



OCR (OPTICAL CHARACTER RECOGNITION)
BY OBSTACLE COURSE RACING

Rapport De Soutenance 2

Participants :

Adam RILLI
Mathis
RABOUILLE
Hugo RATTE
Léo DEVIN

Superviseur

Christophe "Krisboul" BOULLAY
Adjoint à la directrice des études et
responsable Informatique - Classes
préparatoires de l'EPITA

8 Décembre 2021

Sommaire

1	Introduction	3
1.1	Objectif du projet	3
1.2	Méthode de résolution du projet	3
1.3	Déroulement du projet	3
2	Présentation des membres	4
2.1	Adam	4
2.2	Mathis	4
2.3	Hugo	5
2.4	Léo	5
3	Répartition des taches	6
3.1	Pré-traitement	6
3.1.1	Grayscale	6
3.1.2	Gamma	7
3.1.3	BlackScale	7
3.1.4	Sobel	9
3.1.5	Hough	9
3.1.6	Rotation	11
3.1.7	Inversion Couleur	11
3.1.8	Réduction de Bruit	12
3.2	Détection de la grille	13
3.3	Réseau Neuronal	16
3.3.1	X-OR	16
3.3.2	Apprentissage	17
3.3.3	Reconnaissance de caractère	20
3.4	Résolution de la grille	22
3.4.1	Première soutenance	22
3.4.2	Deuxième soutenance	23
3.5	Interface Graphique	23
3.6	Organisation du projet	26
3.6.1	Première soutenance	26
3.6.2	Soutenance final	27
4	Site Internet	27
4.1	Structure du site	27
5	Difficulté rencontrées	29
5.1	Pré-traitement de l'image	29
5.1.1	Soutenance 1	29
5.1.2	Soutenance 2	29
5.2	Découpage de l'image	31
5.2.1	Soutenance 1	31
5.2.2	Soutenance 2	31
5.3	Réseau neuronal	32
5.3.1	Soutenance 1	32
5.3.2	Soutenance 2	32
5.4	Résolution de la grille	33
5.4.1	Soutenance 1	33
5.4.2	Soutenance 2	33

6	Évolution du projet	34
6.1	Première Soutenance	34
6.1.1	Prévisions données lors de la première soutenance	34
6.1.2	Tableau d'avancement	34
6.1.3	Avancement final	35
6.1.4	Recap des éléments présents dans l'application	35
6.2	Représentation graphique du projet	36
7	Expérience personnelle	36
7.0.1	RILI Adam	36
7.0.2	DEVIN Léo	38
7.0.3	RATTE Hugo	38
7.0.4	RABOUILLE Mathis	39
8	Annexes	40
8.1	Bibliographie et références	40

1 Introduction

1.1 Objectif du projet

L'objectif de ce projet est de réaliser un logiciel de type OCR (Optical Character Recognition/Reconnaissance Optique de Caractères) qui résout une grille de Sudoku, en reconnaissant les chiffres en autonomie. L'application prend en entrée une image représentant une grille de Sudoku et affichera en sortie la grille résolue sous forme d'image.

Dans sa version définitive, pour la dernière soutenance, notre application proposera une interface graphique permettant de charger une image dans un format standard, de la visualiser, de corriger certains de ses défauts et enfin d'afficher la grille complètement remplie et résolue. La grille résolue sera également sauvegardée. Notre application présentera un aspect d'apprentissage, qui sera séparé de la partie principale, et qui permettra d'entraîner notre réseau de neurones, puis de sauvegarder et de recharger le résultat de cet apprentissage.

1.2 Méthode de résolution du projet

Pour la première soutenance, nous avons découpé le projet en 4 parties :

→ **Pré-traitement de l'image :**

Dans cette première partie nous allons aborder toutes les opérations nécessaires au traitement de l'image. Donc la gestion des couleurs, des impuretés de l'image et de son orientation.

→ **Segmentation de l'image :**

Ici nous parlons de la découpe de l'image en plus petites images. Dans ce cas précis chaque images contiendra une case de la grille c'est à dire un chiffre. Il faut donc passer par la détection de la grille sur l'image puis par la découpe à la bonne taille de chaque petit carré pour les stocker respectivement dans une nouvelle image.

