

---

Rapport de soutenance  
intermédiaire

REYNET

S3 - 2026

Célian A.    Alexandre C.  
Mathias F.    Hugo SR.

---

## Table des matières

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>Le groupe</b>   | <b>4</b> |
| 1.1      | Constitution . . . . .   | 4        |
| 1.2      | Origine du nom . . . . .   | 4        |
| 1.3      | Présentation des membres . . . . .                                 | 4        |
| <b>2</b> | <b>Répartition des charges</b>                                     | <b>5</b> |
| <b>3</b> | <b>État d'avancement du projet</b>                                 | <b>5</b> |
| <b>4</b> | <b>Les aspects techniques</b>                                      | <b>6</b> |
| 4.1      | L'organisation des fichiers du projet . . . . .                    | 6        |
| 4.2      | La compilation du projet . . . . .                                 | 7        |
| 4.3      | Chargement des fichiers des polices d'écriture et rendu du texte . | 9        |
| 4.4      | Génération/obtention d'un Data Set . . . . .                       | 9        |
| 4.5      | Chargement et sauvegarde des images . . . . .                      | 10       |
| 4.6      | La documentation avec Doxygen . . . . .                            | 10       |
| 4.7      | Prétraitement . . . . .  | 11       |
| 4.8      | Détection des bords/lignes sur une image . . . . .                 | 13       |
| 4.9      | Réseau de neurones (ou exclusif) . . . . .                         | 14       |
| 4.10     | Sauvegarde des données du réseau de neurones . . . . .             | 15       |
| 4.11     | Résolution de la grille . . . . .                                  | 15       |
| 4.12     | La convention de nommage utilisée . . . . .                        | 16       |
| 4.13     | Les fonctions de dessin . . . . .                                  | 17       |
| 4.14     | Le GUI . . . . .   | 17       |

## Introduction

Dans le cadre de notre projet de réalisation d'un OCR nous devons fournir un rapport qui se matérialise par le document que vous êtes en train de lire. Nous sommes "REYNET", un groupe constitué de Alexandre, Hugo, Mathias et Célian, quatre membres de la classe D1 et tous passionnés d'informatique.

Notre projet répond au problème qu'est la réalisation d'un "OCR Sudoku Solver". À travers ce rapport vous trouverez la présentation du groupe, notre répartition des tâches avec notre état d'avancement sur le projet mais aussi toute une partie décrivant nos choix et nos méthodes pour répondre aux différentes problématiques.

Ce projet est un vrai défi technique de par la réalisation d'un réseau neuronal, la coordination de l'équipe, le traitement des images et la résolution du sudoku. Ce document fait état de notre projet, son avancement, nos choix et les problèmes rencontrés.

## 1 Le groupe

Cette partie va présenter notre groupe, du choix de pourquoi nous avons choisi “REYNET” à sa constitution en passant par la présentation des membres.

### 1.1 Constitution

Le groupe est constitué de quatre membres de la classe D1. Nous sommes avant tout des amis ce qui donne de la force à notre groupe car nous pouvons travailler ensemble dans une bonne ambiance tout en ne perdant pas de vue l’objectif du projet et nous nous poussons entre nous pour nous surpasser et nous challenger afin de rendre le meilleur projet qui soit.

### 1.2 Origine du nom

On peut se demander pourquoi nous avons choisi “REYNET” comme nom de groupe. Nous voulions un nom qui sonne technologie à sa prononciation et en tant que fans incontestés du réalisateur Christopher Nolan et de son incroyable film “TENET” nous nous sommes inspirés du nom du film que nous avons légèrement modifié. Un film révolutionnaire tout comme notre projet.

### 1.3 Présentation des membres

#### Célian “kolte200” Aldehuelo-Mateos

J’adore tout ce qui touche à l’informatique et ma passion a commencé avec la programmation depuis le CM1 où j’ai découvert la liberté et diversité de résultats incroyables qu’offrait la programmation. À partir de là, je me suis intéressé à beaucoup d’autres concepts et langages de programmation. Je suis également passionné par la mécanique et l’électronique. Des domaines qui vont souvent bien ensemble dans la robotique.

#### Alexandre “LaGelee” Cornet

Je suis un grand fan d’informatique et surtout de cybersécurité. J’adore détourner les choses de leurs utilités premières et passer par les chemins cachés. Je suis membre du bureau de l’association HDFR et je représente l’école dans des compétitions de cybersécurité internationales et européennes.

#### Mathias “Héphaïsto” Filliol

Je suis passionné par l’informatique et par la musique. Depuis le premier confinement je me perfectionne dans le langage Java en créant des applications diverses. Ce projet est une occasion pour moi de sortir du confort apporté par ce langage et de me pencher sur des langages plus bas niveau tels que le C.



