UNIVERSITY OF WITWATERSRAND

COMPUTER SCIENCE HONORS - DIGITAL IMAGE PROCESSING

Sudoku Solver

Authors:
Philani Mpofu
Goolam Fareed Bangie

Lecturer:
Hairong Bau

June 2021

Declaration

We, Philani Mpofu and Goolam Fareed Bangie, declare that this report is our own, unaided work. It is being submitted for the Digital Image Processing Project for Computer Science Honours 2021.

June 23, 2021

June 23, 2021

Abstract

Sudoku is one of the most popular games for people of all ages. This addictive and competitive game is found in newspapers, online websites and there even exists applications on cell phones just for this game. Sudoku puzzles are thought to have their origins in the mathematical idea of Latin Squares. In the 1780s, Leonhard Euler, a Swiss mathematician, devised a method of organizing numbers so that each number or symbol appears only once in each row or column. The use of digital image processing techniques in relation to sudoku is quite interesting. Digital image processing can be used to extract the puzzle from any image that contains a grid representing a sudoku puzzle, the techniques in image processing together with some machine learning, in particular OCR(Optical Character Recognition), can then be used to extract the cells and numbers represented in each cell. From this the extracted values can be passed onto a sudoku solver which will solve the sudoku. This highlights that digital image processing can be used to automatically solve sudoku puzzles given nothing but an image containing the grid of interest.

Contents

1 Introduction **1**

2 Methodology **3**
2.1 Image Preprocessing 3
2.1.1 Gaussian Blur 3
2.1.2 Thresholding 4
2.1.3 Inversion and Dilation 4
2.2 Cell Extraction 5
2.3 Digit Recognition 7

3 Experimental Results **9**

4 Conclusion **15**

Chapter 1

Introduction

In this section, a brief introduction to the description of the game is discussed, followed by a brief explanation on how image processing and machine learning are used to extract the values from the given image containing the grid. Lastly, the dataset that is used in this research is discussed.

In a sudoku puzzle we start off with a grid, this grid contains 81 cells in a general Sudoku problem, which are organized into nine columns, rows, and regions. The goal now is to fill the vacant cells with numbers from 1 to 9 in such a way that each number appears only once in each row, column, and 3x3 area. A Sudoku puzzle must have at least 17 numbers, but most have 22 to 30. The square root of 81 is 9 which is why only values from 1-9 can be used to fill the vacant cells, however if the grid size was larger then the range of possible values in a cell would increase. In this research however only 9x9 grids will be considered(81 cells).

This research is not focused on solving puzzles but rather obtaining the grids in a matrix format so that it can be passed to a sudoku solver that was implemented by someone else. Image processing and machine learning play vital roles in obtaining the grid from the image that will then be stored in a data structure. The input is an image, the image then undergoes some pre-processing in terms of image processing. Image processing then extracts the grid itself from the image, machine learning then comes into play and uses OCR to extract the values from every cell. The values are then stored in a data structure and are ready to be passed to some sudoku solver. An important note is that if cells are empty in the grid then they are populated with a '0' value in the data structure mentioned above. The methodology section will explain the above process in great detail.

The images that are used in this research are all obtained from a website, 'https://www.rd.com/list/printable-sudoku-puzzles/'. There are about 22 puzzles in the database that we created for this research. The reason that we created our own database was that databases online that had sudoku puzzles all required some sort of payment or the puzzles in the databases were inconsistent in terms of the font and lighting present in the pictures. The images in the database range in difficulty in terms of solving the respective sudoku puzzle. However the main goal is to obtain the values in the grids and in the results section, the images as well as the output matrices will be shown and discussed. An example of one of the images present in the database is shown below.

