An Integrated System For Solving Sudoku Puzzles Based On Digital Image Processing And Convolution Neural Networks

Rizky Yanuar Kristianto , Muhammad Abduh, Hasanuddin Al-Habib [a)]

Data Science Department, Faculty Of Mathematics And Natural Science,
State University of Surabaya Jl. Ketintang, 60231, Surabaya

[a)] Corresponding author: hasanuddinhabib@unesa.ac.id

Abstract. The popular puzzle game Sudoku challenges players to fill a 9x9 grid with digits from 1 to 9 so that each row, column, and 3x3 sub-grid contains all the digits without repetition. This paper explores an integrated approach leveraging advanced image processing and artificial intelligence techniques to automate Sudoku solving. Initially, digit classification using convolutional neural networks (CNNs) identifies and distinguishes digits on the Sudoku board. The system accurately locates and isolates Sudoku squares within images by implementing contour detection and perspective transformation. Finally, a backtracking algorithm ensures efficient and precise placement of numbers to solve the puzzle. Experimental results demonstrate high accuracy in digit recognition and Sudoku solving, showcasing the effectiveness of integrating cutting-edge technologies to enhance problem-solving capabilities in logic-based puzzles.

INTRODUCTION

The puzzle Sudoku has become the passion of many people worldwide in the past few years. The exciting fact about Sudoku is that it is a trivial puzzle. Sudoku, a popular puzzle game, involves filling the boxes with numbers from 1 to 9 so that each row, column, and 3x3 box contains only one number from 1 to 9 [1]. The goal of a Sudoku puzzle is to complete the empty spaces within a 9x9 grid, which is divided into nine smaller 3x3 sections, by placing the digits 1 through 9 so that each digit appears only once in every row, column, and a 3x3 section [2]. This challenge tests logic and problem-solving skills and requires a deep understanding of patterns and order. The complexity of these puzzles is often hidden behind their simplicity, requiring players to use various strategies and techniques to reach a solution. From simple elimination analysis to more advanced search methods, solving Sudoku is an exercise in patience, attention to detail, and critical thinking [3].

In recent years, Sudoku researchers and enthusiasts have developed various algorithms to solve Sudoku efficiently. These algorithms use various techniques, including backtracking, constraint propagation, and dancing links, among others, to find the solution to a given puzzle. However, there is no consensus on which algorithm is the most efficient for solving Sudoku. One interesting approach is to combine image processing and number recognition into the Sudoku-solving process. Based on this research [4] [5], this technology automatically recognizes numbers from Sudoku images taken from various sources, such as photographs or scans. Then, using pattern recognition algorithms and artificial intelligence, we can analyze the image, identify the numbers present, and determine the missing numbers. This approach speeds up the puzzle-solving process and opens up opportunities to understand and apply more complex and efficient strategies. In this paper, we combine these cutting-edge technologies with traditional algorithms like backtracking to build a robust method for solving puzzles efficiently and precisely.

RESEARCH AND METHODS

This research employs a sequence of methodological steps comprising three primary stages: classifying digits to identify and distinguish the numbers on the Sudoku board, detecting the Sudoku board to recognize and pinpoint the positions of Sudoku squares in the image, and solving Sudoku using a pre-configured algorithm. Each stage is meticulously designed to ensure that the data utilized in this analysis is accurate, representative, and primed for subsequent processing using suitable analytical techniques. The primary purpose of each stage is to achieve efficient automated Sudoku puzzle-solving.

