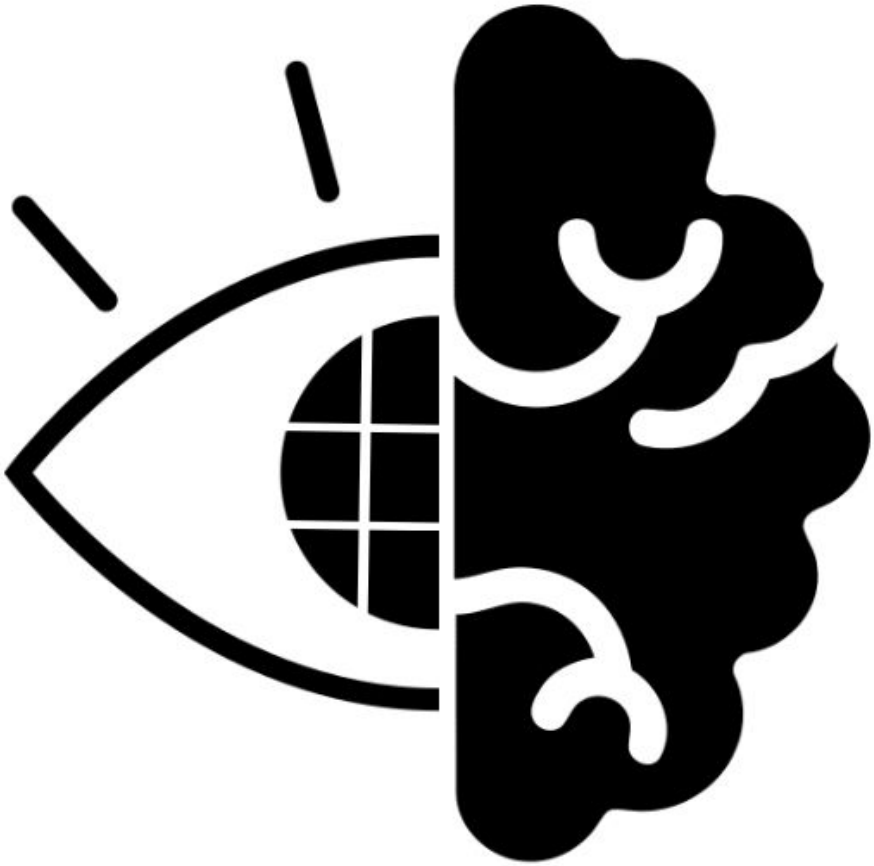
NSDUO - Defence I

Fayçal BEGHALIA . Salma BOUBAKKAR . Yness KHOUADER . Willam LAM

November 4, 2023



Contents

1 Introduction

This paper is the first defense report of NSDUO's Sudoku OCR project.

The aim is to extract and recognize the numbers that make up a classic Sudoku from an image. This program will then solve it, display and save a new image of the solved grid.

Regarding this report, you will be able to find a table of contents bringing together the different points covered in this report. It is also an opportunity for us to provide a written update on our progress, the state of maturation of the project, and our distribution of work so far. It allows us to express our feelings about this project since it began a few weeks ago.

2 Members of NSDUO

2.1 Fayçal Beghalia

Fayçal Beghalia is NSDUO's project leader. He's an average C# enjoyer, but he also likes C. In his life he did some websites, video editing and game development as well as modding for games.

His main role in the project was to coordinate everyone and ensure that all tasks ran as smoothly as possible. He worked on all aspects of the project, but was mainly involved in image processing.

2.2 Salma Boubakkar

Joining the Train Sudoku Challenger (TSC) for the Sudoku Solver project marked a thrilling new adventure for Salma Boubakkar, a second year student. This venture presented a fresh challenge within a dynamic group of developers.

For Salma, this Sudoku-solving mission was more than lines of code. It is an opportunity to elevate her programming prowess and immerse herself in new, complex concepts. However, it is not without its hurdles. The Sudoku Solver project will brought forth a formidable set of challenges that will test her problem-solving skills and push her boundaries.