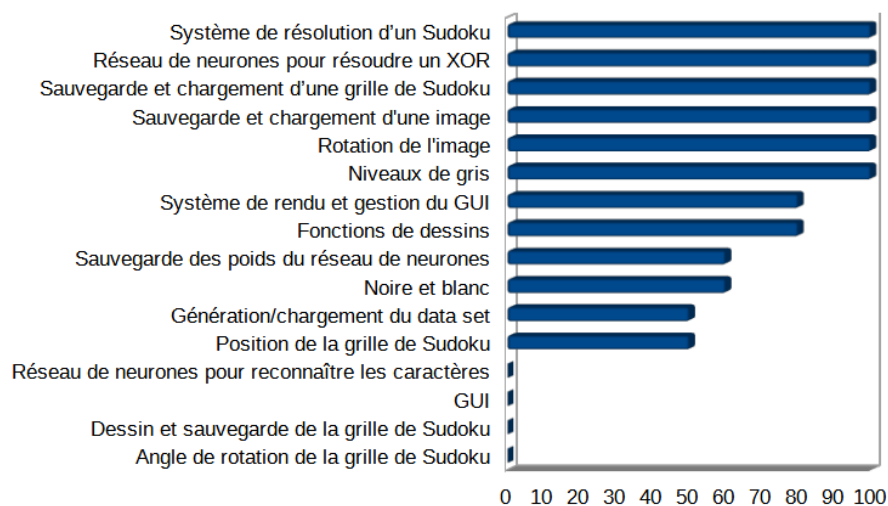
### Hugo “Gollumgogo” Saint-Raymond

Ayant commencé à programmer en Arduino au collège pour faire des effets de vagues avec des rubans de LEDs, je ne me suis toujours pas arrêté dans ma quête de connaissances. Aujourd’hui encore je découvre de nouveaux langages de programmation et en approfondis d’autres.

## 2 Répartition des charges

|                        | Célian A. | Hugo SR. | Mathias F. | Alexandre C. |
|------------------------|-----------|----------|------------|--------------|
| GUI                    | X         |          |            |              |
| Solveur de Sudoku      |           | X        |            |              |
| Détection des lignes   | X         |          |            |              |
| Détection de la grille | X         |          |            |              |
| Makefile               | X         |          |            |              |
| Réseau de neurones     |           |          | X          |              |
| Data Set               | X         |          |            | X            |
| Pré-traitement         |           |          | X          |              |
| Sauvegarde de l’IA     |           |          | X          |              |
| Rendu de la grille     |           |          | X          |              |
| Rédaction de rapport   |           | X        |            | X            |















## 3 État d’avancement du projet















## 4 Les aspects techniques

### 4.1 L'organisation des fichiers du projet

À la racine :

|  |   |
|--|---|
|  .vscode        | La configuration Visual Studio Code (aucun fichier spécifique à la machine)                   |
|  doxygen        | La configuration Doxygen  |
|  rapports       | Les fichiers utilisés pour faire ce rapport de soutenance                                     |
|  samples        | Les images utilisées par les tests  |
|  src            | Le code source du projet  |
|  tests          | Les fichiers de code source et les fichiers d'entrer des tests                                |
|  .gitignore     | .gitignore  |
|  .gitlab-ci.yml | Une configuration spécifique à GitLab (essentiellement pour les tests automatiques)           |
|  AUTHORS        | Les auteurs du projet   |
|  GUI-Tuto.odt   | Un tuto et des explications sur le fonctionnement du système de GUI                           |
|  Makefile       | Le fichier de configuration de GNU Make   |
|  README        | Pareil que "README.md" mais renommé pour respecter l'architecture de fichiers demandé         |
|  README.md    | Les informations de base du projet (notamment la manière de lancer le projet et les tests)    |
|  build.sh     | Un script qui build tout le projet avec le bon nombre de threads (juste pour aller plus vite) |

Pour le code source :

|  |  |
|--|--|
|  graphic      | Sources permettant de charger / sauvegarder une image et contient également le système d'affichage l'interface graphique |
|  image        | Contient le procédé de pré-traitement de l'image   |
|  logic        | Contient la génération de la dataset et le solveur de Sudoku   |
|  utils        | Contient des fichiers bien utiles comme la manipulation de vecteur ou de string non ASCII                                |
|  working_SDL2 | Contient un fix pour l'importation de la SDL 2 sur NixOS   |
|  common.h     | Contient des fonctions et des définitions de type utilisées dans beaucoup de fichiers sources                            |
|  error.c      | Gestion des erreurs (source)   |
|  error.h      | Gestion des erreurs (en-tête)  |
|  main.c       | Contient la fonction main() de l'OCR (permettra plus tard de lancer l'interface utilisateur)                             |
|  ocr.c        | Contiendra l'initialisation de l'interface utilisateur (source)  |
|  ocr.h        | Contiendra l'initialisation de l'interface utilisateur (en-tête)   |
|  solver.c     | Contient la fonction main() du solveur de Sudoku seulement   |

## 4.2 La compilation du projet

### Le Makefile :

Ce fichier a été développé et amélioré continuellement durant cette première partie du développement du projet.

Plusieurs règles de build ont été créées :

- “solver” : Permet de build uniquement l’exécutable du solver
- “reynet” : Permet de build l’entièreté de l’OCR (combinant OCR, GUI et solver)
- “test/nom\_du\_test” : Permet de build et de lancer le test “nom\_du\_test”
- “run/nom\_du\_test” : Permet de lancer le test “nom\_du\_test”
- “build/nom\_du\_test” : Permet de build le test “nom\_du\_test”
- “clean” : Permet de supprimer tous les fichiers générés par les processus de build
- “docs” : Permet de générer une documentation HTML du code source avec Doxygen

Nous avons également ajouté plusieurs variables d’environnement afin de contrôler des paramètres de compilation :

- DEBUG : Par défaut à “0”, et, quand activé (mis à “1”), désactive les flags d’optimisation au profit des flags de debug (comme “-g”).
- USE\_FREEIMAGE : Par défaut à “0”, et, quand activé, permet d’utiliser la bibliothèque “FreeImage” pour charger et sauvegarder les fichiers images plutôt que SDL 2 Image.