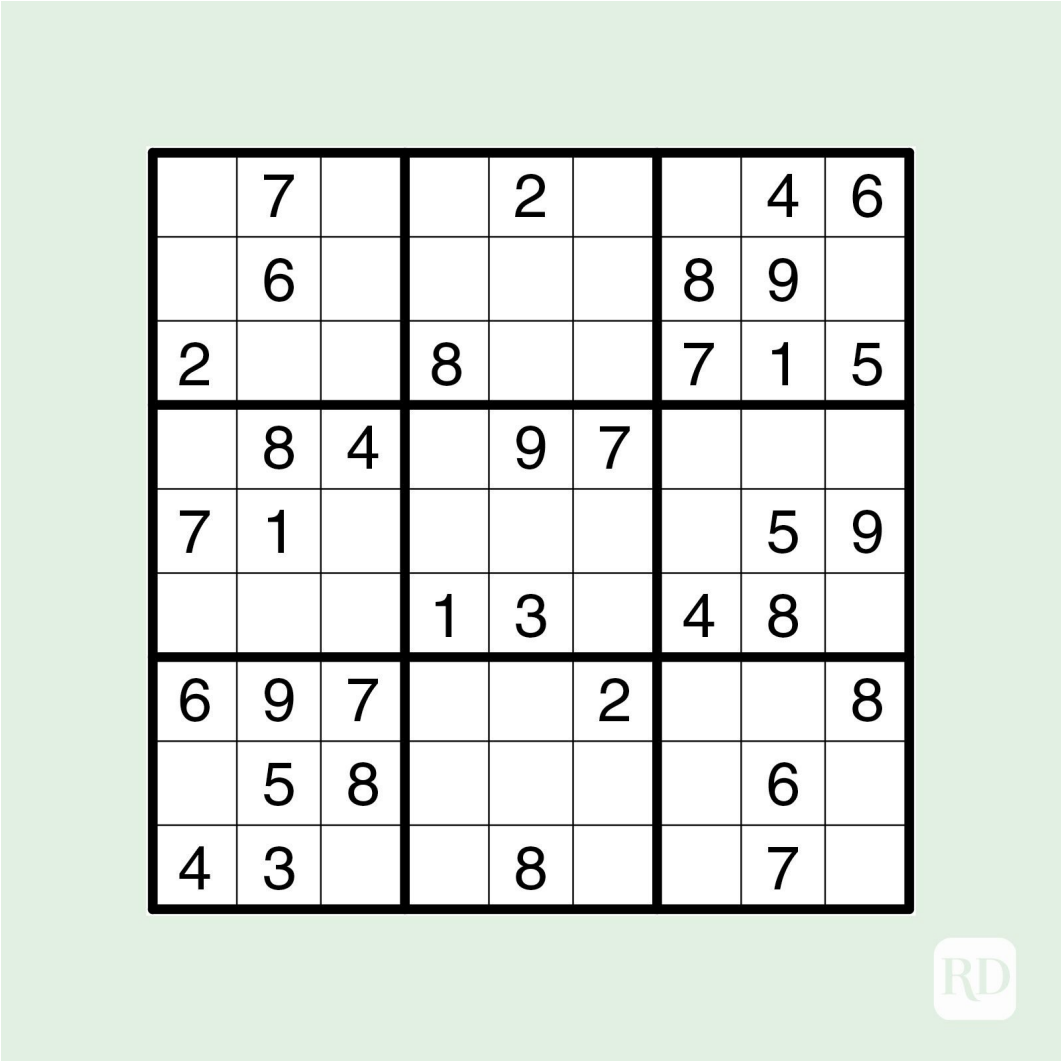


Figure 1.1: Sudoku Image

Chapter 2

Methodology

2.1 Image Preprocessing

After being read into the Jupyter notebook, the images of the Sudoku puzzles are noisy. Noisy images are not very useful because they add uncertainty to the dataset, and this reduces the ability of any model to accuractely classify different classes and labels. This section describes the several techniques implemented (Gaussian Blur, Thresholding, Inversion and Dilation) to decrease the noise in the Sudoku images found in the project database.

2.1.1 Gaussian Blur

A Gaussian Blur will reduce the noise in the input image and make it more appropriate for thresholding. In this project, a 9x9 kernel is used and the standard deviation in the x-direction is set to zero. Figure 2.1 is an image of a Sudoku puzzle after the Gaussian Blur has been applied to it. As you can see, this low-pass filter has sharpened the image, and it is now much clearer.

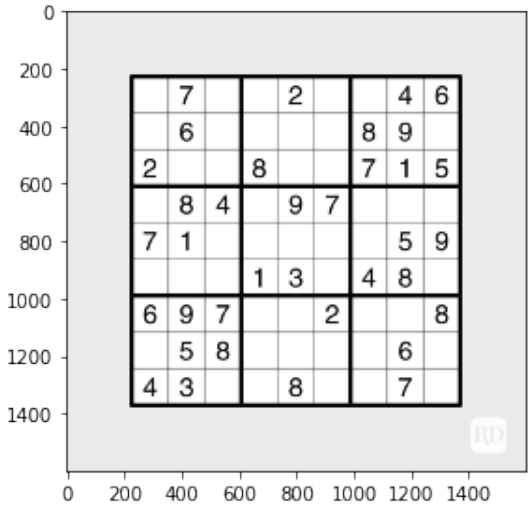


Figure 2.1: Sudoku Image after Gaussian Blur

2.1.2 Thresholding

Following the Gaussian Blurring, thresholding is applied to the image. Thresholding is the process of segmenting an image into several different regions. In a Sudoku puzzle, thresholding segments the image into rows, columns and 3x3 blocks. Figure 2.2 shows the results of segmenting the input image using adaptive thresholding.

