

Rapport de soutenance
Projet S3 EPITA

Nino Zanfonato ; Théo Gille ; Clément Grégoire ; Leandro Tolaini

Novembre 2023

Table des matières

1	Introduction	2
1.1	Présentation du groupe	2
2	Répartition des tâches	4
3	Prétraitement et rotation	5
3.1	Prétraitement	5
3.2	Rotation	8
3.3	Ressenti	8
4	Détection de grille	10
4.1	Définition transformée de Hough	10
4.2	Utilisation de Hough dans le sudoku	11
4.3	Ressenti	12
5	Réseau de neurones	13
5.1	Conception	13
5.2	La fonction XOR	14
5.3	Ressenti	15
6	Solver et UI	16
6.1	Solver	16
6.2	Interface utilisateur	18
6.3	Ressenti	20
7	Conclusion	21

Chapitre 1

Introduction

1.1 Présentation du groupe

Ce projet offrira une opportunité inestimable pour le développement de compétences avancées en programmation en langage C. En outre, il nous permettra d'acquérir une compréhension approfondie du fonctionnement des systèmes d'intelligence artificielle (IA), des techniques de traitement d'image et des principes sous-jacents à la conception d'interfaces graphiques conviviales. Par ailleurs, ce projet revêt une importance significative en ce qu'il renforcera notre capacité à travailler efficacement en groupe tout en perfectionnant notre aptitude à gérer un projet complexe. Cette expérience nous enseignera des compétences essentielles en gestion de projet, telles que la planification, la répartition des tâches, la communication et la collaboration au sein de l'équipe. En somme, ce projet constitue une opportunité d'apprentissage holistique dans le domaine de la technologie et de l'informatique, nous préparant ainsi à des défis professionnels futurs.

Leandro : J'appréhendais quelque peu la réalisation de ce projet, car il me semblait très professionnel, exigeant le développement d'un produit abouti. Heureusement, le fait de pouvoir compter sur une équipe motivée a rapidement dissipé ces inquiétudes. J'ai rapidement compris qu'en travaillant correctement, nous pourrions mener à bien ce projet avec brio. Cependant, la détermination seule ne suffit pas ; une bonne organisation était nécessaire. J'ai donc pris le rôle de chef de projet afin de décharger les autres de cette tâche. Finalement, c'est avec beaucoup d'enthousiasme que je me suis lancé dans sa réalisation, étant convaincu d'apprendre énormément sur un sujet qui m'intéresse : l'intelligence artificielle.

Nino : Le projet OCR semblait initialement très complexe à mes yeux. Il représentait un défi nettement supérieur à celui du projet de création de jeux vidéo au semestre 2. Lorsque j'ai entamé ce projet, mes craintes ont été confirmées, et je me suis vite retrouvé face à des obstacles. En effet, le projet S2 de jeux vidéo avait déjà constitué un défi, et je me suis trouvé confronté à des difficultés similaires dans celui-ci. Cependant, je suis convaincu qu'il était impératif que je m'efforce de donner le meilleur de moi-même pour contribuer au succès de l'équipe. Au fur et à mesure de ma progression dans le projet OCR, j'ai acquis de nouvelles compétences et j'ai pu surmonter certaines des difficultés initiales. J'ai trouvé des moyens de travailler plus efficacement et j'ai pu compter sur le soutien de mes collègues.

Clément : Pour ce projet, ma responsabilité inclut le développement du solveur ainsi que la conception de l'interface graphique. Étant donné que je n'ai aucune expérience préalable dans le domaine de l'interface graphique, cette initiative représente une opportunité précieuse pour acquérir une compréhension approfondie de son fonctionnement. Par ailleurs, ce projet dans son ensemble me permettra de renforcer mes compétences en programmation en langage C, tout en m'offrant une expérience significative en gestion de projets en équipe.

Théo : Depuis tout petit, j'ai toujours été intéressé par l'informatique, quel que soit le domaine, que ce soit les jeux vidéo, la cybersécurité, le traitement d'images, et ainsi de suite. J'ai tout d'abord découvert l'informatique à travers les jeux vidéo, en utilisant de nombreuses consoles comme la plupart des gens. C'est à partir du moment où j'ai eu mon premier PC que l'informatique a vraiment commencé à m'intéresser. Ainsi, vu mon engouement pour l'informatique, j'ai décidé de m'orienter vers une école d'ingénieur en informatique, et voilà où je suis actuellement. C'est lors du Projet S2 que j'ai pu vraiment voir ce qu'est un projet en informatique, car je me suis occupé de tout ce qui concerne le multijoueur, mais aussi du développement de certaines mécaniques. Ainsi, durant l'ensemble de ce projet, malgré certains problèmes, j'ai pu y passer de très bons moments et j'en tire une bonne expérience. Avant de commencer ce projet qu'est l'OCR, je ne savais pas dans quoi nous allions être lancés, car avant de commencer, j'avais pu entendre que, comme moi, celui-ci est compliqué, mais surtout que c'est dans un nouveau langage de programmation que l'on n'avait pas encore fait. Mais je pense que je suis bien entouré avec des personnes motivées, ce qui devrait nous permettre de réussir ce projet.

