FINAL PROJECT: AI-TO SOLVE SIMPLE SUDOKU

Bhanu Teja Giddaluru
Computer Science Department
Kent, USA
[redacted]

Nathaniel Kaduk
Computer Science Department
Kent, USA
nkaduk@kent.edu

***Abstract*— This AI-based Sudoku solver uses artificial intelligence and image processing. The project involves two stages. A convolutional neural network (CNN) for Sudoku puzzle digit identification and classification is developed and trained in the first stage. In the second stage, extracting the contour of the Sudoku puzzles from an image, then splitting and cropping the 9*9 grid into 81 individual grids. use the digit classification model that trained in stage 1, to predict the out of each grid and convert it into a Sudoku Matrix. The Sudoku matrix is solved using constraint propagation algorithms. In addition to solving logical puzzles, the study demonstrates neural networks' versatility and effectiveness in digit recognition. Furthermore, the project's Sudoku puzzle-solving success shows the capabilities of AI to solve complex problems.**

Keywords—Sudoku, CNN, grid, constraint, most constrained, most constraining, matrix

I. MOTIVATION

Developing an AI-based Sudoku solver is driven by technological innovation and practical application. Sudoku, a famous puzzle game, is a great way to demonstrate artificial intelligence's pattern detection, logical reasoning, and problem-solving capabilities.

First, AI can handle Sudoku puzzles' particular challenges. AI excels in pattern recognition, constraint compliance, and logical deduction, which are the game requirements. Integrating AI into Sudoku can show how AI algorithms mimic human cognitive processes in solving complex problems. The project combines image processing and artificial intelligence for solving Sudoku. Digit recognition in Sudoku grids from images requires complex image processing and neural networks.

Solving Sudoku by hand can be as simple as filling in numbers. The task of solving Sudoku efficiently, however, has been the subject of many logisticians, game theorists, and the improving causal-player alike. From the field of AI, the topic of constraint satisfaction has been a hot topic for many problem-solving studies for two main reasons: efficiency and reliability. By using heuristics, constraint satisfaction looks to maximize reliability by minimizing the chances for failure. By using efficient algorithms to solve Sudoku, the solving process becomes faster.

Both of these reasons could be used in defining a “gold standard” for problem solving: the ability to solve a problem both quickly and consistently. As a result, using constraint satisfaction to solve a puzzle such as Sudoku, with both heuristic and consistency, helps establish a baseline that other problem-solving programs can play off from.

Additionally, the project converts an image-based puzzle into an AI-processable numerical matrix. Computer vision for digit recognition and artificial intelligence for problem-solving are integrated in this research.

Finally, this project educates. It shows how AI may be used in real life, making it a great educational tool for students and enthusiasts.

This AI-based Sudoku solver project aims to explore AI's logical problem-solving capabilities, advance image processing and neural network learning techniques, integrate AI subfields, and teach practical AI applications.

II. PROJECT DESCRIPTION

As part of this project, we have done research on many similar works and related papers on AI- To solve simple Sudoku.

Deep learning and state-of-the-art computer vision are investigated in the article "A Deep Learning Approach to Solve Sudoku Puzzle" to solve Sudoku puzzles. It uses optical character recognition (OCR) to identify the numbers and OpenCV for image processing to find the borders of the Sudoku grid in an image.

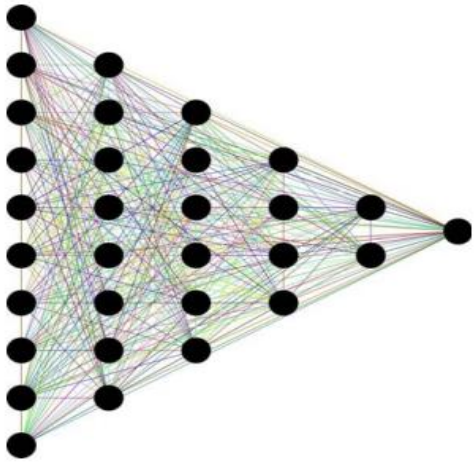

Fig. 1. Sample Neural Network given in the paper

Numerical data collected from photos is processed using a neural network model, in this case TensorFlow. This is the heart

of the solution. This study shows that by integrating deep learning with image processing, it is possible to automate the solution of complicated logical puzzles, such as Sudoku. The area of AI-driven problem-solving has made great strides with this integration of deep learning and computer vision.

According to the research "MNIST Handwritten Digit Recognition using Machine Learning," Convolutional Neural Networks can identify handwritten digits in the MNIST database. Greyscale photographs of handwritten digits (0-9) from MNIST are used to train and test the CNN digit recognition model.

Variations in writing style, size, and orientation make digit recognition difficult. The model accurately converts handwritten digits into machine-readable representations to improve banking procedures, displaying excellent efficacy and reliability. Digital banking, automatic digit recognition, and other sectors benefit from the study.