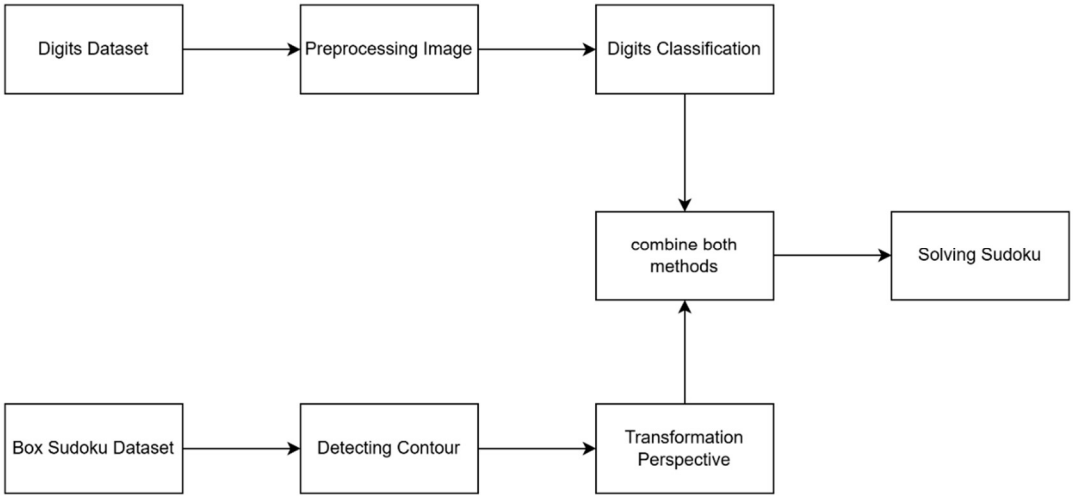


Figure 1. Flowchart Methode

Digits Classification

The first stage in the methodology is digit classification, which is crucial for identifying and differentiating the numbers on the Sudoku board. This step employs advanced image processing techniques to accurately separate and recognize each digit within the Sudoku boxes. The flow of this stage is illustrated in Figure 2.

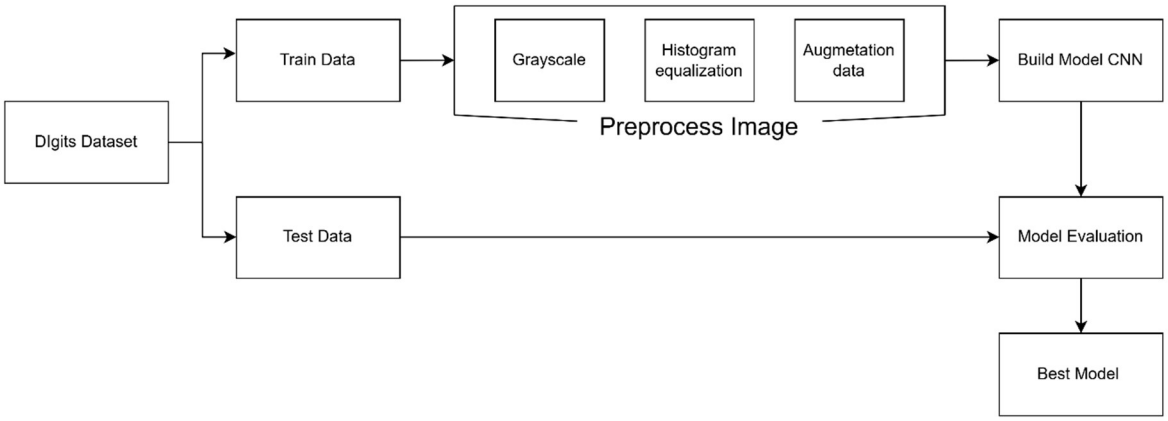


Figure 2. Flowchart Digits Classification

Digits Dataset

For this study, we utilized two primary datasets: Chars74K and MNIST. The Chars74K dataset is designed explicitly for numeric character recognition and comprises over 74.000 digit images sourced from three distinct origins: digits extracted from natural scenes, hand-drawn digits using a tablet PC, and digits generated from computer fonts. Each digit is classified into 10 categories, corresponding to the Hindu-Arabic numerals 0 through 9 [6]. In addition to Chars74K, we employed the MNIST dataset, which is a well-established collection of over 70.000 handwritten digit images [7]. The MNIST dataset is highly regarded in number recognition research due to its reliability and widespread use.

By integrating these various datasets, we achieved higher accuracy in digit recognition. The Sudoku images used in our experiments were sourced from the public domain, ensuring that our model was tested with a broad range of real-world examples. Combining these datasets aims to enhance data richness and boost model performance, much like combining knowledge from different sources to gain a better understanding. This approach helps address data scarcity and improves the model's accuracy in machine learning.

