# Rapport de première soutenance

Vladimir Meshcheriakov Eva Blum Yves-Antoine Gangner
Vincent Massard

26 Octobre 2021


**Les SpeeedyCuber présente**

**l'OCR**

# Table des matières

# 1 Introduction

## 1.1 Qu'est qu'un OCR?

Un OCR (Optical Character Recognition) est un logiciel qui a pour but de traduire une image scannée en un fichier de texte. Autrement dit, le contenu de l'image doit pouvoir être utilisable, ce qui n'est pas le cas sur un fichier image. Ainsi son fonctionnement se déroule en plusieurs étapes distinctes. dans un premier temps, le prétraitement de l'image. Cette étape est essentielle au bon déroulement du reste du programme car elle permet d'obtenir une image 'parfaite'.
Dans un second temps, la segmentation. Elle a pour but d'isoler dans l'image les lignes de la grille et les caractères à l'intérieur des lignes. Enfin, la reconnaissance des caractères. Cette étape est de loin la plus ardue car elle nécessite l'utilisation d 'un réseau de neurones qui servira d'identificateur de caractères mais aussi apprendre de ses erreurs pour devenir plus performant. Ici, notre sujet porte sur le résolution d'un sudoku de dimension 9x9.

# 2 Présentation du groupe

## 2.1 Vladimir Meshcheriakov

Le C est un langage super important pour n'importe quel informaticien. Je suis donc ravi de faire partie de mon groupe cette année, pour améliorer mes connaisances dans ce domaine. L'OCR est un challenge qui vaut les nuits sans sommeils pour parvenir à un resultat attendu.

## 2.2 Eva Blum

Dès mon plus jeune âge, j'ai toujours été passionnée par le monde de l'informatique qui est si vaste et sans limites. Après une année passée à EPITA, mener des projets de bout à bout est très satisfaisant et nous permet de grandement progresser. L'OCR est un projet qui m'intéresse beaucoup car il comprends différentes parties comme le réseau de neurones et le traitement de l'image. Ces parties m'intriguaient beaucoup depuis ma rentrée à EPITA donc je suis très heureuse de faire ce projet accompagnée de mes camarades.

### 2.3 Yves-Antoine Gangner

Je suis passionné d'informatique et adore principalement le python, mais je suis tout autant intéressé par le C et souhaite en apprendre d'avantage sur ce language comme sur les IA. Le projet est une occassion pour moi de travailler et solidifier des éléments tels que la cohésion d'équipe, l'organisation de groupe ou bien ce que peut impliquer un projet sur une période de temps défini.

### 2.4 Vincent Massard

Depuis quelques années, le milieu de l'informatique m'attire et les différentes technologies qui évoluent au cours du temps me fascinent au plus haut point. Cet intérêt ne fait que grandir à force d'expérimentation, d'heure de code, de perte de sommeil et bien entendu d'échec cuissant, sans pour autant abandonné, je trouve toujours intéressant et pratique de réellement se documenter pour un projet et réutiliser ces compétences dans d'autres contextes. J'attends entre autre de l'OCR se même engouement.