Fig. 2. Sample MNIST image

The paper "Construction of Standard Solid Sudoku Cubes and 3D Sudoku Puzzles" introduces SSSCs, and three-dimensional versions of Sudoku tables. Using SSSCs' mathematical and computational base, the research proposes a novel way to build these cubes. The project focuses on using cyclotomic cosets and Latin cubes to construct SSSCs of different orders to create complicated 3D Sudoku puzzles. This novel approach to Sudoku adds a new layer and allows for mathematical and logical puzzle building.

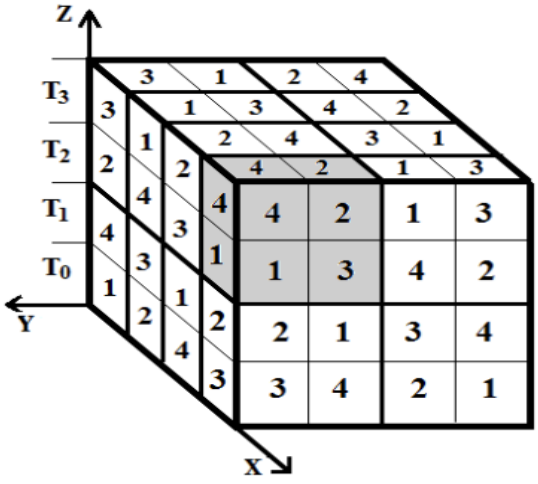

Fig. 3. Construction of Standard Solid Sudoku Cubes

The paper, "A 4x4 Sudoku Solving Model Based on Multi-layer Perceptron" introduces a novel strategy for resolving Sudoku problems by leveraging a Multi-layer Perceptron (MLP) neural network. For smaller-scale puzzles like 4x4 Sudoku, this research forgoes typical reinforcement learning methods in favor of a simpler supervised learning model.

Details of the MLP model's configuration, training methods, and the process of transforming Sudoku puzzles into a format appropriate for MLP classification are provided in the paper. The model's effectiveness in solving these challenges is demonstrated through evaluation and comparison with other methods

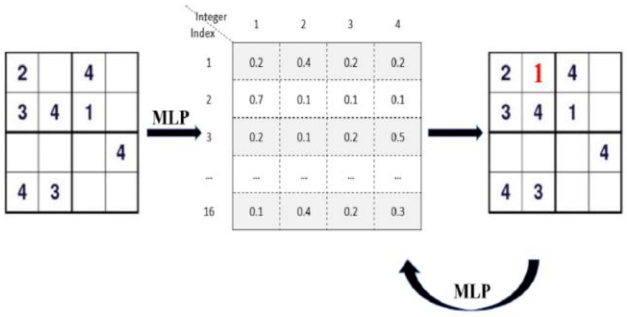

Fig. 4. Decision police flow chart given in the paper

Paper titled "A Novel Automated Solver for Sudoku Images" details an approach to solving Sudoku puzzles using machine learning and image processing protocols. Image processing techniques like thresholding and grid extraction are used to identify the Sudoku grid in a picture.

Fig. 5. Digit extraction output given in the paper

Following an introduction to the k-nearest neighbors technique, the study delves into the topic of digital character identification using a bespoke optical character reader (OCR). The digital Sudoku grid is then used to apply a backtracking-based solving technique in order to obtain the solution. When applied to newspaper photographs of Sudoku puzzles, this method demonstrates encouraging outcomes in recognizing digits and solving the puzzle.

III. PROJECT DESIGN

The AI-based Sudoku solver project has 2 stages: Digit Classification and Sudoku contour extraction from an image. In Stage 1, a convolutional neural network (CNN) is developed to recognize and categorize digits from images. This CNN is trained using a multiclass-digit dataset. In Stage 2, contour detection techniques detect the puzzles in the image and extract the Sudoku grid. by using these two stages. each digit of the 9*9 grid, predict its value using a digit classification model and place it in a puzzle matrix. Finally, a constraint propagation technique uses the most constrained most constraining variables to logically deduce the missing numbers to solve the Sudoku.

3.1. Digit classification model

The digit classification model block diagram shows the process by which digits are recognized from the image. Multiple layers of the CNN are arranged for image-based digit classification. After two sets of convolutional layers, max pooling extracts features and reduces dimensionality. Dropout layers with a rate of 0.5 randomly drop units during training to reduce overfitting. Finally, the network flattens the extracted features to feed into a dense layer with 500 neurons and an output layer with 10 neurons for each of the 10-digit classes, using softmax for multi-class classification. ReLU activation functions help non-linear learning across the model.

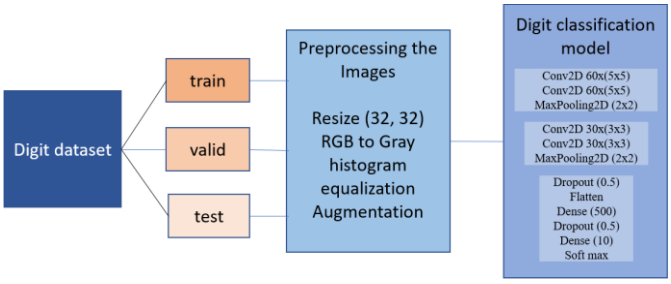

Fig. 6. Block diagram of the Digit classification model

3.1.1 Dataset information:

There are 10 classes of digits 0–9 in the digit dataset, each including 1016 PNG pictures. Consistent 128x128-pixel images help the convolutional neural network (CNN) to understand and classify digits. This large and balanced dataset helps train the AI model to accurately detect digital digits.