Chapitre 2

Répartition des tâches

Pour cette première soutenance, nous avons décidé de nous répartir les tâches comme l'indique le tableau suivant :

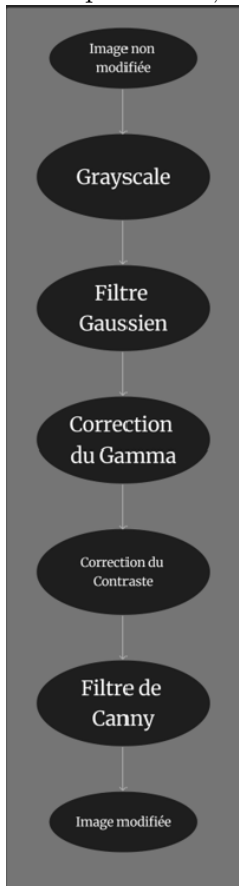
	Nino	Théo	Leandro	Clément
Traitement d'image		X		
Solver				X
IA			X	
Segmentation des lignes	X			
Interface				X
Rotation		X		

Chapitre 3

Prétraitement et rotation

3.1 Prétraitement

Avant de pouvoir détecter les bords du Sudoku et ainsi passer à la phase du solveur, il faut rendre l'image plus lisible pour l'ordinateur et permettre ainsi d'appliquer par la suite différents algorithmes. Pour cette première soutenance, une première partie des filtres a déjà été implémentée, la voici :



Grayscale :

Ce filtre, comme son nom l'indique, va permettre de passer l'image en niveaux de gris. Pour cela, il suffit simplement d'appliquer une formule sur tous les pixels de l'image :

```
color = 0.3 * red + 0.59 * green + 0.11 * blue;
```

	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	

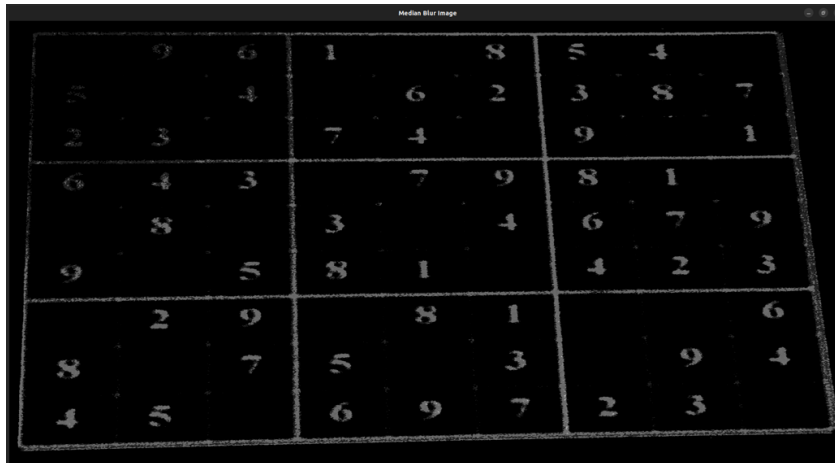
Filtre Gaussien ou Flou Gaussien :

Le filtre gaussien est utilisé pour réduire le bruit de l'image et égaliser les couleurs afin de faciliter le traitement. Le processus de ce filtre est simple. Pour chaque pixel de l'image, il faut calculer une nouvelle valeur à partir de la somme du produit des pixels voisins et d'une matrice gaussienne. Cette matrice est générée à l'aide d'une fonction gaussienne, qui est de la forme suivante :

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Gamma Le réglage de la partie Gamma de l'image permet d'accentuer les contrastes entre les zones claires et sombres de l'image. Ainsi, la formule à utiliser sera :

```
double factor = (259 * (contrast + 255)) / (255 * (259 - contrast));
double new_r = factor * (r - 128) + 128;
double new_g = factor * (g - 128) + 128;
double new_b = factor * (b - 128) + 128;
```

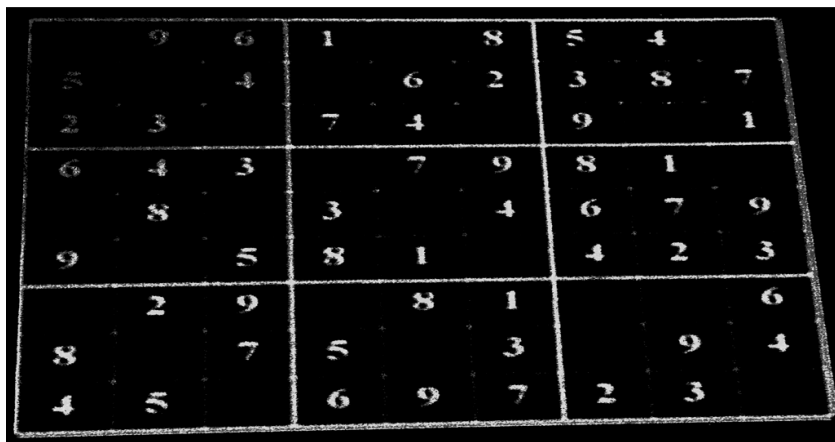


Correction du Contraste :

La correction du contraste de l'image permet d'améliorer la visibilité de certains détails de l'image et donc d'améliorer sa qualité.