### Problèmes de compilation rencontrés :

Durant la création de ce Makefile j’ai été confronté à plusieurs problèmes de compilation sur NixOS.

En effet, bien que la bibliothèque “FreeType 2” (utilisée pour le chargement des polices d’écriture et par la même occasion, est utilisée par GTK et SDL 2 TTF) soit présente sur le système (les fichiers d’en-tête étant trouvable dans /nix/store/), la commande “pkg-config freetype2” qui est censée renvoyer les options de compilation à utiliser pour utiliser FreeType 2, ne trouve pas la bibliothèque.

Pour pallier ce problème il a fallu rechercher manuellement le chemin du dossier contenant les fichiers d’en-tête de cette bibliothèque, ainsi que le chemin vers le dossier contenant la bibliothèque compilée (fichier .a).



Voici un screen d'une partie du Makefile permettant de gérer la compatibilité avec les sessions SPED de Nix OS :

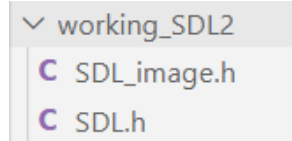
```
ifeq ($(shell grep -i "ID=nixos" /etc/os-release &> /dev/null && echo yes),yes)
FT2_CFLAGS := -I$(shell dirname $(shell find /nix/store/ -name ft2build.h | head -qn1))
FT2_LDFLAGS := -L$(shell dirname $(shell find /nix/store/ -name libfreetype.a | head -qn1)) -lfreetype
else
FT2_CFLAGS := $(shell pkg-config --cflags freetype2)
FT2_LDFLAGS := $(shell pkg-config --libs freetype2)
endif
```

La première ligne permet de tester si nous sommes sur Nix OS.

Un autre problème de compilation auquel on a été confronté sur Nix OS, est l'utilisation de la SDL.

En effet l'inclusion directe "SDL2/SDL.h" ou "SDL2/SDL\_image.h" crée un warning sur Nix OS, donc pour remédier à ce problème nous avons trouvé que définir la macro "\_\_NO\_INLINE\_\_" avant l'inclusion de la SDL était une solution. Cependant, il ne serait pas correct de définir cette macro dans l'ensemble du fichier source utilisant la SDL. C'est pour cette raison qu'il vaut mieux annuler la définition de cette macro juste après l'inclusion de la SDL.

Pour faciliter l'inclusion de la SDL, un dossier "working\_SDL2" a été créé et contient un fichier pour chaque inclusion différente de la SDL 2 utilisé dans le projet.



Voici le contenu de "working\_SDL2/SDL.h" (le code de "working\_SDL2/SDL\_image.h" est très similaire) :

```
1  /*
2  Here #define __NO_INLINE__ is used because there is a bug in wchar_t.h
3  used by SDL2 that create a warning during compilation on NixOS:
4
5  In file included from /run/current-system/sw/include/SDL2/SDL_stdinc.h:72,
6  from /run/current-system/sw/include/SDL2/SDL_main.h:25,
7  from /run/current-system/sw/include/SDL2/SDL.h:32,
8  from grayscale.c:2:
9  /run/current-system/sw/include/wchar.h: In function 'wctob':
10 /run/current-system/sw/include/wchar.h:326:47: warning: comparison of unsigned expression in '>' '0' is always true [-Wtype-limits]
11 326 | { return (builtin_constant_p (wc) && wc >= L'\0' && wc <= L'\x7f'
12  */
13
14 #ifndef __NO_INLINE__
15 #define __NO_INLINE__
16 #include <SDL2/SDL.h>
17 #undef __NO_INLINE__
18 #else
19 #include <SDL2/SDL.h>
20 #endif
```

Le principe est donc d'inclure "working\_SDL2/SDL.h" au lieu de "SDL2/SDL.h" ou "working\_SDL2\_SDL\_image.h" au lieu de "SDL2/SDL\_image.h" à chaque fois que cette bibliothèque doit être inclus dans un fichier quelconque du projet.



### 4.3 Chargement des fichiers des polices d'écriture et rendu du texte

Cette partie n'a pas encore été développée mais est prévue pour la prochaine soutenance.

Il s'agit ici de charger un fichier au format OpenType ou TrueType afin de pouvoir faire le rendu de texte.

Le rendu de texte sera utilisé par le GUI et potentiellement la génération du data set.

Afin de charger ces fichiers nous avons pensé à utiliser la bibliothèque FreeType2, utilisée par SDL 2 TTF et GTK et présente sur NixOs (même si quelques problèmes ont été rencontrés (cf. la section "Problèmes de compilation rencontrés")).

### 4.4 Génération/obtention d'un Data Set

Nous avons ici pensé à plusieurs possibilités pour obtenir les images nécessaires à l'apprentissage du réseau de neurone : générer les images, obtenir des images existantes.