→ **Résolution de la grille :**

Cette troisième partie gère la résolution de la grille et l'affichage de cette dernière. L'algorithme proposera son efficacité face à des grilles de complexités variant du plus simple au plus exigeante.

→ **Réseau Neuronal :**

Pour finir le réseau neuronal aura à l'aide d'une fonction X-OR pour but de permettre à notre programme de pallier à différente erreur de reconnaissance d'un chiffre sur une image. Et ainsi d'avoir une probabilité d'erreur la plus infinitésimale possible.

1.3 Déroulement du projet

Ce projet OCR (Optical Character Recognition)(Reconnaissance optimiser de caractère) a été une expérience à la fois intéressante par la diversité des éléments à implémenter et à découvrir.

2 Présentation des membres

Notre formation de groupe a été assez perturber par les différents changements de classe apparus au début de l'année. Trois de nous, Mathis, Hugo et Léo, étions en C2 et avions former un groupe. Nous avons été ensuite change en B2 où nous avons propose à Adam de se joindre a nous.

2.1 Adam

Première soutenance J'ai été institué du rôle de chef de projet au sein de ce groupe. Pour ce début de projet, les fonction de mon rôle de membre du groupe se sont principalement tourner vers la résolution de la grille et de le gestion des fichiers a transmettre. Dans le projet de S2 mon rôle se tournais vers la même gestion des fichier mais se rajoutait l'intégration d'une Intelligence Artificielle. Je voulais pour ma part explorer d'avantages de possibilités. J'ai donc pris la décision, de porter ma méthode de travail, sur la concentration unilatéral de mon code en respectant l'application d'un seul principe a la fois. Mon travail en tant que membre du groupe achevé; j'ai endosser le poste de chef de projet. Celui-ci implique la relecture des codes de chaque membres de mon groupe, l'organisation de réunion hebdomadaire afin d'identifier de de résoudre tout les problèmes occurrents et ce sur le fondement de l'entraide. Nonobstant les réflexions sur les méthodes de nécessaire dans le but de créer un lien futur entre les différentes partie de notre projet.

Deuxième soutenance Mon expérience en tant que participant a ce projet est positive. Le projet était intéressant et m'a permis d'être plus a l'aise sur différents aspect de réflexions ou d'implantation. Notamment sur un nouveau langage qu'est le C. Contrairement au autre langages de programmation déjà étudié le C a un fonctionnement bien particulier surtout dans ce projet.

2.2 Mathis

Lors de mon travail j'ai eu quelques problèmes au début car je n'étais pas habitué au langage C mais après quelques semaines tout était réglé. Le plus gros du travail a été de découvrir comment bien compiler SDL et ses dépendances, comment les utiliser sans créer d'erreurs et surtout comment les utiliser efficacement. Il y'a eu une longue phase de réflexion sur la façon dont nous allions implémenter ce qui nous était demandé, pour ma part cela a duré environ une semaine. Pour finir le travail des semaine précédentes nous avons également tous du passer pour une phase de débogage qui nous a pris quasiment autant de temps que l'implémentations. Pour moi ce projet est l'occasion de rentrer directement dans le vif du sujet dans un semestre ou nous ne faisons quasiment que du C. Bien qu'il soit un peu dur a appréhender au départ c'est un projet que je trouve assez enrichissant, il nous force a sortir de notre zone de confort. Une fois habitué au nouveaux outils que nous utilisons je pense que le deuxième bénéfice que peut avoir ce projet est de nous apprendre a travailler a plusieurs sur le même projet. Cette expérience nous sera forcément utile au cours des prochaines années ou nous devrons nous servir de ces compétences.

2.3 Hugo

Lors de la réalisation du projet, nous avons rapidement fixé le fait que je fasse la partie découpage des images pour la première soutenance. Tout d'abord, j'ai remarqué petit à petit que le projet était plus complexe que je le pensais. Vivant peut-être dans un monde de bisounours, je me suis dis que le projet allait être facile au départ, et relativement rapide. Puis, j'ai remarqué, les problèmes de documentations du langage C. Malgré tout, la première partie du projet m'a fortement plu puisque en effet, cela m'a aidé à travailler sur le langage C, puisque peu de TP ont eu lieu jusque-là. De plus, j'aimerais ajouter que de façon générale, travailler en groupe apporte rigueur et expérience pour la suite de mes projets personnelles ou professionnelles, que ce soit en durant le cycle ingénieur ou alors pour mon futur. Mes attentes pour la suite sont d'apprendre davantage de choses concernant la programmation générale et aussi et plus particulièrement sur le langage C.

