

# Projet OCR

## Rapport final

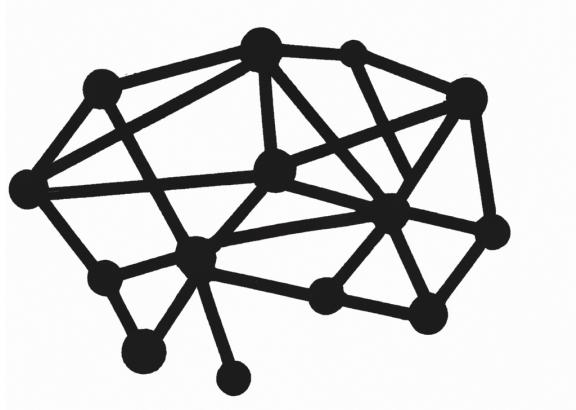
Laure MONTREDON

Irène LIN

Ambroise DURST

Florian FOGLIANI

Décembre 2023



## Sommaire

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Présentation de l'équipe</b>	<b>5</b>
<b>3</b>	<b>Premier cycle de développement</b>	<b>6</b>
3.1	Le pré-traitement . . . . .	6
3.1.1	Rotation de l'image . . . . .	6
3.1.2	Élimination des bruits parasites . . . . .	7
3.2	Détection des lignes et découpage de l'image . . . . .	8
3.2.1	Détection des lignes . . . . .	8
3.2.2	Découpage des cases . . . . .	10
3.3	Réseaux de neurones . . . . .	11
3.3.1	Principe générale . . . . .	11
3.3.2	Notions mathématiques . . . . .	11
3.3.3	Algorithme . . . . .	12
3.4	Solver . . . . .	13
3.4.1	Liste chaînée . . . . .	13
3.4.2	Backtraking . . . . .	13
3.4.3	Formatage . . . . .	15
<b>4</b>	<b>Etat final du projet</b>	<b>15</b>
4.1	Pré-traitement . . . . .	15
4.1.1	Rotation de l'image . . . . .	15
4.1.2	Élimination des bruits parasites . . . . .	16
4.1.3	Renforcement des contrastes . . . . .	19
4.2	Détection des lignes . . . . .	20
4.3	Découpage de l'image	

4.4	Réseaux de neurones . . . . .	24
4.5	Reconstitution de la grille . . . . .	26
4.6	Interface utilisateur . . . . .	28
4.6.1	Glade . . . . .	28
4.6.2	Structure . . . . .	32
4.7	Site internet . . . . .	32
<b>5</b>	<b>Retard(s), imperfections ou non réalisés</b>	<b>34</b>
5.1	Le pré-traitement . . . . .	34
5.2	Rotation automatique de l'image . . . . .	34
5.3	Reconnaissance tapuscrite . . . . .	36
5.4	Interface utilisateur . . . . .	36
<b>6</b>	<b>Conclusion</b>	<b>38</b>
6.1	Commentaires individuels	
	38	
6.1.1	<u>Laure</u> . . . . .	38
6.1.2	<u>Florian</u> . . . . .	38
6.1.3	<u>Irène</u> . . . . .	38
6.1.4	<u>Ambroise</u> . . . . .	39

## 1 Introduction

L'objectif de ce projet est de réaliser un logiciel de type OCR (Optical Character Recognition) capable de résoudre une grille de sudoku. L'application sera en mesure de prendre en entrée une image représentant une grille de sudoku, de la traiter et ensuite d'afficher en sortie la grille résolue.

Pour la première soutenance les objectifs étaient :

- le chargement d'une image et la suppression des couleurs ;
- la rotation manuelle de l'image ;
- la détection de la grille et de la position des cases ;
- le découpage de l'image (sauvegarde de chaque case sous la forme d'une image) ;
- l'implémentation de l'algorithme de résolution d'un sudoku.
- de fournir une preuve de concept de notre réseau de neurones. Pour cette preuve, il est demandé de réaliser un mini réseau capable d'apprendre la fonction OU EXCLUSIF.

A présent pour ce rendu final les objectifs sont:

- Le prétraitement complet ;
- Le réseau de neurones complet et fonctionnel
  - Apprentissage ;
  - Reconnaissance des chiffres de la grille.
- La reconstruction de la grille ;
- La résolution de la grille ;
- L'affichage de la grille ;

## 2 Présentation de l'équipe

OCRnest est une équipe composée de 4 membres :

### Irène

Dans le projet je me charge de tout ce qui touche au pré-traitement, c'est-à-dire, la rotation, la suppression des bruits parasites, des couleurs, et le renforcement de contrastes.

### Ambroise : Chef de projet

C'est avec plaisir que je m'occupe de la partie "IA" du projet. J'ai dû me renseigner sur les méthodes de base du machine learning, à savoir les réseaux de neurones et l'algorithme de descente de gradient. Le sujet est encore plus intéressant que ce que je pensais, bien qu'assez rude en mathématiques (c'est la première fois qu'elle me servent autant) En tout cas j'espere avoir mener à bien, depuis ma partie, l'ensemble du projet :)

### Laure

L'ors de se projet, je me suis occupée du solver. J'ai étudié de nombreux types d'algorithmes différents pour trouver quelle proposition était la plus optimisée. J'ai également pris en charge la reconstitution de la grille après résolution, ainsi que l'interface utilisateur

### Florian

Pour ce projet, mon objectif était la détection des lignes et le découpage des grilles du Sudoku. C'est la première fois que j'implémente des notions de géométrie aussi complexes mais suis heureux de découvrir le fonctionnement du traitement d'images. J'ai été heureux de participer à un projet si

intéressant, complet, et plein d'apprentissages. Je me suis occupé aussi par la suite de notre site internet pour mettre en valeur notre travail.

## 3 Premier cycle de développement

### 3.1 Le pré-traitement

L'objectif de cette partie est d'améliorer la qualité des images traitées, afin de les préparer de manière optimale pour le processus de résolution à venir. Ce module de prétraitement s'engage dans diverses tâches :

- Redressement manuel de l'image (rotation de l'image à partir d'un angle saisi par l'utilisateur)
- Élimination des bruits parasites (grain de l'image, tâches, etc.)
- Renforcement des contrastes.

#### 3.1.1 Rotation de l'image

Il existe diverses méthodes de rotations, mais pour notre projet, cette opération est réalisée par le biais de SDL (Simple DirectMedia Layer), ce qui signifie que l'image d'origine n'est pas altérée directement. Au lieu de cela, SDL génère une nouvelle image orientée en fonction de l'angle de rotation spécifié, sans modifier l'image source. Cette nouvelle image est ensuite sauvegardée sous le nom de "img\_oriented.png" pour une utilisation future. Un angle de rotation de 0 degrés signifie que l'image reste inchangé. Dans ce cas l'image est simplement rendue à l'écran. En revanche, lorsque l'angle est différent de 0, une rotation de l'image est effectuée selon l'angle spécifié.

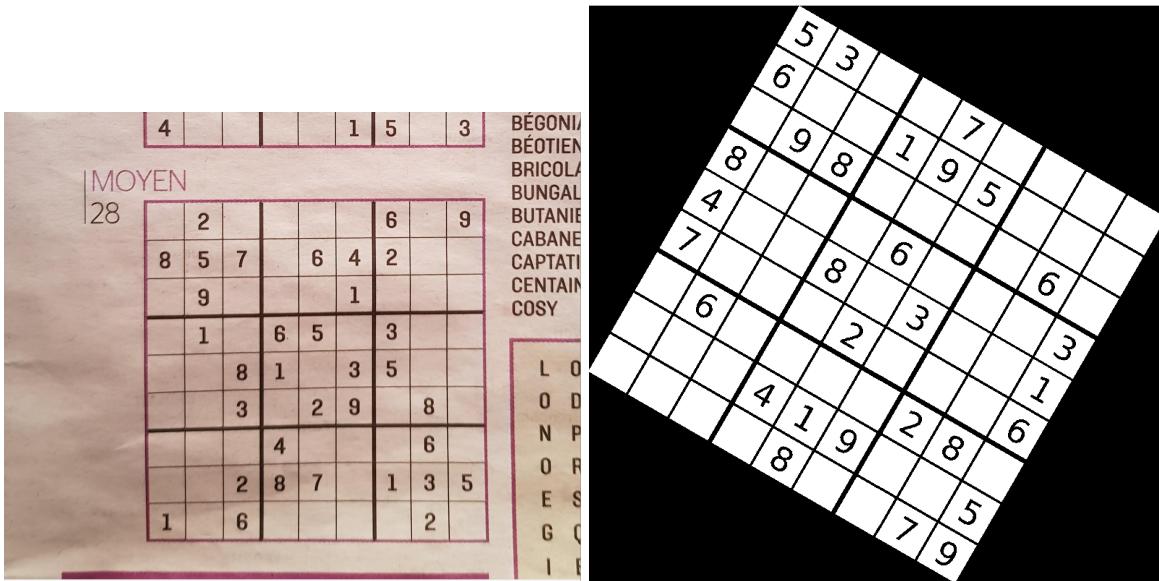


Figure 1 :Avant / après rotation de la grille de sudoku

### 3.1.2 Élimination des bruits parasites

Pour effectuer la conversion en noir et blanc, on parcours chaque pixel de l'image colorée. L'objectif est de remplacer les couleurs en calculant la moyenne des valeurs RGB pour chaque composante de pixel. En d'autres termes, pour chaque pixel P avec des composantes RGB(R, G, B), nous calculons les nouvelles composantes R', G', B', comme suit:

$$R' = G' = B' = \frac{1}{3}R + \frac{1}{3}G + \frac{1}{3}B$$

Figure 1: formule pour obtenir la moyenne des valeurs RGB

Cependant, pour obtenir une image en noir et blanc avec un seuil, nous devons ensuite comparer la valeur de luminance moyenne à ce seuil. Ici, le seuil est de 130, ce qui signifie que si la moyenne de composantes R', G', B' est supérieur à 130, le pixel devient blanc, sinon, il devient noir.