Pour l'obtention d'images existantes, nous avons pensé à utiliser la database du MNIST (lien : <http://yann.lecun.com/exdb/mnist/>) qui présente un grand nombre de caractères écrits à la main.

Pour la génération d'images il est prévu d'utiliser une police d'écriture aléatoire, de faire le rendu du texte, et finalement d'appliquer des post-traitements afin de dégrader la qualité de l'image aléatoirement.

Actuellement seul le chargement du dataset du MNIST a été codé partiellement, en effet les images peuvent être chargées et affichées mais le chargement des "labels" (le chiffre associé à chaque image) n'a pas encore été fait.

Les fichiers du MNIST sont initialement compressés au format GZip et nous avons décidé de garder cette compression car elle diminuait de plus de moitié la taille du dataset. Pour le chargement d'un tel fichier, la bibliothèque zlib a été utilisée.

## 4.5 Chargement et sauvegarde des images

Nous avons décidé de rendre le chargement d'image possible avec 2 bibliothèques : SDL 2 Image et FreeImage. La raison est que FreeImage était le premier choix car cette bibliothèque supporte plus de formats d'image que SDL 2 Image mais dans le cas où cette bibliothèque ne serait pas présente sur les sessions SPE de Nix OS alors SDL 2 Image a été utilisée pour charger les images comme seconde option.

Le choix de la bibliothèque se fait donc au moment de la compilation car selon la valeur de la variable d'environnement "USE\_FREEIMAGE" pendant la compilation avec GNU Make, la macro "USE\_FREEIMAGE" est définie ou non et cela permet de sélectionner le bon code source à compiler dans le fichier "graphic/image.c".

Donc, pour le chargement d'image, les fonctions de SDL 2 Image ou de FreeImage ne sont jamais appelées directement car cela casserait la couche d'abstraction mise en place et la possibilité d'être compatible avec l'utilisation de 2 bibliothèques différentes.

## 4.6 La documentation avec Doxygen

Pour faciliter le travail en équipe sur le projet et son code source, nous avons décidé d'utiliser Doxygen pour générer dynamiquement une documentation du projet. Cela permet de facilement trouver le prototype et la liste des fonctions existantes. Cependant, peu de fichiers possèdent des commentaires décrivant l'utilisation et l'utilité de chacun des paramètres d'une fonction.

Le fichier "image.c" permettant le chargement des images est l'un d'entre eux car ses fonctions sont beaucoup utilisées dans l'ensemble du projet et comme décrit précédemment le chargement des images ne peut pas se faire directement à travers une bibliothèque, donc il se devait d'être documenté pour faciliter le développement du projet.

Voici un aperçu de la documentation du fichier "image.h" (le thème par défaut est utilisé) :

## Reynet 1.0.0

|           |                   |         |
|-----------|-------------------|---------|
| Main Page | Data Structures ▾ | Files ▾ |
|-----------|-------------------|---------|

src graphic

### image.h File Reference

This file provide stuff to load and manipulate images. [More...](#)

```
#include "common.h"
#include "working_SDL2/SDL.h"
```

[Go to the source code of this file.](#)

#### Data Structures

|        |                         |
|--------|-------------------------|
| union  | <a href="#">color_u</a> |
| struct | <a href="#">image_s</a> |

#### Macros

|         |   |   |
|---------|---|---|
| #define | <a href="#">IMG_RED_MASK</a>              | 0xFF000000  |
| #define | <a href="#">IMG_GREEN_MASK</a>            | 0x00FF0000  |
| #define | <a href="#">IMG_BLUE_MASK</a>             | 0x0000FF00  |
| #define | <a href="#">IMG_ALPHA_MASK</a>            | 0x000000FF  |
| #define | <a href="#">RGB_TO_COLOR</a> (vr, vg, vb) | (( <a href="#">color_t</a> ){.r=vr, .g=vg, .b=vb, .a=0xFF}) |

#### Typedefs

|                |                         |                         |
|----------------|-------------------------|-------------------------|
| typedef union  | <a href="#">color_u</a> | <a href="#">color_t</a> |
| typedef struct | <a href="#">image_s</a> | <a href="#">image_t</a> |

## 4.7 Prétraitement

### Niveau de gris :

Pour déterminer la nouvelle image avec des niveaux de gris on calcule le niveau de gris pour chaque pixel de l'image. La formule utilisée pour calculer le niveau de gris d'un pixel (ITU-R 601-2 luma transform) est la suivante :  $0.3*r + 0.59*g + 0.11*b$  ou r, g et b sont respectivement les valeurs de rouge, vert et bleu du pixel (donc cette formule permet de conserver le niveau de luminosité perçue). Nous utilisons cette formule puisque notre œil perçoit mieux le vert puis le rouge puis le bleu. Une fois le niveau de gris calculé on l'applique à chacune des valeurs RGB du pixel.

### Rotation manuelle de l'image :

Cette étape permet de redresser l'image avec un angle donné. Pour effectuer la rotation manuelle de l'image nous avons utilisé une rotation de matrice. En effet les pixels d'une image sont stockés dans un tableau, nous pouvons effectuer une rotation sur ce tableau pour obtenir la nouvelle image.