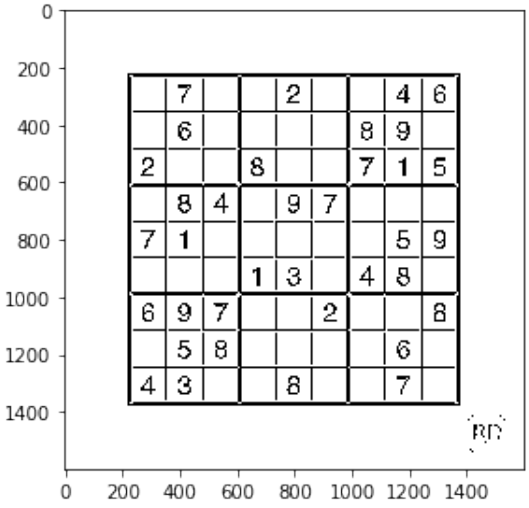


Figure 2.2: Sudoku Image after Thresholding

2.1.3 Inversion and Dilation

In this step of preprocessing, the input image is inverted and dilated. Inversion inverts the grayscale image. This means that all of the white pixels are made black and vice-versa. This is done to improve cell extraction in later sections. In addition, the inverted image is dilated. The Gaussian Blur applied in subsection 2.1.1 sharpened the image, but it did also shrink it. Figure 2.3 illustrates the input image after it is dilated using a 3x3 kernel. In this image, the white pixels are now black, and the black pixels are now white. Furthermore, the lines and borders in the image are thicker due to the dilation.

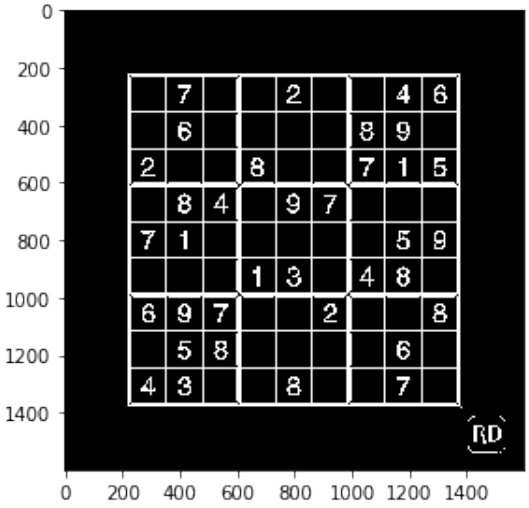


Figure 2.3: Sudoku Image after Inversion and Dilation

2.2 Cell Extraction

The final step of preprocessing is cropping the image. The area outside of the 9x9 grid must be removed from the image because it does not form part of the Sudoku puzzle. We first locate the corners of the Sudoku puzzle in the image. To do this, we find the contours in the thresholded image and sort them by size in descending order. Then, we initialize a contour that corresponds to the outline of the Sudoku puzzle. We implement the approxPolyDP method to find the four largest contours in the image, as they correspond to the four corners of the Sudoku puzzle. Figure 2.4 shows the four located corners of the Sudoku puzzle in the image. After locating the four corners of the puzzle, every pixel outside of the grid is removed from the image and figure 2.5 illustrates what remains of the input image after preprocessing and cropping.