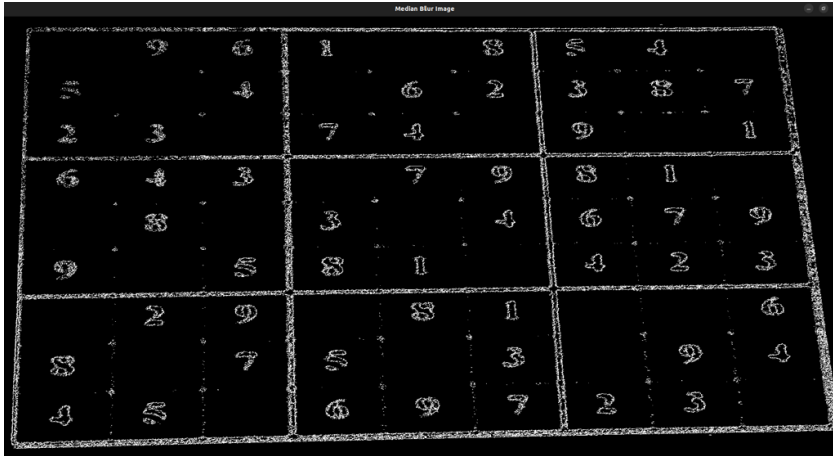
Dans notre cas ici on augmente juste la valeur de chaque pixel par 2 après l'avoir passé en niveaux de gris.

```
red = red * 2;
green = green * 2;
blue = blue * 2;
```



Filtre de Canny :

Le filtre de Canny permet de détecter les bords d'une image, en l'occurrence notre Sudoku. Il regroupe en lui plusieurs autres filtres qui, une fois assemblés, mettent en valeur les bords de l'image. Il est composé d'une réduction du bruit, d'un calcul du gradient d'intensité, de la détermination de la direction des contours, de la suppression des non-maxima, et enfin du seuillage des contours.



3.2 Rotation

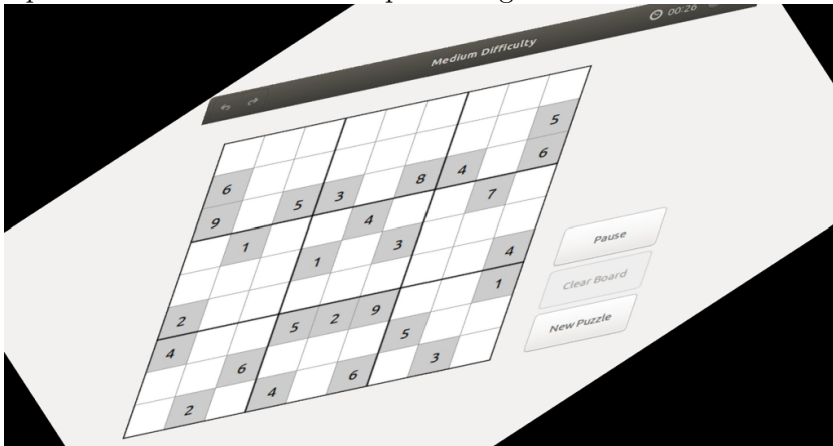
Pour la rotation de l'image, nous avons décidé d'implémenter l'algorithme suivant :

- Faire une copie de l'image actuelle
- Convertir l'angle en radians
- Calculer les cosinus et sinus de l'angle
- Calculer la taille de l'image après rotation
- Calculer le centre de l'image
- Effectuer la rotation des pixels

Pour avoir l'angle en radians il suffit simplement de faire cela :

```
angle_radian = -angle * M_PI / 180.0;
```

Après avoir fait toutes ces étapes l'image donne cela :



3.3 Ressenti

Toutes ces nouvelles compétences que j'ai pu acquérir au cours de ces dernières semaines n'ont pas été une mince affaire, notamment pour la partie de prétraitement, qui a demandé

énormément de recherche sur les filtres et la manière de les implémenter. Malgré le fait que tout ne soit pas parfait, je reste fier de ce que j'ai pu accomplir pour cette première soutenance.

Chapitre 4

Détection de grille

4.1 Définition transformée de Hough

L'algorithme de Hough est une technique de traitement d'image utilisée pour détecter des formes géométriques, en particulier des lignes droites et des cercles, dans une image. Il a été développé par Paul Hough en 1962 et a depuis été étendu pour détecter d'autres formes complexes. L'idée de base derrière l'algorithme de Hough est de représenter chaque point de l'image comme une courbe dans un espace paramétrique correspondant à la forme que l'on souhaite détecter.