Figure 3. Sample Image from Char74 dataset

Preprocess Image

Before training the convolutional neural network (CNN) model, the images were processed to align with the format required by the model. Initially, the images were converted to grayscale using the colour conversions function of the OpenCV library [8]. This efficient process of converting colour images (RGB) to grayscale, a fundamental step in digital image processing, significantly streamlines the process. Grayscale images require just one 8-bit channel per pixel instead of the three channels (Red, Green, Blue) used in RGB images, effectively reducing the file size to about one-third of the original. This reduction simplifies and accelerates the processing of the image. One of the most straightforward methods for converting an image to grayscale involves taking the average of the three colour values (Red, Green, and Blue) for each pixel [9]:

$$Gray = \frac{(R + G + B)}{3} \tag{1}$$

Nevertheless, this method often results in images that look unnatural, as the human eye is more sensitive to green than to red or blue. To produce a more accurate representation, a different algorithm is implemented [10]:

$$Gray = 0.299 \times R + 0.587 \times G + 0.114 \times B \tag{2}$$

In image processing, global thresholding stands out as the most straightforward technique for separating objects from the background. This method involves selecting a threshold value, T, within the range of 0 to 255. The chosen threshold is then used to classify each pixel in the image. Specifically, if the pixel's intensity is less than or equal to T, the pixel is assigned a value of 0, representing black. Conversely, if the pixel's intensity exceeds T, it is assigned a value of 1, indicating white. The resulting binary image DST(x) can be represented as follows:

$$dst(x) = \begin{cases} 0 \; Scr(x) \leq T \\ 1 \; otherwise \end{cases} \tag{3}$$

Gaussian blur is a widely used technique in image processing to effectively reduce noise introduced during camera image capture. This method is based on the Gaussian function, which describes a bell-shaped curve that emphasizes central values and diminishes outliers. The Gaussian function is formulated as follows :

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{4}$$

P(x) represents the Gaussian probability density function, x is the variable of the function, μ is the mean (average) value of the distribution, and σ is the standard deviation, determining the spread or width of the distribution.

Afterwards, we applied a histogram equalization technique to improve the contrast of the images. This process reduced pixel intensity variations, resulting in a more accurate representation for digit recognition. To enhance the model's robustness and detect variations in the data, we implemented data augmentation using the

ImageDataGenerator from Keras. [11], we can modify the training images by dragging, zooming, rotating, and cropping. This helps the model recognize handwritten digits in different positions and conditions.

Build Model CNN

The convolutional neural network (CNN) adopted in this study comprised the following layers:

Convolution Layer 1: The input images were 32×32 pixels in size. The first convolution layer was comprised of sixty 5×5 kernels (also referred to as filters). In other words, there are 5×5×60+60=1,560 weights that need to be updated at the same time in this layer. These 60 filters will generate images of 60 corresponding feature maps. The convolution operations do not alter the dimensions of the images, thus maintaining a resolution of 32×32 pixels.

Convolution Layer 2: The dimensions of the input images remain 32×32 pixels. This layer also employs sixty 5×5 kernels. This implies that there are 5×5×60+60=1,560 weights that must be trained and updated in the second convolution layer. The output remains 32×32 pixels.

The first pooling layer performs the initial downsampling, reducing the sixty 32×32 pixel images to sixty 16×16 pixel images through a max-pooling operation with a pool size of 2×2. The input images now have a size of 16×16 pixels. The third convolution layer employs thirty 3×3 kernels. This implies that there are 3×3×30+30=8703×3×30+30=870 weights that must be trained and updated in this layer. The images remain 16×16 pixels.

Convolution Layer 4: The input images are still 16×16 pixels. This layer also employs thirty 3×3 kernels, resulting in a total of 8703×3×30+30=870 weights that must be trained and updated. The size of the images remains 16×16 pixels.

Pooling Layer 2: The second pooling layer performs a second downsampling operation, reducing the thirty 16×16 pixel images to thirty 8×8 pixel images. This is achieved through a max-pooling operation with a pool size of 2×2 and strides of 2×2. Dropout (0.5) is added to this layer to randomly abandon 50% of the neurons in each training iteration, thus preventing overfitting.

Flattening Layer: This layer converts the thirty 8×8 images output by Pooling Layer 2 into a one-dimensional array (including a total of 8×8×30=1,9208×8×30 = 1,9208×8×30=1,920 data items), which correspond exactly to the 1,920 neurons in the flattening layer.