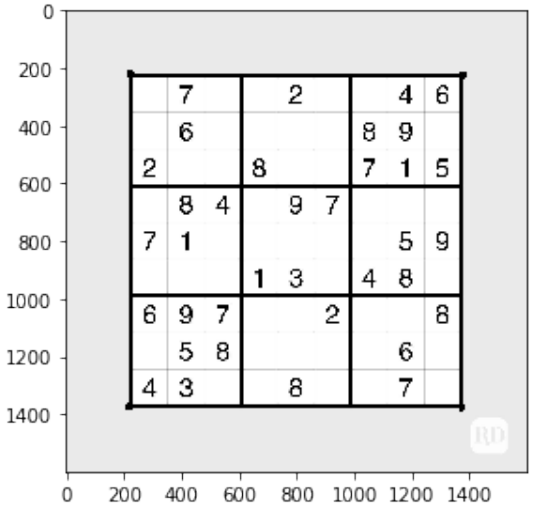


Figure 2.4: Sudoku Image after Corners and Contours Found

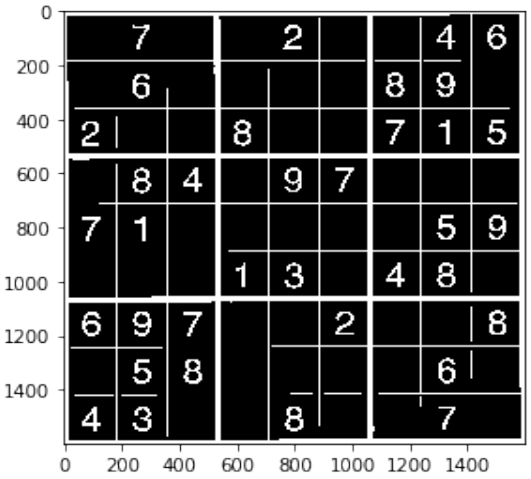


Figure 2.5: Cropped and Processed Input Sudoku Image

After cropping the image, we initialise a 2D list called **finalgrid**. Every element in the **finalgrid** list stores either an image of a number or a blank space. An image in **finalgrid** in position (i,j) corresponds to the number in the sudoku puzzle with the same coordinates. Take, for example, the number 7 in position (0,1) in the sudoku puzzle. Figure 2.6 illustrates how this number is stored in the **finalgrid** list.

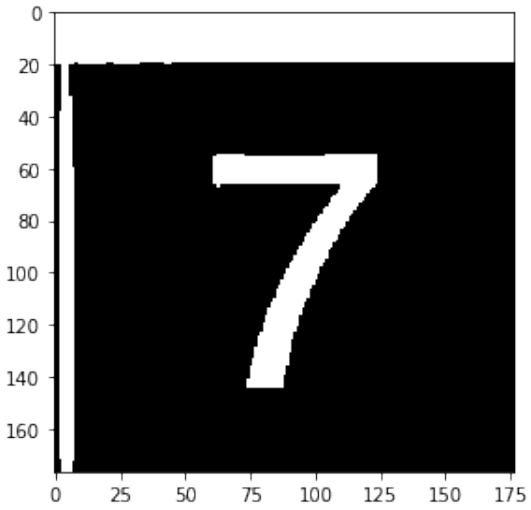


Figure 2.6: Image in position (0,1) of **finalgrid**

Thus, we have successfully extracted the cells from the image into a 2D list. All that remains now is converting the 2D list of images into a 2D array of integers.

2.3 Digit Recognition

In this section, we train a Convolutional Neural Network (CNN) to recognise digits in images. The CNN is trained on the MNIST dataset. Essentially, the MNIST dataset is a substantial database of 60 000 28×28 pixel grayscale images of handwritten single digits between 0 and 9. This dataset is used mainly in image classification problems. Thus, it is an appropriate dataset to use for training in this project. Figure 2.7 details the CNN architecture.