Salma will embrace the difficulties, for it will in overcoming these obstacles that her understanding of programming and algorithmic intricacies deepened. This project is not merely about solving puzzles but also cracking the enigma of efficient algorithms. With each obstacle that will be surmounted, Salma will grew, enriched by the experience of tackling intricate problems within the vibrant, collaborative TSC community.

2.3 Yness Khouader

Yness Khouader is a 20-year-old individual with a deep passion for two things : tennis and artificial intelligence. These two seemingly distinct interests have come to define a significant part of who she is and how she engages with the world around her.

As a tennis enthusiast, she has not only enjoyed the sport for years but have also found it to be a source of inspiration and discipline. Tennis has taught her the values of perseverance, strategy, and maintaining a strong work ethic. It has been both a physical and mental journey that continues to influence her personal growth and approach to challenges.

In parallel, her fascination with artificial intelligence has driven her to explore the intricate world of machine learning, deep neural networks, and the endless possibilities they offer.

These two passions might seem divergent, but she sees them as complementary. Tennis has instilled in her the spirit of competition and the drive to excel, qualities that serve her well in the ever-evolving world of AI.

She is excited to bring this dedication, ambition, and the unique perspective cultivated by her passions to this project she is involved in.

2.4 Willam Lam

William LAM is 19 years old and is currently a student at EPITA, pursuing his education. Outside of his academic pursuits, he is an avid sports enthusiast, with a particular passion for tennis. He started playing tennis at the age of 7 and has been involved in competitive play ever since.

William's love for exploring and learning new things has led him to develop a keen interest in traveling. He finds great joy in discovering new cultures, meeting people from different backgrounds, and immersing himself in unfamiliar environments. This innate curiosity and thirst for knowledge are also what led him to join EPITA, even though he had no prior knowledge or experience in the field of computer science. The excitement of uncovering new skills and embarking on a fresh adventure compelled him to take the leap and dive into this exciting new journey.

EPITA has provided William with the platform to learn and grow, not only academically but also personally. It has pushed him to step outside of his comfort zone and embrace the challenges of acquiring new technical skills. Through dedicated learning, hands-on experiences, and collaboration with his friends and teachers, he has been able to immerse himself in the world of computer science.

In addition to his passion for sports and learning, William values the importance of teamwork and effective communication. He believes that working together, exchanging ideas, and supporting one another are integral to achieving success in any endeavor.

3 Tasks Distribution

Global Advancement					
Section	**Subsection**	**Fayçal**	**Salma**	**Yness**	**Willam**
Image Processing	**Filters**				
	Rotation				
	Segmentation				
	Hough Transform				
	Detection Square				
Machine Learning	**Neural Network**				
Sudoku Solver	**Solver**				

Black for leader - Gray for coworker

4 Image Processing

To solve Sudoku, we first need to extract the grid from the image. However, to do this, we must first process it to rid it of irrelevant information. To do this, we rely on filters.

Filters are algorithms applied to images. Here, we'll use grayscale and threshold to enable our neural network to recognize our digits.

4.1 Grayscale

Grayscale, as the name suggests, is an algorithm used to remove all colors from an image. The result should contain only grayscale. To do this, we iterate over each pixel, then give that pixel the average value of its R, G and B components.

For example, let's say we have a pixel with R=10, V=100, B=0. The average will be (20 + 100 + 0) / 3 = 40. Our new pixel color will therefore be R=40, G=40, B=40.

This is useful when color has no impact on our result; here we only have one value per pixel to check, as color doesn't help us differentiate between numbers. We can go even further, as we'll see with thresholding.

4.2 Threshold

The threshold will allow us to have a black and white image. To do this, we recommend first applying the gray scale filter to our image, then checking whether the pixel value is below or above a threshold value. If it's below, the pixel is black (0), otherwise it's white (255).

4.3 Gaussian

The Gaussian filter is a more advanced way of blurring an image. When processing images for neural networks, we are often confronted with noise in the image. To eliminate it, we can blur the image, but medium blurring leads to a loss of precision.

To alleviate this problem, we use Gaussian blur instead, which uses a kernel to blur the image, i.e. when blurring, pixels that are close to us have more impact than pixels that are further away. The image contours are thus preserved.