Hidden Layer: A hidden layer with 500 neurons is established, requiring 1,920×500+500=960,500 weights to be trained. Dropout(0.5) is added to this layer to randomly abandon 50% of the neurons in each training iteration to prevent overfitting.

Output Layer: This layer contains a total of 10 neurons corresponding to the 10 digits 0-9. The feature values and true values of the pre-processed data in the dataset were input into the CNN model for training. The CNN model is shown in the summary table.

The trained model was tested using a variety of computer fonts to evaluate its predictive capabilities. The results demonstrated that the CNN model had a high accuracy rate, indicating its effectiveness in digit recognition tasks.

Table 1. Summary Model

Layer (type)	*Output Shape*	*Param #*
conv2d_8 (Conv2D)	(None, 32, 32, 60)	1560
conv2d_9 (Conv2D)	(None, 32, 32, 60)	90060
max_pooling2d_4 (MaxPooling2D)	(None, 16, 16, 60)	0
conv2d_10 (Conv2D)	(None, 16, 16, 30)	16230
conv2d_11 (Conv2D)	(None, 16, 16, 30)	8130
max_pooling2d_5 (MaxPooling2D)	(None, 8, 8, 30)	0
dropout_4 (Dropout)	(None, 8, 8, 30)	0
flatten_2 (Flatten)	(None, 1920)	0
dense_4 (Dense)	(None, 500)	960500
dropout_5 (Dropout)	(None, 500)	0
dense_5 (Dense)	(None, 10)	5010

Total parameters: 1081490
Trainable parameter: 1081490
Non-Trainable parameters: 0

The performance of the convolutional neural network (CNN) model was evaluated using key metrics, including training loss, validation loss, training accuracy, and validation accuracy, over 30 epochs.

As illustrated in Figure (4a) below, the training loss decreases significantly in the initial epoch and continues to decline steadily, indicating that the model is learning effectively from the data. The validation loss also demonstrates a consistent decrease, indicating that the model is capable of generalizing well to unseen data. Figure (4b) demonstrates the rapid improvement in training accuracy, which quickly approaches 100%. The validation accuracy also stabilizes at a high level, highlighting the model's ability to accurately recognize digits even in the validation set. Training and validation accuracy over 30 epochs. The graphs below illustrate the progression of these metrics.

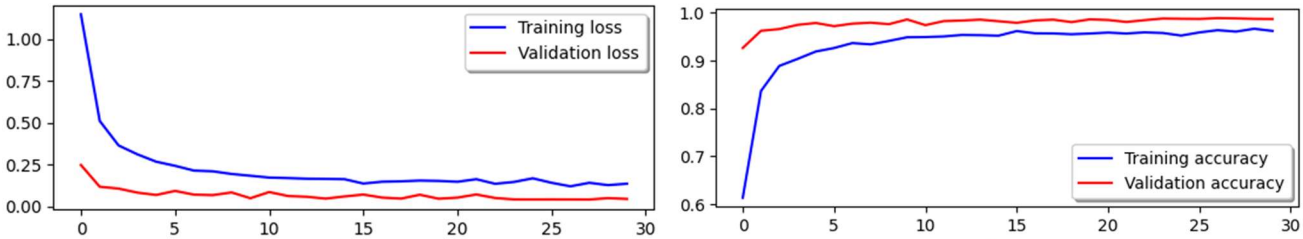


Figure 4. (a) Training and Validation Loss, (b) Training and Validation Accuracy

These results demonstrate that the model is capable of effectively learning from the training data and maintaining strong performance when applied to new, unseen examples. This balance between training and validation performance is crucial for the successful deployment of the model in real-world scenarios.

Detecting Sudoku Box

In the second stage of the methodology, the primary objective is to identify Sudoku squares through digital image processing. The goal is to ensure that each number on the Sudoku board is readable by the pre-built digit classification model. This process involves sophisticated steps such as detecting contour, transforming perspective, and splitting cells to identify the boundaries of each square with high accuracy. Proper detection is essential for the digit classification model to work efficiently and accurately. Errors in the detection of boxes can affect the accuracy of digit classification and, in turn, the success of automatically solving Sudoku puzzles. Figure 5 provides an overview of the flow of this stage.