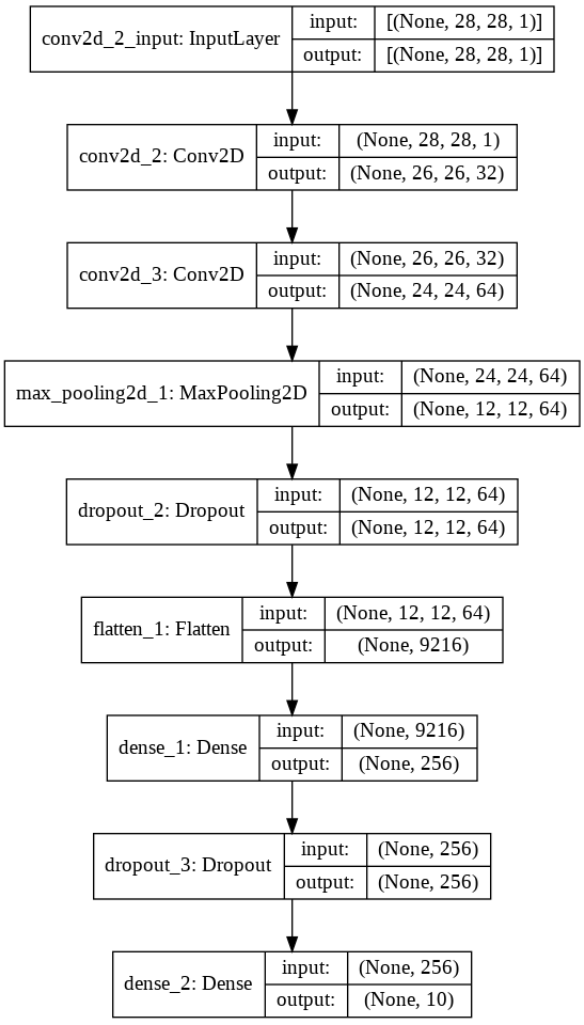


Figure 2.7: Visualization of CNN architecture

After training the CNN architecture, we initialise a 9x9 numpy array called **grid** and it will be used to store the integer representation of the 2D image list **finalgrid**. From here, we loop through every element in the **finalgrid** list and use the trained CNN to predict the number at **finalgrid**[i,j]. If the image at **finalgrid**[i,j] is a blank space, then the CNN predicts that number as 0. Then we set

$$\textbf{grid}[i,j] = predict(\textbf{finalgrid}[i,j])$$

Following this, we are left with a 9x9 array that looks like figure 2.8.

```
[[0 7 0 0 2 0 0 2 0]
 [0 6 0 0 0 0 8 9 0]
 [2 0 0 8 1 0 7 1 5]
 [0 8 2 0 0 7 0 0 0]
 [7 1 0 0 1 0 0 5 9]
 [0 1 0 1 3 0 4 8 0]
 [6 0 7 0 0 2 0 0 0]
 [0 5 8 0 1 0 1 6 0]
 [4 3 1 0 8 0 1 7 0]]
```

Figure 2.8: Extracted Sudoku Puzzle

A discussion of the methodology is presented in the next section.

Chapter 3

Experimental Results

The goal of this research was to take in an image as input, do some digital image processing as well as some machine learning, mainly OCR, and display the sudoku grid present in the image as a matrix representation of the grid. In this chapter, a few randomly selected puzzles from our dataset will be presented together with the matrix representation of the grid from the sample input. Following this, an explanation will be given to the results achieved.

Puzzle 1

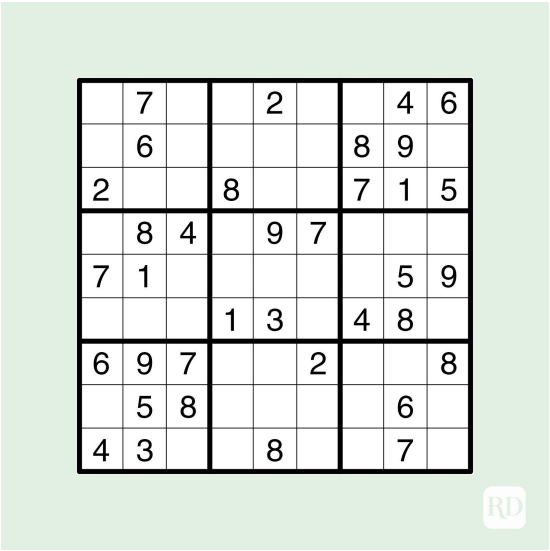
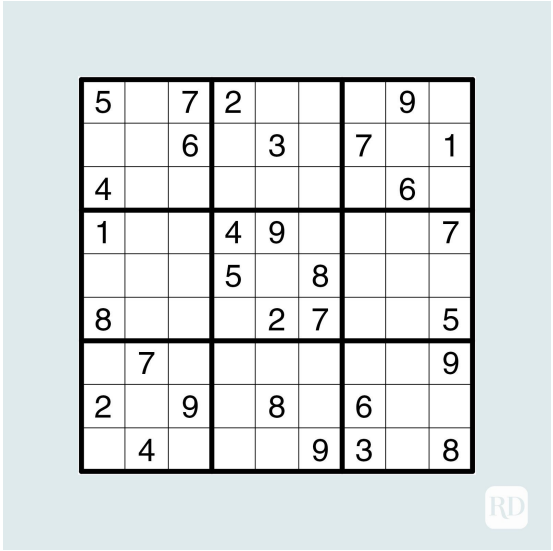