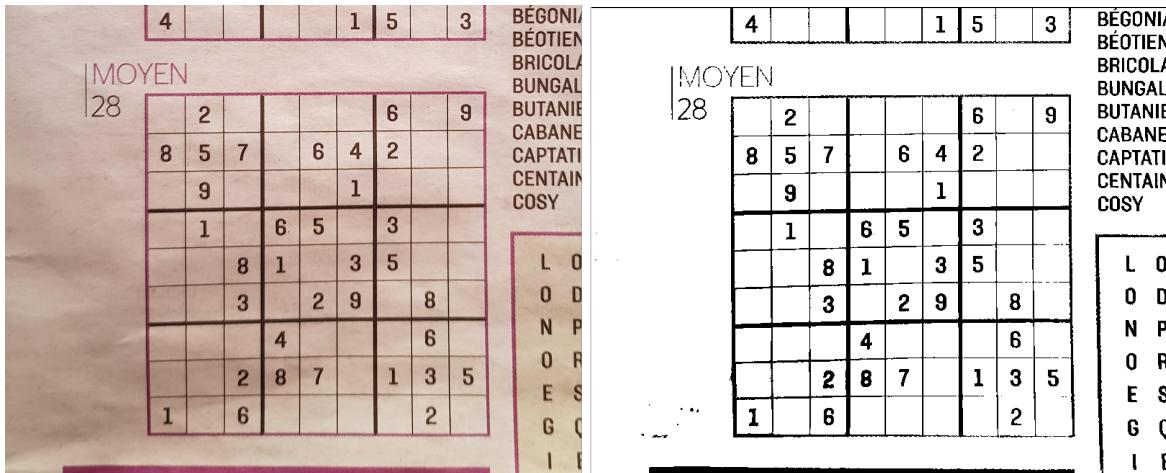


Figure 2: Avant/après conversion en noir et blanc

## 3.2 Détection des lignes et découpage de l'image

### 3.2.1 Détection des lignes

Pour la détection des lignes nous avons décidé d'utiliser la transformée de Hough. La transformée de Hough fonctionne sur le principe suivant : pour chaque pixel de l'image, l'on va étudier l'ensemble des droites passant par celui-ci et dans une matrice de correspondance on incrémentera la case de la droite associé en coordonnée polaire. Une droite en coordonnée polaire est représenté par le couple ( $\rho$ ,  $\theta$ ) tel que  $\rho$  est la distance de la droite à l'origine du repère et  $\theta$ , l'angle que fait la perpendiculaire à la droite avec l'axe x.

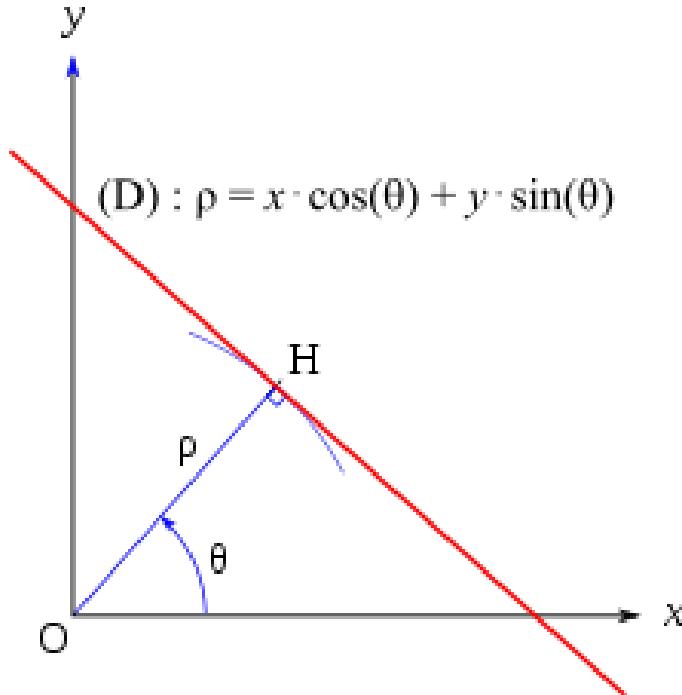


Figure 3 : Cordonnées polaires d'une droite

La matrice de correspondance représente en ligne l'intervalle de rho allant de  $[-p, +p]$  résultat d'une formule en fonction de l'angle théta et de la position du pixel et en colonne l'angle théta allant de  $[-90, +90]$ . Ainsi l'on va pour chaque pixel, étudie les droites d'angles  $-90$  jusqu'à  $+90$  tout en calculant leurs rho associé. Ce couple  $[rho, theta]$  représentera une case dans la matrice de correspondance.

Ensuite, nous détectons les maximums dans la matrice de correspondance. Ces maximums représentent les droites qui passent par le plus de points dans l'image. Dans le cas du sudoku cela représente la grille avec ses lignes horizontales et verticales.

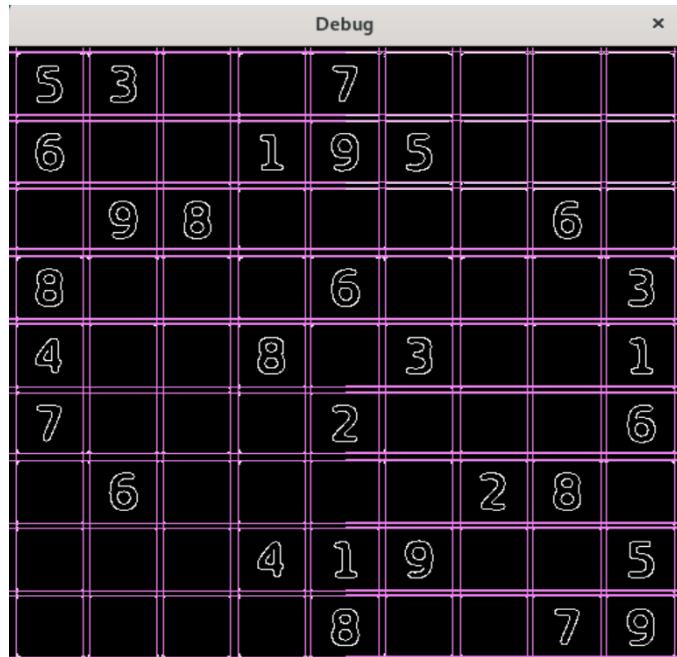


Figure 4 : Résultat de la détection des droites

### 3.2.2 Découpage des cases

Pour le découpage nous avons décidé de récupérer du transformée de Hough, l'ensemble des lignes horizontales d'un côté et l'ensemble des lignes verticales de l'autre. Ensuite, l'on cherche les 10 lignes verticales et les 10 lignes horizontales correspondantes à la grille de Sudoku, à distances égales deux à deux. Pour cela, l'on va étudier leurs distances par rapport à l'origine (leurs coordonnées rho). En premier temps, nous allons triés les lignes horizontales et verticales par ordre croissantes. Ensuite, nous déterminons les 10 lignes horizontales et 10 lignes verticales à distance égales deux à deux. Ainsi, nous obtenons la grille de Sudoku uniquement. Enfin, nous allons étudier les points d'intersections des droites entre elles pour obtenir l'ensemble des côtés de chaque case. Grâce à SDL, nous enregistrerons chaque case sous la forme d'une image grâce à leurs bords supérieurs droits et leurs bords inférieurs gauche. Le format de fichier enregistrés sont au format mat\_ligne\_colonne.

### 3.3 Réseaux de neurones

#### 3.3.1 Principe générale

Plusieurs notions ont dû être acquises pour implémenter correctement le traitement qui servira à la reconnaissance des nombres. Des méthodes mêlant algorithmie et mathématiques sont entrées en jeu, en voici donc un résumé structuré:

Nous utilisons du machine learning, cette branche de l'IA utilise une structure de donnée clef autour de laquelle toute la phase de traitement va tourner, à savoir un réseau de neurones. Ce réseau de neurones, initié par un programme, est premièrement constitué de valeurs choisies au hasard. Puis en suit une phase d'apprentissage, durant laquelle grâce à différents algorithmes, les valeurs du réseau sont modifiées, jusqu'à obtenir un réseau opérationnel. Pour finir, il ne reste plus qu'à tester le réseau pour s'assurer de son bon fonctionnement.

Ainsi, la création de notre réseau de neurones peut se résumer en 3 phases: Initialisation, Apprentissage, Vérification.

A présent, en gardant en tête ces trois phases, nous pouvons davantage entrer dans les détails, en effet pour implémenter ce système, des notions mathématiques, et des algorithmes, ont dû être intégrés.

#### 3.3.2 Notions mathématiques

Des matrices sont utilisées pour représenter le réseau de neurones, ces matrices facilitent et accélèrent les calculs, permettant de simplifier et d'optimiser le programme.

Une loi normale de distribution (ou loi gaussienne), est utilisée pour initialiser aux hasards, mais avec une certaine cohérence, les valeurs du réseau de neurones. Le calcul de dérivées partielles, nécessaire aux fonctionnement des algorithmes, a été implémenté.

### 3.3.3 Algorithme

A l'état actuel du projet, un seul algorithme est utilisé pour la phase d'apprentissage: la descente de gradient. Le principe général de la descente de gradient, consiste en le calcul d'un gradient, qui correspond à la liste des petites variations que chaque valeur du réseau doit subir pour améliorer la performance générale. Ainsi, on calcule ce gradient puis on applique ces petites variations à chaque valeurs, et répète l'opération autant de fois que nécessaire pour obtenir un réseau fonctionnel.