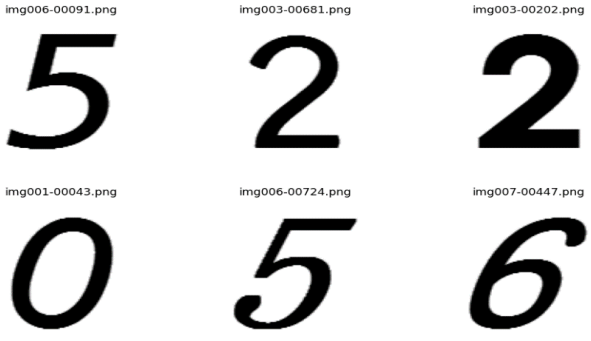

Fig. 7. The digit dataset images

The classification model's training digit images are shown in Figures, Digit dataset multi-class image counts in a bar plot. This graphic helps visualize the dataset information.

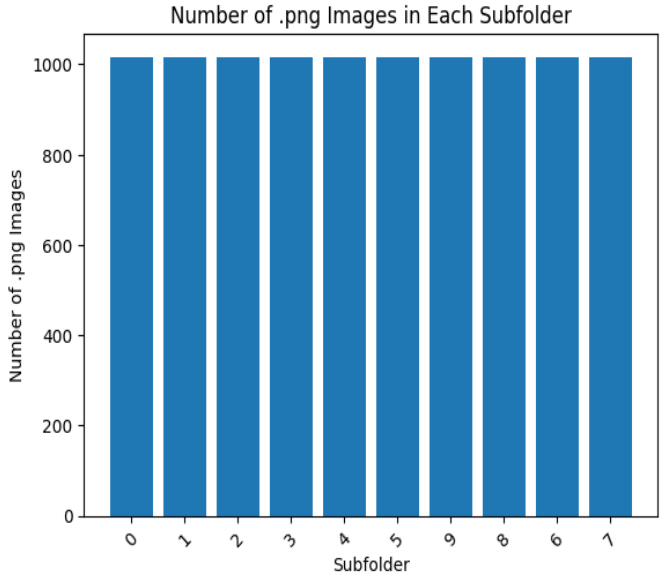

Fig. 8. Digit dataset multiclass bar plot

3.1.2 Dataset preparation:

I. Data splitting:

The dataset of 10160 images from 10 classes was split using train-test-split into 5% for testing, 95% for training and 20% for validation. This classification balances model training, testing, and validation for a complete evaluation framework.

II. Dataset preprocessing:

Images are converted to grayscale, histogram equalization is applied for contrast enhancement, pixel values are normalized, and then the input size is resized to 32*32, in the preprocessing. Data augmentation methods like shifting, zooming, and rotating improve the training dataset, improving the model. One-hot encoding encodes categorical labels for classification, finishing data preparation. The sequential model is trained on the data using the optimizer RMSprop and compile the model on categorical cross-entropy.

3.2. Contour detection.

Sudoku contour detection, where image processing methods identify the Sudoku grid borders in an image. It separates the grid from the background and segments it into cells using a detection procedure and image thresholding. Each digit is accurately found in the grid using contour detection, allowing precise extraction and puzzle matrix arrangement.

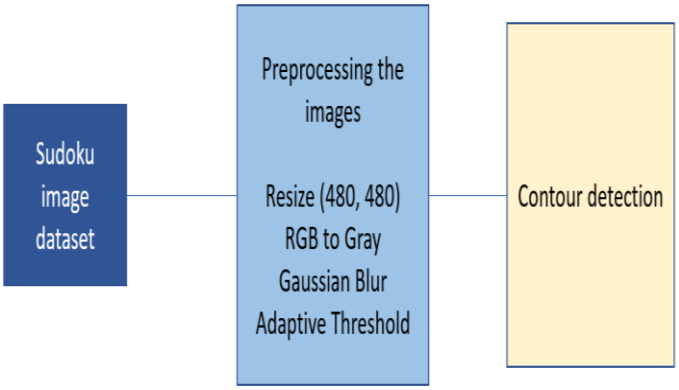

Fig. 9. Block diagram of the Contour detection

In the contour detection block diagram, the Sudoku image dataset is preprocessed before contour detection, as shown in the block diagram. Resizing images to 480x480 pixels and converting them from RGB to grayscale reduces image data and speeds up processing. By reducing noise and improving adaptive thresholding, Gaussian blur helps differentiate grid lines from the rest of the image. These stages provide contour identification and Sudoku grid extraction.