4.4 Sobel

The Sobel filter is an image processing technique used for edge detection. It calculates the intensity gradient at each pixel to identify and highlight the edges of an image. It consists of two convolution kernels, one for detecting horizontal changes and the other for vertical changes in intensity. The combination of these kernels determines the magnitude and direction of the gradient, which, when thresholded, produces a binary map of edges. The Sobel filter is essential for tasks such as object detection and image segmentation.

5 Segmentation

5.1 Image Partitioning

In this section, we will discuss the cut-and-save function, which is primarily designed for processing SDL surfaces, often representing images. We use this function to partition the input surface into smaller cells and save the non-empty cells as individual image files within an "output" directory. We begin by creating the "output" directory, ensuring its existence.

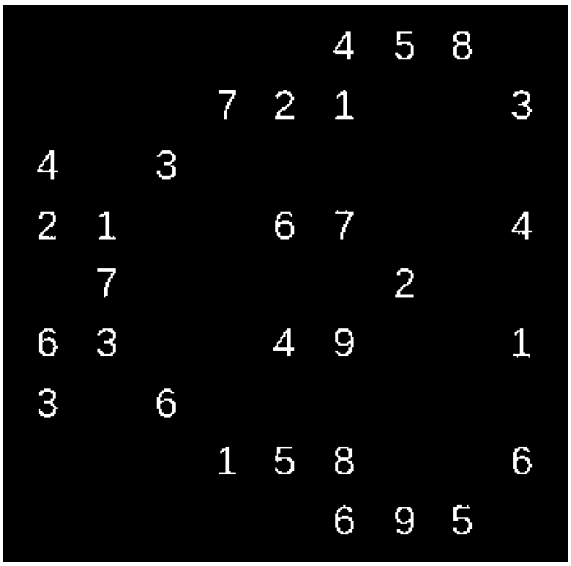
The next crucial step involves calculating the dimensions of each cell based on the original surface's dimensions and the provided value of n. This calculation determines the size of the individual cells within the grid.

Two nested loops are then employed to iterate through the cells, effectively traversing the rows and columns of the grid. For each cell, a new SDL surface (image) is created to represent an individual cell.

To prevent conflicts when accessing pixel data, we lock the image surface for manipulation. We then proceed to copy pixel data from the corresponding region of the input surface to the image surface within nested loops, effectively isolating each cell from the original image.

A crucial aspect of this function lies in checking the center of each cell for the presence of white pixels, determining whether the cell is empty or non-empty. If a cell is found to be non-empty, we save it as an individual PNG image within the "output" directory. The naming of these files is based on the position of the cell within the grid.

Finally, after processing each cell, the image surface is unlocked, and its resources are freed to prevent memory leaks.