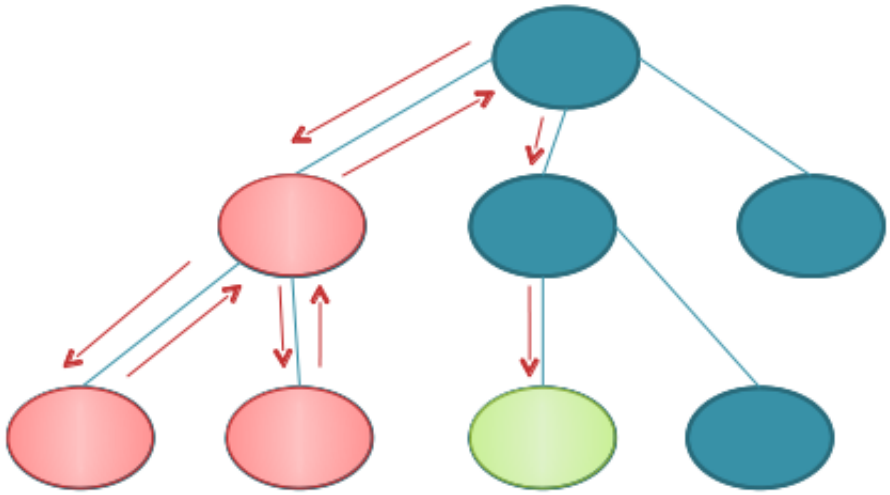
### 2.5 Organisation

| Tâches | Eva | Vladimir | Vincent | Yves-Antoine |
|---|---|---|---|---|
| Pré-traitment | x | x | | |
| Segmentation | | x | | |
| Solver du sudoku | | | x | |
| Interface utilisateur | | | x | |
| XOR | | | | x |
| Réseau de neurones | | | | x |
| Rotation de l'image | x | | | |
| Découpage de l'image | | x | | |

# 3 Résolution du Sudoku

## 3.1 Algorithme du solver

Comme précédemment explicité, notre projet consiste à la lecture d'une image, la détection d'un grille de sudoku, ses différentes cases (vides ou non), et bien évidemment sa résolution. De manière simple, nous avons opté pour une méthode récursive (ou de backtracking) ; Le backtracking est une forme de parcours en profondeur d'un arbre avec des contraintes sur les noeuds. L'idée est de partir du noeud parent, descendre dans le premier noeud fils satisfaisant la contrainte. Ce noeud fils devient alors un noeud parent et l'on parcourt ensuite ses noeuds fils sous le même principe. Lorsque l'on a parcouru tous les noeuds fils d'un noeud et qu'aucun ne satisfait la contrainte, on remonte alors au noeud parent et on descend dans le noeud fils suivant. Si l'on arrive au dernier fils du premier noeud parent et qu'il ne satisfait pas la contrainte alors il n'existe pas de solution. La solution est identifiée lorsque l'on arrive à un noeud qui satisfait la contrainte et qui n'a pas de noeud fils.



Ainsi on peut se concentrer sur les éléments importants de l'algorithme de résolution. On classe les cases de celles ayant le moins de possibilités à celles en ayant le plus.
On place ce classement dans une liste. On parcours la liste jusqu'à arriver à la derniere cellule de la liste.
Pour chaque cellule de la liste:
- On teste les valeurs de 1 à $n^2$:
- si la valeur est possible :

- on l'inscrit dans la cellule et on passe à la suivante
- sinon :
- on remontre à la cellule suivante et on reprend le test des valeurs de 1 à $n^2$ à partir de la valeur déjà inscrite dans la cellule.
- dans le cas où aucune valeur correspondante n'est trouvé, on considère le sudoku insoluble.

# 4 Prétraitement de l'image

## 4.1 Chargement de l'image

Notre programme se basant sur une ou plusieurs images en entrée. La première chose à faire était donc de réussir à charger une image afin de l'exploiter. Pour cela, nous exploitons la librairie SDL. En utilisant SDL, nous avons accès à chaque pixel d'une image sous la forme d'un tableau à deux dimensions. Par exemple la fonction SDL_SaveBMP nous permet de sauvegarder une image sous le format de BMP pour ensuite manipuler nos images. La justification de l'utilisation de ce format, est assez simple: nous avons voulu simplifier la manipulation d'une image et BMP est un parfait format pour débuter.

## 4.2 Nuance de gris

Avant tout traitement, il faut transformer l'image en noir et blanc ce qui nous permettra de récupérer plus facilement les caractères.
Mais dans un premier temps, il est nécessaire de passer en niveau de gris. Pour cela, nous nous sommes basés sur ce que nous avons vu en TP en début d'année. Pour rappel on calcule la luminosité d'un pixel grâce à la fonction suivante :

$$\text{Luminosité} = 0.3*r + 0.59*g + 0.11*b$$

Grâce au fait que les pixels de notre image sont ramenés d'un triplé de 8 bits vers une unique valeur de 8 bits, on réduit suffisamment l'information de l'image pour ne pas allourdir les calculs.

