. The dynamic glow effects not only contribute to the cyberpunk theme but also enhance the visibility of critical components, ensuring a user-friendly and visually engaging experience.
- *Dark Theme for Comfortable Interaction:* The incorporation of a dark theme in EnigmaScan aligns with the cyberpunk genre and serves a practical purpose. The dark background minimizes eye strain during prolonged usage, prioritizing user comfort. This design choice enhances the overall usability of the application, especially for users who engage with puzzles for extended periods.

Algorithm 3: Algorithm for Sudoku Solver

Data:

```
1. Node structure: Node *left, Node *right, Node *up, Node *down, Node *head, int size, int rowID[3]
2. Constants: $MAX_K$=1000, $SIZE$=9, $SIZE_SQUARED=SIZE \times SIZE$, $SIZE_SQRT= sqrt((double)SIZE)$, $ROW_NB=SIZE \times SIZE \times SIZE$, $COL_NB$=4 $\times SIZE \times SIZE$
3. Global Variables: Node Head, $Node * HeadNode = \&Head$, $Node * solution[MAX\_K]$, $Node * orig_values[MAX\_K]$, $boolmatrix[ROW_NB][COL_NB]$=0
```

Procedures:

```
1. **procedure** COVERCOLUMN(col)
2. Cover the specified column in the matrix
3. **end procedure**
4. **procedure** UNCOVERCOLUMN(col)
5. Uncover the specified column in the matrix
6. **end procedure**
7. **procedure** SEARCH(k)
8. Recursive backtracking search for a solution
9. **Parameters**: $k$ (current depth in the search)
```

```
end procedure
procedure BUILDSPARSEMATRIX(matrix)
    Build the initial sparse matrix containing all possibilities
    Parameters: matrix (2D array representing the sparse matrix)
end procedure
procedure BUILDLINKEDLIST(matrix)
    Build a toroidal doubly linked list out of the sparse matrix
    Parameters: matrix (2D array representing the sparse matrix)
end procedure
procedure TRANSFORMLISTTOCURRENTGRID(Puzzle)
    Covers values that are already present in the grid
    Parameters: Puzzle (2D array representing the Sudoku grid)
end procedure
procedure MAPSOLUTIONTOGRID(Sudoku)
    Map the solution to the Sudoku grid
    Parameters: Sudoku (2D array representing the Sudoku grid)
end procedure
procedure PRINTGRID(Sudoku)
    Print the Sudoku grid
    Parameters: Sudoku (2D array representing the Sudoku grid)
end procedure
procedure SOLVESUDOKU(Sudoku)
    Solve the Sudoku puzzle using the Dancing Links algorithm
    Parameters: Sudoku (2D array representing the Sudoku grid)
end procedure
```

- *Typography for Readability and Aesthetic Harmony:* Typography is a pivotal element in EnigmaScan's UI design. Carefully selected right-sized fonts strike a balance between readability and aesthetic harmony. The fonts contribute to a cohesive visual identity while ensuring that users can effortlessly comprehend the presented information. This meticulous consideration of typography enhances the overall user experience.
- *Layout Simplicity to Reduce Cognitive Load:* The UI layout in EnigmaScan adheres to simplicity principles to minimize cognitive load. By presenting information in a straightforward manner, users can navigate the application with ease. The

streamlined layout contributes to stress-free interactions, allowing users to focus on puzzle-solving without unnecessary distractions.

- *Gaming-Inspired UI for Enhanced User Satisfaction:* EnigmaScan's UI design goes beyond mere functionality, aiming to elevate the puzzle-solving experience. By integrating gaming-inspired UI elements, the application immerses users in an environment that is both enjoyable and visually stimulating. This thoughtful design approach aligns with established UI principles, ultimately enhancing usability and contributing to overall user satisfaction.

4.0 Experimental Results and Discussion

In this study, we conducted an experiment to evaluate the performance of various puzzle-solving algorithms. Our goal is to choose the most effective algorithm for our puzzle-solving web application, allowing users to solve Sudoku and word search puzzles. As for crosswords, we have not yet decided on the use of any specific algorithms. The primary objective of this experiment is to assess the accuracy and efficiency of different algorithms in solving puzzles. We aim to identify an algorithm that not only provides accurate solutions but also does so in a timely manner. The selected algorithm will be integrated into our web application to enhance the user experience. The experiment encompasses two main puzzle types, i.e., Sudoku and word search puzzles. Each of these puzzles presents distinct challenges and requires specialized solving techniques.