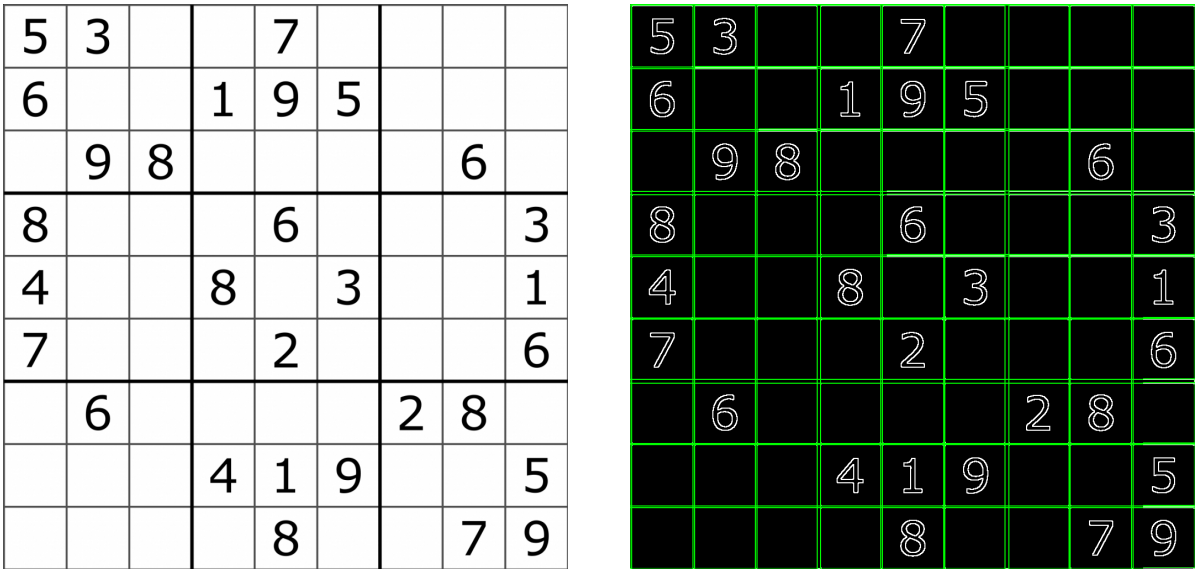


(a) **Before cutting**

(b) **Some of the generated images**

5.2 Hough Transform

The Hough Transform is a image processing algorithm used to detect geometric shapes in images. The idea behind the Hough Transform is to represent these shapes (here only lines) as mathematical equations in the hough space depending of rho and angle theta. Then to identify a line, each point in the input image is converted into a vote in the Hough space, and the parameters that receive the most votes correspond to the detected shapes in the original image. Here we are using the Hough transform to detects Sudoku cells. With it we will be able to draw the lines detected in the image. The next step is to detect the Sudoku grid square and crop the image.



(a) **Before Hough transform** (b) **After Hough transform**

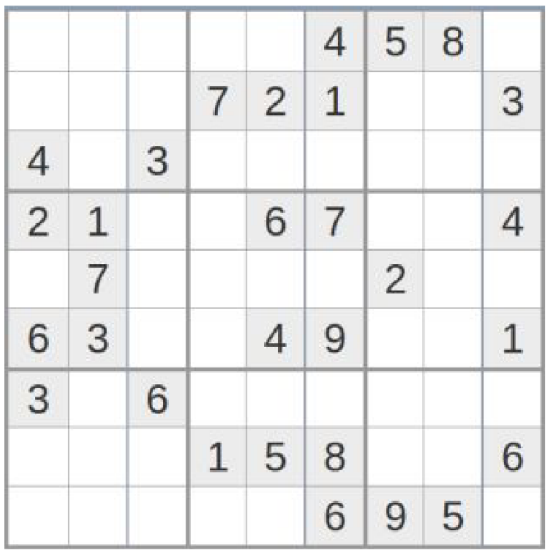
5.3 Rotation

In the rotation part, we performs a rotation transformation on an SDL image. The process begins by converting the user-provided angle from degrees to radians, ensuring it's compatible with trigonometric functions. The new dimensions of the rotated image are then calculated with precision, accounting for the angle of rotation. This is done by using trigonometric functions such as cosine and sine to calculate the width and height of the new canvas.

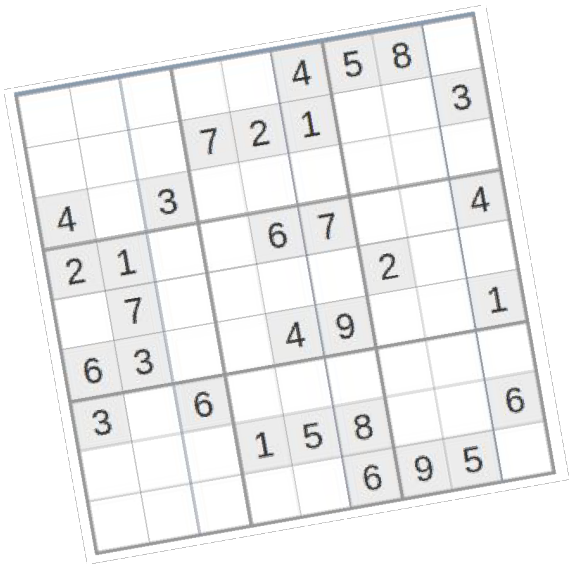
The pivotal point in this process is the pivot point, which is set at the center of the original image. It acts as the axis around which the rotation occurs.