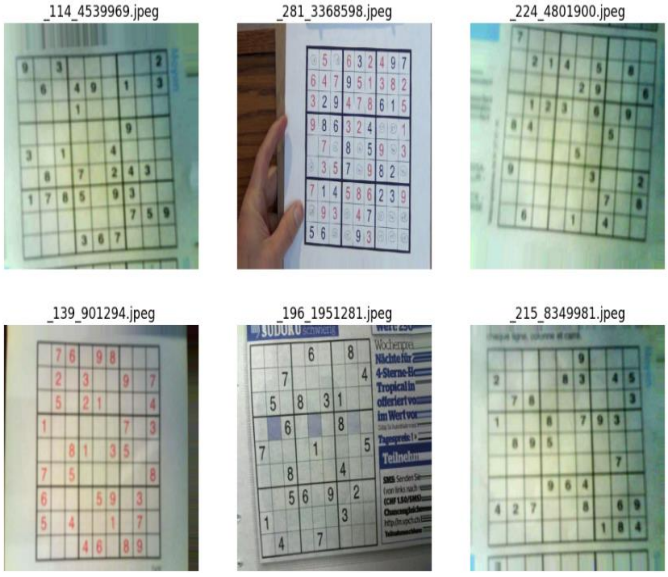

Fig. 10. Sudoko dox dataset images

The Sudoku Box Detection dataset is a collection of single-class images and contains 600x600 of 2620 JPEG images.

5.2.1 Data preprocessing

Resizing to 450x450 pixels, converting to grayscale, using Gaussian blur to reduce noise, and utilizing adaptive thresholding to create a clear, binary Sudoku image are preparation steps. After processing, this image displays enhanced features for contour recognition and analysis.

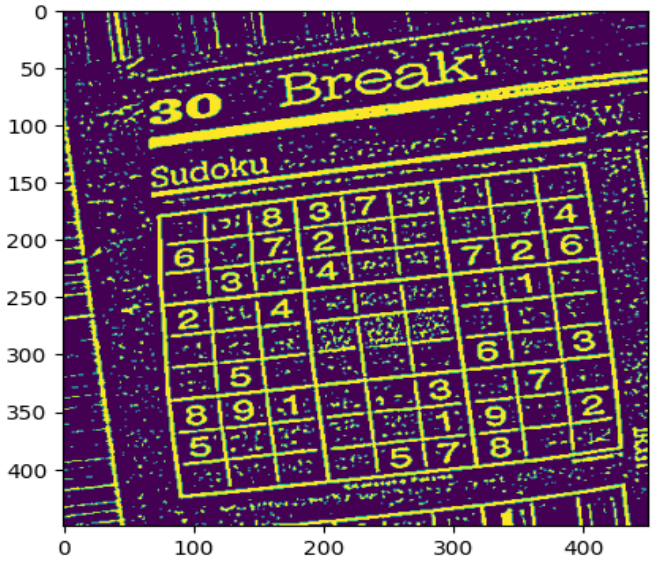

Fig. 11. Pre-processed Sudoku image

A pre-processed Sudoku puzzle with enhanced contrast and sharpness due to adaptive thresholding clearly defines the grid and numbers for digit extraction.

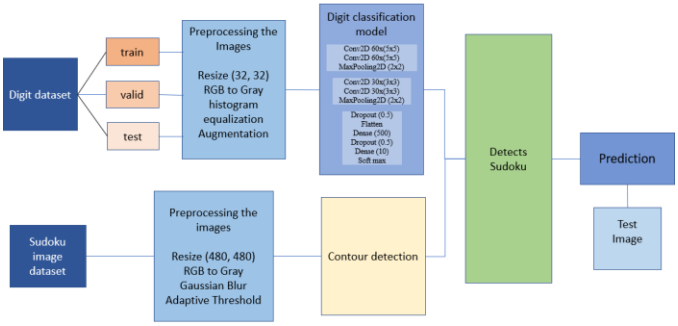

Fig. 12. AI-TO SOLVE SIMPLE SUDOKU ARCHITECTURE

IV. Problems Faced

Throughout the project, several challenges impeded progress, beginning with searching for the Sudoku image dataset. In stage 1, the Digit classification model is highly dependent on the input images, they should be high resolution and the dataset should be normalized for every class of 0 to 10 digit classes. For stage 2, Sudoku box contour detection the dataset should be general.

The size of the datasets used required a lot of GPU, so we used Google Collab. Coincidently, managing enormous amounts of data requires careful storage and processing. TensorFlow requires compatibility with the Python version in order to provide a seamless library experience.

A major challenge during contour identification was separating the Sudoku grid from image backgrounds and noise.

To highlight grid lines without capturing unnecessary information, the border detection and thresholding parameters had to be fine-tuned. Making accurate digit recognition a top priority in Sudoku prediction was no easy task, particularly when dealing with a wide variety of fonts and distorted images. Misclassifying one digit might cause incorrect puzzle solutions. It was also tough to turn the contours into a Sudoku matrix. Ensuring each digit was in the right matrix cell required accurate alignment and scaling.