The performance and efficacy of the proposed puzzle solver are evaluated in terms of accuracy and time efficiency. Accuracy refers to the ability of an algorithm to provide correct solutions, while time efficiency measures how quickly the algorithm can generate these solutions. We have selected a range of algorithms written in two major programming languages for evaluation, which are C++ and Python. Each algorithm is assessed based on its performance in solving the specified puzzle types. Table 2 presents the performance of the proposed algorithms in terms of accuracy and time compared to existing algorithms. Processes a given word search grid and a list of words, systematically searching for those words both horizontally and diagonally in multiple directions. When a word is found, it is highlighted by surrounding it with asterisks ('*') and converting it to uppercase. To ensure thorough search coverage, the function also rotates the grid in different directions. The result is a modified word search grid with the found words clearly marked, aiding in their identification.

Designed to search for a given word in a 2D grid (matrix) in all eight possible directions (horizontally, vertically, and diagonally). It uses a nested loop to iterate through each cell in the grid and checks for matches in all directions starting from that cell. If a match is found, it returns the position where the word was found; otherwise, it reports that the pattern was not found in the grid. The algorithm uses a simple backtracking algorithm to solve a 9× Sudoku puzzle. It repeatedly attempts to fill empty cells with numbers from 1 to 9 while ensuring that the placement is valid. If it encounters a conflict, it backtracks to the previous cell and continues the search. If a solution exists, it prints the solved Sudoku board; otherwise, it reports that no solution can be found. In solving Sudoku puzzles, the algorithm can be employed to efficiently search for solutions. The grid of a Sudoku puzzle is transformed into an exact cover problem, where each cell represents a constraint that must be satisfied. By converting the Sudoku grid into a matrix of constraints, the Dancing Links algorithm [11] then systematically searches for solutions by employing a technique known as "Algorithm X." This approach enables the algorithm to efficiently explore possible solutions and backtrack as needed, ultimately leading to the successful solving of Sudoku puzzles.

4.1 Time savings comparison

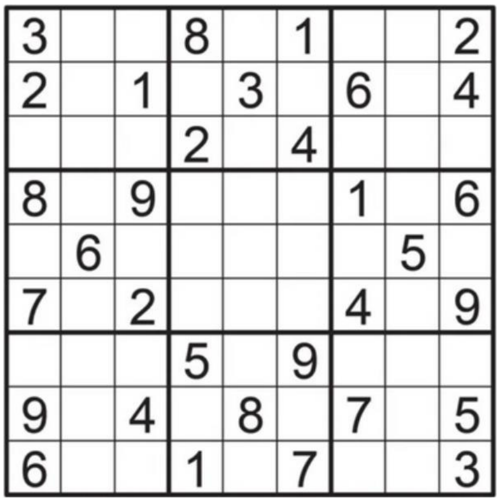
EnigmaScan vs. Manual Solving Taking into account user interactions, including confirming and editing puzzles, the entire process with EnigmaScan is streamlined to take only a few seconds. This efficient workflow aims to provide users with rapid solutions, minimizing the time spent on puzzle-solving tasks.

As a point of reference for manual solving, insights from a Medium article by [21], an average puzzle player, have been included. According to Vezina, he typically solves hard puzzles in under 10 minutes, with an average time of around 12 minutes. This comparison serves to highlight the significant time savings achieved through EnigmaScan, emphasizing its efficiency in providing quick and accurate puzzle solutions.

After careful consideration and performance evaluation, we have decided to choose the dancing links algorithm for our Sudoku solving feature, and for word search, we decided to use our algorithm based on a nested loop grid search. The decision is based on their high accuracy and the least execution time, and the efficiency they demonstrate in solving the respective puzzles. The combination of high accuracy and fast solving speed ensures a reliable and user-friendly experience for our app's users.