Ensuite, concernant la deuxième partie, celle-ci m'a davantage plu. En effet, malgré de nombreuses heures passées à l'école où j'essayais de résoudre mes problèmes, j'ai eu la satisfaction du rendu du projet.

2.4 Léo

Première soutenance J'ai pu beaucoup apprendre lors de mes précédents projets notamment lors du projet s2 où nous devions former un groupe soude et productif afin de finalisé le projet. Mon rôle dans le projet a été de me pencher sur la partie réseau neuronal, ce qui m'a vraiment fait peur car c'est une partie assez technique. Le projet m'intéresse beaucoup, il reflète bien la difficulté qui monte d'un cran au s3.

Seconde Soutenance Lors de cette seconde partie de projet, le travail fut bien plus intense malgré qu'il l'ai été auparavant. Nous avons du bien plus nous coordonner sur le projet vu qu'il fallait tout relier et ce n'était pas une mince à faire. En effet plusieurs problème nous ont barré la route

3 Répartition des taches

Voici maintenant l'explication de fonctionnement précis de notre programme dans l'ordre logique de son exécution.

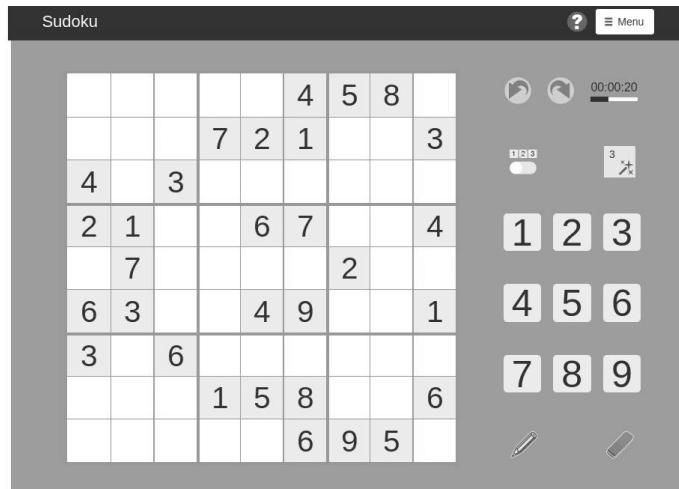
D'abord le pré-traitement

3.1 Pré-traitement

Lors du pré-traitement le programme va s'occuper de récupérer l'image dans l'endroit où on l'a enregistrer à partir d'un argument donné à l'exécution du programme. On vas superposer plusieurs filtre afin de supprimer un maximum d'impuretés et permettre une visualisation de l'image la plus efficace pour le traitement qui vas lui s'occuper du découpage de l'image.