Concerning the Sudoku solver, problems arose in a variety of ways. The first version of the Sudoku solver only used a "runSet" function, which solved simple Sudokus by applying singular legal values to their respective cells. In future versions, valid numbers were tried in a breadth-first manner. Once all the valid numbers for a cell were found, all of the Sudoku boards corresponding to that value were generated, but there was a catch: Python used the same board to update, meaning that values in one iteration were replaced by the next. As a result, deep-copy was implemented.

Not surprisingly, breath-first with deep-copy led to agonizingly slow results (runSet helped, but it wasn't enough). It was at this point schemes to prune the search space were implemented, like heuristics. The most constraining heuristic greatly improved speed, but hard Sudokus wouldn't solve correctly. Other attempts, including a completely redone Sudoku solver went by, but still the hard test Sudoku used refused to become solved.

Then, the hard Sudoku was analyzed as if a human was tying to solve it. In this moment, the problem was found. The possible values were expanding too much, but a consistency checker could prune the possible continuations by as much as one-half in the early stages. Hours upon hours later, the consistency checker was implemented, and the solver seemed to work at its full potential.

Only the comprehensive test said otherwise. The solver failed at detecting which Sudokus were unsolvable, and took extremely long to solve Sudokus which had differing values than the ones obtained online. As a result, an extra checker was implemented to identify unsolvable Sudokus (and with the help of runSet could even identify non-consistent Sudokus). Furthermore, the entire scheme of the solver was redone, using recursive depth-first search instead, which allowed it to operate much faster.

Finding the most efficient pathways needed careful programming and testing. Solver precision was crucial. To provide proper data into the solver, our digit classification model has to be highly precise to avoid unsolvable cases from picture recognition of incorrect digits. Lastly, we needed to make sure the solver could withstand extreme cases without breaking. This required adding checks and balances to the solving algorithm to handle odd issues configurations.

V. Implementation and Results

5.1 Digit classification model

Our Sequential CNN model is designed to classify 32*32 pixel grayscale images into 10 classes, the initial 2 Con2d layer with 60 filters with 5*5 filter size and have activation function ReLU and same padding. These layers extract features, recognize patterns, and create spatial hierarchies in input images. Two Max-Pooling layers with a 2x2 pool size compress the input and allow the network to focus on the most important features.

Followed by two Convolutional layers with 30 3x3 filters, Max pooling and ReLU activation. As network depth rises, these layers extract more complex patterns. To avoid overfitting, the network randomly drops neurons during training with dropout layers at 0.5 after pooling and before the fully connected layer. The model then flattens multi-dimensional feature maps to form a single long feature vector that feeds into a dense layer with 500 neurons and an output layer with 10 neurons for each 10-digit class. The output layer's softmax activation function assigns probabilities to each class.

For the training, we used the optimizer RMSprop with a learning rate of 0.001 and compiled the model using the categorical cross-entropy for 32 batches of 30 epochs, for each batch, we used the augmented images to improve the model efficiency. Our model test accuracy for 10 classes is 0.998.

5.2 Sudoku contour detection:

5.2.1 Detecting the Sudoku Contour

Image processing is used to extract Sudoku puzzles. First, contours are recognized in the preprocessed sudoku box image. Assuming the largest quadrilateral in the image has the Sudoku puzzle outline, identify its contour. once obtaining the contour, then reframe the output with the size of the contour outline, and then use the perspective correction to get sudoku at the output. then splitting the main Sudoku contour into 81 individual contour cells.

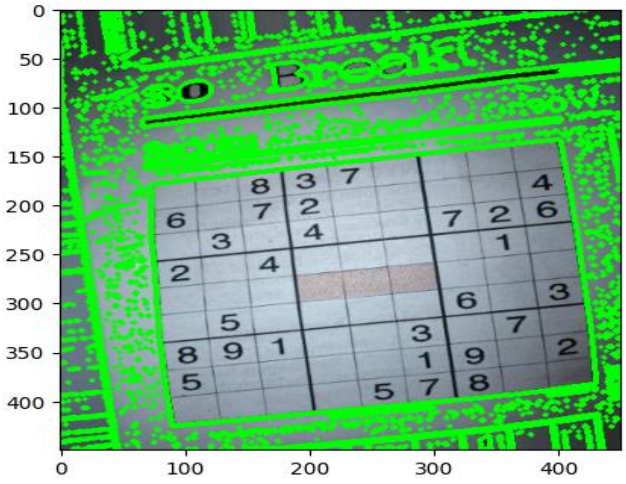

Fig. 13. Sudoku Box Contour detection

The Fig.13 is the contour detection of the input image and identifying the sudoku puzzle's main outline. After obtaining the main outline the output displays images sized to the contour size. And split the contour into 81 individual contours to form a 9*9 Sudoku grid.