Figure 3.1: Sample Image with Sudoku Grid

```
[[0 7 0 0 2 0 0 2 0]
 [0 6 0 0 0 0 8 9 0]
 [2 0 0 8 1 0 7 1 5]
 [0 8 2 0 0 7 0 0 0]
 [7 1 0 0 1 0 0 5 9]
 [0 1 0 1 3 0 4 8 0]
 [6 0 7 0 0 2 0 0 0]
 [0 5 8 0 1 0 1 6 0]
 [4 3 1 0 8 0 1 7 0]]
```

Figure 3.2: Output Grid in Matrix Format

Puzzle 6



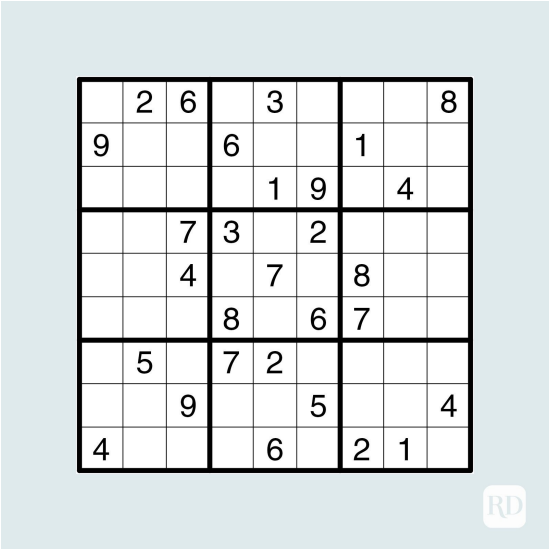
5		7	2				9	
		6		3		7		1
4							6	
1			4	9				7
			5		8			
8				2	7			5
	7							9
2		9		8		6		
	4				9	3		8

Figure 3.3: Sample Image with Sudoku Grid

```
[[5 0 7 2 0 0 0 0 0]
 [0 0 6 0 3 0 7 0 1]
 [4 0 0 1 1 0 1 6 0]
 [1 0 0 2 0 0 0 0 7]
 [0 7 0 5 1 8 0 1 0]
 [8 1 0 1 2 7 1 1 5]
 [0 7 0 0 0 0 0 0 0]
 [2 1 0 0 0 0 6 1 0]
 [0 4 1 0 1 0 3 1 0]]
```

Figure 3.4: Output Grid in Matrix Format

Puzzle 10



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

Figure 3.5: Sample Image with Sudoku Grid

```
[[0 2 6 0 3 0 0 0 8]
 [0 0 0 6 0 0 1 0 0]
 [0 0 0 1 1 9 1 4 0]
 [0 0 7 3 0 2 0 0 0]
 [0 7 4 0 7 0 8 1 0]
 [0 1 0 8 1 6 7 1 0]
 [0 5 0 7 2 0 0 0 0]
 [0 1 0 0 1 5 1 1 4]
 [4 1 1 0 6 0 2 1 0]]
```

Figure 3.6: Output Grid in Matrix Format

Puzzle 15

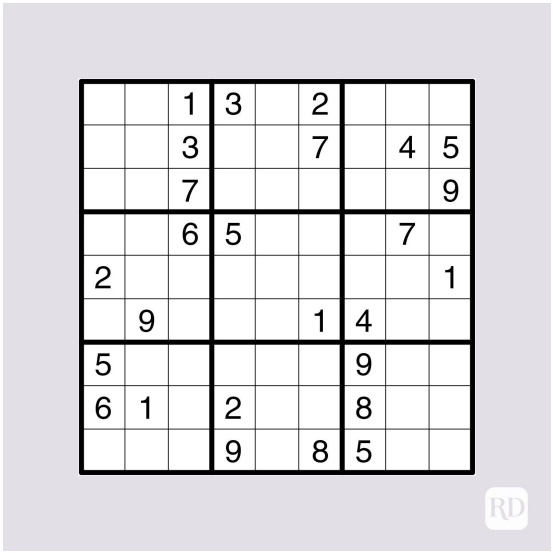


Figure 3.7: Sample Image with Sudoku Grid

```
[[0 0 1 3 0 2 0 0 0]
 [0 0 3 0 0 7 0 4 5]
 [0 0 7 1 1 0 1 0 0]
 [0 0 6 5 0 0 0 7 0]
 [2 7 0 0 1 0 0 1 1]
 [0 0 0 1 1 1 4 1 0]
 [5 0 0 0 0 0 0 0 0]
 [6 1 0 2 1 0 0 1 0]
 [0 1 1 0 1 8 5 1 0]]
```