Voici les étapes de base de l'algorithme de Hough pour détecter des lignes droites :

- Conversion de l'image en une représentation binaire : Avant de commencer, l'image d'origine est généralement convertie en une image binaire, où les pixels appartenant à l'objet d'intérêt sont mis à 1 (blanc) et le reste des pixels à 0 (noir).
- Création de l'espace de Hough : L'espace de Hough est une représentation paramétrique des lignes. Pour détecter des lignes, on utilise généralement une représentation polaire des lignes, où chaque ligne est représentée par deux paramètres : la distance r de l'origine au point le plus proche de la ligne et l'angle θ entre la ligne et l'axe horizontal. Un tableau appelé "accumulateur de Hough" est utilisé pour stocker les votes pour chaque combinaison de r et l'angle θ .
- Parcours de l'image : Pour chaque pixel blanc dans l'image binaire, on parcourt l'espace de Hough et on incrémente les valeurs correspondantes dans l'accumulateur de Hough en fonction de r et θ .
- Détection des lignes : Une fois que tous les pixels blancs ont été pris en compte, on examine l'accumulateur de Hough pour trouver les valeurs de r et θ qui ont reçu le plus de votes. Ces valeurs correspondent aux paramètres de la ligne détectée.
- Tracé des lignes détectées : On peut utiliser les valeurs de r et θ pour tracer les lignes détectées sur l'image d'origine.

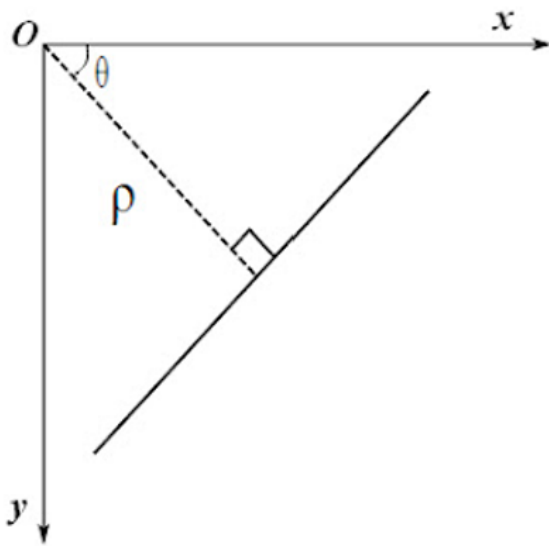
L'algorithme de Hough peut être adapté pour détecter d'autres formes géométriques, comme des cercles ou des ellipses, en modifiant la manière dont l'espace de Hough est défini. L'algorithme de Hough est robuste aux bruits et aux interruptions dans les lignes, ce qui en fait un

outil précieux pour la détection de formes dans les images.

Il existe des variantes de l'algorithme de Hough, telles que la Transformée de Hough probabiliste (ou Hough probabiliste) pour améliorer l'efficacité en ne traitant qu'un sous-ensemble aléatoire de points de l'image, ou la Transformée de Hough généralisée pour détecter d'autres formes.

4.2 Utilisation de Hough dans le sudoku

Une ligne peut être représentée comme $\rho = x \cos \theta + y \sin \theta$, où ρ est la distance perpendiculaire de l'origine à la ligne, et θ est l'angle formé par cette ligne perpendiculaire et l'axe horizontal. Cette direction varie en fonction de la façon dont vous représentez le système de coordonnées. Cette représentation ci-dessus est utilisée dans OpenCV. Toute ligne verticale aura un angle de 0 degrés, et les lignes horizontales auront un angle de 90 degrés pour θ . Normalement, le point d'origine sera le point en haut à gauche de l'image, de sorte que l'angle θ variera de 0 degré (minimum) à 90 degrés (maximum).



Nous savons que chaque point sur la ligne satisfait toujours cette équation : $\rho = x \cos \theta + y \sin \theta$ (1). Si nous connaissons une paire de (ρ, θ) , nous pouvons toujours dessiner une ligne. Donc, l'idée de la Transformée de Hough consiste à créer un tableau 2D M comme une matrice pour stocker les valeurs de deux paramètres (ρ et θ). Les lignes correspondent aux valeurs de ρ , et les colonnes correspondent aux valeurs de θ . Nous supposons qu'il existe une ligne d avec $\rho = \rho_1$ et $\theta = \theta_1$ dans l'image. Si un point A(x1, y1) se trouve sur d et satisfait (1) $\rho_1 = x_1 \cos \theta_1 + y_1 \sin \theta_1$, nous augmentons la valeur de $M[\rho_1][\theta_1]$ de 1. Nous continuerons ce processus pour chaque point sur la ligne, en augmentant la valeur de $M[\rho_1][\theta_1]$ de 1 si ce point satisfait l'équation (1). À la fin, la valeur de $M[\rho_1][\theta_1]$ sera beaucoup plus élevée que celle d'une autre cellule dans le tableau 2D M. Et la valeur de $M[\rho_1][\theta_1]$ sera également la longueur de la ligne. La taille du tableau 2D dépend de la précision dont vous avez besoin. Supposons que vous voulez une précision d'angles de 1 degré, vous aurez besoin de 180 colonnes. Pour ρ , la distance maximale possible est la longueur de la diagonale de l'image. Donc, en prenant une précision d'un pixel, le nombre de lignes peut être la longueur de la diagonale de l'image.