### Conversion en noir et blanc :

Ce procédé consiste à transformer une image RGB en image contenant seulement du noir et du blanc, mais contrairement au niveau de gris il n'y a pas de niveau intermédiaire de couleur entre le noir et le blanc, donc chaque pixel peut être stocké sur seulement 1 bit.

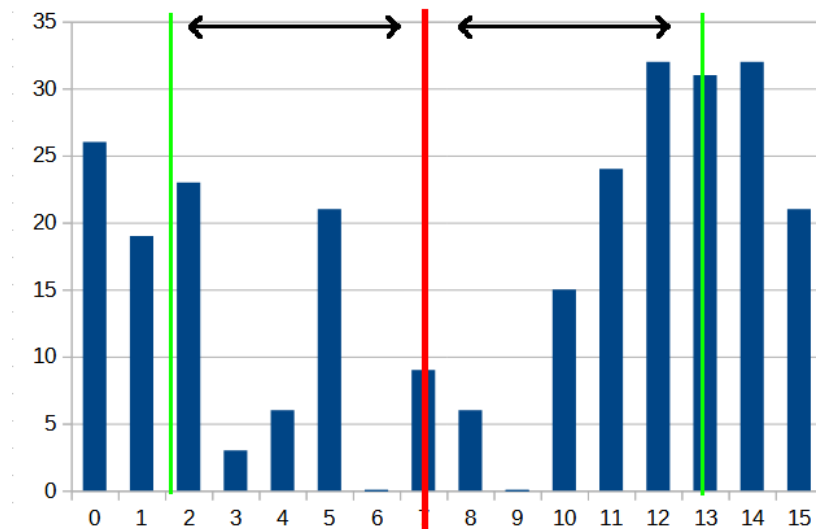
Pour arriver à ce résultat il suffit de définir un seuil à partir duquel le pixel est blanc et est noir autrement. Ce seuil est une valeur entre 0 et 255 (incluses) et pour chaque pixel ce seuil est comparé à la valeur de niveau de gris elle aussi comprise entre 0 et 255.

Mais se pose alors un problème, car on ne connaît pas la meilleure valeur possible de ce seuil. En effet une valeur trop faible aura pour effet d'avoir trop de blanc et une valeur trop élevée va donner trop de noir et dans les deux cas on perd de l'information.

Cette valeur peut tout à fait être codée en dur mais elle ne sera alors par adaptée en fonction de l'image, et certaines images trop sombres ou trop claires seront alors transformées en image complètement blanches ou noires.

J'ai donc cherché une méthode pour calculer automatiquement la meilleure valeur de seuil en fonction de l'image. Voici la méthode que j'ai utilisée :

Le critère sur lequel je me suis basé pour savoir si un seuil est bien ou pas est "l'éloignement" du seuil des couleurs moyen de part et d'autre de ce seuil. Donc plus cet éloignement est élevé pour chaque côté, mieux c'est. Voici une image pour l'illustrer :



Ici la ligne rouge est le seuil et les lignes vertes la couleur moyenne de part et d'autre du seuil.

L'axe des abscisses est la couleur et l'axe des ordonnées la quantité de pixels de cette couleur.

L'avantage de cette méthode est que la quantité de noir ou de blanc ne rentre pas en compte dans le calcul du seuil, mais seulement la répartition de ce noir et de ce blanc (ici est considéré comme noire toute valeur inférieure au seuil et comme blanc toutes celles supérieures).

## 4.8 Détection des bords/lignes sur une image

### Détection des points constituant les lignes :

Afin de détecter les bords, j'ai d'abord opté pour détecter tous les points séparant deux couleurs ainsi que calculer les caractéristiques suivantes pour chaque point :

- La largeur de la transition entre les deux couleurs en pixels
- La visibilité de cette transition
- La vecteur directeur normalisé de la transition

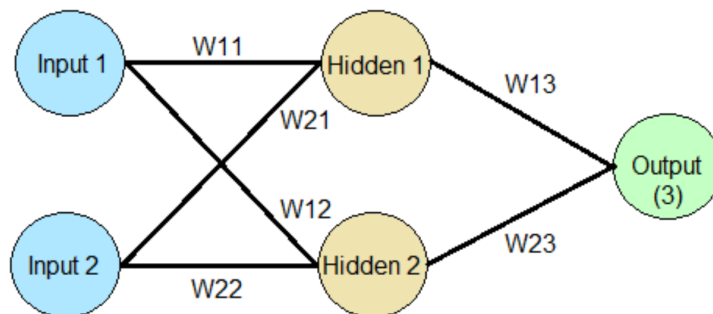
Une ligne est une bordure entre deux couleurs différentes et est constituée d'une suite de points, mais la difficulté réside dans le fait que la largeur de cette bordure peut s'étendre sur plusieurs pixels de long. On doit donc vérifier pour chaque pixel si, en le comparant avec un autre pixel proche de lui, sa différence de couleur est élevée. Sachant que cet autre pixel avec lequel on compare, n'est pas forcément adjacent. En effet, comme dit précédemment, une ligne peut faire plusieurs pixels de large. Le but de cet algorithme est donc de placer plein de points sur l'image, et chacun de ces points doit décrire une séparation de deux couleurs avec plusieurs caractéristiques telles que : la différence de couleur, la largeur de la séparation, la visibilité (ou force) de cette séparation. En général, sur la séparation entre deux couleurs, plus on éloigne les deux pixels à comparer et plus on a de chance d'avoir une différence élevée. Ceci est un problème, car la largeur des séparations aura tendance à être très élevée. Donc pour éviter cela, j'ai trouvé trois méthodes applicables : définir une largeur maximum, vérifier si chaque pixel séparant les deux pixels à comparer, est continu (pas de variation à sens opposé d'un pixel à l'autre) et chercher à récupérer le segment avec la plus grande variation moyen en considérant également la variation totale. Pour ce faire, il a fallu trouver une fonction mathématique permettant de déterminer la meilleure longueur séparant les deux pixels :  $\text{score} = \text{moy\_of\_diff\_between}(\text{pixels}) * \text{diff}(\text{pixel\_1}, \text{pixel\_2})$  Où le but est d'avoir le score le plus haut possible.

