SUDOKU SOLVER

Sahil Sharma, Shikhar Tiwari

B.Tech. in Computer Science Engineering, Indraprastha Institute of Information Technology Delhi

ABSTRACT

Solving sudoku is fun, but it turns out to be rather frustrating sometimes. There can be many variations for this puzzle, but the most common one is with a nine by nine grid having 81 cells. Our project can solve these complex puzzles for users. The user inputs an image of a sudoku puzzle having reasonable clarity, and our sudoku solver displays the solution by overlaying the computed values on the original image. The solution of the puzzle is computed by extracting the puzzle from the image, extracting and recognizing the digits from the puzzle to create a sudoku board, and finally solving the board using a sudoku solving algorithm.

Keywords— image processing, convolutional neural network, perspective transform, contours, digit recognition.

1. INTRODUCTION

Sudoku puzzles can be found in newspapers, magazines, etc. This puzzle may look simple but can be very complex at times. Users have to follow some set of rules like no digit should be repeated in any row or column. Grid is further divided into 3X3 sub-grids, and no subgrid should have repeating values. Our project uses some basic image processing techniques like thresholding, blurring, perspective transform, etc., to find the grid of the puzzle in the given sudoku image, and applies a custom Convolutional Neural Network(CNN) model to recognize the digits in the puzzle. Using the obtained digits, we can easily solve the puzzle by using backtracking.

2. METHODOLOGY

Our project can be divided into subparts. Each of which has been discussed below.

2.1. Processing the input image

First, we converted the three channeled RGB input image into one channeled grayscale image. Then we used Gaussian blurring on the grayscale image to remove unwanted noise. After this, we wanted a binary image for finding the contour of the sudoku grid. Therefore, we used adaptive thresholding to get the binary image, which we used in the next step (See Figure 1 [a], [b]) [3][4].

2.2. Extracting the grid

Now, we used the thresholded image and found all the contours in it. In order to find the contour of the grid, we sorted the contours according to their area and chose the largest one having a length of 4, which signifies that it is a rectangle (See Figure 1 [c]). Then we used the selected contour to find the coordinates of its four vertices and the side of the grid. Using the sides, we can determine the coordinates of the required grid image corresponding to each of the previous 4 coordinates of the grid in the input image. Now, we can find the perspective transformation matrix using the above 4 points mapping and apply it to the input image to get the warped image of the grid (See Figure 1 [d]) [3][4].

2.3. Making CNN model for Digit Recognition

We used Sequential class of the keras API to make a CNN model. We then tuned the number of sequential layers and their corresponding hyperparameters to get the best results. We then compiled the model and trained it on the standard MNIST dataset. Finally, we saved the trained model and used it for digit recognition [3][5][6][7][8][9].

2.4. Extracting digits from the grid

To extract the digits, we first need to divide the grid into 81 cells (See Figure 2 [a]). Then, for each cell, we thresholded the cropped image to remove unwanted noise and cleared the borders of the cell (See Figure 2 [b]). Now, we found all the contours which were present in the cell. To find the contour of the digit, we selected the contour with the largest area. Then, we drew the obtained contour on a black image with the same size as that of a cell (See Figure 2 [c]). After the above steps, we resized the obtained image accordingly to use it for prediction (See Figure 2 [d] to [l]). Finally, we used our saved CNN model to predict the corresponding digit in the cell [3][5].

2.5. Making sudoku board

We initialized the sudoku board with empty cells and used the model obtained from step 2.3 to predict the digits and

put them at their corresponding cell locations (See Figure 3 [a]). Then, we used this board to solve the puzzle [3].