Enfin, plusieurs paramètres sont manipulables pour adapter l'apprentissage du réseau:

- Le nombre de neurones dans le hidden layer
- La vitesse d'apprentissage (rate)
- La nombre de cycle d'entraînements (epochs)

Pour cette première soutenance, le réseau est entraîné pour reproduire le comportement d'une porte XOR.

Cette tâche simple pour le réseau nécessite que 3 neurones dans le hidden layer, une vitesse d'apprentissage de 0.01 à 5, et 100 cycles d'entraînements

```
EPOCH 90 : 0.428583
EPOCH 91 : 0.000001
EPOCH 92 : 0.001135
EPOCH 93 : 0.000000
EPOCH 94 : 0.000072
EPOCH 95 : 0.000000
EPOCH 96 : 0.000000
EPOCH 97 : 0.000000
EPOCH 98 : 0.980223
EPOCH 99 : 0.000000
[0.000000, 0.000000] ==> 0.000000
[0.000000, 1.000000] ==> 0.999659
[1.000000, 0.000000] ==> 0.999827
[1.000000, 1.000000] ==> 0.000000
```

Figure 5 : Apprentissage du Xor

## 3.4 Solver

Le solver résout le sudoku, en alliant liste chaînée et backtraking la plupart des résolutions prennent environs 0,0005s mais certaines grilles peuvent prendre jusqu'à 0.002s

### 3.4.1 Liste chaînée

Afin d'optimiser le programme de résolution par backtracking, nous avons implémenté notre propre système de liste chaînée. Cette liste contient le nombre de valeurs que peut prendre chacune des cases vides du sudoku et est triée par ordre croissant. De cette manière, les cases contenant le moins de possibilités sont traitées en priorité.

Pour cela nous avons crée une structure contenant la position de la case, le nombre de valeurs qu'elle contient et un pointeur vers l'élément de la liste chaînée.

Cette implémentation de la liste chaînée posséde trois fonctions: new\_elt,insert et free\_l.

-New\_elt crée un nouvel élément indépendant de la liste chaînée principale. Pour cela elle prend en paramètres : la position et le nombre de valeurs que peut prendre la case.

-Insert insert un nouvel élément dans une liste chaînée. Elle le placera par ordre croissant selon le nombre de valeurs possibles.

-Free\_l free la mémoire d'une liste chaînée passée en paramètre.

### 3.4.2 Backtraking

Le programme va remplir progressivement la grille en vérifiant que celle-ci est toujours potentiellement valide. Si le fait de placer un nombre rend la grille fausse alors on change le nombre, s'il n'y a aucune solution possible : on change le nombre de l'étape précédente, sinon : on fait un appel recursif

de la fonction sur la nouvelle grille.

Afin d'augmenter la rapidité du traitement on remplit les cases selon le nombre de possibilités qu'elle possède. Pour ce faire on crée une liste chaînée contenant le nombre de possibilités pour toutes les cases vides. En effectuant un backtraking depuis les cases avec un minimum de solutions vers les cases avec un maximum de solutions on minimise significativement la complexité temporelle.

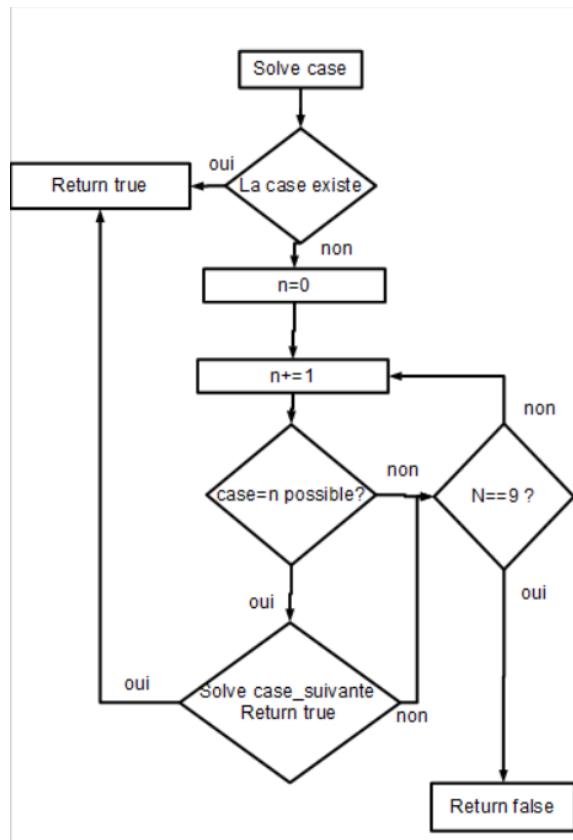


Figure 6: Algorithme du solver

### 3.4.3 Formatage

Le fichier passer en entrée est un fichier texte de type :



Figure 7: format des fichiers texte

Ce fichier est lu et est entré dans un tableau d'entiers de taille 81. Les points sont transformés en zéros pour la résolution du solveur. Ensuite, le programme enregistre le résultat dans un nouveau fichier texte avec le même format que le texte d'entrée.

## 4 Etat final du projet

### 4.1 Pré-traitement

#### 4.1.1 Rotation de l'image

Il existe plusieurs méthodes de rotations, mais pour notre projet, cette opération est réalisée par le biais de SDL (Simple DirectMedia Layer), ce qui signifie que l'image d'origine n'est pas altérée directement. Au lieu de cela, SDL génère une nouvelle image orientée en fonction de l'angle de rotation spécifié, sans

modifier l'image source. Cette nouvelle image est ensuite sauvegardée sous le nom de "img\_oriented.png" pour une utilisation future. Un angle de rotation de 0 degrés signifie que l'image reste inchangée, tandis qu'un angle différent de 0 entraîne une rotation de l'image.

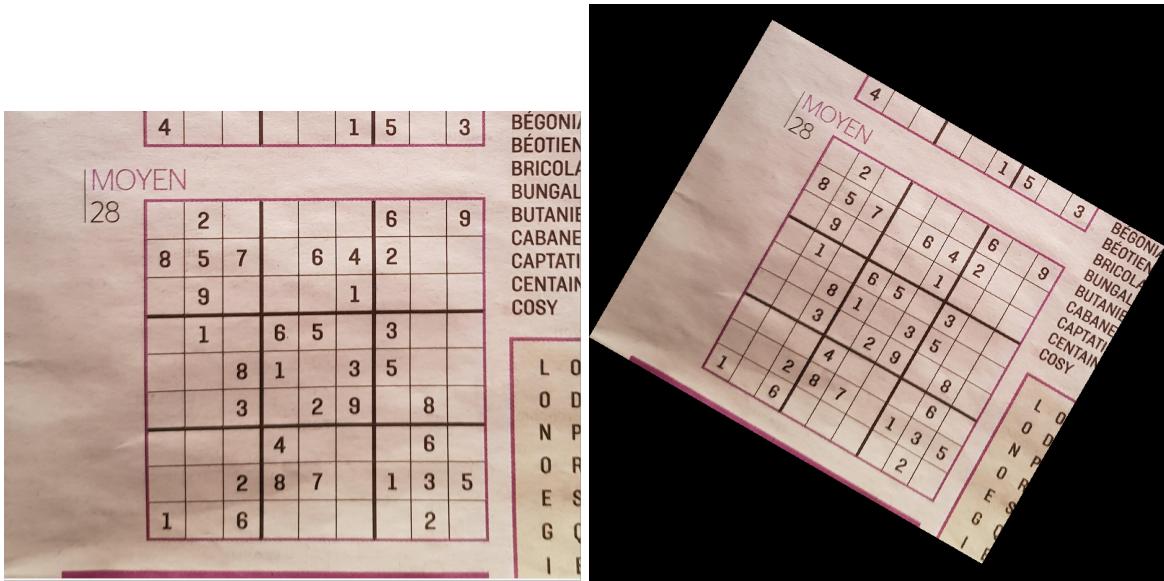


Figure 8 :Avant / après rotation de la grille de sudoku

#### 4.1.2 Élimination des bruits parasites

L'élimination des bruits parasites se fait en deux parties :

1. Conversion en niveaux de gris: Chaque pixel de l'image colorée est parcouru. L'objectif est de remplacer les couleurs en calculant la moyenne des valeurs RGB pour chaque composante du pixel. En d'autres termes, pour chaque pixel P avec des composantes RGB (R, G, B), nous calculons les nouvelles composantes R', G', B' comme suit :

$$R' = G' = B' = \frac{1}{3}R + \frac{1}{3}G + \frac{1}{3}B$$

Figure 1: formule pour obtenir la moyenne des valeurs RGB

2. Le seuillage : La binarisation consiste à transformer une image de niveaux de gris une image noire et blanche.

Dans notre projet, nous avons choisi d'utiliser la méthode globale : Ce processus commence par calculer la moyenne des valeurs global du sudoku en analysant l'histogramme de l'image, ce dernier permet de visualiser la fréquence d'apparition des différentes intensités de gris, permettant ainsi de déterminer un seuil approprié.

Considérons différents scénarios d'images et comment notre approche de seuillage pourrait s'adapter à chaque situation :

- Image floue : L'histogramme présente une distribution relativement uniforme des niveaux de gris, indiquant une image où les contrastes sont moins prononcés. Le seuil moyen pourrait être sélectionné pour distinguer les zones claires et sombres de manière équilibrée.
- Image nette : L'histogramme présente une distribution bien définie avec des pics clairs et sombres, caractéristique d'une image nette. Le choix d'un seuil entre les pics peut efficacement séparer les éléments clairs des éléments sombres.
- Image avec de l'ombre : l'histogramme est biaisé vers les valeurs plus sombres, indiquant la présence d'ombres. Un seuil plus bas pourrait être préférable pour prendre en compte l'ombrage et éviter la perte de détails dans les zones sombres.
- Image Lumineuse : L'histogramme est ici biaisé vers les valeurs plus claires, suggérant une image lumineuse. Un seuil plus élevé peut être choisi pour bien séparer les zones claires et sombres.