#### 4.9 Réseau de neurones (ou exclusif)

Pour la première soutenance nous avons juste besoin d'une preuve de concept de notre réseau de neurones. Pour cette preuve, nous avons réalisé un mini réseau capable d'apprendre la fonction OU EXCLUSIF. Ce réseau comporte 2 entrées et 1 sortie car la fonction logique OU EXCLUSIF est construite ainsi. La porte logique OU EXCLUSIF (XOR) renvoie 1 uniquement si une de ses entrées est à 1, autrement dit renvoie 1 si ses deux entrées sont différentes. Voici sa table de vérité :

| A | B | XOR |
|---|---|-----|
| 0 | 0 | 0   |
| 0 | 1 | 1   |
| 1 | 0 | 1   |
| 1 | 1 | 0   |

Pour effectuer le réseau de neurones nous allons utiliser la fonction sigmoid ainsi qu'une seule couche cachée. Voici une représentation du réseau de neurones XOR :



Notre taux d'apprentissage est ici fixé à 0.2 car il est suffisamment grand pour que notre réseau apprenne mais pas assez petit pour que ça prenne beaucoup d'itérations.

Notre jeu de données d'entraînement se limitera aux valeurs possibles du XOR (celle du tableau ci-dessus). D'après l'algorithme du gradient stochastique nous devons mélanger aléatoirement les échantillons de l'ensemble d'apprentissage.

Pour chaque valeur de test nous allons calculer l'activation de la couche cachée puis calculer celle de la couche de sortie. Afin que notre réseau apprenne nous allons utiliser le principe de rétropropagation. Notre marge d'erreur sera ici la différence entre la sortie attendue et celle obtenue. En testant notre XOR nous avons trouvé qu'un nombre d'époque supérieur à 3000 permettait d'avoir 100% de réussite (en arrondissant les valeurs). Il en faudrait beaucoup plus pour avoir exactement 1 ou 0.

#### 4.10 Sauvegarde des données du réseau de neurones

Pour ne pas perdre les données du réseau de neurone, nous devons les sauvegarder. Pour l'heure nous avons choisi de les sauvegarder sous la forme binaire. Nous avons 4 tableaux à sauvegarder ; deux pour la couche cachée et deux pour la couche de sortie. Il y a en tout 2 tableaux à une dimension de double et deux tableaux à deux dimensions de double. On sauvegarde les tableaux les uns à la suite des autres et pour les récupérer il nous suffit de lire le fichier du début vers la fin et de récupérer les valeurs binaires des tableaux. Pour l'instant cette méthode marche bien car la taille des tableaux est fixe mais lorsque nous implémenterons le réseau de neurones pour les chiffres nous allons probablement faire varier le nombre de nœuds dans la couche cachée. Pour se faire nous allons sûrement passer par un en-tête comportant les données que nous devons récupérer pour créer nos tableaux.

#### 4.11 Résolution de la grille

Comme nous avons déjà fait un résolveur de sudoku au cours de l'année de SUP, nous avons une base de secours au cas où on ne trouverait pas mieux. Le problème de la version que nous avons déjà codée était que, dans certains cas, la résolution pouvait durer plusieurs secondes car c'était un algorithme de brute force (backtracking). Son implémentation est toutefois la plus facile, mais pas la plus rapide. Nous avons donc recherché les différentes techniques de résolutions et en avons retenu une qui utilise le concept de singletons nus. Un singleton nu correspond tout simplement à une case qui ne peut accepter qu'un seul chiffre, case que l'on va ainsi remplir directement avec cette valeur comme dans l'exemple suivant :

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

Toutefois l'algorithme n'est pour l'instant pas déterministe du fait que nous pouvons tomber sur un cas où nous ne trouvons plus de singleton nu. C'est pour

cela que dès que nous n'en trouvons plus, nous passons à une phase où l'algorithme cherche la case qui a le moins de possibilités différentes pour sa valeur et appelle récursivement à la première phase en changeant la valeur de ladite case par ses différentes valeurs possibles une à une jusqu'à trouver une résolution intégrale de la grille. Nous avons ainsi notre algorithme qui est déterministe, car il trouvera une solution à la grille s'il en existe une et s'arrêtera dans le cas échéant.