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

Donc on sort avec une image avec les lignes principales mises en avant.

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

4.3 Ressenti

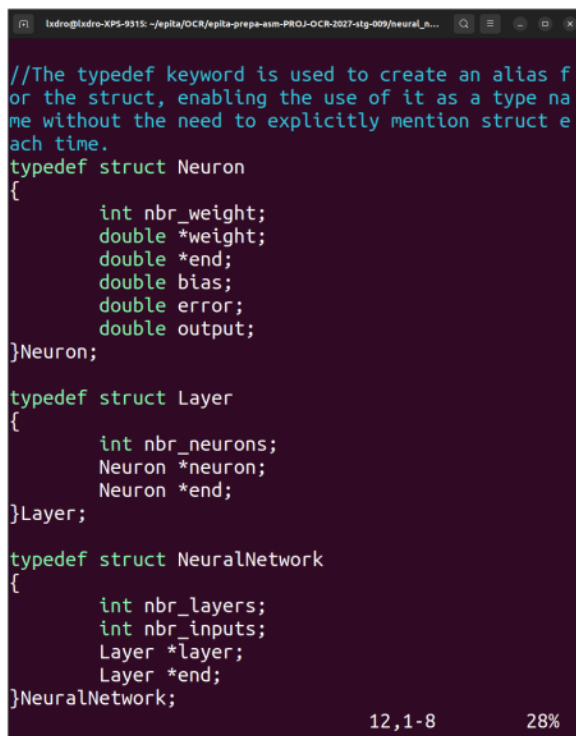
Ces nouvelles techniques de programmations ont été compliquer a comprendre et a manier, j'ai donc du me renseigner sur plusieurs sites et forum pour pouvoir avancé. Mes craintes de du début ont été confirmer puisque je me suis confronté à des difficultés très rapidement et à des défis a réaliser.

Chapitre 5

Réseau de neurones

5.1 Conception

Pour commencer, j'ai essayé d'implémenter les neurones, les couches et le réseau neuronal à la manière de la Programmation Orienté Objet du C# étudié l'année dernière. En effet, les structures C et le mot clé "typedef" m'ont permis de créer mes propres types, reliés à des structures et donc des attributs.



```
//The typedef keyword is used to create an alias for the struct, enabling the use of it as a type name without the need to explicitly mention struct each time.
typedef struct Neuron
{
    int nbr_weight;
    double *weight;
    double *end;
    double bias;
    double error;
    double output;
}Neuron;

typedef struct Layer
{
    int nbr_neurons;
    Neuron *neuron;
    Neuron *end;
}Layer;

typedef struct NeuralNetwork
{
    int nbr_layers;
    int nbr_inputs;
    Layer *layer;
    Layer *end;
}NeuralNetwork;
```

A partir de là, je devais réussir à initialiser chacun de ces attributs, notamment les poids et les biais. Pour ça j'ai implémenté une fonction qui génère un double suivant une distribution Gaussienne, grâce à la formule suivante :

$$\sqrt{-2 \cdot \log(u1)} \cdot \cos(2 \cdot \pi \cdot u2)$$

où u1 et u2 sont compris entre 0 et 1.

En manipulant les attributs avec des pointeurs et des tableaux, j'ai pu initialiser correctement

mon réseau de neurones. Maintenant il me restait à implémenter les fonctions pour le faire évoluer.

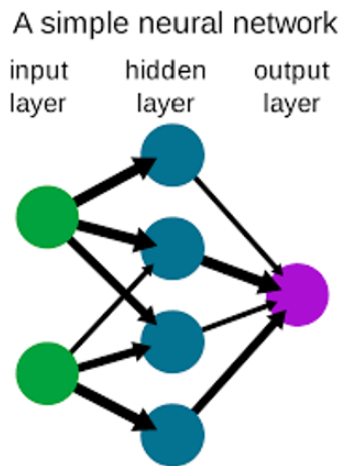
Grace à l'ebook fournit dans le sujet et la documentation, j'ai implémenté la fonction sigmoïde et sa dérivée, un algorithme de backpropagation, et de la fonction feedforward pour tester les résultats du réseau.

En suivant les instructions et en manipulant correctement mes structures de données, l'IA pouvait maintenant apprendre à partir d'exemples.