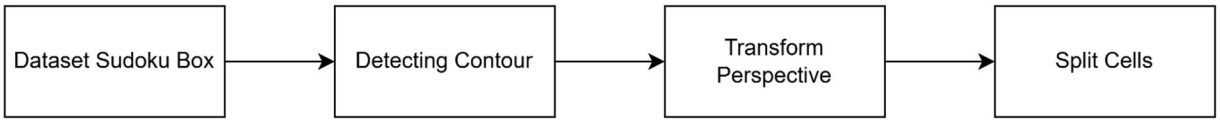


Figure 5. Flowchart Detecting Sudoku Box

Detecting Contour

Before executing the entire puzzle detection process, RGB images captured from a camera or imported from a media gallery must undergo a series of preprocessing steps. Initially, the image is converted to grayscale by calculating the average intensity value of each colour band for each pixel. Subsequently, Gaussian blurring is applied to the image, smoothens it, reduces noise, and minimizes inaccurate edge detection. These preprocessing steps are essential for the image to be optimized, thereby ensuring that the subsequent puzzle detection process can be performed accurately.

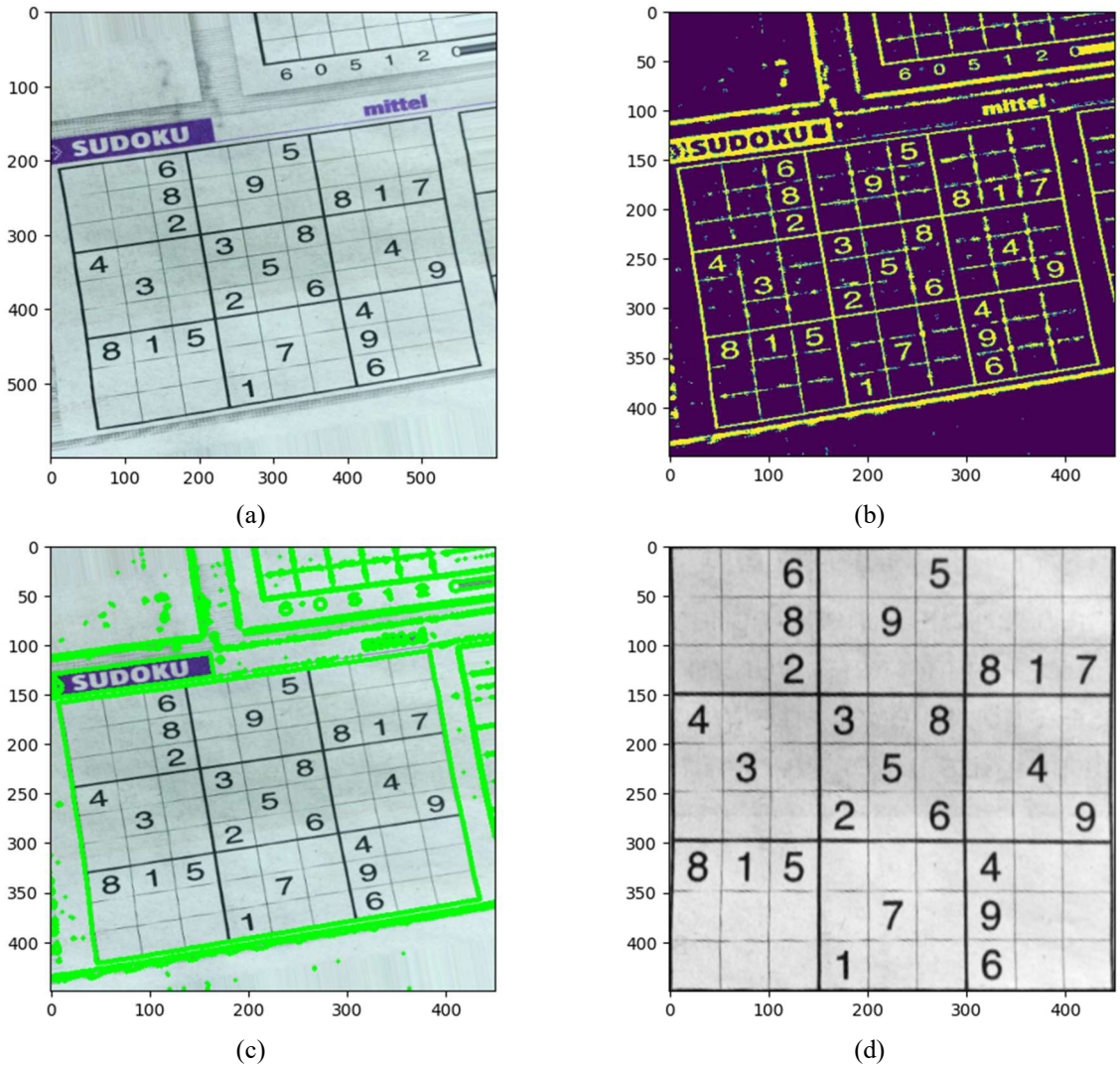


Figure 6. (a)Sample Image, (b)Processing Image, (c)Detecting Contour, (d)Transform Perspective

The application of a Gaussian blur to an image serves to smoothen the image, as illustrated in Figure 6(b). This process is designed to reduce the likelihood of improper edge detection, as demonstrated by Equation 5 [12].

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x-\mu_x)^2}{2\sigma_x^2} + \frac{-(y-\mu_y)^2}{2\sigma_y^2}} \tag{5}$$

Where: σ represents the standard deviation for each of the variables x and y. The reduction of noise that may interfere with the identification of true edges in the image is a crucial aspect of this process. By drowning out fine details and irrelevant intensity fluctuations, Gaussian blur helps to highlight key structures that are important for further analysis.

Adaptive thresholding, which uses the average of the local neighborhood, will be applied to binarize the image, as shown in Equation 2 [13].

$$T = \text{mean}(IL) - C \tag{6}$$

(T): The threshold value to be used. (mean(IL)of the local sub-region (IL) in the figure, and (C): A constant that can be adjusted to set the threshold value (T).

Contour detection represents a fundamental technique for the isolation of homogeneous regions within an image. The algorithm traces a sequence of coordinates along the border between connected components. The OpenCV library employs the contour-finding algorithm developed by Suzuki and Abe. In this study [14], the largest contour, representing the border of the entire Sudoku puzzle, is detected first. This is followed by the detection of the four corners using the Hough transform [15]. Once the four corners are identified, a perspective transform is applied to correct any warped orientation of the image; otherwise, the image is discarded. The entire process is illustrated in Figure 6.

