Sudoku Solver

1. Jay Nagaria 1001564213 **2. Kush Raina 1001567809**

3. Naveen Kumar Ramakrishna 1001371633

Abstract

The Project recognize a sudoku puzzle from a picture. We'll make use of simple image processing algorithms (edge detection, thresholds) and character recognition using the K-Nearest Neighbors (KNN) algorithm. It is a very simple but effective algorithm for solving multi-class classification problems. This puzzle matrix is a 9x9 array of known numbers 1-9 or 0s where the number is unknown This implementation is divided into two parts. The first part is related to computer vision and the second part is related to machine learning. To complete this task it is necessary to use a computer vision library, in this case we are going to use OpenCV library.
Finally, the Sudoku Solver will solve the predicted grid from the previous step.

1 Introduction

In this project we are attempting to solve the sudoku puzzle from scratch. Sudoku, the popular number puzzle that involves filling a 9x9 grid with the digits one to nine in a way that each number can only appear once in the cells in each row and column. This topic is relevant to Machine Learning as it requires involvement of both machine learning algorithms and computer vision. Wide variety of work has been done on sudoku solving using traditional deep neural networks. But ,in our project we are attempting to solve sudoku using kNN which is a classic machine Learning algorithm. So , purpose of this project is to take a new sudoku image as an input and recognize the digits and give us the solved sudoku as output. The project is divided into below list of phases.

- Take an image
- Convert image to greyscale
- Sudoku outline detection
- Sudoku grid detection
- Digit Recognition
- Solving sudoku

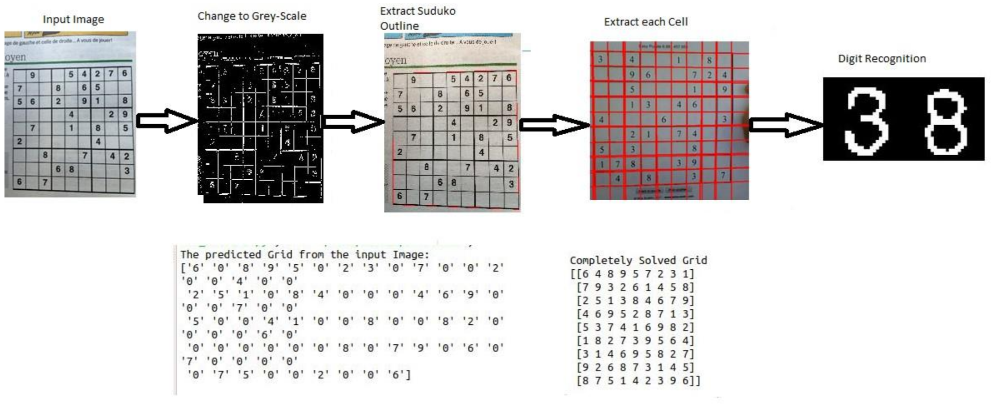
In the below pages we will elaborate on each of these steps which we had mentioned above in Method section and showing the results of our execution in Experiment section.

2 Method

The steps involved in designing and implementing this project are below.

1. Take an image
    - Take a picture from any source .png , .jpg , .jpeg
2. Convert image to grey scale
    - Convert the image to black and white so that the image outlines are prominent. Light pixels are the paper and dark pixels are the ink.
3. Sudoku outline detection
    - Once we obtained grey scale image we have to detect the sudoku lines.
    - We assumed the puzzle is the biggest square in the image and the puzzle will be orientated reasonably correctly.
    - To detect edges, we used an adaptativeThreshold to extract the edge of the image (for each pixel in the image take the average value of the surrounding area).
4. Sudoku grid detection
    - Once you have the outline we need to find individual cells of the sudoku grid.
    - Now, as we know the size of the image and size of the boundary, we can divide the puzzle into a 9*9 grid. Each cell in the grid will correspond (approximately) to a cell on the puzzle. The correspondence might not be perfect, but it should be good enough.
5. Digit recognition
    - Directly using this cells generated by the above step might not give us a good accuracy, so we first do a bit image cleaning and noise removal.
    - Cleaning, rotates the image, so we wrote a code for rotating the image, taking a transpose and then mirror of the image grids for all the 81 cells.
    - Now we obtain s segmented numbers in a grid
    - Use Machine learning tools for digit recognition inside the individual box of sudoku grid.For this we used KNN, and trained using dataset we created for 16*16 pixel images of every digit.
6. Solve sudoku
    - Wrote an algorithm to solve the sudoku.

3 Experiment



```
The predicted Grid from the input Image:
['6' '0' '8' '9' '5' '0' '2' '3' '0' '7' '0' '0' '2'
'0' '0' '4' '0' '0'
 '2' '5' '1' '0' '8' '4' '0' '0' '0' '4' '6' '9' '0'
'0' '0' '7' '0' '0'
 '5' '0' '0' '4' '1' '0' '0' '8' '0' '0' '8' '2' '0'
'0' '0' '0' '6' '0'
 '0' '0' '0' '0' '0' '0' '8' '0' '7' '9' '0' '6' '0'
'7' '0' '0' '0' '0'
 '0' '7' '5' '0' '0' '2' '0' '0' '6']
```

```
Completely Solved Grid
[[6 4 8 9 5 7 2 3 1]
 [7 9 3 2 6 1 4 5 8]
 [2 5 1 3 8 4 6 7 9]
 [4 6 9 5 2 8 7 1 3]
 [5 3 7 4 1 6 9 8 2]
 [1 8 2 7 3 9 5 6 4]
 [3 1 4 6 9 5 8 2 7]
 [9 2 6 8 7 3 1 4 5]
 [8 7 5 1 4 2 3 9 6]]
```

Figure 1: Steps

4 Conclusion

The recognition of the pattern of the numbers is very accurate because the training data is from the image was modified, specifically to achieve a better efficiency in training.

Currently the algorithm works for limited set of images. This can be further improved by running for different size of image and change the training accordingly.In this example we have used a holistic method, all the image is introduced in the classifier. But adding new features to the classifier, for example the number of holes, number of straight lines or length of the lines, could improve the accuracy of the classifier.

5 References

http://www.shogun-toolbox.org/static/notebook/current/Sudoku_recognizer.
html http://docs.opencv.org/ https://github.com/prajwalkr/SnapSudoku
https://caphuuquan.blogspot.com/2017/04/building-simple-sudoku-solver-from.
html https://github.com/JulienPalard/grid-finder
http://www.stackoverflow.com