Figure 8bis : Comparaison des Histogrammes : variations selon les caractéristiques des images

Chaque pixels est ensuite évalué individuellement ; s'il est supérieur à la moyenne (seuil calculé), il est convertie en noir, sinon en blanc. Cette approche de seuillage crée une distinction nette entre les éléments clairs et sombres de l'image, facilitant ainsi les étapes ultérieures de traitement. Cette image finalement convertie en noir et blanc sera enregistré à part sous le nom de "img\_toCut.png".

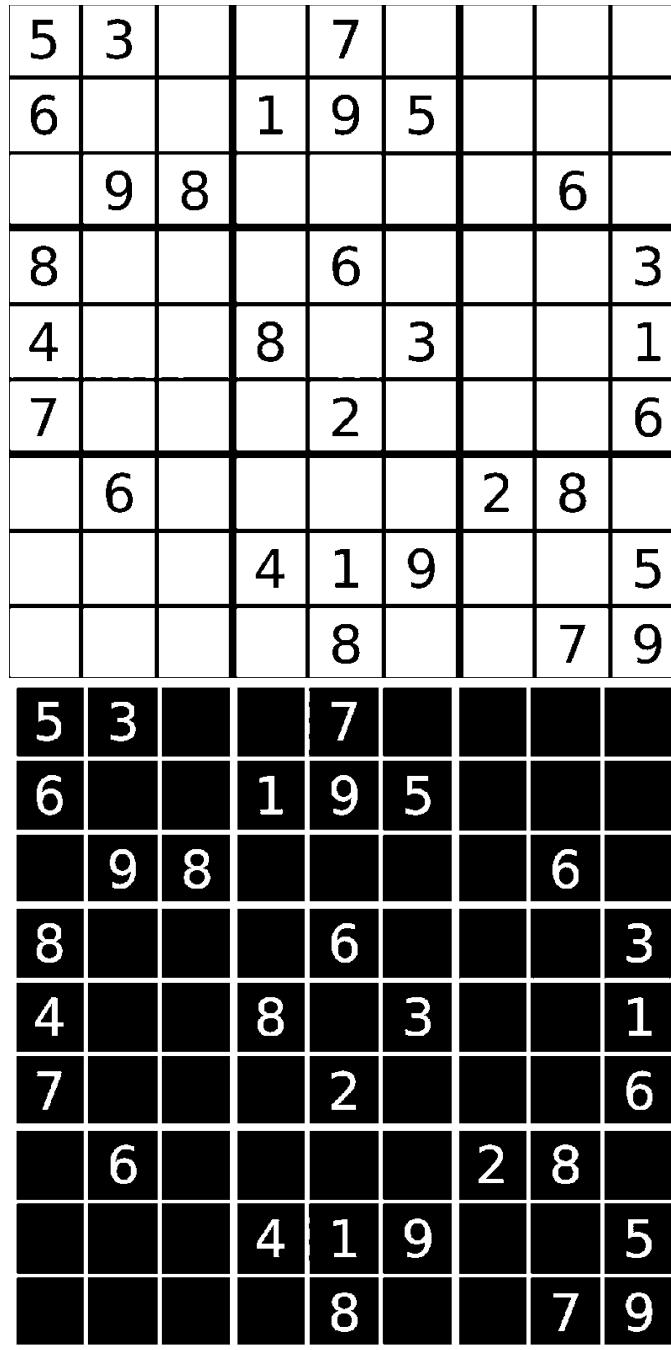


Figure 9: Avant / après conversion en noir et blanc

#### 4.1.3 Renforcement des contrastes

Le renforcement des contrastes s'inscrit dans une logique spécifique visant à accentuer les frontières entre les objets présents dans une image. Chaque pixel est examiné, si au moins l'un de ses huit voisins est noir, cela suggère

la présence d'un contour, potentiellement celui d'un chiffre ou de la grille, et le pixel est laissé en blanc. En revanche si aucun voisin n'est noir, cela indique probablement que le pixel en question ne se trouve pas sur le bord d'un chiffre ou de la grille, et le pixel est alors rendu noir. Le résultat final de cette opération sera ensuite sauvegardé sous le nom de "img\_analyse.png" pour le découpage de l'image et la reconnaissance des caractères.

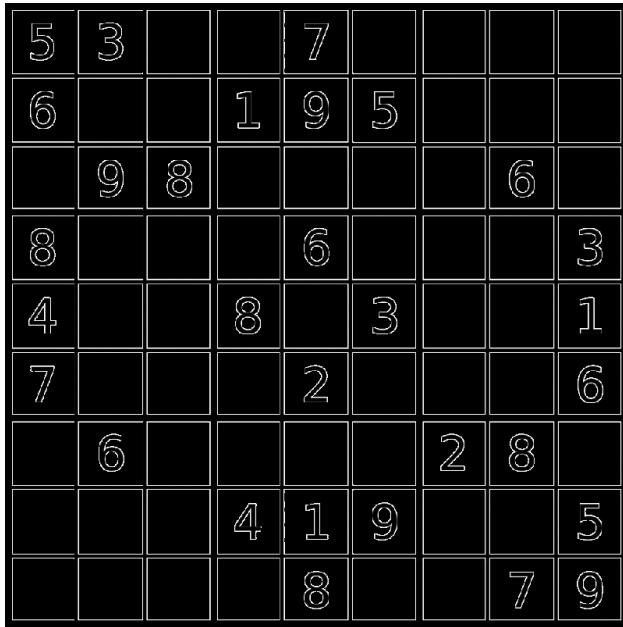


Figure 10 : Renforcement des contrastes

## 4.2 Détection des lignes

A l'issue de la première soutenance, nous avons remarqués que la détection des lignes pouvait être améliorée.

Pour cela, nous avons décidé dans premier temps d'ajuster le seuil de maximum. En effet, lors du transformée de Hough le maximum est enregistré dans une variable. Ce maximum nous permet de détecter toutes les lignes passant par le plus de points dans notre image. Le précédent seuil était de 0.2, c'est à dire que l'ensemble des lignes dont la valeur dépassaient  $\text{max} * 0.2$  étaient enregistrées en tant que lignes possiblement valides pour le découpage. Mal-

heureusement, de nombreuses lignes dépassaient ce maximum. Nous avons donc décidés d'ajuster ce seuil à 0.65. En effet, après de nombreux tests ce seuil semblait être le plus efficace. Il était en même temps assez restrictif pour pouvoir sélectionner que les lignes nécessaires au découpage et assez flexible pour lui permettre de sélectionner les 10 lignes horizontales et les 10 lignes verticales essentielles au découpage. Si le seuil dépassait la valeur de 0.65, il pouvait arriver que sur certaines images avec de nombreux défauts, une erreur de segmentation fault pouvait se produire car moins de 10 lignes pouvaient être détectées. Cette amélioration nous a permis de réduire la complexité temporelle et donc de rendre le traitement plus rapide.

Ensuite, nous avions remarqués que dans le cas du tri des lignes horizontales, certaines lignes diagonales pouvaient se glisser dans la sélection de ces lignes. Nous avons donc décidés d'effectuer des vérifications supplémentaires de l'angle Theta en le bornant entre  $80^\circ$  et  $100^\circ$  pour permettre de récupérer seulement les lignes dont nous sommes sûrs qu'elles sont horizontales.

Enfin, nous avions aussi remarqués que sur certaines images la sélection des 10 lignes horizontales et des 10 lignes verticales, toutes égales deux à deux, pouvait posée problème car le seuil de tolérance de distance entre deux lignes était parfois trop petit. Pour résoudre ce problème, nous avons tout simplement ajustés le seuil de tolérance. Malheuresement, un autre problème est apparu sur les images à nombreux défauts. Ce problème était que la sélection des lignes pouvait parfois par erreur croire qu'un défaut était une ligne de la grille. En conséquence, nous avons dû augmenter le nombre d'itérations sur les lignes horizontales et les lignes verticales. Nous pensions donc que la complexité allait fortement augmenter, mais ça ne fut pas le cas car la sélection des 10 lignes horizontales et des 10 lignes verticales s'effectuait toujours au bout de quelques itérations et ne nécessitait pas une nombre trop élevé d'itérations successives.

En conclusion, l'avancement final fut surtout des ajustements pour extraire de façon la plus efficace et la plus net possible les différentes lignes composants la grille du Sudoku.