The heart of the rotation lies within nested loops. These loops iterate over each pixel of the new canvas, computing the corresponding coordinates in the original image using trigonometric transformations. If these coordinates fall within the bounds of the original image, the corresponding pixel is copied to the rotated canvas.

This pixel-by-pixel transformation results in a visually rotated image, showcasing the power of mathematics and code in image manipulation.



(a) **Before rotation**



(b) **After rotation of 10°**

6 Machine Learning

6.1 XOR

Creating an artificial neural network able to learn the XOR (or exclusive) function is a classic task in machine learning. To do this, we used a two-layer neural network, also called a feedforward neural network, because information flows in only one direction, from input to output. Here is an overview of the process of creating this network:

1. Neural network setup: - We created a neural network with two layers : an input layer and an output layer. Each layer contains a certain number of neurons, usually three in this case. - The input layer has two neurons, one for each input value (0 or 1), representing the operands of the XOR. - The output layer has a single neuron which gives the XOR response.

2. Initialization of weights: - The weights of the connections between neurons are initially defined randomly. This step is crucial because this is where the learning will take place. The weights are adjusted as the network learns.

3. Activation function: - We use an activation function, generally the sigmoid function, to introduce non-linearity into the network. The sigmoid function takes as input the weighted sum of the inputs and produces an output between 0 and 1.

4. Network training: - To learn the XOR function, we need to provide the network with a training dataset that contains all possible combinations of the input values (0,0), (0,1), (1,0) and (1 ,1) with their expected outputs (0, 1, 1, 0). - Training involves adjusting the weights of connections between neurons iteratively using a learning algorithm, such as gradient back propagation.

5. Validation and adjustments: - After training, we need to check if the network is able to correctly predict the XOR output for new inputs. If not, further adjustments may be necessary, such as increasing the network size or increasing the number of training iterations.

Ultimately, after successful training, the neural network should be able to take the input values (0 or 1) and produce the corresponding XOR output (0 or 1) accurately. Creating such a neural network demonstrates the ability of machine learning to solve complex problems, even when dealing with non-linear functions like XOR.

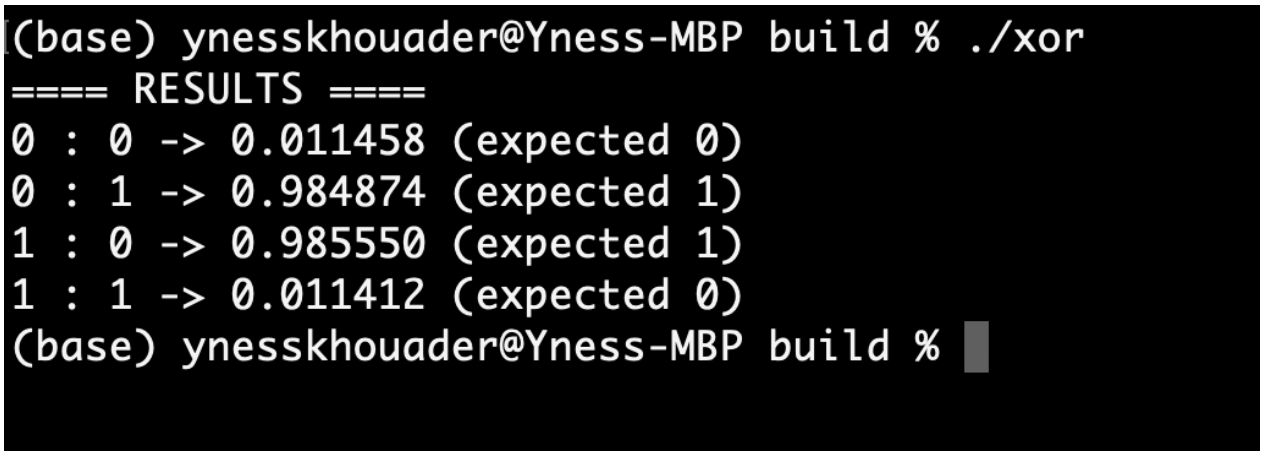


Figure 4: **Results of the neural network**

7 Sudoku Solver

In this section, we will take a more detailed look at the core functions that comprise the Sudoku solver. These functions collectively handle the essential tasks of loading the Sudoku grid, employing a backtracking algorithm for puzzle-solving, and saving the completed puzzle to a file. They form the backbone of the Sudoku-solving process.