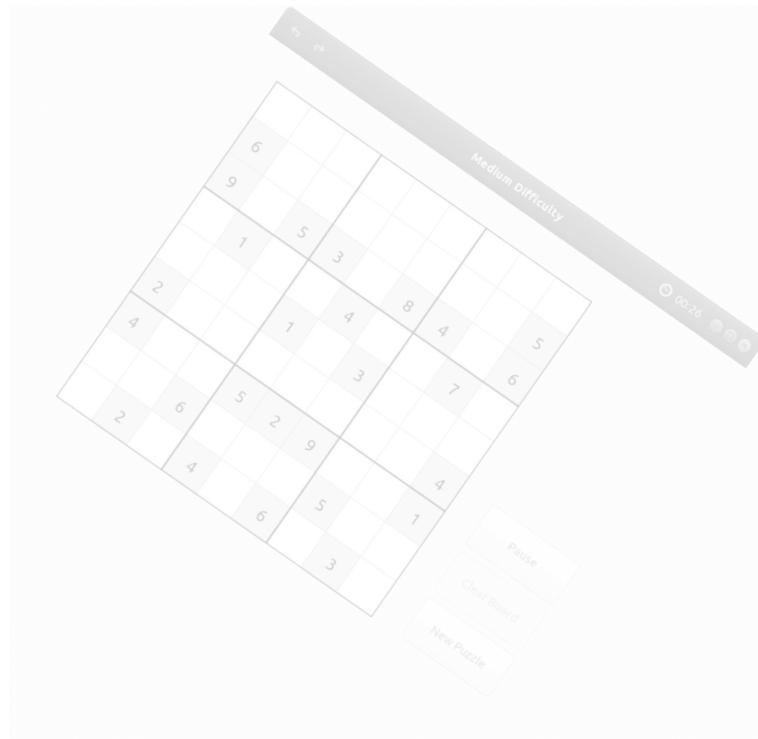
3.1.1 Grayscale

La fonction Grayscale consiste simplement en la récupération des constantes R,G,B du pixel traité puis les remplacer par la valeur average = $0.3*R + 0.59*G + 0.11*B$. La fonction Blackscale quant à elle sert à faire en sorte que tous les pixels deviennent soit noirs soit blanc ce qui est indispensable pour la reconnaissance de la grille est des caractères. Pour ce faire la fonction fait la moyenne des 3 composantes du pixel traité et si cette moyenne est supérieure à 127 alors le pixel est changé en blanc sinon il est changé en noir. On applique donc tout d'abord un "grayscale" qui vas nous permettre de différencier les pixels de couleurs ou noir de ceux en blanc. Voici ci dessous un exemple de l'application de cette fonction sur une image qui comportait des couleurs.



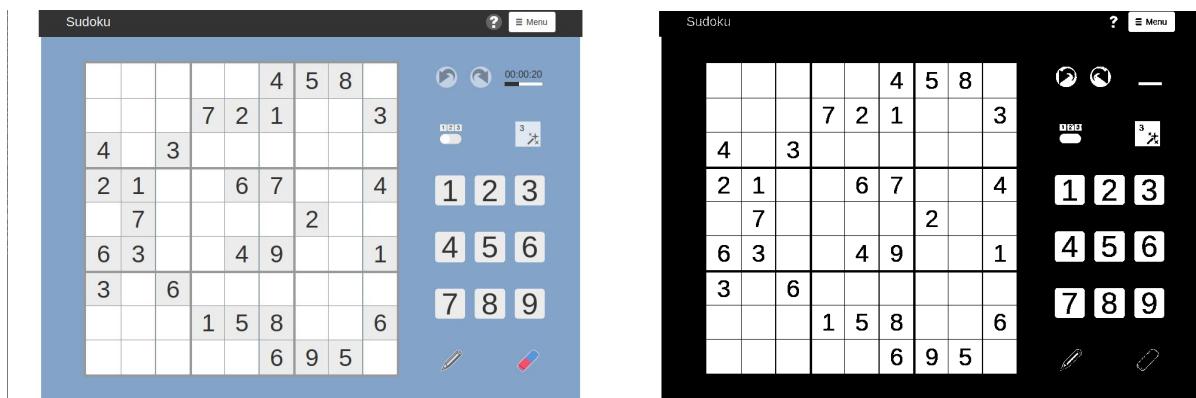
3.1.2 Gamma

Ensuite on augmente le "Gamma" à un contraste le plus efficace pour la fonction qui suit qui est un "BlackScale", cette fonction récupère également les constantes R,G,B du pixel traite et cette fois ci les utilise pour calculer trois nouvelles valeurs $\text{averageR} = 255 * \text{pow}((\text{R}/255.), .1)$; , $\text{averageG} = 255 * \text{pow}((\text{G}/255.), .1)$; et $\text{averageB} = 255 * \text{pow}((\text{B}/255.), .1)$; qui remplaceront respectivement R,G et B. Comme dit au dessus l'intérêt de cette fonction est de limiter l'apparition de certains types de bruits une fois que l'on applique la prochaine fonction.