5.2 La fonction XOR

Pour cette première soutenance, mon avancement sur le développement du réseau de neurones devait se traduire par un exemple concret : faire apprendre à l'intelligence artificielle une fonction simple, celle du XOR.

Pour y parvenir, mon réseau de neurones devait donc être initialisé de manière à avoir cette forme-là :



- 2 input, pouvant chacun contenir 1 ou 0
- 1 output, pouvant contenir 1 ou 0
- une couche cachée (une seule est nécessaire pour ce genre de fonction simple) contenant un nombre arbitraire des neurones

Je devais me munir d'une base de données pour entrainer mon programme. Dans ce cas, cela n'a pas été compliqué étant donné la simplicité de la fonction à reproduire.

a	b	s
0	0	0
0	1	1
1	0	1
1	1	0

Fig. 19. - Table de vérité du
OU exclusif.

Pour chaque paire d'input, je connaissais l'output attendue. J'ai donc pu répéter le processus d'entraînement un nombre assez conséquent de fois jusqu'à que les poids de chaque lien et les biais de chaque neurone soient parfaitement ajustés. Mon premier réseau de neurones à appris la fonction XOR, et pour sauvegarder son résultat, il me suffisait de sauvegarder la forme du réseau (nombre d'input/output/hidden layers/neurones...) et enfin de stocker les poids de chaque lien et les biais de chaque neurone !

5.3 Ressenti

Toutes ses nouvelles notions à assimiler d'un coup m'ont quelque peu découragé au début. En effet la théorie autour des réseaux de neurones est assez fournie et complexe, de plus je n'avais encore aucune notion à ce sujet. Mais en me plongeant sérieusement dedans, et à force de tester, j'ai réussi à comprendre de mieux en mieux et éclaircir les points les plus flous. Cela m'a pris beaucoup plus de temps que ce à quoi je m'attendais, la partie du projet sur le réseau de neurone est plus dense que ce que je pensais et je n'ai donc pas pu me focaliser sur d'autres tâches.

Chapitre 6

Solver et UI

6.1 Solver

Pour le solveur de Sudoku, le processus commence par l'extraction des informations contenues dans un fichier texte. Chaque caractère représente une cellule, avec les points symbolisant les cellules vides. Les informations sont extraites et stockées dans une matrice de dimensions 9 par 9.

```
... ..4 58.
... 721 ..3
4.3 ... ...

21. .67 ..4
.7. ... 2..
63. .49 ..1

3.6 ... ...
... 158 ..6
... ..6 95.
```

Une fois cette première étape accomplie, des fonctions sont définies pour faciliter la résolution de la grille. La première fonction est "checkLline". Cette fonction a pour objectif de vérifier si un chiffre est déjà présent dans une ligne donnée. Elle prend deux paramètres, un entier "line" correspondant au numéro de la ligne, et un entier "x" représentant le chiffre recherché dans la ligne. Cette fonction parcourt la matrice et renvoie 1 si "x" peut être placé dans la ligne, sinon elle renvoie 0.

```

int check_line(int line, int x) //if x can be on line return 1, else return 0
{
    int i = 0;
    while (i < 9 && grid[line][i] != x)
    {
        i++;
    }

    return i == 9;
}

```

La fonction "checkCol" fonctionne sur le même principe, mais elle vérifie la présence de "x" dans la colonne plutôt que dans la ligne.

La fonction "checkBloc" vérifie si un chiffre "t" est présent dans le bloc de 3x3 cases. Elle prend un entier "t" et deux entiers "x" et "y" pour représenter la position de la cellule à vérifier. La fonction parcourt la petite matrice 3x3 et renvoie 0 ou 1 en fonction de la possibilité de placer "t" dans la cellule aux coordonnées (x, y).

La fonction "gridPrinter" stocke la grille "grid" dans une chaîne de caractères "p" dans le même format que la grille du fichier d'origine. Cette fonction facilite la détection d'erreurs et permet de formater la grille pour la sauvegarder dans le fichier ".result" à la fin du programme.

Une fonction précédant la dernière étape est "findEmpty", qui prend en argument deux pointeurs vers des entiers "xVide" et "yVide". Elle sert à localiser une case vide dans la matrice Sudoku en modifiant les pointeurs "xVide" et "yVide" pour les utiliser dans les autres fonctions.

```

int findEmpty(int* x_vide, int* y_vide)
{
    *x_vide = 0;
    *y_vide = 0;
    int flag = 0;
    while (flag == 0 && *y_vide < 9)
    {
        while (flag == 0 && *x_vide < 9)
        {
            if(grid[*y_vide][*x_vide] == 0)
            {
                flag = 1;
                break;
            }
            *x_vide += 1;
        }
        if(flag == 0)
        {
            *y_vide += 1;
            *x_vide = 0;
        }
    }
    return flag;
}

