

# OCR – Sudoku Solver

SudokUwU

**Johan TRAN**

**Adrian GRILLET**

**Virgile COUDERT**

**Simon THUAUD**

**26 October 2021**

**Responsible**

David BOUCHET

## Contents

	<b>2</b>
<b>1 Group presentation</b>	<b>3</b>
1.1 Adrian . . . . .	3
1.2 Simon . . . . .	3
1.3 Johan . . . . .	3
1.4 Virgile . . . . .	4
<b>2 Task distribution</b>	<b>4</b>
<b>3 Progress Made</b>	<b>5</b>
3.1 Sudoku solver . . . . .	5
3.2 Image processing . . . . .	5
3.3 Image Splitting . . . . .	5
3.4 Neural Network . . . . .	6
<b>4 Technical Aspects</b>	<b>7</b>
4.1 Image processing . . . . .	7
4.2 Image splitting . . . . .	12
4.3 Neural Network . . . . .	14
4.4 Sudoku solving algorithm . . . . .	14
<b>5 Conclusion</b>	<b>16</b>

## 1 Group presentation

### 1.1 Adrian

Since I entered EPITA, every project I was asked to do was more interesting than the last and this one follows that rule. After reading the project's book of specifications, I knew it was going to be great. The neural network part seemed so interesting I knew that I wanted to do that part.

This feeling quickly faded as getting started was really challenging and it took quite some time to get something going. But as time went by, the neural network started to resemble what you would expect and getting the first results really revived my motivation that was slowly fading away.

### 1.2 Simon

I have always been interested in group projects! To me it represents the best experience of what the real life is. In fact working in group is a simulation of what we will encounter in the future.

In this project I mainly worked on the image-processing, at first I was really confused of what I needed to use, what kind of filter I had to apply to my image and how to do so. I have watched and read a lot of articles about image treatment and it started slowly to be clearer. Image treatment and image-processing is something that has always attracted me, we concluded that I will work on that!

### 1.3 Johan

As far as I can remember, I always play and have fun with new technologies. First, I used them for their main goal. However, when I was bored of the object, I tried to push it over its limit. I was truly fascinating but I broke it after that. Even if I did not know what I wanted to do when I was younger, I knew that I have this attraction to computer science and new technologies in general.

When I discovered EPITA, it was for me a school that had everything I wanted to do and I naturally join the school in the English class because it's mandatory for the future. In general, I'm a person that feel really frustrated when I do not succeed to do something I'm really into, it's why I try to always give my best when I want to succeed.

## 1.4 Virgile

During this year I was expecting to do the same project as the past year (a text recognition OCR). I was pleasantly surprised to learn that it changed and is now an OCR sudoku solver, which is very similar, but I find the sudoku component more interesting than plain text. I was very happy to work on another project than a video game like we did with my group in S2 and have a more technical experience. When doing the task repartition, We agreed that I would do the sudoku solver algorithm for the first defense. I also plan on finding ways to improve the solving algorithm I did.

## 2 Task distribution

At first, we thought that the neural network would be the hardest part so Adrian and Johan started working on it together. But as time passed, we realized how challenging was the image processing part and so Johan joined Simon on this dreadful task. Virgile was in charge of the sudoku solver and the file reading / writing. This is how the tasks were distributed :

	Simon	Johan	Adrian	Virgile
Image treatment	X			
Image splitting		X		
Neural network			X	
Sudoku solver				X

## 3 Progress Made

### 3.1 Sudoku solver

As we have seen in this project, we need to solve a Sudoku from a given image. We first had to create a program to solve Sudoku, once it's been transformed into a readable text file. As shown in 4.3 figure 8. This implementation with periods made it difficult to handle the transfer from text to the integer array used in the solving algorithm. We implemented this program in C as the rest of the project.

### 3.2 Image processing

Image processing is an important of this OCR project. Since we need to detect a sudoku grid, our image needs to be perfect. In order to do that we apply a lot of different filters. Actually we have all of them:

- Gamma correction
- Contrast correction
- Grayscale filter
- Adaptive threshold (image binarization)
- Noise correction (median filter)

We also have the manual rotation which allows us to rotate manually with a certain angle an image. For example the image 5 needs to be rotated by 35 degrees.

### 3.3 Image Splitting

Image splitting is a main part of the project, because it's where we detect the grid and we cut it to send information to our Neural Network. We have done the line detection and the split of an image that is perfectly cut. We store it in a folder for our Neural Network. We will do a better line detection to have a better split.

### 3.4 Neural Network

For now, we have a program that after inputting all the needed information, is able to learn anything (limited by size since everything needs to be inputted by hand).