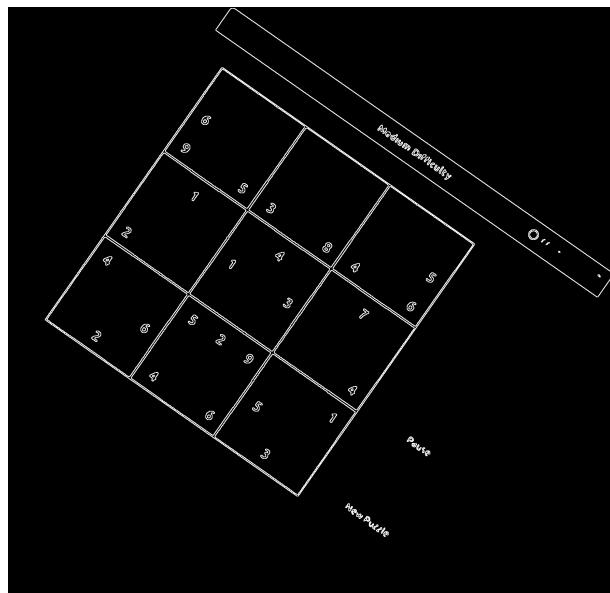
3.1.3 BlackScale

Afin de ne prendre bien que les parties qui on de l'écriture ou des formes comme les lignes de la grille et non des impuretés qui pourrait se trouver sur l'image. Cette fonction est extrêmement simple, elle permet simplement de séparer les pixels sous un certain seuil que nous choisissons, les pixels sous ce seuil sont noirs et le reste blancs. Ce traitement a pour but de binariser l'image c'est à dire de faire en sorte qu'il n'y ait que deux sortes de pixels, les pixels blancs et les pixels noirs. Cela permet de simplifier grandement l'utilisation des images par les autres membres du groupes.



3.1.4 Sobel

Afin d'utiliser Hough il est nécessaire de modifier l'image de base en passant par une inversion des couleurs une fois les traitements décrits précédemment effectués. L'image sera donc une image sur fond noir et écriture blanche. Pour faire simple, l'opérateur calcule le gradient de l'intensité de chaque pixel. Ceci indique la direction de la plus forte variation du clair au sombre, ainsi que le taux de changement dans cette direction. On connaît alors les points de changement soudain de luminosité, correspondant probablement à des bords, ainsi que l'orientation de ces bords. En termes mathématiques, le gradient d'une fonction de deux variables (ici l'intensité en fonction des coordonnées de l'image) est un vecteur de dimension 2 dont les coordonnées sont les dérivées selon les directions horizontale et verticale. En chaque point, le gradient pointe dans la direction du plus fort changement d'intensité, et sa longueur représente le taux de variation dans cette direction. Le gradient dans une zone d'intensité constante est donc nul. Au niveau d'un contour, le gradient traverse le contour, des intensités les plus sombres aux intensités les plus claires. De surcroît, Sobel est en relation directe avec la réduction de bruit notamment les impuretés sur les chiffres. En effet, en regardant à chaque opération les 9 pixels autour (hors les bords) du pixel traité nous donne l'information d'un pourcentage de pixel via une matrice de passage de noir vers blanc et donc enlever des possibles pixels qui sont des impuretés dans l'image.