```

La dernière fonction, "solve", est une fonction récursive qui résout le Sudoku en effectuant des essais. Si aucune case vide n'est trouvée, le Sudoku est résolu. Sinon, la fonction essaie séquentiellement de remplir toutes les cases vides avec des valeurs de 1 à 9 tout en vérifiant leur compatibilité à l'aide des fonctions "checkLine", "checkCol" et "checkBloc". La fonction est appelée récursivement, et si elle ne renvoie pas 1, cela signifie que le Sudoku ne peut pas

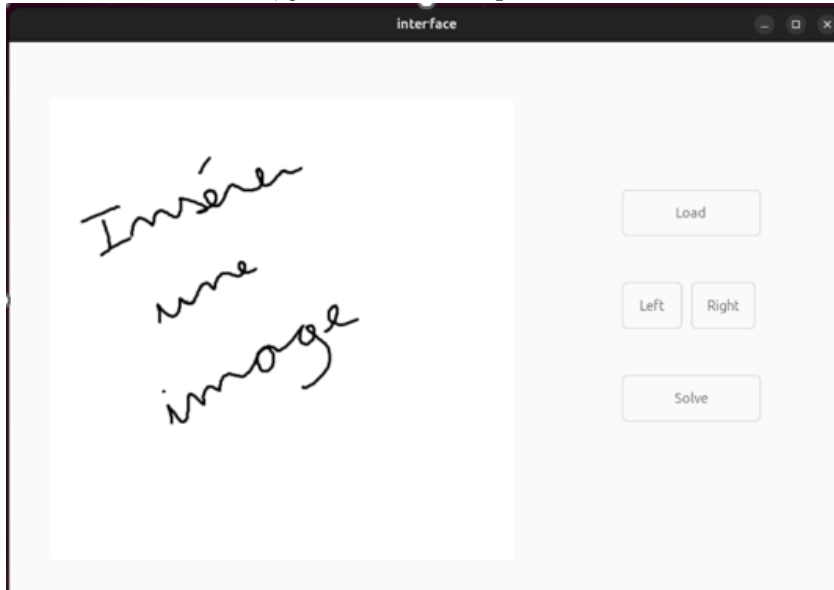
être résolu avec la valeur actuelle dans la case, donc la valeur est réinitialisée à 0, et la recherche continue.

```
int solve()
{
    int line = 0;
    int col = 0;
    if(findEmpty(&col, &line) == 0) return 1;
    for(int i = 1; i < 10; i++)
    {
        if(check_line(line, i) == 1 && check_col(col, i) == 1 && check_bloc(col, line, i) == 1)
        {
            grid[line][col] = i;
            if (solve()) return 1;
            grid[line][col] = 0;
        }
    }
    return 0;
}
```

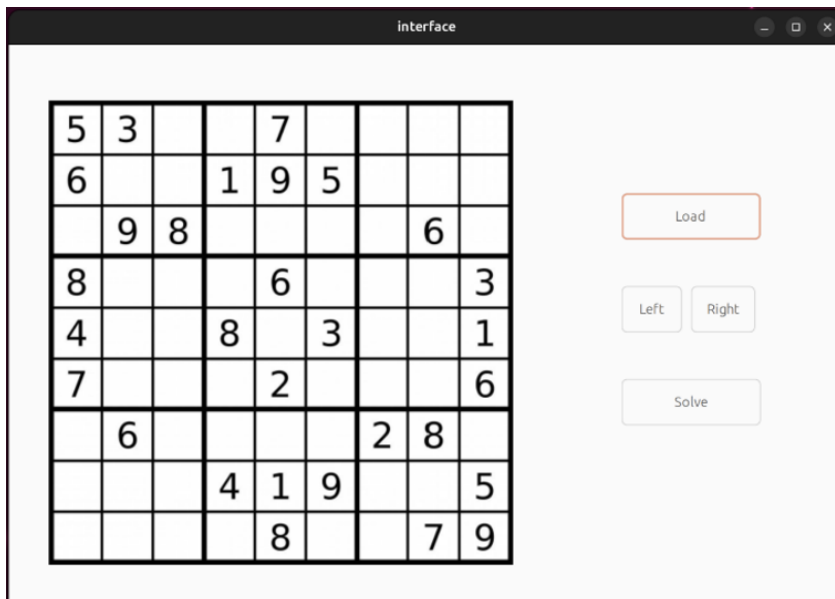
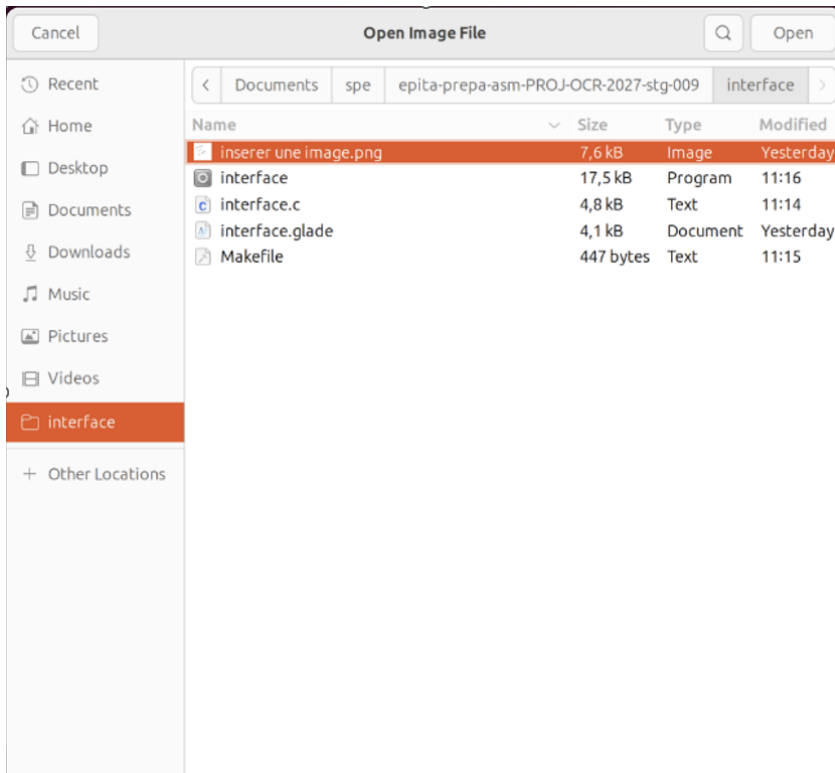
Une fois le Sudoku résolu, la fonction "gridPrinter" est utilisée pour stocker le résultat dans une chaîne de caractères, qui peut ensuite être enregistrée dans un fichier ".result".