The inputs needed (for the XOR) are as follow:

- Number of layers (3)
- Number of neurons per layer (2, 4, 1)
- Learning rate (0.15)
- Number of training examples (4)
- Inputs for all the training examples (0 0, 0 1, 1 0, 1 1)
- Desired outputs for all the training examples (0, 1, 1, 0)

Then the program should train the network for 20000 epochs and inform you when it is finished. It then lets you test the results if you input an example. You can therefore check if the training has been successful.

## 4 Technical Aspects

### 4.1 Image processing

The loaded image goes through a lot of different algorithm in order to have the best image for the grid detection and the grid splitting. It goes through all the filter below:



**Figure 1.** Gamma correction

The first filter applied to the image is the gamma filter. It might be useless but in fact it has a great impact on the noise reducing and the edge detection. We calculate the new R, G and B value by using the following formula:

$$I' = 255 * \left( \frac{I}{255} \right)^\gamma$$

In this formula  $I'$  will be the new pixel color value (i.e: We change the red value,  $I'$  will be the new number of red component and  $I$  is the one before the calculation, gamma is the power of the gamma that we want to apply).

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

**Figure 2.** Contrast correction

Contrast is used to bring out the contours. We use a mathematical formula to compute the contrast correction:

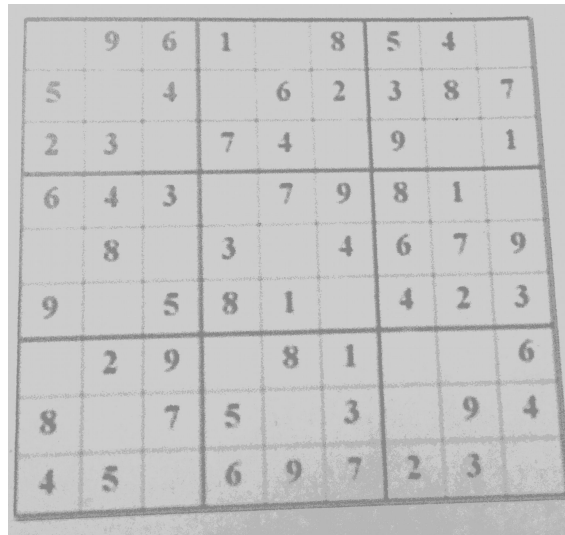
$$F = \frac{259 * (C + 255)}{255 * (259 - C)}$$

This is the factor (the intensity of the contrast we want). It will be used to modify the value of the red pixel with the formula:

$$R' = F * (R - 128) + 128$$

The R (red) value of the current pixel will be changed by R'. The combination of these two processes gives us a more detailed image, edges are more defined and we have a little less of noise. In fact it just prepares the image for what is coming!





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

**Figure 3.** Grayscale correction

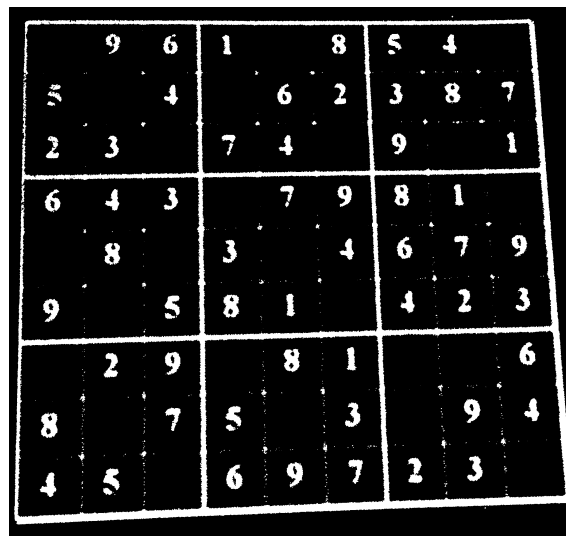
The grayscale filter is important for the next filter. Because we compare only one color component of each pixel in the adaptive threshold.

$$average = 0.3 * r + 0.59 * g + 0.11 * b$$

This average value will remplace all the RGB values of the current pixel, which will turn it gray. We use this formula instead of the basic one because by only doing the sum of the RGB component divided by 3 we can have an overflow. But it is also due to the fact that we are more sensitive to green light, less sensitive to red light, and the least sensitive to blue light. That is why we use the weighted method!

For the binarization of the image we do not use a basic black and white algorithm. We use an adaptive thresholding technique which consist of: the adaptive thresholding technique is a simple extension of Wellner's method. The main idea in Wellner's algorithm is that each pixel is compared to an average of the surrounding pixels. Specifically, an approximate moving average of the last  $s$  pixels seen is calculated while traversing the image. If the value of the current pixel is  $t$  percent lower than the average then it is set to black, otherwise it is set to white.

