OneToNine – Project Report

Abigaëlle Panhelleux, Rémy Le Bohec
Hugo Sibony, Antoine Leyglene

September 2022 - December 2022

Contents

1 Introduction

Our goal for this project is to build an O.C.R. software (Optical Character Recognition) the main function of which will solve a Sudoku grid. This is the second major team project that we, as EPITA students, are subjected to. Thus, we hope that it will help us develop our teamwork skills, discover new fields of work that we have not studied before, as well as learn a new skillset that will most likely be useful later in our lives.

2 Group Presentation

2.1 Abigaëlle Panhelleux

My name is Abigaëlle Panhelleux and my role in this group is to create the whole neural network that we will need for the project.

For the first defense, my main role is to learn and create a neural network able to learn the xor operator. In the last defense, I will have completed the final neural network used for our project to learn how to take the image of a sudoku and solve it.

I chose to take the neural network because I wanted to try something different. In SUP year, we all did algorithms throughout TP that are similar to the project we are doing now (e.g. checkers), and I wanted to step out of my comfort zone. I knew that before I coded the actual network, I would have to do a lot of research, as I knew nothing about neural networks.

2.2 Rémy Le Bohec

My name is Rémy Le Bohec and I will be the lead student for the image preprocessing section of the project.

The reason why I chose to take care of the preprocessing is because I am not very fond of neural networks but I am more interested in manipulating the input image to remove its impurities and I enjoyed working with SDL during our prog practicals.

I am looking forward to getting more familiar with the C language throughout this project as well as understanding more thoroughly the use of libraries in order to improve my skills and not get lost in the piscine next year.

2.3 Antoine Leyglene

My name is Antoine Leyglene and I will be in charge of processing the grid, separating the different cells to extract the values, then returning a usable representation of the original grid.

I had the responsibility for this first defense to finish the grid separation of the image, as well as the segmentation. I did not choose to work on this part of the project for a personal reason, but because I knew that all the different tasks would allow me to learn new things, and therefore choose after the others.

2.4 Hugo Sibony

I'm Hugo Sibony, at the beginning I was in charge of all the aspects of the Sudoku part: the solver, the display board, the generator and so on. But thanks to my previous experience in this field, I finished my work quite quickly, which allowed me to be available to help in other parts of the project.

Regarding the chosen part, I have already worked on neural networks on professional and personal projects, so I wanted to expand my expertise by letting another teammate choose this topic, but as said before, I am actively helping these teammates with my experience.

3 Task Repartition

Given our diverse programming knowledge and interests, we have chosen to divide the workload into four main categories. Although a lead student works on each task, we try to work together as much as possible to avoid easy mistakes. Therefore, this table is only indicative.

3.1 Assigned Roles

	Antoine	Rémy	Hugo	Abigaëlle
Neural Network			+	⊗
Image processing		⊗		
Grid detection	⊗			
Sudoku			⊗	

3.2 Completion of tasks

Task	Expected	Current
Image loading and color removal	100%	
Image rotation	100%	
Grid detection	100%	
Segmentation	100%	
Sudoku solver binary	100%	100%
Neural Network XOR	100%	100%
Save / Load NN's Weights	33%	50%
Dataset for learning	33%	100%
File manipulation for result saving	33%	40%
Display board	0%	66%

4 Image Preprocessing (Rémy)

The image preprocessing stage I will attempt to implement for the OCR project is split in multiple milestones.

For this defense, I have worked on manual deskewing of the image (rotation from an angle entered by the user) to try and grasp the concepts of SDL and image rotation. Furthermore, I also have started working on a function to convert the original image to grayscale by manipulating the original RGB values of the present image.

4.1 Rotation of an image

The process of manual rotation consists of creating a new surface of width and height equal to the original image, but where each pixel's coordinate is recalculated.

The calculations are as follows:

Let dx be the value corresponding to the x coordinate of the pixel minus the x coordinate of the center of the image and dy the value corresponding to the y coordinate of the pixel minus the y coordinate of the center of the image.

The new x coordinate is equal to dx times the cosine of the angle plus dy times the sine of the angle plus the x coordinate of the center of the image.

The new y coordinate is equal to dy times the cosine of the angle minus dx times the sine of the angle plus the y coordinate of the center of the image.