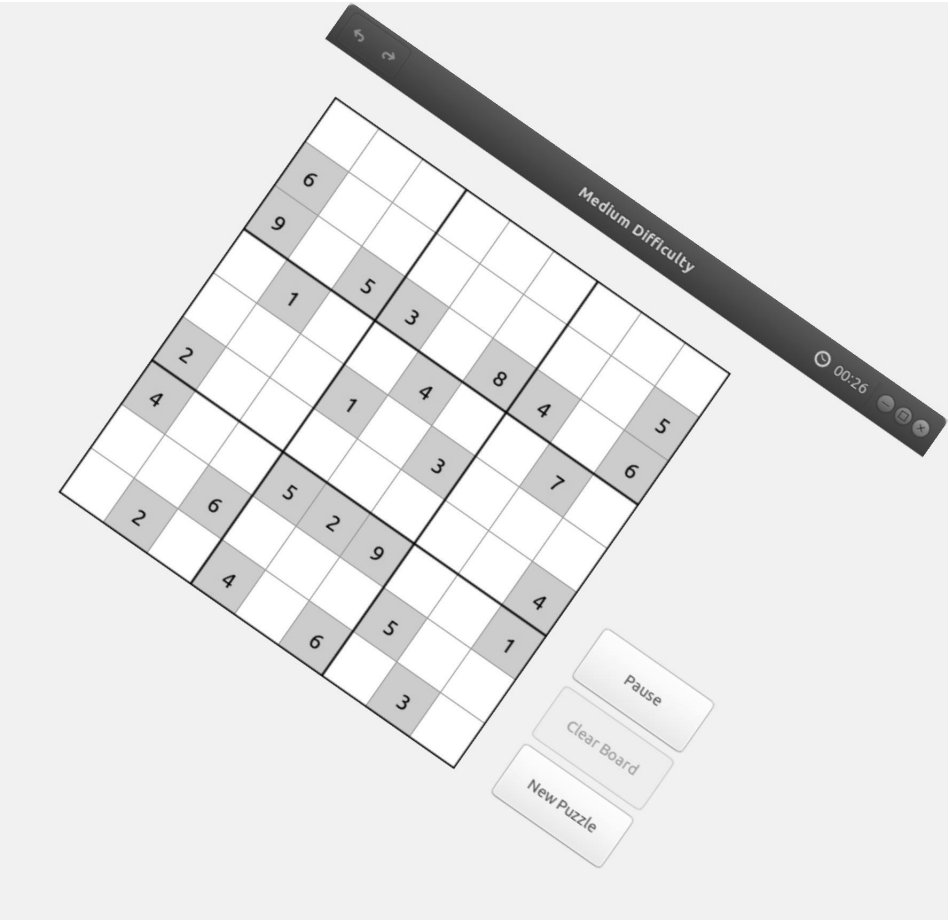



Figure 1: Démonstration du greyscale

### 4.3 Binarisation

Il existe un nombre conséquent d'algorithmes de binarisation. Mais nous avous essayé d'implémenter ceux qui nous ont semblé les plus efficaces et optimisés. Par exemple, la binarisation de Otsu qui parcourt l'image et établit un histogramme, dans lequel il va chercher un seuil. Celui-ci fera la distinction claire entre l'arrière plan de l'image et son premier plan. Ce dernier a donc pour but de souligner davantage les objets exposés en premier plan.
Néanmoins Otsu ne donne pas toujours de très bons resultats. On fait donc recours à une autre technique qui est un seuil adaptable à chaque image. Il donne donc généralement de meilleurs résultats que Otsu mais est plus coûteux en compilation. L'algorithme parcourt des zones de l'image et adapte un seuil à chaque zone.