7.1 Grid Loading

The load-grid function plays a crucial role in the Sudoku solver. Our primary responsibility is to read the initial Sudoku grid from a file specified by filename and convert it into a data structure that we can work with. We open the file, iterate through its content, and construct a 2D array (grid) that represents the Sudoku puzzle.

We correctly interpret characters in the file, treating digits as values and dots as empty cells. If the file cannot be opened or is not properly formatted, we return an error code.

7.2 Empty Cell Locator

The find-empty-cell function is a utility function that helps us locate the first empty cell in the Sudoku grid. We do this by scanning each cell in a row-major fashion until we find an empty cell (a cell with a value of '0'). Once an empty cell is found, we record its row and column indices.

If there are no more empty cells, we return an error code. This function is used by the backtracking algorithm to determine the next cell to fill.

7.3 Valid Number Placement

The is-umber-placement-valid function is an essential part of our logic. It checks the validity of placing a specific number in a given row and col within the Sudoku grid. It enforces the rules of Sudoku, ensuring that the same number does not appear in the same row, column, or the 3x3 subgrid.

By examining the row, column, and square containing the current cell, this function helps us make informed decisions about number placement.

7.4 Puzzle Solver

The solve-grid function is the heart of the Sudoku solver. We employ a recursive backtracking algorithm to explore possible solutions for the puzzle. We start by calling find-empty-cell to locate the first empty cell. If an empty cell is found, we attempt to place numbers from 1 to 9 in that cell, one at a time, and recursively call ourselves to continue solving the puzzle.

If a valid solution is discovered, the function returns. If not, we backtrack by resetting the cell to empty and exploring other possibilities. This process continues until a solution is found or all possibilities are exhausted.

7.5 Grid Saving

The save-grid function is responsible for saving the solved Sudoku grid to a file with a ".result" extension. We construct the result filename, open the file for writing, and write the solved grid to the file while applying proper formatting. The grid is written as characters, with digits representing values, spaces for cell separation, and line breaks to format the grid properly. If the file cannot be opened for writing, the function returns an error code.

8 Interface

For our final defense, we will focus on refining and showcasing the interface of our Sudoku Solver project. The user experience and interaction will take center stage as we aim to deliver a polished and user-friendly product.

9 Global Advancement

Progression Schedule			
Section	**Subsection**	**First**	**Final**
Image Processing	**Filters**	**80%**	**100%**
	Rotation	**70%**	**100%**
	Segmentation	**100%**	**100%**
	Hough Transform	**100%**	**100%**
	Detection Square	**10%**	**100%**
Machine Learning	**Neural Network**	**75%**	**100%**
Sudoku Solver	**Solver**	**100%**	**100%**

9.1 Some words

During our first defense, we encountered challenges in implementing the square detection feature. Despite our best efforts, this aspect remained incomplete. However, we chose to invest our time wisely in perfecting the grid detection mechanism.

This decision proved beneficial as it allowed us to achieve remarkable results in accurately identifying and interpreting Sudoku grids. While we didn't accomplish everything we initially set out to do, our commitment to ensuring the core functionality of grid detection was a strategic choice.

It reflects our dedication to delivering a reliable solution, even if it meant temporarily setting aside certain features. Our first defense might have had its setbacks, but our grid detection success set a solid foundation for future developments.

10 Conclusion

In conclusion, at the midpoint of this project, it is evident that our team is functioning cohesively and efficiently. We have demonstrated a strong ability to work together, with each member contributing their unique strengths and skills to the project. Furthermore, we have consistently met our project deadlines, which is a testament to our commitment and time management.

The results achieved thus far are indeed promising. Our collective efforts have produced tangible progress, and we are on track to meet our project goals and objectives. This positive momentum serves as a testament to our dedication and the effectiveness of our collaborative approach.

As we move forward in the project's second half, we can approach the remaining tasks with confidence, knowing that our teamwork, commitment, and adherence to timelines will continue to drive our success. With the current trajectory, we are well-positioned to achieve the desired outcomes and deliver a successful project.