Figure 3.8: Output Grid in Matrix Format

Puzzle 19

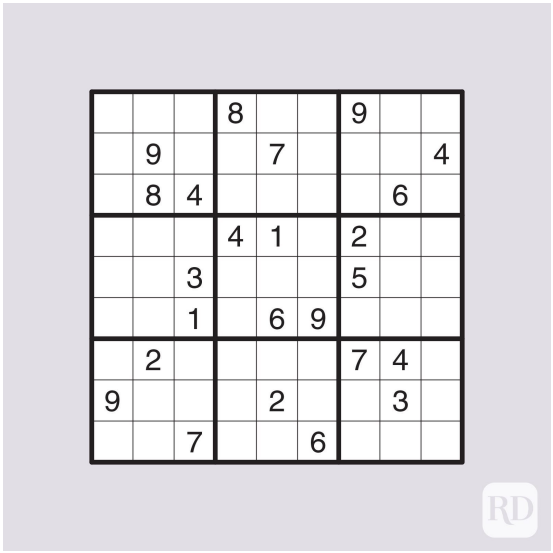


Figure 3.9: Sample Image with Sudoku Grid

```
[[0 0 0 8 0 0 0 0 0]
 [0 0 0 0 7 0 0 0 4]
 [0 8 4 0 1 0 1 6 0]
 [0 1 0 2 1 0 2 0 0]
 [0 0 3 0 1 0 5 1 0]
 [0 1 1 1 6 0 1 1 0]
 [0 2 0 0 0 0 7 4 0]
 [0 1 1 0 2 0 1 3 0]
 [1 1 7 0 1 6 1 9 0]]
```

Figure 3.10: Output Grid in Matrix Format

Discussion

From the sample images above and their corresponding output matrices, it is quite apparent that the methodology adopted and explained in the sections above works. It allows us to take a sample image and retrieve the values from a grid and store them in a matrix format. This matrix can then be passed to a sudoku solver which will return the results for a particular puzzle. One problem was encountered in this research study, certain values were misclassified as other values during the machine learning process of extracting the values from the grid using OCR. This issue occurs due to the way numbers are displayed in images, certain numbers have quite a few similarities and this confuses the machine learning algorithm and as such the incorrect number is returned. The numbers the algorithm struggles with when doing the classification are 0,1,9 and 8. This is due to the way the numbers are presented and because the input to the machine learning algorithm are pictures representing each cell, certain misclassifications occur unfortunately. Accuracies were calculated for each of the above 5 puzzles. 85%,72%,80%,75%,73% were the accuracies recorded for each of the puzzles above respectively. It was also noticed that the classification of the cells occured best for the border cells.

Chapter 4

Conclusion

The Sudoku game is a well known game throughout the world and as alluded to in the introduction, the game is played using various sources such as mobile applications and newspapers. It is however also something of interest to automatically solve these puzzles without need for human interaction. The goal of this report was to do all the steps prior to solving the sudoku grid given an input image. This aspects is important as it will allow the sudoku solving library to receive input that it can use to solve the grid as no current sudoku solving libraries exist that taken in images as input. The method adopted in this report works very well with the database that we composed. It performs all the necessary steps as outlined in the methodology section and retrieves the values present in the grid for a particular image, it then stores these values in a matrix which corresponds to the grid in the image in terms of dimensionality. As seen above in the results section some problems were encountered when converting the cell image into a value which gets stored in a matrix and this is due to the similarities in the representation of the numbers in the images. This is also due to the fact that no machine learning algorithm is perfect and in order to extract perfect grids from images into a corresponding matrix, a lot of time would need to be put into building a very complex model and also obtaining a ton of image samples so that the model can distinguish images that contain these numbers perfectly. Unfortunately time is a major factor here and these changes would require a lot of time, which is not at our disposal. The method above does however perform the task of extracting a grid from an image into output text(a matrix) and does it with decent accuracies for the sample images in the database, reaching accuracies of 85%. Therefore the digital image processing and machine learning methods used above are quite good and for further work in this field, more complex models and more input sample images can be obtained to try and improve the accuracies obtained above.