Nous aurions pu encore améliorer le temps de résolution en faisant entrer en compte les principes de paires nues, triples nus, etc., mais ça aurait été se rajouter beaucoup de difficulté pour finalement peu d'intérêt dans notre situation. En effet, à terme, notre projet n'aura à résoudre qu'une seule grille de sudoku et non des milliers d'affilé. Une optimisation de quelques fractions de secondes ne nous est donc d'aucune véritable utilité. Cependant, l'option d'améliorer un peu l'algorithme de résolution pourrait revenir sur la table s'il nous reste du temps supplémentaire avant le rendu final.

D'autre part, le format de stockage de la grille de sudoku dans les fichiers texte est la suivante :

| test_00 110 bytes |             | test_00.result 110 bytes |             |
|-------------------|-------------|--------------------------|-------------|
| 1                 | ... ..4 58. | 1                        | 127 634 589 |
| 2                 | ... 721 ..3 | 2                        | 589 721 643 |
| 3                 | 4.3 ... ..  | 3                        | 463 985 127 |
| 4                 |             | 4                        |             |
| 5                 | 21. .67 ..4 | 5                        | 218 567 394 |
| 6                 | .7. ... 2.. | 6                        | 974 813 265 |
| 7                 | 63. .49 ..1 | 7                        | 635 249 871 |
| 8                 |             | 8                        |             |
| 9                 | 3.6 ... ..  | 9                        | 356 492 718 |
| 10                | ... 158 ..6 | 10                       | 792 158 436 |
| 11                | ... ..6 95. | 11                       | 841 376 952 |
| 12                |             | 12                       |             |

Avec à gauche une grille non résolue et à droite le fichier obtenu après avoir utilisé la commande `"/result chemin_du_fichier"` avec le chemin du fichier `test_00`.

#### 4.12 La convention de nommage utilisée

Les noms des fonctions sont écrits en "lower\_snake\_case".

Les noms des constantes sont écrits en "UPPER\_SNAKE\_CASE".

Les noms des variables sont écrits en "lower\_snake\_case".

Cette convention est basée sur celle du C.





La majorité des fonctions a un préfixe afin de les classer par catégorie. Le nom d'une fonction est donc de manière générale : `catégorie_fonction()` Voici la liste des catégories actuelles de fonction et leurs préfixes :

- Images : `img`
- Erreurs : `err`
- Journalisation : `log`
- Détection des lignes : `edge`
- Détection de la grille : `grid`
- Fonction de dessin : `draw`
- Vecteurs : `vect`
- Solveur : `solve`

De manière plus générale le terme le plus courant dans le nom de fonctions est placé au début.

#### 4.13 Les fonctions de dessin

Afin de faciliter le dessin de la grille de sudoku pour le projet final ainsi que de facilement visualiser le résultat de certains tests, des fonctions de dessin ont été créées telles que :

- `draw_rect()` : Pour dessiner un rectangle (plein)
- `draw_circle()` : Pour dessiner un cercle (plein)
- `draw_line()` : Pour dessiner une ligne

Actuellement seule la fonction permettant de dessiner une ligne ne fonctionne pas correctement, le reste est tout à fait fonctionnel.

#### 4.14 Le GUI

Pour calculer la position des éléments, afficher et calculer les interactions du GUI, un système personnalisé sera créé. Une grande partie du code a déjà été écrite et le système permettant de renseigner comment positionner les éléments du GUI a déjà été imaginé.

Une structure hiérarchique en se basant sur un arbre général premier fils - frère droit a été utilisée pour organiser les différents éléments du GUI.

Les éléments les plus hauts dans la hiérarchie sont dessinés en premier, donc les enfants sont dessinés après l'élément actuel. Cet ordre de rendu permet, par exemple, d'utiliser un fond quelconque contenant plusieurs éléments.



En effet il existe un élément racine qui contient l'ensemble du GUI. Cette élément racine est toujours de la même dimension que la fenêtre de rendu du GUI.

Pour généraliser le système permettant de positionner un élément du GUI quelqu'il soit (image, bouton, texte, etc.), la position de chaque élément est définie par un rectangle.

Voici la structure de données utilisée pour placer les éléments du GUI à l'écran tout en gardant l'aspect responsive de l'application :

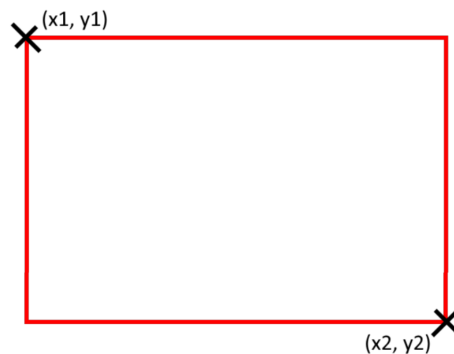
```
typedef struct gui_posval_s {
    float rel;
    float self;
    int16_t abs;
    uint8_t rel_ref;
    uint8_t abs_ref;
} gui_posval_t;

typedef struct gui_posdim_s {
    gui_posval_t val;
    gui_posval_t min;
    gui_posval_t max;
} gui_posdim_t;

typedef struct gui_pos_s {
    gui_posdim_t x1;
    gui_posdim_t y1;
    gui_posdim_t x2;
    gui_posdim_t y2;
} gui_pos_t;
```

Ici “gui\_pos\_t” est la structure principale qui contient toutes les informations définissant les positions d'un élément quelconque.