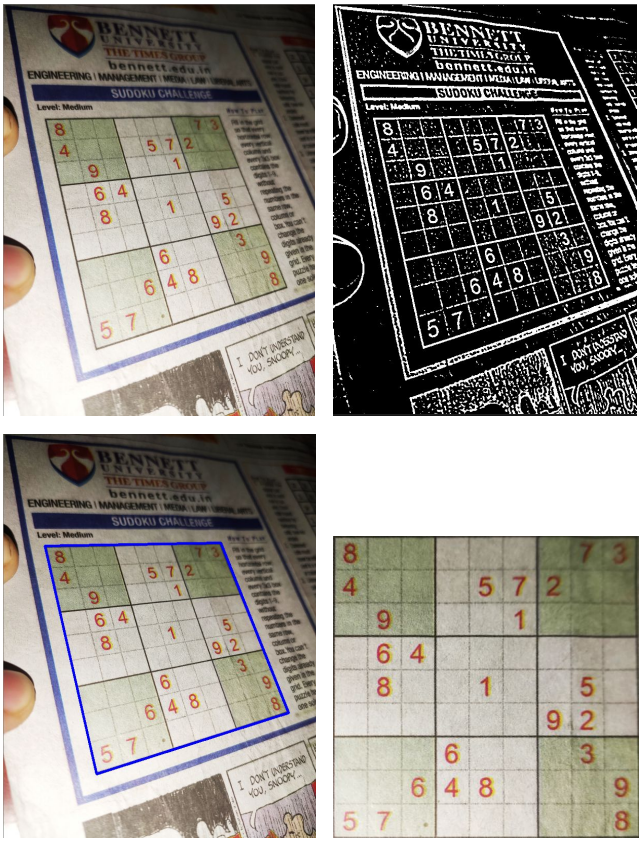

Figure 1. [a] [b] [c] [d]
[a]. *Input Image* [b]. *Image after gaussian blurring and Adaptive thresholding* [c] *Required Contour of the grid* [d]. Grid Image

Figure 2. [a] [b] [c] [d] [e] [f] [g] [h] [i] [j] [k] [l]
[a]. *Cell Image* [b]. *[a] after Thresholding* [c]. *Displaying the largest contour in [b].* [d][e][f][g][h][i][j][k][l]. *Resized images*

2.6. Solving the puzzle

For solving the puzzle, we keep on finding empty boxes and check if we can fill any number in it. Before putting a number, we check if it follows all the rules of sudoku. After putting the number, we again find empty boxes. If we see that we have a box that cannot be filled now, we backtrack and remove our previously entered values [2]. All this is done until we get a solved sudoku.

Figure 3. [a] [b] [c] [d]
[a] *Unsolved Sudoku Board* [b]. *Solved Sudoku Board* [c]. *Solution overlaid on the warped grid.* [d]. *Final Result*

2.7. Displaying the output image

The solution we got from step 2.6 (See Figure 3 [b]) is then overlaid on the warped image, which we got after step 2.2 (See Figure 3 [c]). We then transformed this image back to the input image. After that, we combined this image with our input image to get our final output (See Figure 3 [d]).

3. RESULT

We successfully extracted the puzzle grid from the input image, and after resizing the extracted cells of that grid, we predicted digits using our CNN model. The predicted puzzle was solved correctly, and final output was shown successfully.

4. REFERENCES

[1] https://web.stanford.edu/class/ee368/Project_Spring_1415/Reports/Wang.pdf
[2] https://www.techwithtim.net/tutorials/python-programming/sudoku-solver-backtracking/

[3] https://www.pyimagesearch.com/2020/08/10/opencv-sudoku-solver-and-ocr/
[4] https://aakashjhawar.medium.com/sudoku-solver-using-opencv-and-dl-part-1-490f08701179
[5] https://aakashjhawar.medium.com/sudoku-solver-using-opencv-and-dl-part-2-bbe0e6ac87c5
[6]https://missinglink.ai/guides/keras/using-keras-flatten-operation-cnn-models-code-examples/
[7] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
[8] https://www.tensorflow.org/api_docs/python/tf/keras/Model
[9] https://www.kaggle.com/cdeotte/how-to-choose-cnn-architecture-mnist