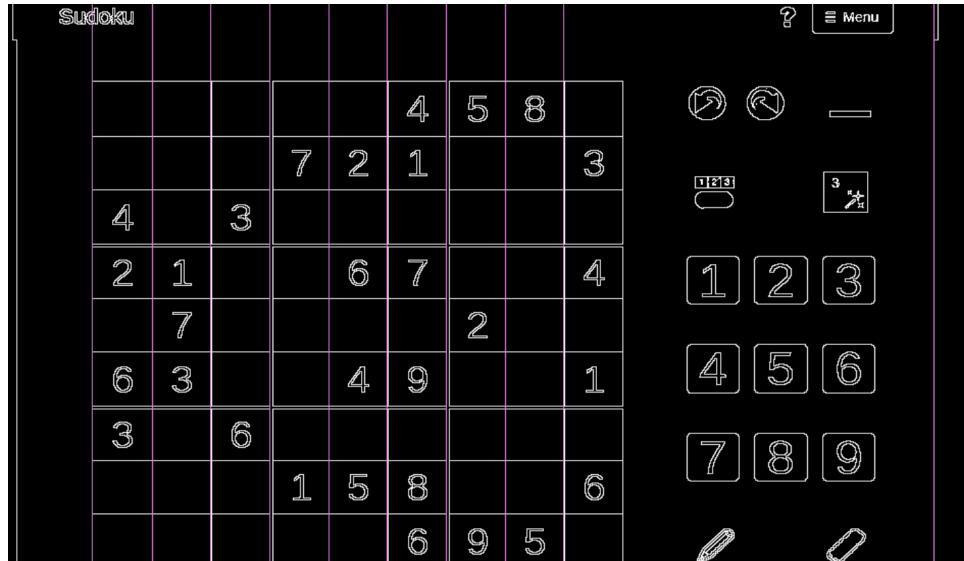


Figure 11 : avant ajustement de seuil

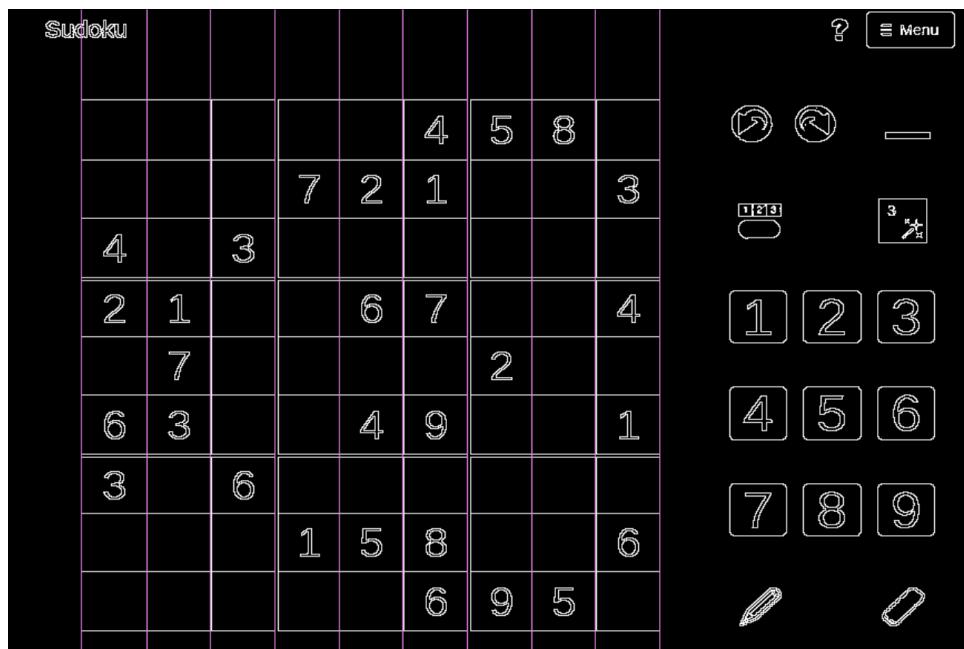


Figure 12 : après ajustement de seuil

### 4.3 Découpage de l'image

A la suite de la première soutenance, nous avons dû effectuer quelques modifications concernant notre découpage de l'image et l'enregistrement de nos cases.

La problématique fut l'adaptation des images résultantes du découpage des cases pour la détection des chiffres par le réseau neuronal. En effet, le réseau neuronal est entraîné pour des images au format 28x28 pixels. Il a donc fallu adapter nos cases découpés pour un seul format unique. Pour cela, nous avons d'abord enregistré l'image total de la grille au format 252x252 pixels. Enfin, nous avons découpés cette image en plusieurs cases distinctes de format 28x28 pixels. Ainsi, nous avons donc obtenu des images au bon format pour la détection des chiffres par le réseau neuronal.

En conclusion, les améliorations apportées au découpage de l'image étaient surtout nécessaires pour interconnecter les différentes parties du projet entre elles, notamment avec le réseau neuronal de détection des chiffres.

#### 4.4 Réseaux de neurones

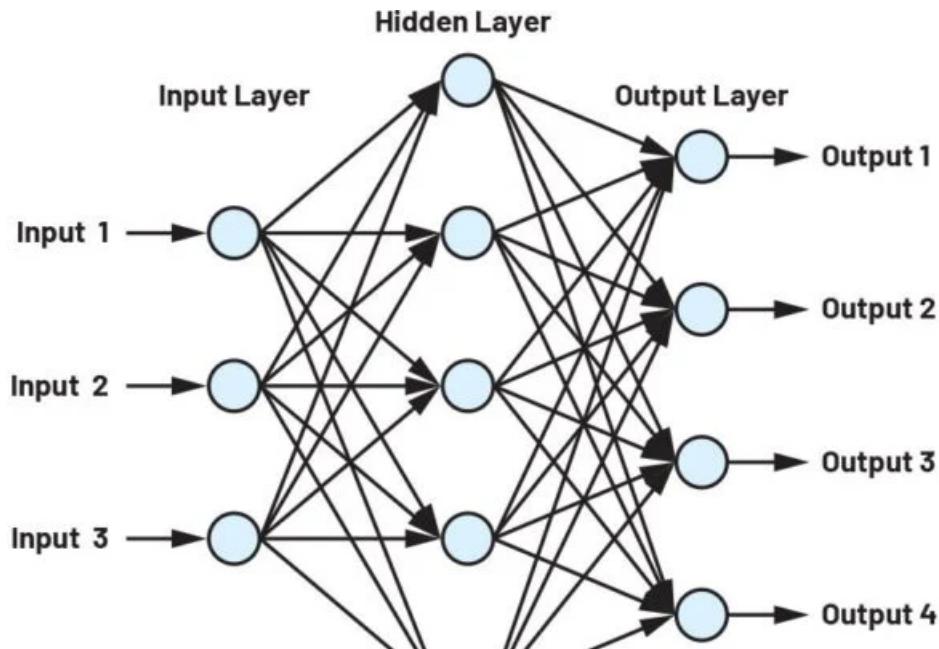


Figure 13 : réseau neuronal

Pour la reconnaissance des chiffres à travers un réseau de neurones, une amélioration des techniques d'apprentissage du réseau déjà établi a été nécessaire.

En effet, le premier réseau devant simplement imiter une porte XOR, il suffisait d'appliquer le gradient de la fonction coût - préalablement calculé - avec un taux constant.

Cependant, lorsque ce système a été appliqué pour la reconnaissance des chiffres, deux problèmes majeurs ont été rencontrés : la vitesse de calcul et la précision.

En effet, jusqu'à présent, le gradient de la fonction coût était calculé en appliquant une dérivée partielle sur chaque variable du réseau, et ce, à chaque image du dataset d'entraînement, ce qui prend énormément de temps de calcul et donc une complexité temporelle élevée.

Aussi, il était beaucoup plus compliqué, voire impossible, puisque de toute

façon trop long, d'atteindre un pourcentage de réussite significatif.

Ainsi, deux ajouts ont été faits pour améliorer la phase d'apprentissage du réseau :

Premièrement, un algorithme dit de "backpropagation", basé sur le principe du chaînage de dérivées, a permis de calculer le gradient en environ 100 fois moins de temps.

De plus, plutôt que d'utiliser une descente de gradient classique, une descente de gradient stochastique a été choisie. En effet, en prenant au hasard des sous-ensembles plus petits de données dans le dataset, on augmente la vitesse et on diminue les chances de tomber dans des minimums locaux (paradoxalement grâce à une légère perte de précision).

Enfin, un taux d'apprentissage décroissant à chaque époque a permis de gagner encore plus en précision.

Par la mise en place de ces différentes techniques, nous avons pu entraîné un réseau jusqu'à un taux de réussite de 91% pour la reconnaissance des chiffres manuscrits.

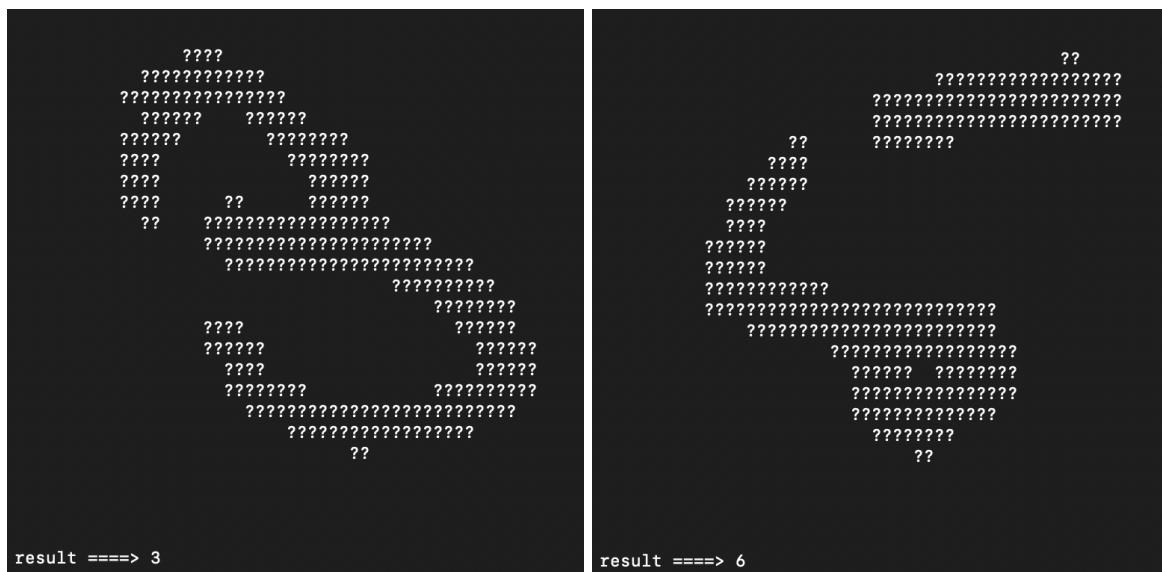




Figure 14 : reconnaissance des chiffres manuscrits

Un autre réseau a également été entraîné pour reconnaître des chiffres tapuscrit, plus adapté aux grilles de sudoku.