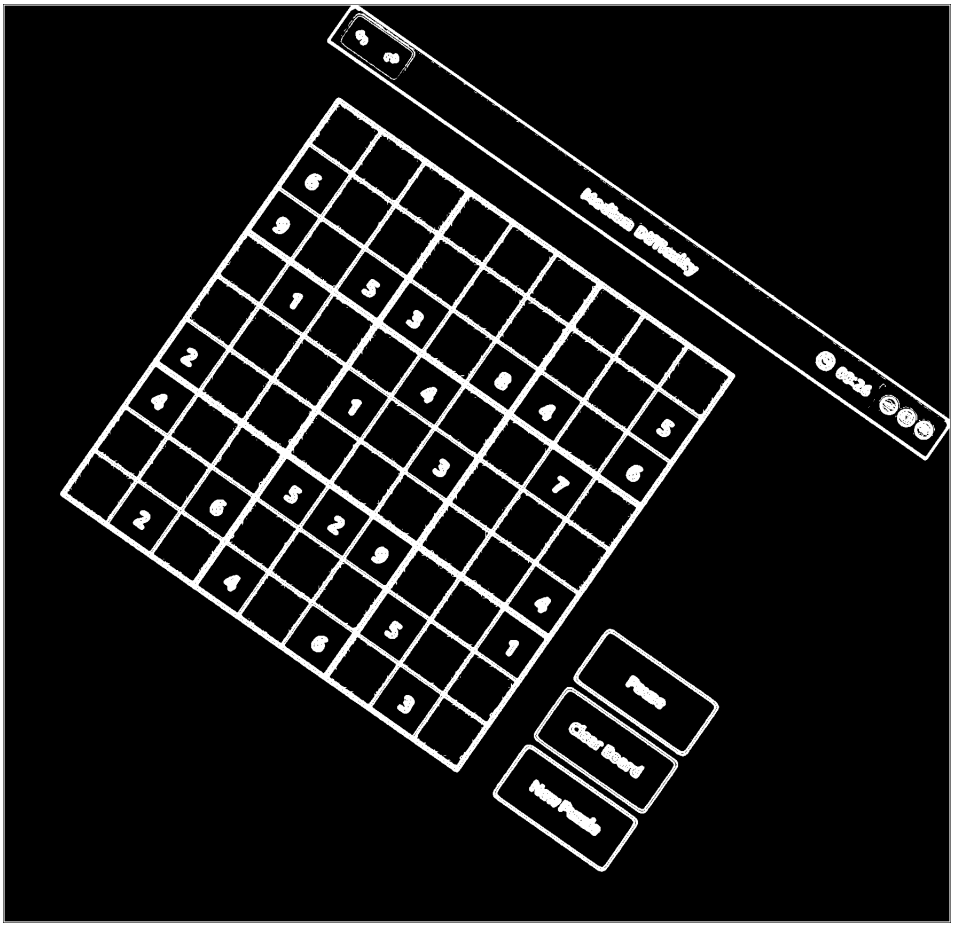


Figure 2:

### 4.4 Élimination du bruit

Le bruit d'une image est le nom donné a la présence d'informations parasites qui viennent s'ajouter de façon aléatoire aux détails d'une image numérique. La principale conséquence de cela est la perte de netteté de l'image dû a l'apparition de pixel parasite nommé artefact.
Comme tout logiciel permettant un traitement d'image, nous nous sommes rendu compte de la nécessité d'utiliser des filtres. Nous avons, dans un premier temps, concentré nos efforts sur un filtre permettant d'éliminer les pixels parasites (créés par la binarisation et la rotation) puis nous avons également créé un filtre de contraste pour contrebalancer le flou créé par le filtre précédent.
Ainsi après avoir effectué des recherches nous avons trouvé des méthodes différentes pour éliminer le bruit: le filtre médian, le gaussien, le sobel, la méthode d'Otsu et le treshold. Bien évidement ces différents filtres ont pour but d'éliminer les artefacts. Nous avons donc décidé d'implémenter ces types de filtres pour pouvoir choisir le plus adapté.