This method works because comparing a pixel to the average of nearby pixels will preserve hard contrast lines and ignore soft gradient changes. The advantage of this method is that only a single pass through the image is required. Wellner uses  $1/8t$  of the image width for the value of  $s$  and 15 for the value of  $t$ .



**Figure 4.** Black and white correction (threshold

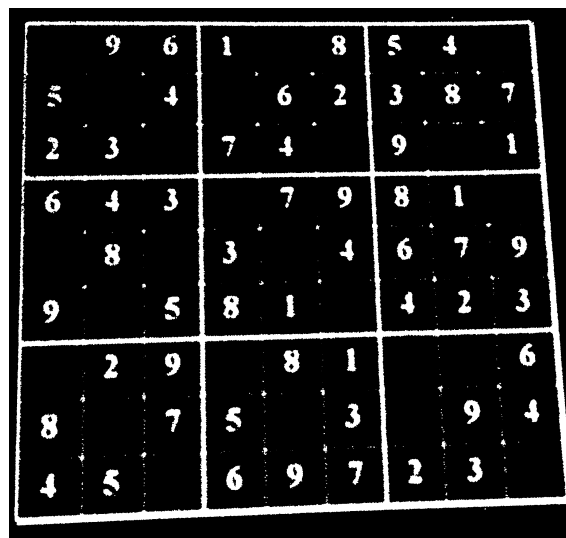
This method uses a two dimensional array in which we store the sum of the current pixel red component value and the last cell in the two dimensional array. And we have then a comparison between the current pixel and the value in the two dimensional array.

To finish, we apply a basic median filter to all the image. It really is one of the easiest algorithm for noise reduction.

6	7	16
22	<i>Pixel</i>	67
7	12	2

The matrix above is just a representation of the current pixel value and the values around. The process is as follow: we take all the 8 pixels value surrounding our current pixel. We store them all in an array which is then sorted in increasing order. We take the median value and we replace our actual current pixel value by the median one.

It has not such a big impact on the final image since it only takes the 8 surrounding pixel but it smooth the image. We will try to implement a better noise reducer algorithm for the next defense.

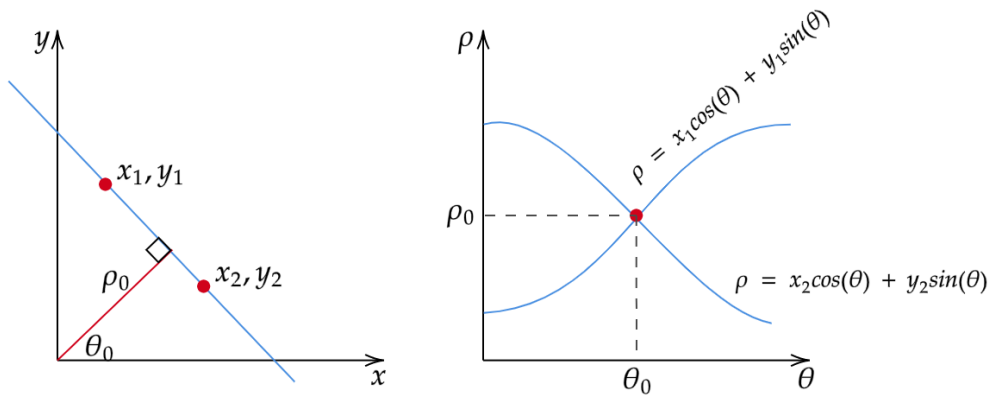


**Figure 5.** Noise correction

## 4.2 Image splitting

We can divide the image splitting in two parts.

- The first one is the edge detection. We use Hough Transform to create lines that will be used for detecting the edge. In theory, almost every line of the form  $y = ax + b$  is represented as a point in the Hough Space (Horizontal axis is the slope and vertical axis represent the intercept)  $(a,b)$ . The major issue with representing line with the equation  $y = ax + b$  is when there's a vertical edge. At this moment "a" can't be determined. To solve this problem you have to switch to polar space  $(\rho, \theta)$ .



**Figure 6.** Two ways of representing a line in a space

Now that we have different coordinates in our polar space, we can detect lines that have an amount of intersections greater than a limit fix.

Let's dive into the algorithm:

First, we have to decide of a range for rho and theta, (theta is often between  $[0,180]$  and rho is between  $[-d,d]$  with d the length of the image diagonal.

With an accumulator that's represented by a 2D array of dimension (num-rhos, num-thetas) each time you detect an edge pixel, you find the values rho and theta and you can increment at the index corresponding.