Backtracking method to Solve Sudoku Puzzle

The final stage of this methodology is Sudoku solving using the backtracking algorithm. The focus at this stage is to solve the Sudoku puzzle automatically and efficiently. After the numbers on the Sudoku board are classified and the positions of the Sudoku boxes are identified through digital image processing, the next step is to apply the backtracking algorithm to determine the right number on each empty box. Backtracking is an algorithmic technique that builds a solution incrementally, exploring one piece at a time, and backtracks as soon as it determines that the current path cannot lead to a valid solution [16]. This approach is especially useful for solving constraint satisfaction problems, like Sudoku. Backtracking examines each empty cell and tries to fill it with a digit from its list of allowable candidates. If placing a digit leads to a conflict, the algorithm removes the digit and tries the next candidate. This process continues until a complete solution is found or all possibilities are exhausted. By discarding invalid paths early, backtracking efficiently narrows down the search space.

Backtracking is a powerful algorithmic technique used to find solutions by exploring possible paths in a structured way, using the Depth First Search (DFS) approach. Unlike exhaustive search methods that check every possible option, backtracking focuses only on promising paths, making it more efficient. This method is beneficial for solving problems like puzzles or games where multiple constraints need to be solved. In the backtracking algorithm, the process begins by selecting a starting point (like an initial node or an empty cell in a Sudoku puzzle). It then moves forward step by step, trying to build a complete solution. If it encounters a point where the current path cannot lead to a valid solution, it "backtracks" to the previous step and tries a different option. This process continues until a solution is found or all possibilities have been explored [17].