**Filtre Gaussien :**
La première méthode, consiste à appliquer une matrice de convolution 3 * 3, sur chaque pixel de l'image. Chaque composante de couleur du pixel est modifiée en fonction du filtre utilisé et des pixels environnants. Cette méthode se base sur le fait que les pixels d'une image sont en interaction les uns avec les autres. On applique donc la matrice sur le pixel, en sommant les produits des composantes colorées des pixels voisins avec la valeur de la matrice correspondante. On divise ensuite le résultat par la somme des valeurs de la matrice. Voici la matrice de convolution généralement utilisé pour ce filtre :

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

**Filtre Médian :**
Cette méthode consiste à appliquer une matrice de convolution 3 * 3, sur chaque pixel de l'image, comme le précédent. A la différence qu'on utilise la matrice suivante :

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

On remplace ensuite la valeur du pixel par la valeur médiane des pixels voisins, après application du masque. Le principe de ce modèle est le suivant : On parcourt l'image pixel par pixel, en appliquant le masque sur chaque pixel et sur ses voisins. On applique la formule suivante sur chacun des pixels voisins : luminosité = 0.3 * rouge + 0.59 * vert + 0.11 * bleu. Une fois le masque parcouru, on classe les valeurs du tableau, et on prend la valeur médiane. On remplace ensuite les composantes du pixel par celles du pixel correspondant.

**Filtre de Sobel :**
Nous utilisons également le filtre de sobel qui est un calcul du gradient de l'intensité de chaque pixel pour indiquer les variations du clair au sombre pour la détection des contours.