We went through the accumulator and if the value is greater than a threshold, we can convert it into a line on our image.



**Figure 7.** Hough Transform

We can now detect the position of each intersection point and have the grid of the Sudoku.

- We can now crop the the grid into 81 cases to separate each number of the grid. We save them, with 81 names that are always name the same in the same order to facilitate the neural network and the creation of the Sudoku in the file.

### 4.3 Neural Network

In order to recognize the digits that are already present on the grid, a neural network will be needed. Indeed even if you and I can easily recognize the digits, it is significantly harder for a machine to do so. After the image is processed, the pixels can be either black or white which is like 1s and 0s.

Depending on the size of the picture, we will have a different number of inputs corresponding to the number of pixels. To make the calculations smoother, we will implement a hidden layer in the network. Then, the output will be 10 numbers, either 1 or 0. If the network works, inputting the picture of a 1 will return the following output : 0100000000 (approximately, values will be close to 1 and 0 with a unnoticeable margin for error i.e. 0.998, 0.006).

The activation function we are using for this network is the sigmoid function for the output layer:

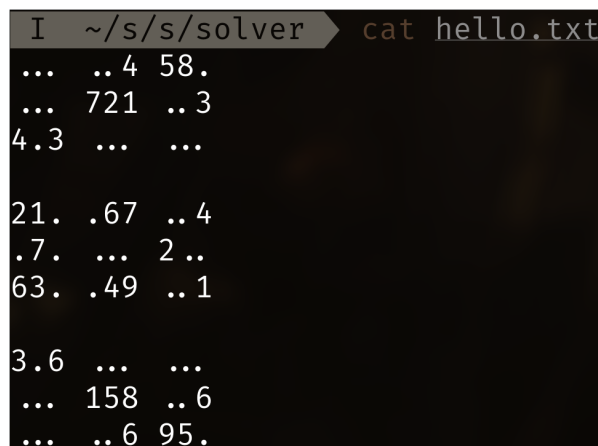
$$S(z) = 1 / 1 + e^{-z}$$

For hidden layers, the Relu function will be used:

$$R(z) = \begin{cases} z & z > 0 \\ 0 & z \leq 0 \end{cases}$$

### 4.4 Sudoku solving algorithm

The solver takes only a file name as input, the file itself contains the Sudoku that we want to solve, with periods representing boxes that we have to fill.

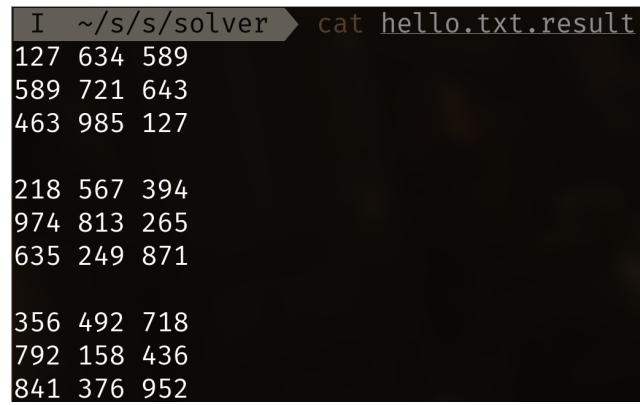


```
I ~/s/s/solver cat hello.txt
... ..4 58.
... 721 ..3
4.3 ... ..
21. .67 ..4
.7. ... 2..
63. .49 ..1
3.6 ... ..
... 158 ..6
... ..6 95.
```

Figure 8. Input file

We must subsequently open this file and transform it into an array of integers, to facilitate solving the Sudoku.

We do this by first allocating memory for the 9 by 9 integer array and reading each character from the input file into the array, skipping newlines and spaces.



```
I ~/s/s/solver > cat hello.txt.result
127 634 589
589 721 643
463 985 127

218 567 394
974 813 265
635 249 871

356 492 718
792 158 436
841 376 952
```

**Figure 9.** Output file

We also make sure to transform the periods into the integer (-1), that will represent an empty space in our array.

We then use a recursive algorithm to solve the integer array, and then write it to a file we created, using the original file name followed by ".result"

## 5 Conclusion

We are proud to see how far we have been able to go and what we were able to learn during this first session. We were able to fulfill almost all the objectives that were set. We only miss a part of the grid detection, but overall everything is done and well done.

For the next defense we will do:

- Neural network that recognize digits (digits that are in the sudoku).
- Complete post-processing.
- The final grid (the resolved sudoku) not as a string printed in a terminal but as an image.
- The save of the result.
- A graphical interface to use all the features listed above.

It is with enthusiasm and determination that we will continue to work on this project! We will do our best to have the best end product.