The typical steps involved in backtracking are:

1. Building a Path from Start to End: The algorithm constructs potential solutions by tracing paths from the root (starting point) to the leaves (possible endpoints). Nodes currently being explored are called live nodes, and those being expanded for further exploration are known as E-nodes (Expand nodes).
2. Identifying Dead Nodes: If an E-node leads to a path that cannot produce a solution, it is marked as a dead node, indicating that no further exploration will be done on this path.
3. Backtracking from Dead Nodes: When the algorithm encounters a dead node, it retraces its steps to the parent node to explore other potential child nodes. If there are no unexplored child nodes left, the algorithm backtracks further up the tree until it finds a new path to explore.
4. Determinating the Search: The search process stops either when a solution is found or when there are no remaining live nodes to explore.

By efficiently narrowing the search space and avoiding unnecessary paths, backtracking provides a systematic approach to solving complex problems effectively and efficiently.

RESULT METHODS

The combination of the three approaches described above digit classification, Sudoku board detection, and solving using a backtracking algorithm demonstrated highly satisfactory results. The first step involved capturing the image

of the Sudoku board using OpenCV, ensuring that the image was appropriately pre-processed for further analysis, as shown in Figure 7.

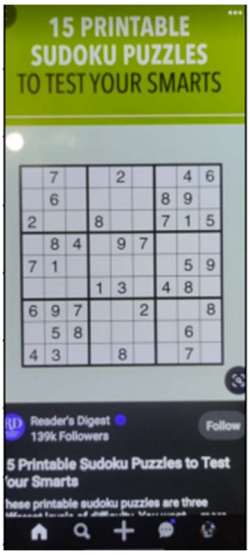
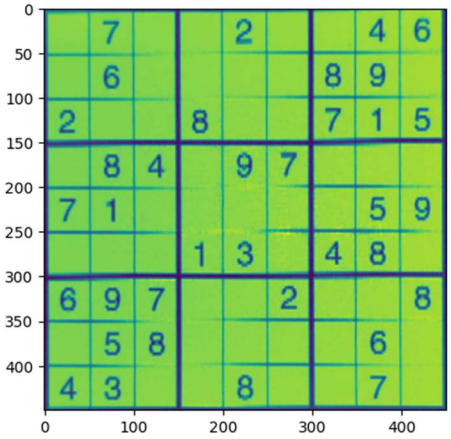


Figure 7. Capture Sudoku

Next, the digits on the Sudoku board were successfully identified with a high degree of accuracy using sophisticated image processing techniques. Subsequently, Sudoku board detection was performed to identify and determine the positions of the Sudoku squares in the image. By using image processing algorithms such as the Hough transform and edge detection, the system successfully detected the contours of the board with high precision, as illustrated in Figure 8(a).

This step ensures that each number on the Sudoku board can be correctly processed for the subsequent stage. Utilizing a convolutional neural network (CNN), the system was able to accurately recognize the digits despite variations in handwriting or suboptimal lighting conditions. If the system encounters a square on the Sudoku board that does not contain a recognizable digit, it predicts the value as 0, indicating an empty cell. This allows the system to maintain consistency in processing and ensures that all cells are accounted for, either with a digit or marked as empty. The result of the digit prediction is shown in Figure 8(b), demonstrating the effectiveness of the CNN in handling various challenges presented by real-world images.



(a)

0	7	0	0	2	0	0	4	6
0	6	0	0	0	0	8	9	0
2	0	0	8	0	0	7	1	5
0	8	4	0	9	7	0	0	0
7	1	0	0	0	0	0	5	9
0	0	0	1	3	0	4	8	0
6	9	7	0	0	2	0	0	8
0	5	8	0	0	0	0	6	0
4	3	0	0	8	0	0	7	0

(b)

Figure 8. (a)Sudoku Board With Digits, (b) Digit Prediction Result

Once the board and the position of the squares are identified, a backtracking algorithm is applied to solve the puzzle. Through backtracking, the system is able to efficiently place the correct number in each empty square, even in puzzles with high difficulty levels. The end result is illustrated in Figure 9 below, showcasing a completely solved Sudoku puzzle with all numbers correctly placed. This comprehensive methodology underscores the potential of combining advanced image processing techniques with robust algorithms to automate complex problem-solving tasks

accurately and efficiently. The success of this integrated approach highlights its potential application in various real-world scenarios where accurate digit recognition and problem-solving are crucial.