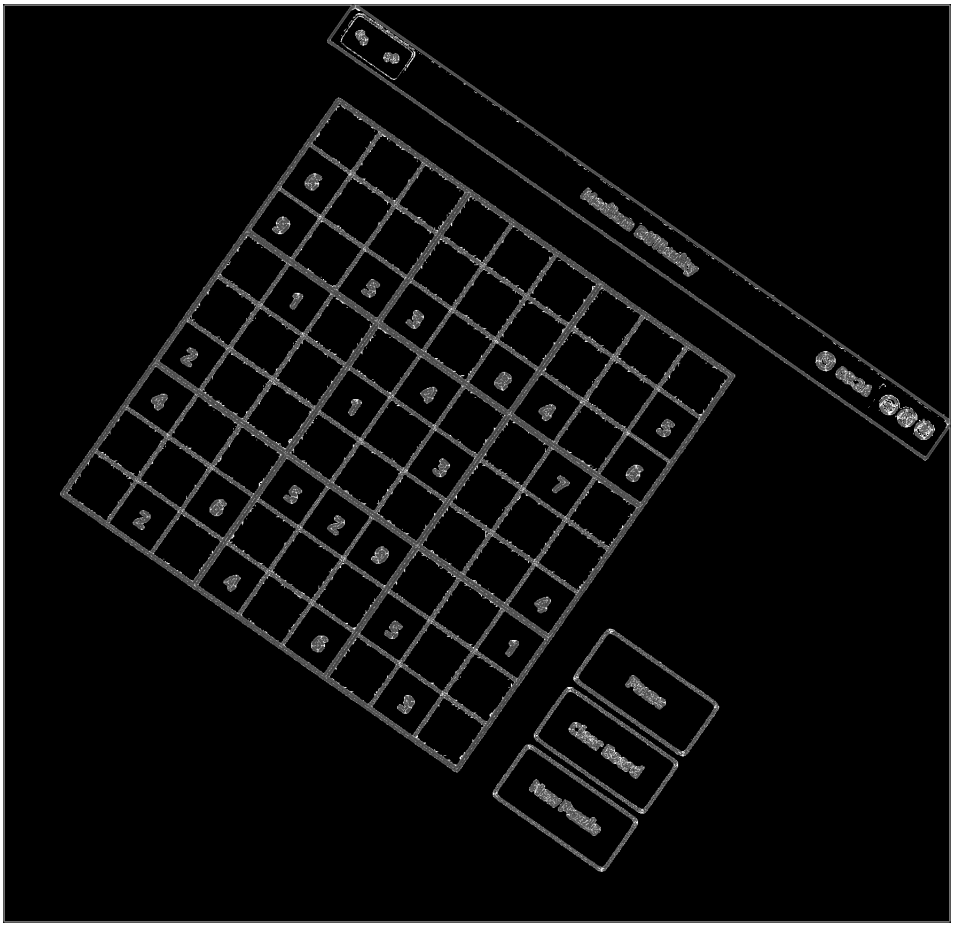


Figure 3: Démonstration du sobel

# 5 Segmentation et rotation

## 5.1 Préparation de le segmentation

Pour cette partie, le Sobel ne suffit point. If faut donc ajouter un autre type de filtre convolutionnel comme par exemple le Canny suivi d'un Hysteris. Cela va nous permettre d'obtenir une image dont la grille de sudoku sera plus épaisse et donc plus facile pour l'algorithme de la détecter.

## 5.2 Détection de lignes et colonnes

Cela représente la plus difficile de la partie de la rotation de l'image. Pour déterminer l'angle de la rotation, nous avons utilisé le procédé de la transormée de Hough. C'est une technique qui permet de reconnaître des formes dans le traitement d'images numériques. Ici, nous l'utiliserons pour détecter les lignes et colonnes de notre sudoku.

### 5.3 Détection de l'angle

A l'aide de l'algorithme de Hough, nous pouvons néanmoins connaitre l'angle theta qui apparaît le plus souvent sur un image. Nous en déduisons aisément que cette occurence fréquente d'un angle theta permet de distinguer d'autres angles. Donc nous en connaissons l'angle de la rotation de notre sudoku.

### 5.4 Rotation de l'image

Cet angle est donc simplement utilisé par notre fonction de rotation qui va aligner naturellement notre grille. Cette dernière étant allignée, elle nous donne la possibilité de la segmenter.