In order to rotate the image, we simply need to apply this calculation over all pixels of the original image to place the said pixels onto the new surface.

4.2 Grayscale

In order to convert the image to grayscale, a simple trick is performed based on the RGB values of the pixels.

Indeed, we only need to compute the weighted average of the RGB components of each pixel in order to obtain a shade of gray.

The formula is as follows:

$$\text{average} = 0.3 \times r + 0.59 \times g + 0.11 \times b \tag{1}$$

Finally, we only need to replace each pixel's RGB component by a 3-tuple (r', g', b') where r'=g'=b'=average.

4.3 Conclusion

For the next defense, I plan on working on removing the noise of the image (grains, stains and so on) as well as automating the deskewing process uing skew angle detection and finally a "better" grayscale algorithm that will make the image entirely black and white to facilitate the recognition of the characters.

5 Grid Segmentation

5.1 Cell Isolation

In most documents that I read about grid segmentation, the first part always consisted in a "cut" of the grid. This meant going through the original picture, analyzing the entire picture and isolating each individual cell and what it contains. I was not familiar with this kind of algorithm so I attempted to search for pseudo-code on the Internet. I was not successful, as the majority of videos or tutorials on this topic used Python libraries, like OpenCV, that are not available in C which slowed down my progression. However, my search for a standardized method helped me understand what was expected. The idea was to first estimate the width and height of the grid and deduce the size of a cell, then find the top left corner of the grid. After that, simply go through the determined coordinates of the cell(using the size of the cell), and store an isolated image of the cell, and do so for the entire grid. After recovering every cell, we move on to the next part.

5.2 Cell Treatment

The next step is to recognize what is contained in each cell, a number or a blank. In theory, this should be quite simple. Since the Sudoku grids that we are using are printed ones, we don't need to analyze any different handwriting, as numbers should have a standard shape. Now the only important things would be to match the number in the cell to all the possible numbers and assign the newly found number to the cell. After going through all of them, we create a double array, in which we progressively add the values to the array, using 0 as the value for blanks.

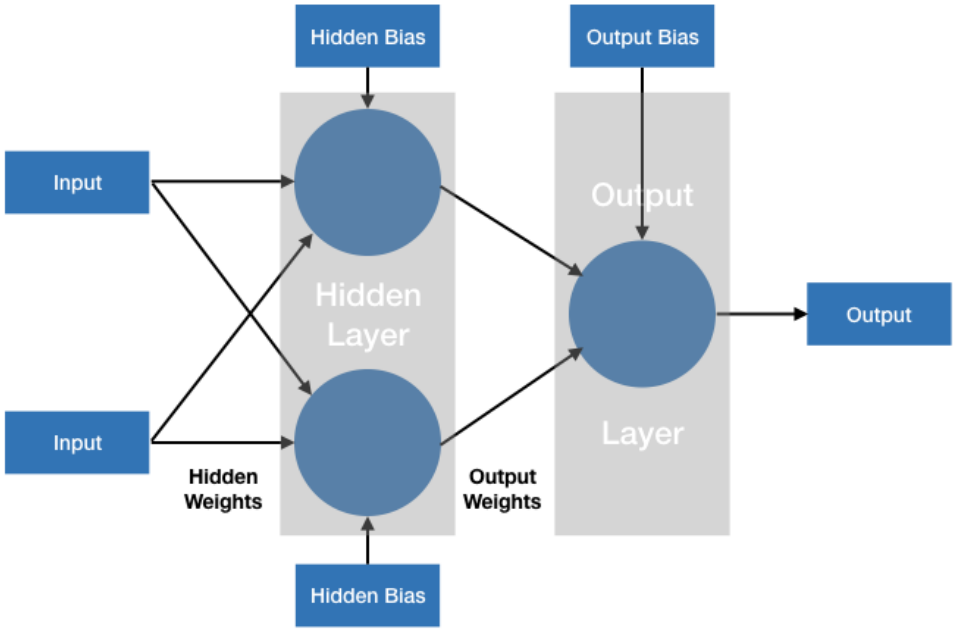
5.3 Future updates

I was not able to properly implement those algorithms, thus for the next defense I need to write a functioning version of those two steps. Then, I will work with Abigaëlle in order to use the Neural Network in order to streamline the number recognition process.