5.2.2 Splitting and Cropping

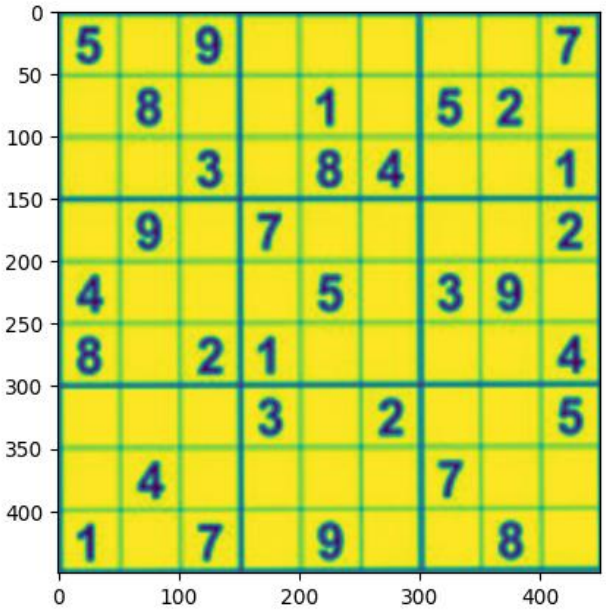

Fig. 14. Contour detection on a test image for the splitting and cropping

5.2.3 Detecting the Sudoku Puzzle

```
[[5, 0, 9, 0, 0, 0, 0, 0, 7],
 [0, 8, 0, 0, 0, 0, 5, 2, 0],
 [0, 0, 3, 0, 8, 4, 0, 0, 0],
 [0, 9, 0, 7, 0, 0, 0, 0, 2],
 [4, 0, 0, 0, 5, 0, 3, 9, 0],
 [8, 0, 2, 0, 0, 0, 0, 0, 4],
 [0, 0, 0, 3, 0, 2, 0, 0, 5],
 [0, 4, 0, 0, 0, 0, 7, 0, 0],
 [0, 0, 7, 0, 9, 0, 0, 8, 0]]
```

Fig. 15. Sudoku Image from Digit classification and Contour detection

Then after cropping the Sudoko in 81 sub-grids each grid is resized to 32*32 and sent to the digit classification model and then predections are converted into a 9*9 matrix. This is the Sudoku puzzle from the image. Fig. 11 is an example Sudoku puzzle matrix obtained by these processes.

5.3 Results

The accuracy plot of a digit classification model shows that training and validation accuracies immediately converge to high values during the first few epochs and remain consistent throughout training, indicating a well-fitting model without overfitting.

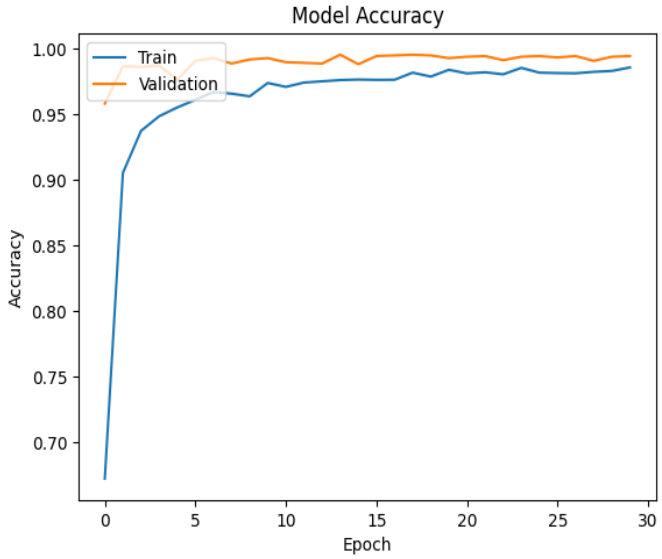

Fig. 16. Digit classification model accuracy plot

The model loss during 30 epochs for digit classification, showed a significant decrease in loss initially and stabilization, indicating effective learning and good generalization. The training and validation loss curves are closely matched.