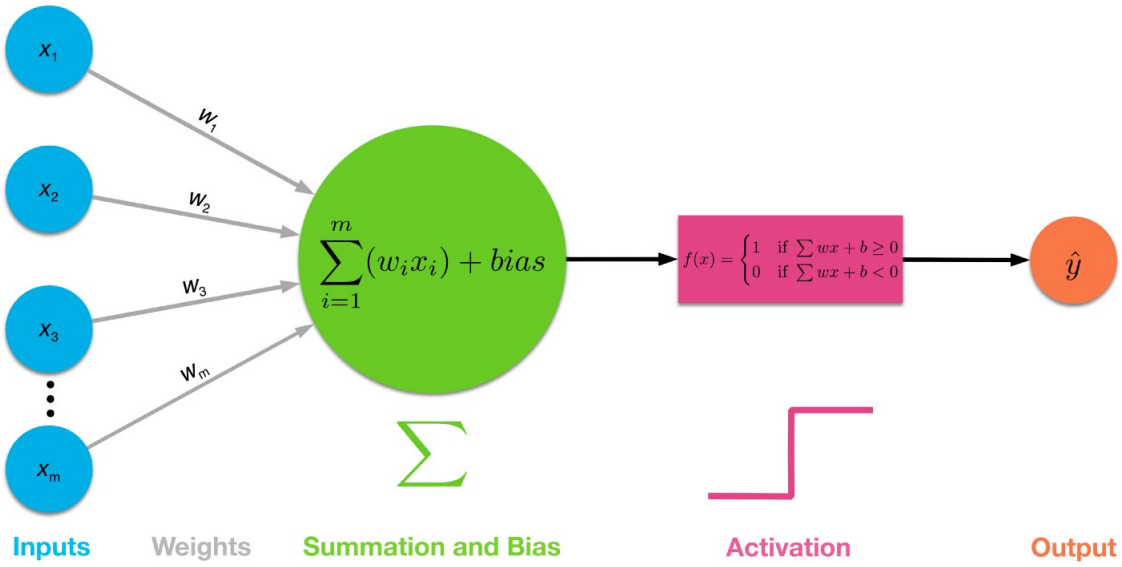
### 5.5 Segmentation

Nous avons développer un algorithme qui nous segmente notre grille. Son concept, même intrinsequement mathématique, reste simple à utiliser et à comprendre. Une image est parcourue verticalement et horizontalement. Des marqueurs de couleurs sont mis dessus tout en repérant les occurences les plus répandues de séquences grille/non-grille. Or comme les cases à segmenter sont des cubes, il est simple de comprendre que toutes les cases carrées seront detectées et segmentées.

# 6 Réseau de neurones

## 6.1 Qu'est ce que c'est?

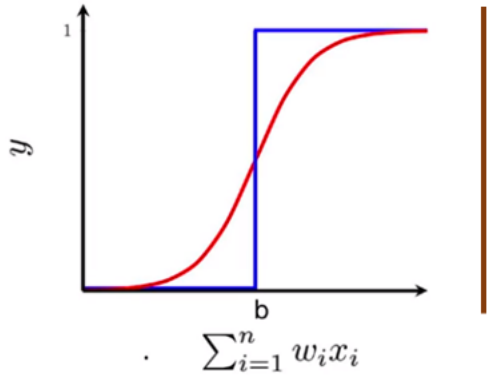
Un réseau neuronal est un ensemble de neurone, et dans le contexte une modélisation mathématique d'un neurone biologique, prenant plusieurs entrées mais ne renvoie qu'une sortie. Un neurone est entre autre relié par un synapse, le noyau recevant les informations des entrées x1, x2, ..., xn. Pour chacune de ces entrées, on va leur attribuer un poids synaptique (w1, w2, ..., wn) puis effectue une somme de ces poids. Si le nombre dépasse un certain seuil, l'information est transmise à la couche suivante, une fonction d'activation. Si le poids est inférieur au seuil, l'information n'est pas transmise.




## 6.2 Implémentation

L'implémentation la plus adaptée à l'exercice est l'implémentation de neurones sous forme de perceptron / neurone sigmoïde. Le modèle perceptron prend plusieurs entrées à valeur réelle et donne une seule sortie binaire. Dans le modèle de perceptron, chaque entrée xi est associée à un poids wi. Les poids indiquent l'importance de l'entrée dans le processus de décision. La sortie du modèle est décidée par un seuil Wo si la somme pondérée des entrées est supérieure au seuil Wo la sortie sera 1 sinon la sortie sera 0. En d'autres termes, le modèle se déclenche si la somme pondérée est supérieure au seuil. D'après la représentation mathématique, on peut dire que la logique de seuillage utilisée par le perceptron est très sévère.
Nous introduisons les neurones sigmoïdes dont la fonction de sortie est beaucoup plus lisse que la fonction en escalier. Dans le neurone sigmoïde,

une petite variation de l'entrée ne provoque qu'une petite variation de la sortie, contrairement à la sortie en escalier. Il existe de nombreuses fonctions ayant la caractéristique d'une courbe en forme de "S", appelées fonctions sigmoïdes. La fonction la plus couramment utilisée est la fonction logistique.




$$y = \frac{1}{1+e^{-(w^T x+b)}}$$

Nous ne voyons plus de transition nette au seuil b. La sortie du neurone sigmoïde n'est pas 0 ou 1. Il s'agit plutôt d'une valeur réelle comprise entre 0 et 1 qui peut être interprétée comme une probabilité.