On peut remarquer qu'un rectangle est défini par la position de deux coins opposés, tel qu'illustré ci-dessous :



Chaque coordonnée sur son axe est définie par une structure de type "gui\_posdim\_t", définissant un minimum "min", maximum "max" et une valeur optimal "val". Chacune de ces valeurs est définie par une structure de type "gui\_posval\_t".

Cette structure contient plusieurs champs décrits ci-dessous :

- rel : Le coefficient à appliquer à la référence relative
- self : Un coefficient de la taille initiale/optimale de l'élément manipulé
- abs : Une valeur de décalage absolue (en pixels)
- rel\_ref : Le point de référence relative (ce par quoi la valeur du champ "rel" est multipliée)
- abs\_ref : La référence absolue (où se situe le point (0,0))

Voici les points de référence actuellement pensés ainsi que leurs descriptions :

Référence absolue :

|                      |  |
|----------------------|--|
| GUI_REF_ABS_UNSET    | Valeur par défaut et identique à GUI_REF_ABS_PARENT  |
| GUI_REF_ABS_PARENT   | (0,0) est placé sur le coin supérieur gauche du parent   |
| GUI_REF_ABS_PREV     | Un décalage de 0 par rapport à cette référence est placé sur le dernier le côté le plus proche de l'élément précédent  |
| GUI_REF_ABS_SELF     | Un décalage de 0 par rapport à cette référence est placé sur le côté opposé au côté courant  |
| GUI_REF_ABS_BASELINE | (0,0) est placé sur la ligne de texte de base du dernier caractère de l'élément précédent (si l'élément précédent n'a pas de texte alors il s'agit du coin inférieur droit de l'élément précédent) |
| GUI_REF_ABS_INLINE   | (0,0) est placé sur le coin inférieur droit du dernier caractère de l'élément précédent (si l'élément précédent n'a pas de texte alors il s'agit du coin inférieur droit de l'élément précédent)   |

Référence relative :

|                    |  |
|--------------------|--|
| GUI_REF_REL_UNSET  | Valeur par défaut et identique à GUI_REF_REL_PARENT  |
| GUI_REF_REL_PARENT | Utilise la largeur (pour la définition de x1 ou x2) ou hauteur (pour y1 ou t2) du parent comme multiplicateur du champ "rel"                   |
| GUI_REF_REL_PREV   | Utilise la largeur (pour la définition de x1 ou x2) ou hauteur (pour y1 ou t2) de l'élément précédent comme multiplicateur du champ "rel"      |
| GUI_REF_REL_SELF   | Utilise la largeur (pour la définition de x1 ou x2) ou hauteur (pour y1 ou t2) initial/optimal de lui-même comme multiplicateur du champ "rel" |
| GUI_REF_REL_RATIO  | Utilise la hauteur (pour la définition de x1 ou x2) ou largeur (pour y1 ou t2) de lui-même comme multiplicateur du champ "rel"                 |

Donc la valeur finale d'un "gui\_dimval\_t" est calculée comme suit :  $rel * reference\_relative + abs + self * sa\_propre\_largeur\_ou\_longueur\_initial + reference\_absolue$  Ensuite la valeur d'un "gui\_dimpos\_t" est calculée en bornant la valeur calculée de "val" entre celles de "min" et "max". Puis finalement avec nos 4 "gui\_dimpos\_t" l'ont option les coordonnées de 2 coins du rectangle définissant la position de notre élément.

L'on peut remarquer ici que "GUI\_REF\_REL\_RATIO" poserait un problème s'il était utilisé pour définir à la fois, la largeur et la hauteur d'un élément, car dans ce cas chaque côté dépendrait de l'autre. Pour remédier à ce problème, il suffit de calculer en deux étapes, d'abord calculer x1, x2, y1, y2 en considérant le champ "rel" à 0 ensuite ajouter à chaque valeur "rel \* largeur\_du\_coter\_adjacent".

L'on peut cependant remarquer que ce système offre une très grande flexibilité et un bon moyen de faire un GUI très facilement adaptable à toute dimension d'écran.

## Conclusion

Finalement, le projet a beaucoup avancé. Nous avons mis en place beaucoup de procédures pour nous faciliter le développement, notamment par un Makefile complet pour build tout le projet ou seulement certaines parties et qui permet aussi de générer des tests et tout cela est paramétrable avec des variables d'environnement. Nous avons aussi une documentation sur le projet pour nous aider. Nous utilisons le Data Set du MNIST et des fonctions sont en développement pour améliorer ce Data Set.

Les fonctions de base ont été implémentées comme prévu, niveau de gris, rotation, conversion noir/blanc, détection... Une partie du GUI a déjà été codée, mais l'implémentation se fera ultérieurement. Notre démonstration du réseau neuronal basé sur l'apprentissage du XOR est fonctionnel. Notre projet avance dans les temps et résolvant les problèmes un par un. Chaque fonction a été pensée et développée pour être la plus optimisée et répondre au mieux au problème et à nos besoins.

Nous sommes en avance sur sa réalisation et il sera complet dans les temps. Nous avons pour le moment réussi à passer tous les problèmes rencontrés, mais il nous reste encore du travail à réaliser pour finir le projet.