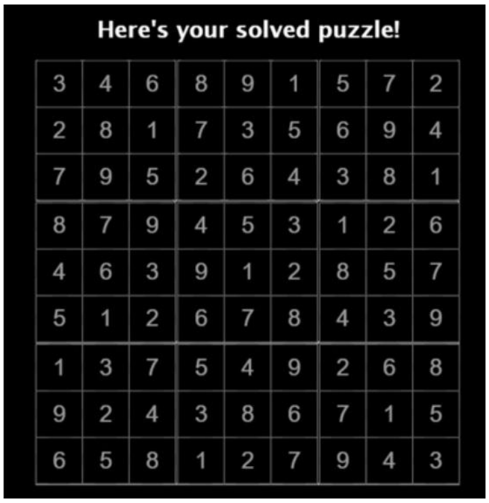
Figure 3: Sudoku Puzzle Used and Sudoku Puzzle Solved

(a)



3			8		1			2
2		1		3		6		4
			2		4			
8		9				1		6
	6						5	
7		2				4		9
			5		9			
9		4		8		7		5
6			1		7			3

(b)

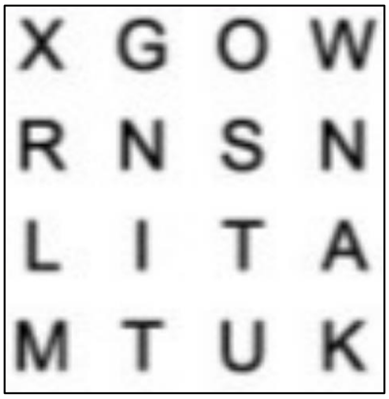


Here's your solved puzzle!

3	4	6	8	9	1	5	7	2
2	8	1	7	3	5	6	9	4
7	9	5	2	6	4	3	8	1
8	7	9	4	5	3	1	2	6
4	6	3	9	1	2	8	5	7
5	1	2	6	7	8	4	3	9
1	3	7	5	4	9	2	6	8
9	2	4	3	8	6	7	1	5
6	5	8	1	2	7	9	4	3

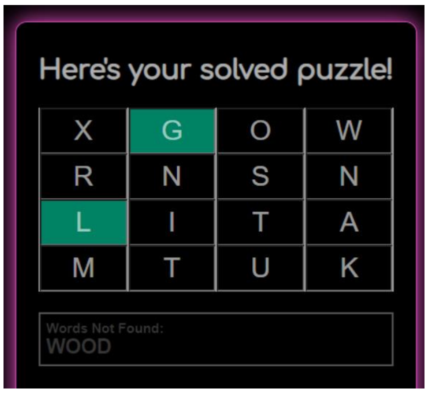
Figure 4: Word Search Puzzle Used and Word Search Puzzle Solved

(a)



X	G	O	W
R	N	S	N
L	I	T	A
M	T	U	K

(b)



Here's your solved puzzle!

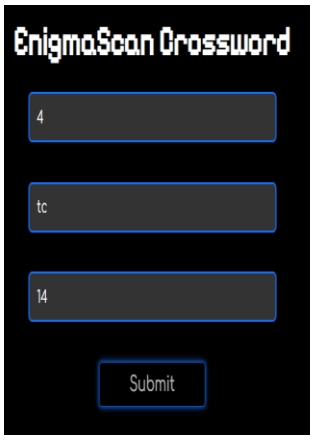
X	G	O	W
R	N	S	N
L	I	T	A
M	T	U	K

Words Not Found:
WOOD

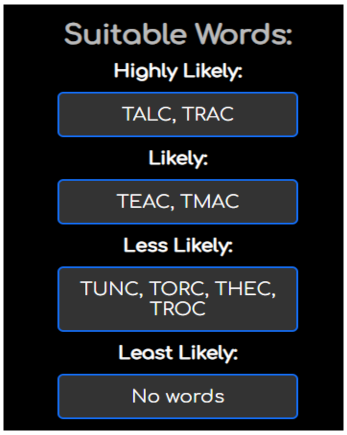
EasyOCR Implementation: We used EasyOCR for our Optical Character Recognition (OCR) feature, which is an open-source OCR library designed to simplify the process of extracting text from images. It is built on top of the PyTorch deep learning framework and provides pre-trained models for various languages. EasyOCR employs a combination of deep learning techniques, particularly Convolutional Neural Networks (CNNs), to recognize and understand characters in images. The library works by segmenting an input image into individual characters or text regions and then using the trained neural network models to recognize the content of each segment. EasyOCR supports a variety of languages and can be fine-tuned for specific use cases. The following algorithm illustrates the implementation within the SudokuOcr class:

Figure 5: Crossword Puzzle Used and Crossword Puzzle Solved

(a)



(b)


Suitable Words:

Highly Likely:

TALC, TRAC

Likely:

TEAC, TMAC

Less Likely:

TUNC, TORC, THEC, TROC

Least Likely:

No words

Table 2: Performance Comparison of Puzzle Solving Algorithms (Including Language)

Algorithm	Language	Puzzle Type	Accuracy	Time (s)
Dancing Links	Python	Sudoku	100%	$4.7e^{-2}$
Dancing Links	C++	Sudoku	100%	$1.7e^{-3}$
Our Algorithm	Python	Word Search	100%	$9.7e^{-5}$
Our Algorithm	C++	Word Search	100%	$1.8e^{-5}$

In EnigmaScan, EasyOCR is implemented as part of a larger puzzle-solving application. The class from our EasyOCR code is instantiated with the language parameter set to 'en' (English), indicating that the OCR is configured for English language recognition. The readtext method of the class is then utilized to perform OCR on cropped cell images extracted from a Sudoku puzzle. This method returns a list of detected text and their corresponding bounding boxes. The code utilizes the OCR output to extract recognized digits and form a matrix representing the Sudoku puzzle. By incorporating EasyOCR, the implementation simplifies the integration of OCR capabilities into the puzzle-solving process, abstracting away the complexities of training a custom OCR model and making it easier to apply OCR in different scenarios. Figure 3, 4, and 5 show the examples of how our proposed puzzle solver solves the sudoku, word search and crossword puzzles respectively.

5.0 Conclusion

To cater to the ardent puzzle-solving community and simplify this experience, we propose the creation of a web application, EnigmaScan. This innovative application harnesses the power of advanced image recognition technology to swiftly decipher and present solutions for puzzles uploaded by users. In conclusion, our web application provides a comprehensive solution for puzzle enthusiasts, integrating word search, Sudoku, and crossword puzzle solvers with the help of an innovative OCR feature. The key achievements include a user-friendly interface, successful puzzle-solving modules, and the practicality of extracting puzzles from images. The incorporation of OCR not only enhances accessibility but also introduces a level of innovation to the puzzle-solving domain. Regarding a comparative study, it is essential to clarify that EnigmaScan is built upon what we consider the best existing algorithms available during the development phase. While we acknowledge the absence of a direct comparative study with other existing methods, EnigmaScan stands as a testament to its uniqueness and commitment to providing users with an efficient and accurate puzzle-solving experience. As we move forward, the application stands as a testament to the synergy of entertainment and technology. Future enhancements may include refining algorithms, improving the OCR feature, introducing additional puzzle types, and collaborating with the community for continuous improvement.

Algorithm 4: Sudoku OCR Algorithm

```
Data:
    1:  image_path, reader, grid_size, cell_images
Functions:
    1:  procedure INIT
    2:  Initialize image_path, reader, grid_size, and call extract_cell_images
    3:  end procedure
    4:  procedure EXTRACT_CELL_IMAGES
    5:  Load Sudoku image, determine grid dimensions, and calculate cell size
    6:  Initialize empty list cell_images and loop through grid cells
    7:  Crop cells and add to cell_images
```

```
8:  **Returns:** cell_images
9:  **end procedure**
10: **procedure** CROPIMAGE(input_image)
11: Resize input_image and crop using specified dimensions
12: **Returns:** cropped image
13: **end procedure**
14: **procedure** SOLVE_SUDOKU
15: Initialize empty list matrix
16: **for** each cell_image in cell_images **do**
17: Use EasyOCR to read text and append result to matrix
18: **end for**
19: **Returns:** result of list_to_matrix(matrix, grid_size)
20: **end procedure**
21: **procedure** LIST_TO_MATRIX(1st, n)
22: Initialize empty list matrixq and loop through 1st with step n
23: **Returns:** matrixq
24: **end procedure**
```

References

[1] Berg, B., Høgskar, D., & Torbjørnsen, Ø. (2003). A fast parallel algorithm for solving sudoku puzzles. In *Proceedings of the Fourth International Symposium on Artificial Intelligence and Mathematics*.