### 6.3 Apprentissage

Le but de l'apprentissage est de permettre au réseau de neurone de s'améliorer. Cela consiste simplement à modifier les poids de chaque connexion, mais cela nécessite plusieurs étapes pour y parvenir à obtenir les nouvelles valeurs de poids. Plus on répète de fois cette opération, plus les poids des connexions devraient optimaux, en fait les poids que l'on va obtenir vont chacun converger vers leur valeur optimale. L'algorithme utilisé pour entrainer le réseau est l'Algorithme du gradient stochastique. Çela consiste à calculer la différence entre le résultat donné par le réseau de neurones et le résultat attendu. On remonte ensuite les étapes de calculs avec la dérivée de la fonction sigmoid et avec les résultats obtenus on règle les valeurs de neurones pour diminuer l'erreur jusqu'à obtenir le meilleur résultat possible.

### 6.4 Réseau de neurone: XOR

Nous avons réalisé un réseau de neurones qui apprend la porte XOR, le réseau que nous utilisons est un 2-1-1 avec connexions entrées-sorties c'est-à-dire qu'il existe deux neurones à la couche d'entrée et sont reliés directement à la sortie, il a une seule couche cachée contenant un neurone, et enfin un neurone en couche de sortie.
On utilise aussi deux neurones de biais (un connecté au neurone caché et un connecté à la sortie) pour que l'apprentissage fonctionne mieux. On peut observer que sur les premiers tests le réseau peut se tromper, mais plus on a effectué la rétro-propagation, plus les poids seront juste et on aura une vrai porte XOR.

# 7 Ce qu'il reste à faire

## 7.1 Interface graphique

En préparation, nous avons commencé à nous intéresser à Toolkit GTK afin que notre interface corresponde aux spécifications demandées. Toolkit GTK est un widget multi plateformes pour créer des interfaces graphiques. De plus, l'utilisation de GTK Glade nous permettra également de faire une mise en forme de l'interface. A l'heure, un début est déjà implémenté en interne sur une ouverture de fenêtre avec un serveur host pour que par la suite nous puissions introduire l'image de sudoku et effectuer chaque tâche.

## 7.2 Apprentissage renforcée et complet

Pour le réseau neuronal, nous nous concentrerons sur son apprentissage renforcée afin de le rendre le plus performant possible et complet. Nous allons donc tester sur un grand nombre de valeur de case notre réseau afin qu'il puisse avoir un cycle d'apprentissage plus rapide sans forcément reprendre de zéro son entraînement avec de nouvelles images.

## 7.3 Preprocessing

Le preprocessing correspond en grande partie à l'amélioration de nos premières étapes de pré-traitement de l'image donnée en entrée.
• Amélioration du redressement automatique de l'image;
• Amélioration de l'élimination des bruits parasites (grain de l'image, tâches, etc.) ;
• Renforcement des contrastes.

## 7.4 Sauvegarde d'image

Cette partie consistera à la récupération d'une image résultante du preprocessing, passant par une reconstruction de la grille puis de la résolution du sudoku, un affichage de la grille final et pour finir une sauvegarde de notre image résolu.

# 8 Conclusion

Cette première partie de notre projet a été riche en expérience, en effet la chronologie des cours et du suivi du projet nous ont pris au dépourvu mais nous avons tenu à rattraper le train en direction d'une soutenance parfaite. Même si elles peuvent encore être améliorées, nos fonctions de prétraitement de l'image sont bien avancées, une segmentation correcte et un réseau neuronal efficace sous Porte XOR. Ce projet nous a appris beaucoup sur le langage C mais aussi sur les réseaux de neurones ou bien encore sur les algorithmes de détection de caractères.
En ce qui concerne la seconde soutenance, nous comptons mettre les bouchées doubles pour avancer sur les éléments à terminer et ainsi vous présenter un OCR parfaitement opérationnel.