## 4.5 Reconstitution de la grille

Pour reconstituer la grille, nous utilisons SDL. La reconstitution se fait en deux étapes : tout d'abord, le tracé de la grille, puis celui des nombres.

Pour effectuer le tracé de la grille, j'utilise la fonction `SDL_RenderDrawLine()` 16 fois, verticalement et horizontalement. Cela me permet de tracer les dix lignes nécessaires pour les deux dimensions et d'accentuer les lignes formant les 9 blocs.

Afin de tracer les nombres, nous récupérons d'abord les résultats renvoyés par le solveur. Pour chaque case, nous exécutons la fonction de dessin du nombre correspondant en fonction de l'angle supérieur gauche, de la hauteur et de la longueur d'une case.

De plus, la grille affichée peut être dimensionnée par l'utilisateur et sera sauvegardée sous cette forme.

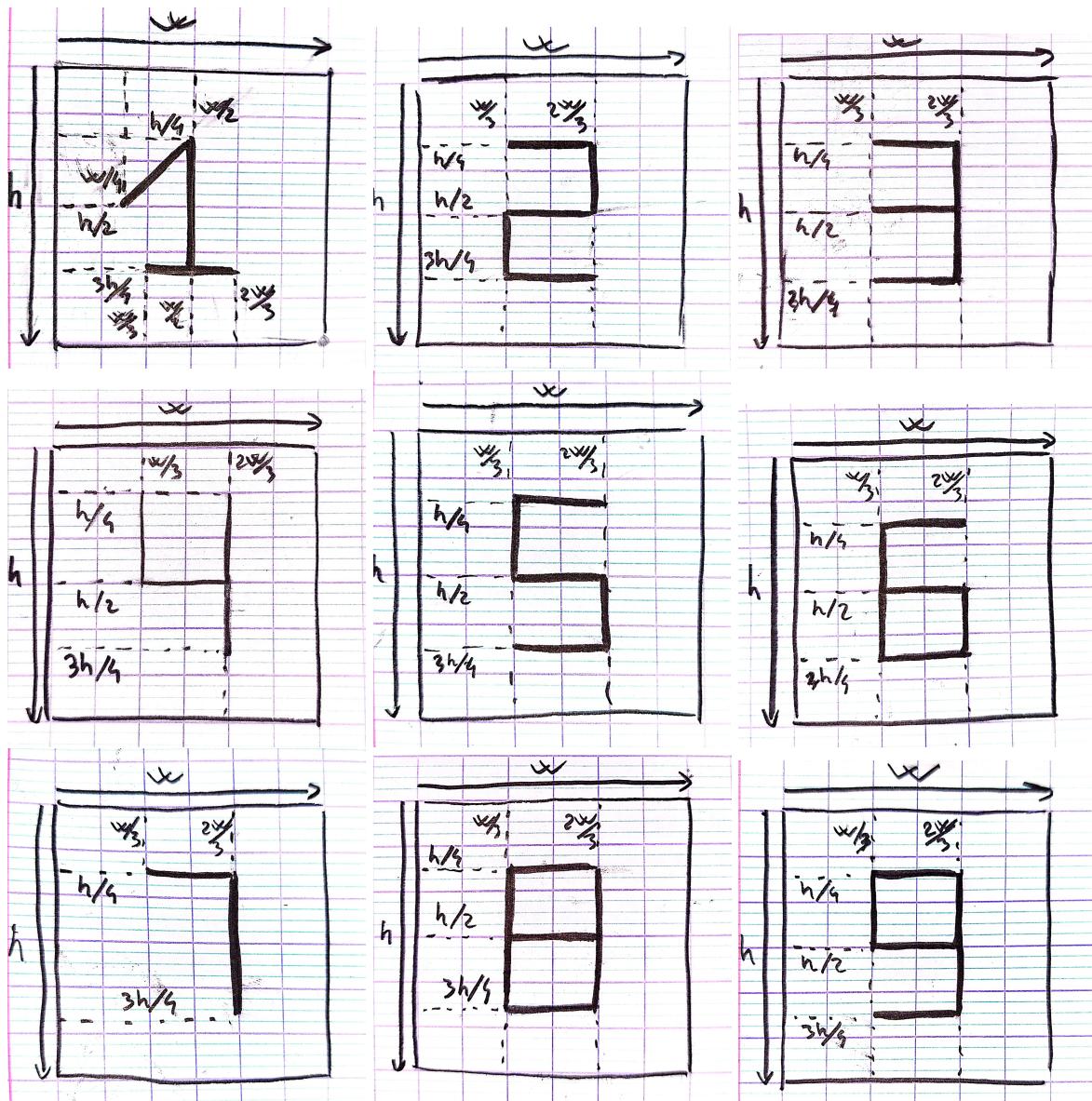


Figure 15 : dessins des chiffres

Enfin, pour afficher les nombres déjà présents dans une couleur différente, nous utilisons un double pointeur, nous indiquant ainsi à la fois le nombre contenu dans la grille et la présence ou l'absence du dit nombre avant la résolution.

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

Figure 16 : reconstitution de la grille

## 4.6 Interface utilisateur

### 4.6.1 Glade

Le fichier Glade est constitué d'un en-tête et d'une pile contenant les différentes pages. Chaque page contient une image, des boutons et du texte. Ces pages permettent de visualiser les différentes étapes du traitement d'un sudoku. Voici les différentes pages que nous avons créée :

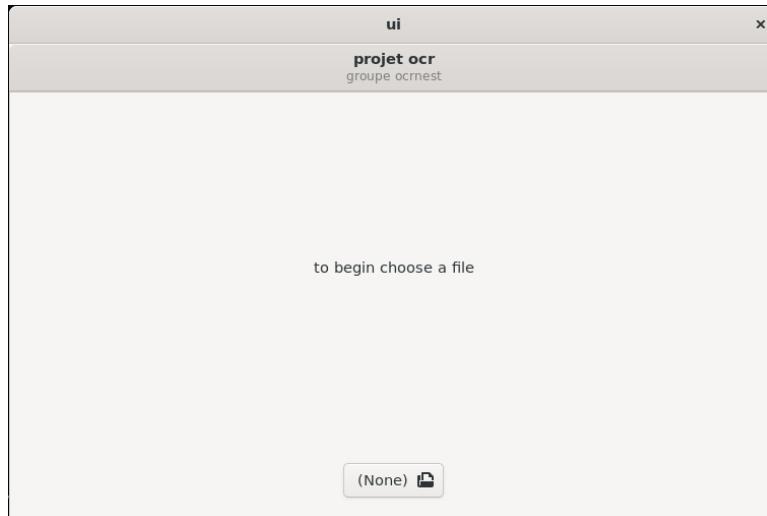


Figure 17 : UI page1

La page 1 contient un sélecteur de fichier et invite à sélectionner un fichier pour commencer la résolution du Sudoku.

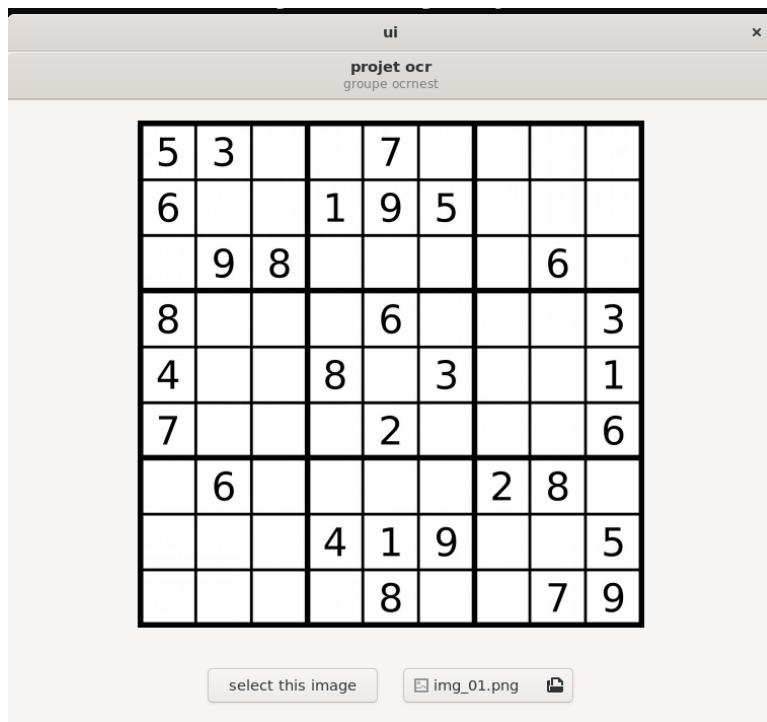


Figure 18 : UI page2

La page 2 contient une image, un sélecteur de fichier et un bouton de confirmation. Cette page permet de confirmer ou de modifier notre sélection

d'image tout en l'affichant.