6 Neural Network (Abigaëlle)

6.1 Learning the XOR operator

This section will explain how the XOR learning algorithm works. The perception.c file is mainly separated into two parts : the learning part, that contains the learning algorithm, and the main part, that is "accessible" to the user. The main part is mostly explained in the 'How to use it' part.



For the algorithm, we have :

- The hidden layer, that has 2 nodes. Its weight/bias is hiddenWeights and hiddenLayerBias.
- The output layer, that has 1 node. Its weight/bias is outputWeights and outputLayerBias.
- The input layer, that has 2 nodes.
- The input layer : the input to the hidden layer.
- The hidden layer : the input to the output layer.

The file starts with some basic functions that will be needed by the learning algorithm. We need the algorithm to use the inputs in a random number to get the maximum effect on it. So we have a shuffle function that will shuffle the order the algorithm will take the inputs. At the very beginning of the learning, the weight needs to be random, so the

init_weight function() does it. The sigmoid and dsigmoid (the derivative of sigmoid) are used to get the new values of the output and hidden layer.

In this neural network, I decided to use what is mostly used in these types of problem: the stochastic gradient descent. I shuffled the inputs, and I need to update the hidden layer using the activation sub function, then update the output layer.

Then, the next step is to change the weights and the bias of both the hidden layer and the output layer so that the algorithm makes less and less error. We loop all of this for each set of inputs, then for each epoch.

6.2 How to use it

To see if the learning algorithm works, the only thing the user needs to do is compile the perception.c file. After that, he needs to use the line ./perception <numEpoch>, with <numEpoch> the number of tries that he wants the learning algorithm to make. The terminal will then show the user the comparison between what the algorithm found and what he should have found. So that the user can see precisely at what percentage the algorithm is right, the main function will show the exact float number that the learning function returned. If the number is below 0.5, the result is supposed to be 0, else it is 1. Also, the user can see, for each try, the error that the algorithm makes. If it is below 0.1, it is marked green, else it is marked red. No matter if the error is green or not, the biases and weight will be saved in the xor.nn file.

6.3 The making

This section will talk about the struggles that I faced and how I managed to learn and code the neural network.

I was scared at the beginning because I didn't know where to start. I never tried to do anything on my own with no specific guidance like the TPs we normally have. All I knew was that I didn't know what was a neural network. So I started with that. With all of the hints that were given, I started my research. Later, I watched the YouTube channel that Hugo recommended to me.

After I knew what every vocabulary I needed mean, I started trying to figure out how to use them and in what order. So I started looking on the internet how people would use all this knowledge to create a learning algorithm. After figuring out how to put everything together, I started coding.

The coding section wasn't were hard. With the knowledge that I had now, I could easily figure out how to do everything. I helped my coding with two websites, one of them using python, that wanted to code the same problem as me.

After I coded everything, Hugo came to help me test it. He helped me debug my algorithm and created for me the main function that would help the user know if the learning function works. He made some adjustments with my learning algorithm and now everything works!