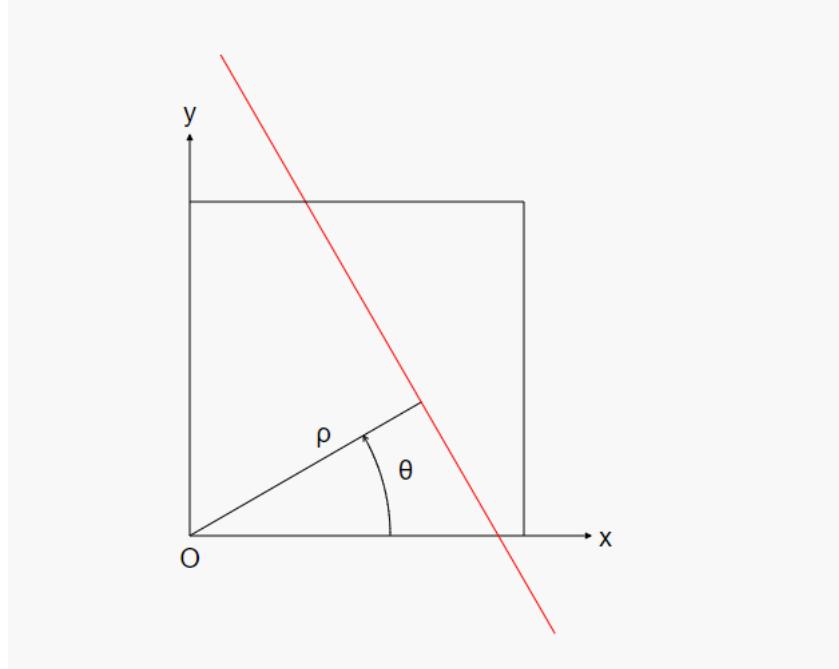


3.1.5 Hough

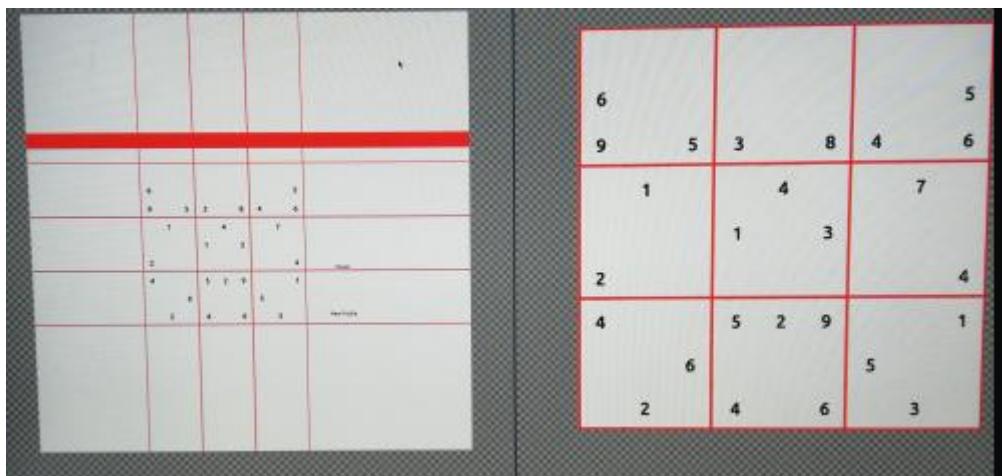
Considérons l'équation d'une droite dans un plan, avec deux paramètres (a,b) : $ax + by + 1 = 0$. Toute droite traversant une image peut être mise sous cette forme. Chacune de ces droites peut être représentée par un point dans le plan (a,b) . Soit $M(x,y)$ un point quelconque de l'image. L'ensemble des droites passant par ce point est représenté par une droite dans le plan (a,b) . La méthode de Hough repose sur l'utilisation d'un accumulateur dans le plan (a,b) . L'accumulateur est une matrice qui correspond à un domaine rectangulaire du plan (a,b) . Pour chaque pixel $M(x,y)$ de l'image, la droite d'équation $ax+by+1=0$ est tracée sur l'accumulateur : plus précisément, chaque pixel rencontré est incrémenté d'une unité. Lorsque tous les points de l'image sont traités, les éléments de l'accumulateur les plus peuplés correspondent à des droites détectées. Le paramétrage des droites par les coefficients (a,b) n'est toutefois pas adapté à cette méthode, car les valeurs possibles de a et b ne sont pas bornées. Il est pratiquement impossible, avec un échantillonnage périodique, de couvrir à la fois les petites, les moyennes et les grandes valeurs de ces coefficients. Pour cette raison, Hough a utilisé le paramétrage suivant des droites, utilisant les coordonnées polaires :

$$\rho = x \cos(\theta) + y \sin(\theta)$$

La signification géométrique de p et θ est montrée sur la figure suivante :