[2] Bradski, G., & Kaehler, A. (2000). Opencv: A general-purpose computer vision library. *Dr. Dobb's Journal of Software Tools, 25*(11), 120-123.

[3] Chen, H., Tsai, S.S., Schroth, G., Chen, D.M., Grzeszczuk, R., & Girod, B. (2011). Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In *2011 18th IEEE International Conference on Image Processing, 2609-2612*. IEEE.

[4] Colton, S., Bacardit, J., & Ross, P. (2007). An efficient and flexible algorithm for solving crossword puzzles. *AI Communications, 20*(3), 181-189.

[5] Crawford, B., Castro, C., & Monfroy, E. (2009). Solving sudoku with constraint programming. In *20th International Conference, MCDM 2009, Chengdu/ Jiuzhaigou, China, June 21-26, 2009*. Proceedings (pp. 345-348). Springer.

[6] Damas, F.B.V., & Oliveira, M.H.M.B. (2004). A survey of constraint programming techniques for the sudoku puzzle. In *European Conference on Artificial Intelligence* (pp. 735-739).

[7] Haythorpe, M. (2016). Reducing the generalised sudoku problem to the hamiltonian cycle problem. *AKCE International Journal of Graphs and Combinatorics*, *13*(3), 272-282.

[8] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

[9] Hutchings, R.A., & Pothen, A. (2019). Solving large sudoku puzzles faster. *arXiv preprint arXiv:1905.12965*.

[10] Jaderberg, M., Simonyan, K., & Zisserman, A. (2014). Reading scene text in deep convolutional sequences. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[11] Knuth, D.E. (2000). Dancing links. arXiv:cs.DS/0011047 v1.

[12] Kusakari, K., & Hashimoto, J. (2010). Solving and rating large-scale sudoku problems via partitioning. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming* (pp. 297-310).

[13] Littman, M.L., Keim, G.A., & Shazeer, N. (2002). A probabilistic approach to solving crossword puzzles. *Artificial Intelligence*, *134*(1-2), 23-55.

[14] Jam Soft LLC (n.d.): Asolver>let's solve the puzzle. Retrieved from https://apps.apple.com/us/app/asolver-lets-solve-the-puzzle/id1505663005

[15] Mishra, A., Alahari, K., & Jawahar, C.V. (2013). Robust text detection in natural images with edge-enhanced maximally stable extremal regions. arXiv preprint arXiv:1312.4894.

[16] Nguyen, A.D., Do, T.T., Nguyen, T., & Cheung, N.M. (2017). Sudokuvision: Solving and explaining sudoku puzzles using vision. arXiv preprint arXiv:1704.03004.

[17] Prosser, P. (1998). Solving crossword puzzles as a constraint satisfaction problem. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, (vol. 98, pp. 429-433).

[18] Rosebrock, A. (2020): Opencv sudoku solver and ocr. Retrieved from https://pyimagesearch.com/2020/08/10/opencv-sudoku-solver-and-ocr

[19] Shazeer, N.M., Littman, M.L., & Keim, G.A. (1999). *Solving crossword puzzles as probabilistic constraint satisfaction*. AAAI Press, Palo Alto: California USA.

[20] Abto Software (2019). Computer vision powers automatic jigsaw puzzle solver. Retrieved from https://www.abtosoftware.com/blog/computer-vision-powers-automatic-jigsaw-puzzle-solver

[21] Vezina, M. (2017). On project estimation or how long does it take to solve a sudoku. Retrieved from https://medium.com/@m_vezina/on-project-estimation-or-how-long-does-it-take-to-solve-a-sudoku-881dc41d341e