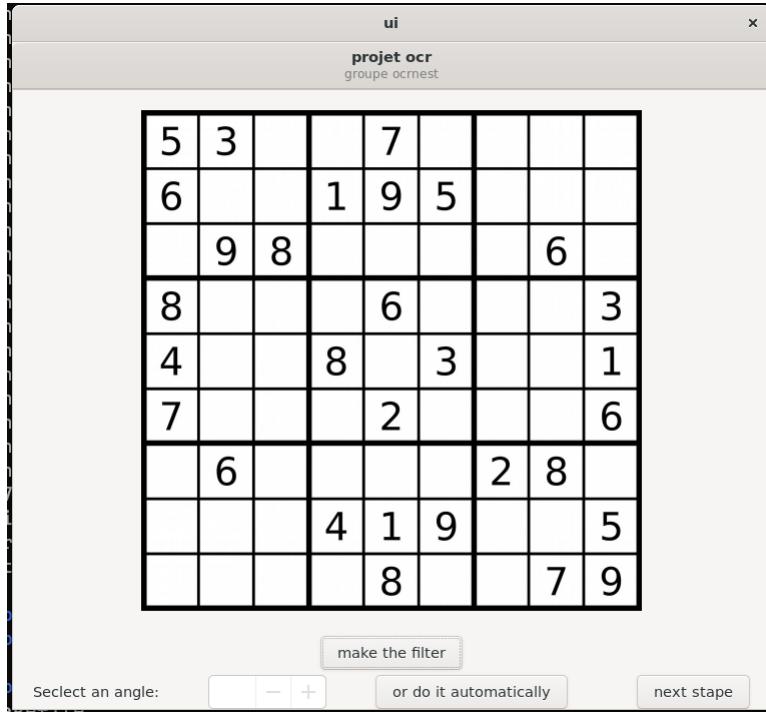


Figure 19 : UI page3

La page 3 contient une image, un SpinButton pour effectuer une rotation manuelle, et trois boutons 'cliquer' : le premier pour appliquer le filtre noir et blanc à l'image, le second pour effectuer une rotation automatique, et le troisième pour passer à la page suivante. Cette page présente donc les différents aspects du prétraitement et permet à l'utilisateur de découvrir l'envers du décor.

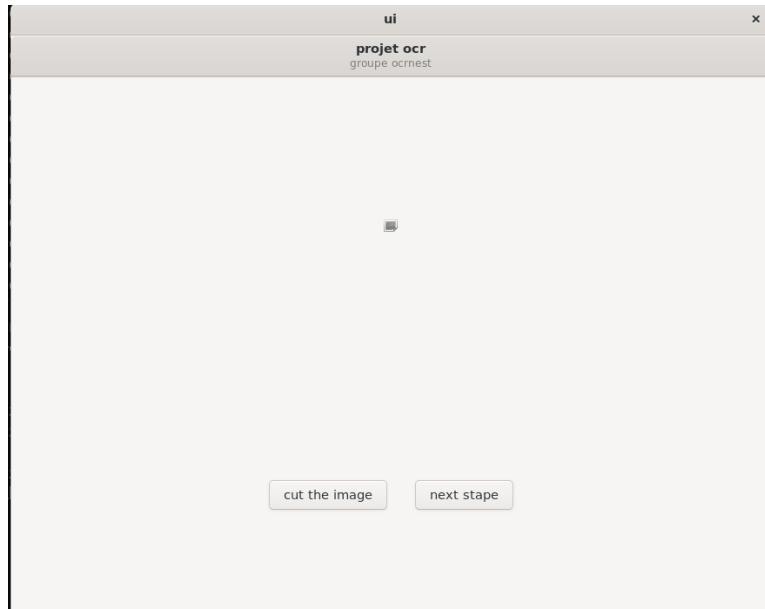


Figure 20 : UI page4

La page 4 contient une image et deux boutons cliquer : un pour effectuer le découpage des cases et un pour passer à la page suivante. Cette page présente les différents aspects du découpage des cases.

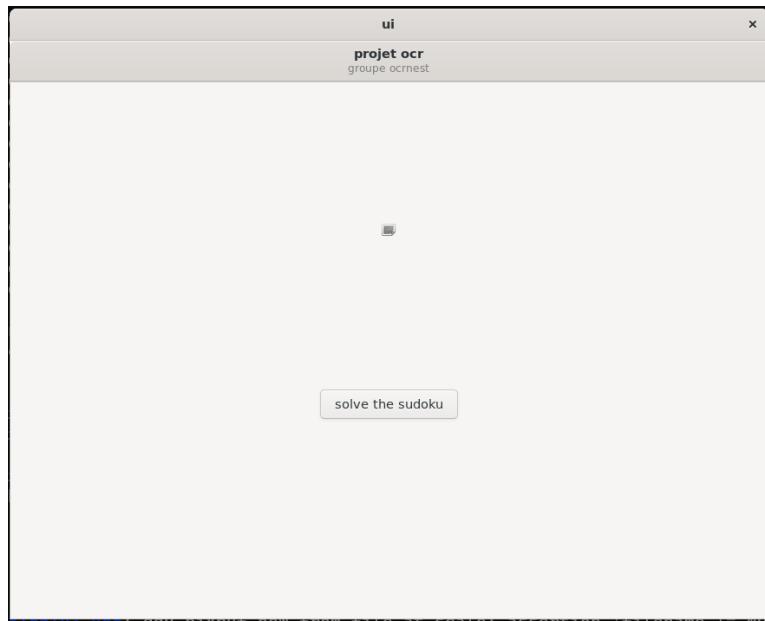


Figure 21 : UI page5

La page 5 contient une image et un bouton cliquer permettant de résoudre

le sudoku. Cette page utilisera le réseau de neurones, le solveur et la reconstitution de la grille pour afficher le résultat final.

#### 4.6.2 Structure

Afin d'implémenter l'interface utilisateur (UI), nous avons créé de nombreuses structures que je vais détailler ici :

- Controls: cette structure contient tous les boutons et contrôles de l'UI.
- Page: cette structure contient un "Viewport" et une image
- Pages: cette structure contient les cinq structures "Page" de l'UI.
- Images : cette structure contient les différents chemins d'accès aux différentes images à afficher. Il y a donc un chemin vers l'image originale, un vers l'image filtrée, un vers l'image réorientée, un vers l'image résultante du découpage des cases et un vers l'image résolue.
- Main Window : cette structure contient une structure 'Images', une 'Stack', une fenêtre 'Window', une structure 'Pages', une structure 'Controls' et un entier indiquant la page actuelle.

#### 4.7 Site internet

Pour mettre en valeur notre projet, nous avons décidé de créer un site internet avec plusieurs rubriques :

- Accueil : une petite présentation du projet et de ses objectifs.
- Fonctionnement : des cartes défilantes présentant chacune les différentes étapes du projet (le prétraitement, le découpage, la détection, la résolution).
- Installation : présentant l'ensemble du processus d'installation avec les dépendances nécessaires.
- Auteurs : avec une photo et le nom/prénom de chaque membre du groupe OCRnest.

Accueil Fonctionnement Installation Auteurs

### Qu'est-ce qu'OCRnest ?

Do esse labore magna consequat tempor velit. Fugiat anim enim sit ullamco. Elit deserunt sunt laborum ad sint eliti dolore ut id sunt aliquip nostrud aliqua ullamco. Sint consequat cupidatat in ullamco et veniam dolore ad id. Sint consequat cupidatat in ullamco et veniam dolore ad id. Sint consequat cupidatat in ullamco et veniam dolore ad id.

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

### Fonctionnement

- Prétraitement
- Découpage
- Déttection
- Résolution

Cillum reprehenderit commodo amet exercitation aliqua est incididunt. Cillum reprehenderit commodo amet exercitation aliqua est incididunt.

### Installation

```
$ echo "deb [trusted=yes] https://apt.dev /" | sudo tee /etc/apt/sources.list.d/ocrnest.list
$ sudo apt-get update
$ sudo apt-get install ocrnest
```

### Auteurs

- Prénom Nom
- Prénom Nom
- Prénom Nom
- Prénom Nom

Figure 22 : maquette du site internet

Pour accéder au site, veuillez saisir l'URL suivante dans la barre d'adresse de votre navigateur : <https://florian-fogliani.github.io/>

## 5 Retard(s), imperfections ou non réalisés

### 5.1 Le pré-traitement

Cependant, malgré les améliorations apportées par le processus de prétraitemet, des imperfections subsistent. En particulier, l'observation révèle que la grille représentée après le renforcement des contrastes n'est pas parfaitement alignée et nette. Ces déformations peuvent potentiellement avoir un impact négatif sur le découpage précis des cases.

De plus, lorsque la rotation est appliquée, comme illustré dans la figure 3, les traits deviennent irréguliers, perdant ainsi leur netteté d'origine.

Cette altération peut compromettre la qualité de la représentation de la grille et, par conséquent, influencer défavorablement les étapes ultérieures du processus de traitement.

### 5.2 Rotation automatique de l'image

La rotation automatique de l'image n'a pas pu être effectuée pour le rendu final de notre projet. Nous avons étudié de nombreuses pistes, mais aucune ne nous a permis de tendre vers un résultat convenable. Une piste fut particulièrement intéressante, mais malheureusement, elle n'a pas pu être assez approfondie pour le rendu final.

Nous avions réfléchi à l'utilisation de la transformée de Hough pour la détection du degré de rotation nécessaire. L'idée était d'utiliser l'angle Theta lié à chaque droite. Si cet angle n'était pas compris entre  $80^\circ$  et  $100^\circ$  (correspondant aux lignes horizontales) et entre  $-10^\circ$  et  $+10^\circ$  (correspondant aux lignes verticales), nous pouvions l'enregistrer comme une droite potentiellement liée à une image n'étant pas bien orientée. Après cette sélection de droites, nous souhaitions trier celles qui semblaient être à des distances égales deux à deux,

au nombre de 10, pour les considérer comme des lignes horizontales ou verticales.