7 Sudoku Solver

7.1 Choose the right algorithm

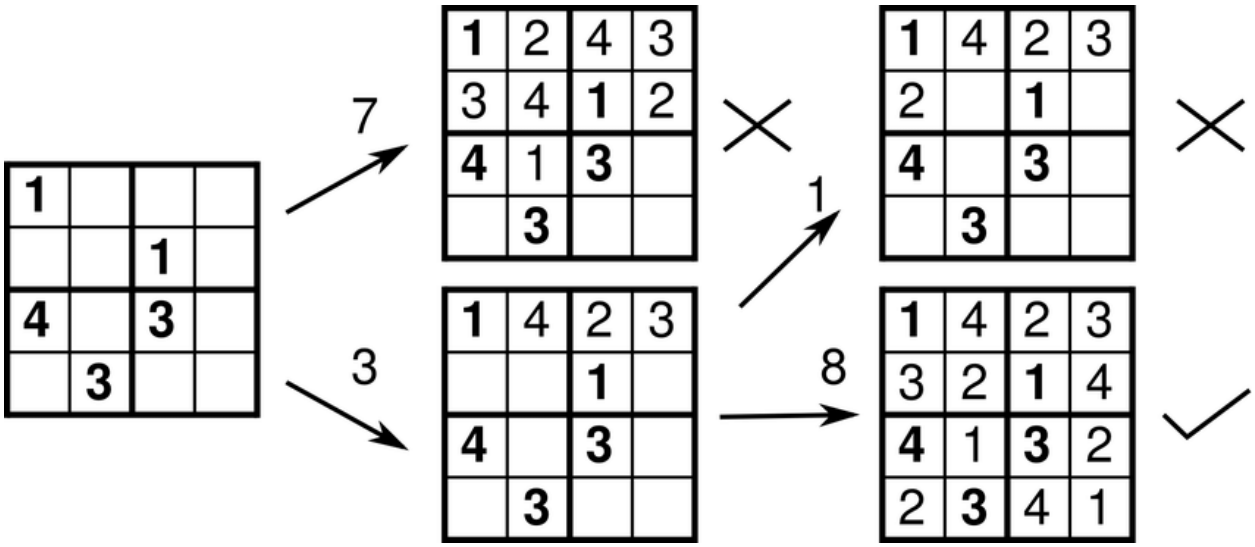
To begin implementing Sudoku, some research was needed. During my quest for the best Sudoku algorithms, I discovered not only that there were countless different ways to deal with this problem, but also that there was a giant community, spread across the globe, working together to come up with innovative ideas to solve problems like the Sudoku solver.

Two hours scrolling endlessly into StackExchange and OpenClassrooms, three algorithms stood out to me :

1. Backtracking
2. Rule-based
3. Boltzmann machines

On the one hand, the Boltzmann machine method is much more fun and interesting with its stochastic behavior, but on the other hand, it is much more difficult to implement and understand, in addition to being unpredictable. The rule-based approach is the best in theory, but we decided to use the backtracking algorithm (for the first defense at least) because it is the perfect compromise between the two methods. It is easy to implement and understand, for this use case its lack of performance is acceptable. Its deterministic behavior is what we need for this project.

7.2 Backtracking



The backtracking algorithm is a recursive algorithm that tries to find a solution to a problem by brute force: trying all possible solutions until one is found. It is based on the following steps:

1. Find an empty cell
2. Try all numbers from 1 to 9
3. Check if this number is valid in the current cell
4. If the number is valid, fill the cell with this number and recursively try to fill the next empty cell
5. If we can't find a valid number for the current cell, we return false and go back to the previous cell
6. If we have filled all the cells, we return true

8 Conclusion

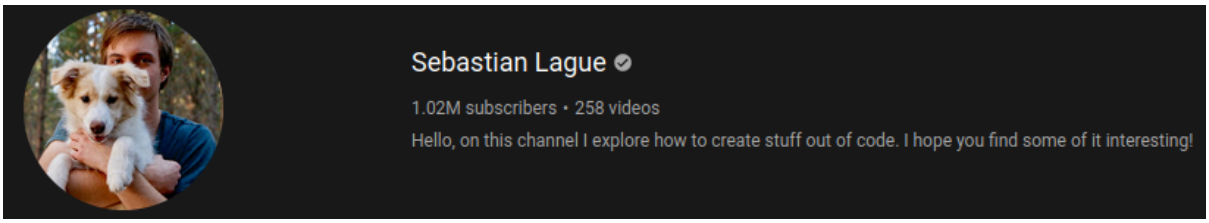
To conclude, this first defence allowed us to start working on what has to be done for our O.C.R. to be functional. For the next defense, it will be necessary to correct the mistakes that were not fixed, and work as a team to implement the last remaining functionalities so that our software works as intended.

OneToNine

9 Appendices

This section contains some of the resources used in the creation of the project.

- The best coding channel EVER :



Each of his videos is a masterpiece that explains in depth a rather complicated subject while remaining entertaining: namely neural networks.
This helped a lot understanding the main principles of Neural Network

- TDS "Simple neural network implementation in C" and "Understanding Basics of Deep Learning by solving XOR problem" were also very helpful in the coding part of the neural network. The image used in 'Learning the XOR operator' comes from the first website.