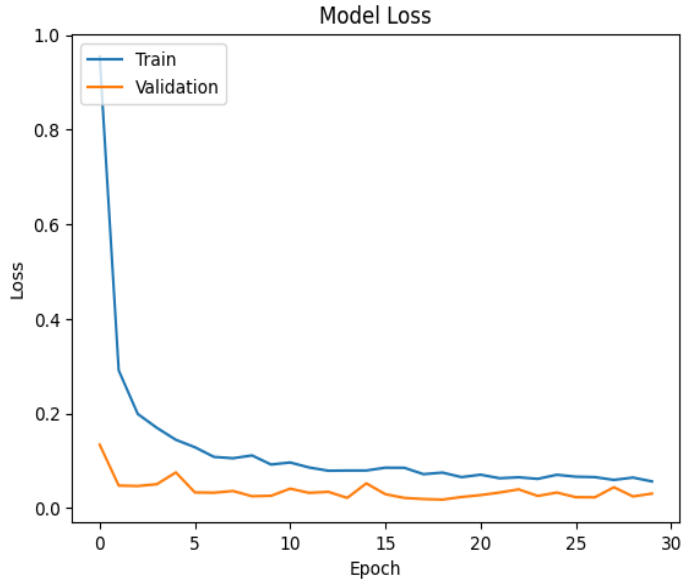

Fig. 17. Digit classification model loss plot

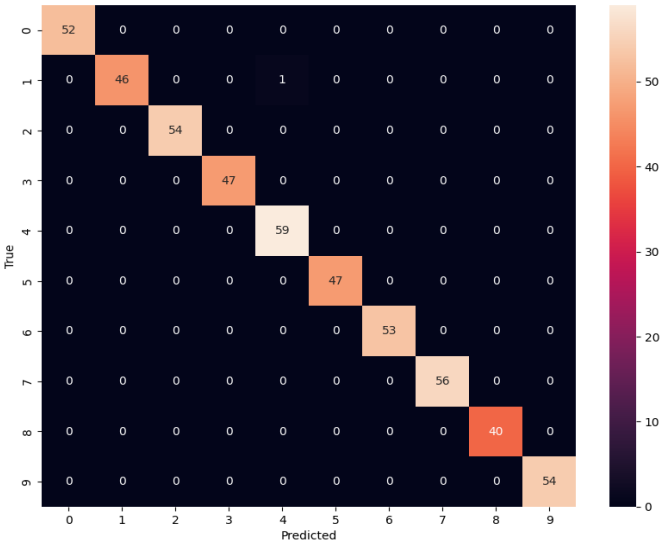

Fig. 18. Digit classification model Confusion Matrix

The confusion matrix for the digit classification model shows high accuracy across all classes, with the main diagonal showing strong true positive rates, indicating that the model accurately predicts the various digit classes with minimal confusion.

Table I shows our digit classification model's Sudoku digit classification accuracy metrics.

Precision: Precision measures the model's ability to properly recognize a digit from all its instances. High precision is the ratio of true positive predictions to total positive predictions and suggests fewer false positives.

$$Precision = \frac{TP}{TP + FP}$$

Recall: Also known as sensitivity, recall tests the model's capacity to classify all instances of a digit in the test dataset. Divide the number of true positives by the sum of true positives and false negatives.

$$Recall = \frac{TP}{TP + FN}$$

F1-Score: The harmonic mean of precision and recall balances both metrics in a single score, especially beneficial when class distribution is uneven. For perfect precision and recall, it approaches 1, but as either decreases, it approaches 0.

$$F1 - Score = \frac{2(Precision \times Recall)}{Precision + Recall}$$

TABLE I. EVALUATION METRICS

Digits	Precision	Recall	F1- score	Support
0	1.0	1.0	1.0	52
1	1.0	0.98	0.99	47
2	1.0	1.0	1.0	54
3	1.0	1.0	1.0	47
4	0.98	1.0	0.99	59
5	1.0	1.0	1.0	47
6	1.0	1.0	1.0	53
8	1.0	1.0	1.0	56
9	1.0	1.0	1.0	40
10	1.0	1.0	1.0	54

5.4. Solving the Sudoku

```
Algorithm: Solve
Input: Sudoku grid, gridS
Output: Solved Sudoku grid or failure

if(gridS == failure ∨ ¬runSet(gridS)) {return failure;}
MCC = mostConstraining(gridS);
if(MCC == ∅) {return gridS;}
valid = getValid(gridS, MCC);
for(∀v ∈ valid){
    currentGrid = copy(deepGrid);
    currentGrid_MCC = v;
    if(runSet(currentGrid)==0 ∨ ¬checkConsistency(currentGrid)){continue;}
    currentGrid = solve(currentGrid);
    if(currentGrid != failure) {return currentGrid;}
}
return failure;
```

Fig 19. Algorithm to solve Sudoku

The Sudoku solver implements a "runSet" function, which checks if there is a single value that can be placed in each cell. If all of the rows, columns, and subgrid agree that there is only one possible value, it assigns that value to the cell. Simultaneously, it also checks whether two values are in the same row, column, or subgrid, to ensure that if a mistake was made during the detection steps, an error message will be raised.

But this strategy doesn't work on all Sudokus, as many hard-expert Sudokus require trial and error. For this reason, a depth-first search is implemented to try different values to see if they solve the Sudoku. To do this, the most constraining cell is chosen (to restrict the other cells as much as possible), and all of the valid numbers are generated for that cell. If each valid number is consistent, or assigning that cell with one of those numbers doesn't take away the possibility for there to be 1-9 in every row/column/subgrid, then that continuation is possible to succeed and the depth-first process repeats on this Sudoku. At each step, "runSet" is called, and if it returns false, the search

stops for that particular branch. If the majority of the Sudoku has been filled using the depth-first search, and is solvable, "runSet" will instantly solve the Sudoku.

VI. Discussion

The AI model's digit classification and Sudoku puzzle identification skills were evaluated against various layout challenges throughout the project. The model performed well due to careful dataset preparation and data augmentation.

To adapt the model to Sudoku puzzles, these steps were crucial. Through iterative testing and optimization, contour detection, a major early challenge, was improved, improving digit recognition accuracy. The model's ability to learn and improve suggests that extending the dataset and using more advanced image processing methods could improve it.

In addition, a chance for the model to progress beyond simple puzzle-solving was brought to light throughout the discussions. Due to artificial intelligence advances, extending the model to real-time solving and more difficult Sudoku variants look realistic. The model would show its scalability and adaptability to dynamic problem-solving circumstances.