Malheureusement, cette implémentation s'est avérée plus complexe que prévu, nécessitant une réflexion plus approfondie sur des notions de trigonométrie liées aux rotations, notamment sur les images pouvant avoir des angles de rotations supérieurs à  $180^\circ$ . Nous avons exploré de nombreuses pistes, mais malheureusement, aucune n'a été adéquate pour l'usage que nous souhaitions en faire.

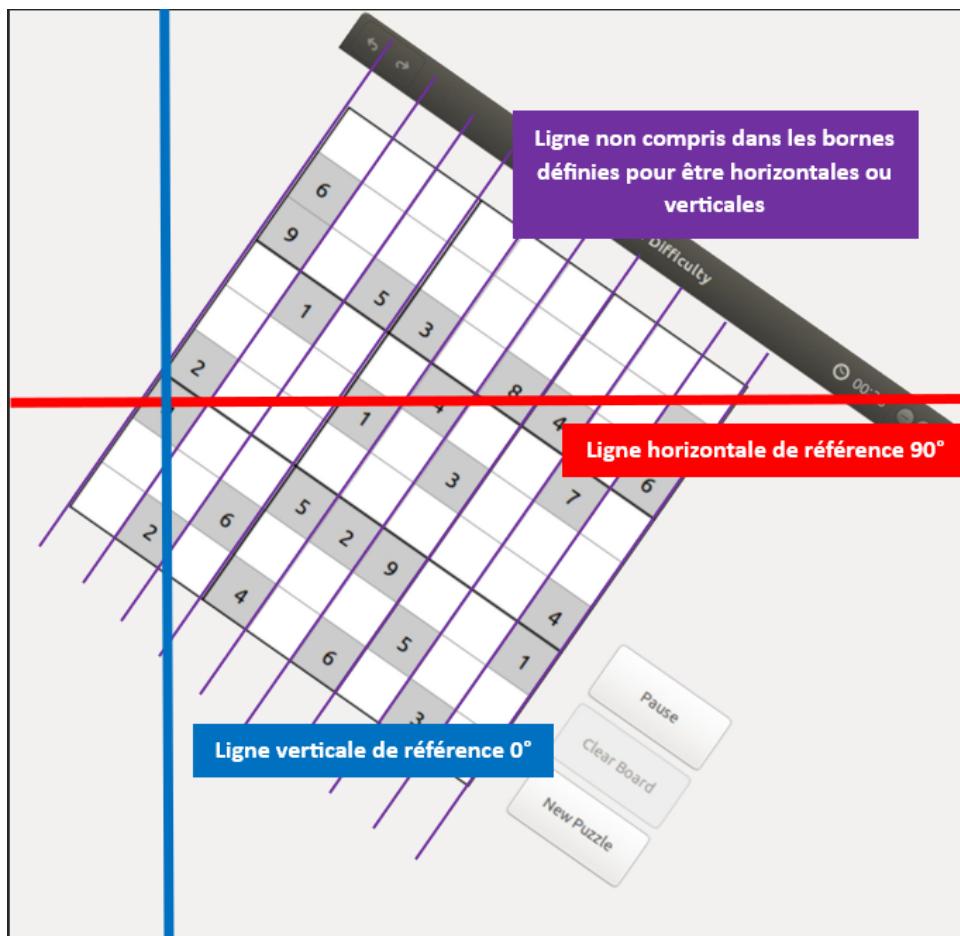


Figure 23 : Schéma de réflexion sur l'implémentation de la rotation automatique

### 5.3 Reconnaissance tapuscrite

Plusieurs difficultés ont été rencontrées pour la reconnaissance des chiffres tapuscrits.

En effet, il a été réalisé tardivement que un réseau performant sur des chiffres manuscrits ne l'est pas forcément pour ses congénères tapés au clavier.

Ainsi il a fallu créer un dataset personnalisé puisque contrairement au MNIST il n'y a pas de set public de chiffres tapuscrits. Ainsi un réseau a été entraîné mais n'a pas pu atteindre un pourcentage assez élevé ( 80%) pour être utilisé.

Enfin des imperfections lors du prétraitement (présence de lignes du cadrillage) ont rendu très complexe la reconnaissance des chiffres).

### 5.4 Interface utilisateur

Malheureusement, l'interface utilisateur n'est pas fonctionnelle, car elle ne contient aucune des étapes du traitement du sudoku en raison de bugs liés à la mise en commun du projet, notamment à cause d'erreurs d'importation de bibliothèques que nous n'avons pas réussi à résoudre avant la soutenance finale. Le squelette de l'UI est bien présent : on peut sélectionner un fichier, le sauvegarder et passer à la page suivante.

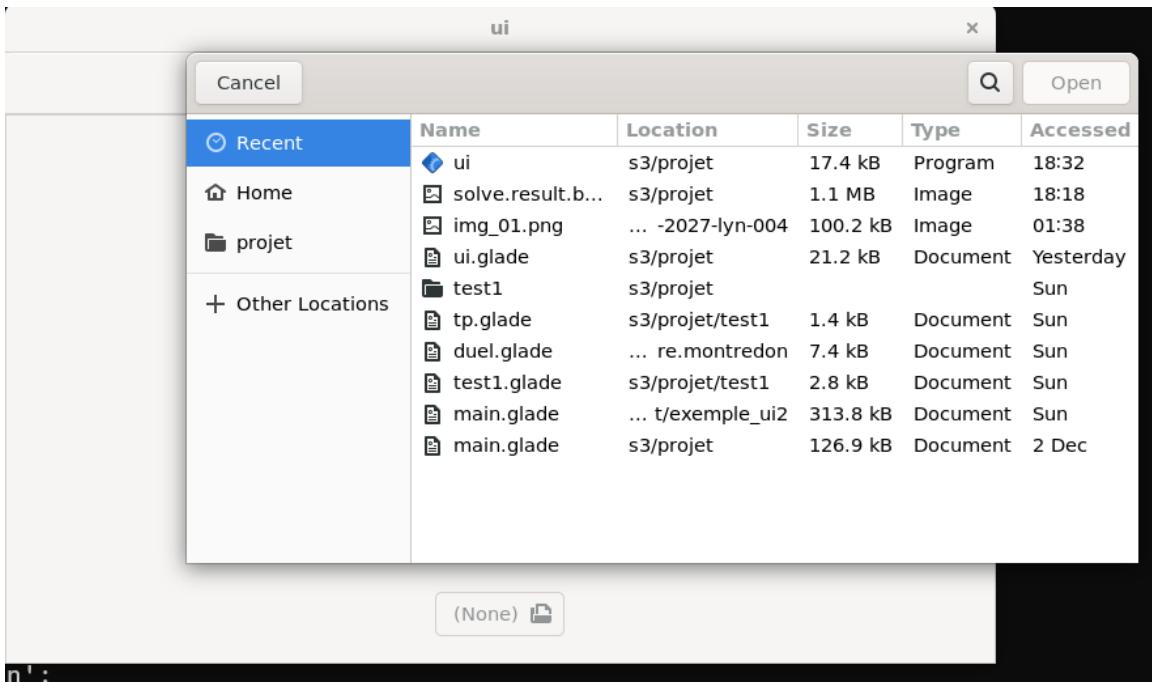


Figure 24 : Sélection du Sudoku à résoudre

## 6 Conclusion

Nous sommes aujourd’hui très fier(e)s de vous présenter notre projet, fruit de nombreux mois de travail. Ce projet fut enrichissant pour chacun de nous, tantôt au niveau des compétences techniques et humaines.

### 6.1 Commentaires individuels

#### 6.1.1 Laure

Je suis fière, aujourd’hui, de vous présenter notre projet. Nous n’avons pas atteint tous les objectifs fixés, mais chacun a su donner le meilleur de lui-même pour faire avancer le groupe. Ce projet était pour moi déroutant en raison de la précision de l’objectif à atteindre, parallèlement à la liberté de réalisation offerte, mais je pense m’en être bien sortie.

#### 6.1.2 Florian

Je suis heureux aujourd’hui de pouvoir vous présenter notre projet. Même s’il n’a pas pu aller jusqu’à son terme, je suis content d’avoir pu y participer. Il a été enrichissant dans l’apprentissage de nouvelles compétences en programmation. La liberté qui nous a été donnée dans l’implémentation de nos idées semblait déroutante à première vue, mais elle s’est révélée enrichissante par la suite. J’ai pu apprendre à faire par moi-même (et à chercher au fin fond de Google le fonctionnement de certains principes). La transformation de Hough fut un programme long et complexe à implémenter, difficile à comprendre et à cerner. J’ai bien remarqué cela lorsque le seul lien l’expliquant de façon claire était un site de l’École supérieure des sciences et techniques de Tunis, mais c’était un défi que je suis content d’avoir pu relever.

#### 6.1.3 Irène

En cette dernière étape de notre projet OCR, je suis ravie de présenter notre

travail, même s'il n'a pas tout à fait atteint l'objectif prévu, le parcours a été instructif, marqué par des défis inattendus et des moments d'apprentissage intenses. La collaboration au sein du groupe a été précieuse, chaque membre apportant ses compétences uniques à notre projet.

Dans l'ensemble, le projet demeure intéressant, mais il est également essentiel de souligner que la contrainte temporelle a eu un impact sur sa réalisation complète. Cette expérience servira de base pour améliorer notre capacité à gérer le temps.

#### **6.1.4 Ambroise**

Ce projet aura été pour moi un superbe premier pas dans le domaine de l'intelligence artificielle. J'ai dû assimiler beaucoup plus de notions que ce que j'imaginais, et cela a été très intéressant. J'aurais aimé aller plus loin pour ce dernier rendu, mais je suis quand même fière de ce que j'ai pu accomplir pendant ces quelques mois, en parallèle des cours.