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

Figure 9. Solve Sudoku

CONCLUSION

This research is focused on improving the efficiency of CNN by using contour method to recognize the sudoku puzzle and solve it by using backtracking method. The approach relies on convolutional neural networks to recognize the numbers on the Sudoku board and uses a backtracking algorithm to accurately place the numbers. The integration of these technique not only speeds up the puzzle-solving process, but also opens the opportunities for broader application to problem solving that requires pattern recognition and accurate logical completion.

REFERENCES

[1] Peter Gordon and Frank Longo, "Mensa guide to solving sudoku: hundreds of puzzles plus techniques to help you crack them all," Sterling Publishing Company, 2006.

[2] J. F. Crook, "A pencil-and-paper algorithm for solving Sudoku puzzles," *Notices of the American Mathematical Society*, vol. 56, Jul. 2009.

[3] P. Berggren and D. O. Nilsson, "A study of Sudoku Solving Algorithms.," 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:60691555

[4] P. S. Tsai, T. F. Wu, J. Y. Chen, and J. F. Huang, "Integrating of Image Processing and Number Recognition in Sudoku Puzzle Cards Digitation," *Journal of Internet Technology*, vol. 23, no. 7, pp. 1573–1584, 2022, doi: 10.53106/160792642022122307012.

[5] A. Narayanaswamy, Y. P. Ma, and P. Shrivastava, "Image Detection and Digit Recognition to solve Sudoku as a Constraint Satisfaction Problem," 2019.

[6] T. E. , B. B. R. , & V. M. De Campos, *Character recognition in natural images*. Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal, 2009.

[7] Y. LeCun, C. Cortes, and C. J. C. Burges, "The MNIST database of handwritten digits," http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/.

[8] OpenCV, "OpenCV Color Conversions Documentation." Accessed: Jul. 02, 2024. [Online]. Available: https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html
[9] I. Ahmad, I. Moon, and S. J. Shin, "Color-to-grayscale algorithms effect on edge detection — A comparative study," in *2018 International Conference on Electronics, Information, and Communication (ICEIC)*, 2018, pp. 1–4. doi: 10.23919/ELINFOCOM.2018.8330719.
[10] C. Kanan and G. W. Cottrell, "Color-to-grayscale: Does the method matter in image recognition?," *PLoS One*, vol. 7, no. 1, Jan. 2012, doi: 10.1371/journal.pone.0029740.
[11] F. Chollet, "Keras Image augmentation layers Documentation." Accessed: Jul. 02, 2024. [Online]. Available: https://keras.io/api/layers/preprocessing_layers/image_augmentation/
[12] J. Reycian, B. Saraosos, M. Anthony, and J. B. Regis, "Development of an Android-Based Visual Sudoku Solver Using Contour Finding and Backtracking Algorithm," 2018.
[13] S. G. Qin Fang and X. Qiao, "Adaptive Functional Thresholding for Sparse Covariance Function Estimation in High Dimensions," *J Am Stat Assoc*, vol. 119, no. 546, pp. 1473–1485, 2024, doi: 10.1080/01621459.2023.2200522.
[14] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following," *Comput. Vis. Graph. Image Process.*, vol. 30, pp. 32–46, 1985, [Online]. Available: https://api.semanticscholar.org/CorpusID:205113350
[15] P. E. Hart, "How the Hough transform was invented [DSP History]," *IEEE Signal Process Mag*, vol. 26, no. 6, pp. 18–22, 2009, doi: 10.1109/MSP.2009.934181.
[16] B. Indriyono, N. Pamungkas, Z. Pratama, E. Mintorini, I. Dimentieva, and P. Mellati, "Comparative Analysis of the Performance Testing Results of the Backtracking and Genetics Algorithm in Solving Sudoku Games," *International Journal of Artificial Intelligence & Robotics (IJAIR)*, vol. 5, no. 1, pp. 29–35, Jul. 2023, doi: 10.25139/ijair.v5i1.6501.
[17] V. S. Widjaja and D. Z. Sudirman, "Implementasi Algoritma Backtracking dengan Optimasi Menggunakan Teknik Hidden Single pada Penyelesaian Permainan Sudoku," Seminar Nasional Aplikasi Teknologi Informasi 2013, Jul. 2013.