VII. Conclusion

In conclusion, the project's accomplishment of developing an exceptionally precise AI model for solving Sudoku puzzles demonstrates the promise of AI in cognitive areas. The model has the potential to be used in instructional and problem-solving tools due to its exceptional performance in overcoming layout issues and accurately recognizing numbers. Looking forward, the project paves the way for real-time problem-solving and the investigation of increasingly complex puzzle variations through the utilization of artificial intelligence. Based on the results of this effort, artificial intelligence (AI) has the potential to be an effective tool for tackling a variety of cognitive problems, not limited to numerical riddles.

VIII. Team roles

TABLE II. Team Roles

	Al Sudoku – Group #7
Team Members	**Roles**
Bhanu Teja Giddaluru	CNN model design to extract Sudoku from an image, Optimize and fine-tune the CNN model. Image processing techniques for puzzle recognition.
Nathaniel Kaduk	Implemented advanced Constraint solver algorithm, DFS, and pattern recognition. Fine-tune the solver for complex Sudoku.

IX. References

[1] K. Poloziuk and V. Yaremenko, "Neural networks and Monte-Carlo method usage in multi-agent systems for sudoku problem solving," Technology Audit and Production Reserves, no. 6 (2 (56)), pp. 38-41, 2020, doi: 10.15587/2706-5448.2020.218427.

[2] Shrivastava, A., Jaggi, I., Gupta, S., & Gupta, D. (2019, October). Handwritten digit recognition using machine learning: a review. In 2019 2nd International Conference on Power Energy, Environment and Intelligent Control (PEEIC) (pp. 322-326). IEEE.

[3] H. Du, L. Gao and X. Hu, "A 4×4 Sudoku Solving Model Based on Multi-layer Perceptron," 2021 International Conference on Electronic Information Engineering and Computer Science (EIECS), Changchun, China, 2021, pp. 306-310, doi: 10.1109/EIECS53707.2021.9587969.

[4] Arbelaez, P., Maire, M., Fowlkes, C., & Malik, J. (2010). Contour detection and hierarchical image segmentation. IEEE transactions on pattern analysis and machine intelligence, 33(5), 898-916.

[5] Jain, M., Kaur, G., Quamar, M. P., & Gupta, H. (2021, February). Handwritten digit recognition using CNN. In 2021 International Conference on Innovative Practices in Technology and Management (ICIPTM) (pp. 211-215). IEEE.

[6] Ghosh, M. M. A., & Maghari, A. Y. (2017, October). A comparative study on handwriting digit recognition using neural networks. In 2017 international conference on promising electronic technologies (ICPET) (pp. 77-81). IEEE.

[7] J. Redding, J. Schreiver, C. Shrum, A. Lauf and R. Yampolskiy, "Solving NP-hard number matrix games with Wisdom of Artificial Crowds," 2015 Computer Games: AI, Animation, Mobile, Multimedia, Educational and Serious Games (CGAMES), Louisville, KY, USA, 2015, pp. 38-43, doi: 10.1109/CGames.2015.7272959.

[8] J. Jilg and J. Carter, "Sudoku evolution," 2009 International IEEE Consumer Electronics Society's Games Innovations Conference, London, UK, 2009, pp. 173-185, doi: 10.1109/ICEGIC.2009.5293614.

[9] Chel, H., Mylavarapu, D., & Sharma, D. (2016, March). A novel multistage genetic algorithm approach for solving Sudoku puzzle. In 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) (pp. 808-813). IEEE.

[10] Ali, S., Sakhawat, Z., Mahmood, T., Aslam, M. S., Shaukat, Z., & Sahiba, S. (2020, August). A robust CNN model for handwritten digits recognition and classification. In 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA) (pp. 261-265). IEEE.

[11] Gurav, R. M., & Kadbe, P. K. (2015, May). Real time finger tracking and contour detection for gesture recognition using OpenCV. In 2015 International Conference on Industrial Instrumentation and Control (ICIC) (pp. 974-977). IEEE.

[12] Zhan, H., Lyu, S., & Lu, Y. (2018, August). Handwritten digit string recognition using convolutional neural network. In 2018 24th International Conference on Pattern Recognition (ICPR) (pp. 3729-3734). IEEE.

[13] Yokoyama, M., & Poggio, T. (2005, October). A contour-based moving object detection and tracking. In 2005 IEEE international workshop on visual surveillance and performance evaluation of tracking and surveillance (pp. 271-276). IEEE.

[14] Kamal, S., Chawla, S. S., & Goel, N. (2015, December). Detection of Sudoku puzzle using image processing and solving by Backtracking, Simulated Annealing and Genetic Algorithms: A comparative analysis. In 2015 third international conference on image information processing (ICIIP) (pp. 179-184). IEEE.

[15] A. B N, S. Ravikumar, V. Jason, T. A. Sarika, S. Adnan and T. G, "Use of Propositional Logic in building AI Logic and Game Development," 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), Delhi, India, 2023, pp. 1-6, doi: 10.1109/ICCCNT56998.2023.103082