6.2 Interface utilisateur

Pour la conception de l'interface graphique, j'ai utilisé le logiciel Glade afin de créer l'interface utilisateur, et la bibliothèque GTK pour implémenter la logique de l'application. Grâce à l'utilisation de Glade, j'ai élaboré une première version visuelle de l'application.



L'interface se compose de quatre boutons : un bouton "Load" permettant le chargement d'une image dans l'application, un bouton "Left" et un bouton "Right" permettant de faire pivoter l'image afin d'aligner la grille de Sudoku si nécessaire, et enfin un bouton "Solve" destiné à résoudre le Sudoku et afficher la solution sur l'interface. La partie gauche de l'interface est dédiée à l'affichage de l'image. Lorsque l'utilisateur appuie sur le bouton "Load," il a la possibilité de choisir une image depuis son ordinateur, et cette image s'affiche dans la zone prévue à cet effet.



Il est à noter que les autres boutons ne sont pas encore fonctionnels à ce stade. Cette interface représente une version préliminaire, et d'autres boutons seront vraisemblablement ajoutés ultérieurement, notamment pour permettre la découpe de l'image afin de ne conserver que la partie contenant la grille de Sudoku.

La fenêtre possède une taille préétablie et n'est pas redimensionnable. Cette décision vise à maintenir tous les éléments de l'interface à leur emplacement initial, évitant ainsi la nécessité de les repositionner pour les centrer dans la fenêtre.

6.3 Ressenti

Pour la conception du solveur de sudoku, l'algorithme de backtracking de base est amplement documenté sur Internet. J'ai ainsi rencontré relativement peu de difficultés lors de sa mise en œuvre. Cependant, j'ai été confronté à un problème spécifique : le programme ne fonctionnait pas correctement sur la machine de la salle, alors qu'il fonctionnait correctement sur mon ordinateur personnel. Par conséquent, j'ai dû réviser une partie du code depuis la salle de machine afin de garantir son bon fonctionnement.

En ce qui concerne l'interface utilisateur, l'utilisation du logiciel Glade s'est avérée relativement intuitive. Néanmoins, lorsqu'il s'agissait d'implémenter la partie du code en utilisant la bibliothèque GTK, j'ai éprouvé des difficultés à trouver des ressources en langage C pour réaliser ce que je souhaitais accomplir. La syntaxe employée par cette bibliothèque ne m'a pas paru particulièrement accessible. Malgré ces obstacles, j'ai tout de même réussi à mettre en place une interface fonctionnelle conforme à mes attentes.

Chapitre 7

Conclusion

En ce qui concerne notre première soutenance, nous avons réussi à respecter les échéances établies pour notre projet. Les filtres d'image sont opérationnels, notre intelligence artificielle démontre une capacité de reconnaissance du "ou exclusif", le solveur est fonctionnel, la détection de grille est également opérationnelle, et nous avons d'ores et déjà commencé la mise en place d'une interface utilisateur.

Pour notre deuxième soutenance, notre objectif essentiel est d'achever l'intégralité du projet, afin de présenter un solveur de sudoku pleinement fonctionnel, capable de gérer toutes les étapes du processus, de la résolution à l'interface utilisateur.