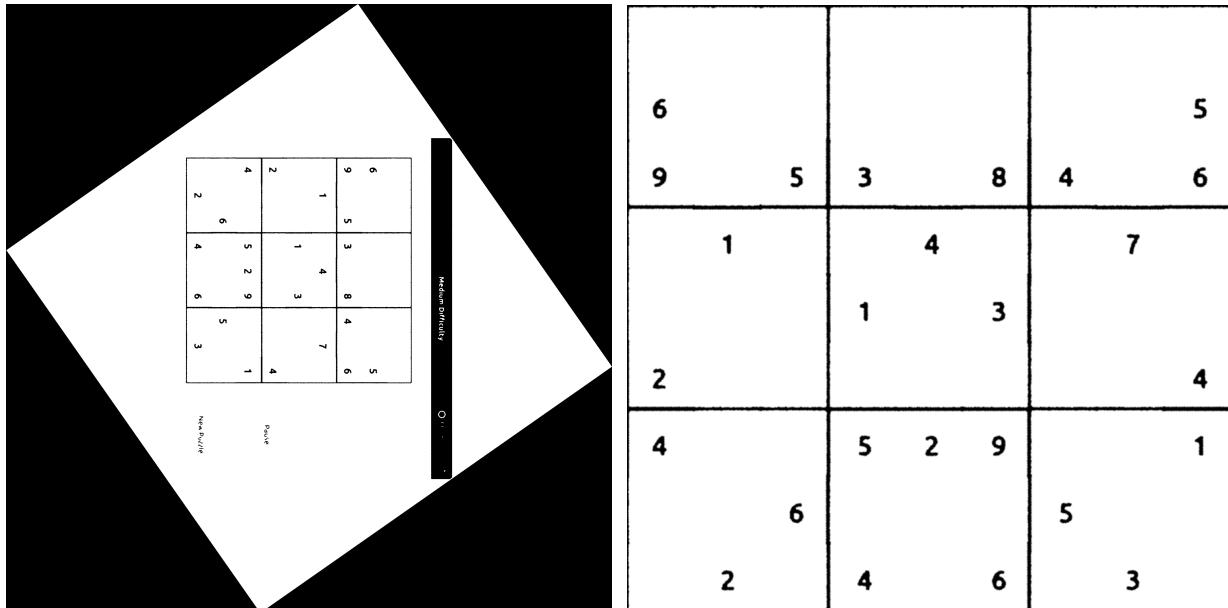


Suite à cela on cherche sur toutes les pixels blanc de la surface de Sobel afin de trouver un angle commun à plusieurs lignes tracer sur l'image. Cette angle va ensuite être l'angle de rotation de notre image. Néanmoins, dans l'implémentation de notre code il y a une possibilité qu'il tourne de 90 degrés une image droite car il va comprendre que l'image est droite mais va faire une rotation car c'est ce qu'on lui demande.



3.1.6 Rotation

Nous utilisons simplement la fonction de la librairie GFX de SDL qui est Rotozoom cette fonction permet de faire tourner dans le sens trigonométrique à partir d'un angle donné. Cette angle étant calculé par Hough. Une fois la grille trouvée et tournée elle est isolée c'est à dire que nous ne gardons que la grille.

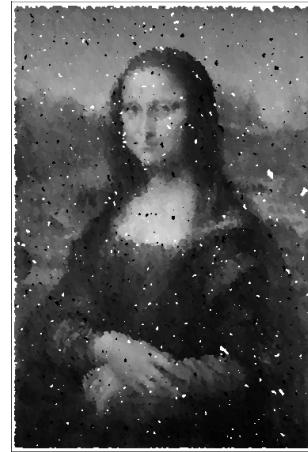
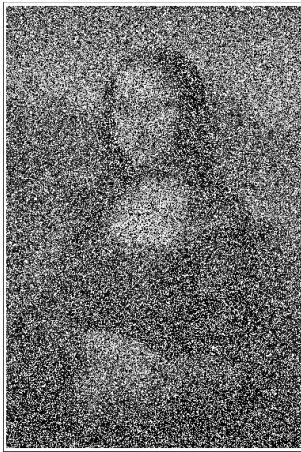


3.1.7 Inversion Couleur

Une fois que la transformé de Hough a été réalisé et que la grille a été mise droite sur l'image, nous inversons une nouvelle fois les couleurs pour que la grille soit en noir. Toutefois, afin de permettre une détection de la grille parfaite, nous faisons la moyenne des 3 composantes du pixel traité et si cette moyenne est supérieure à 30 alors le pixel est changé en noir sinon il est changé en blanc. Cela inverse donc les couleurs de l'images, c'est à dire le blanc devient noir et le noir devient blanc.

3.1.8 Réduction de Bruit

Il s'agit d'un des traitements les plus complexe de cette partie en comparaison avec les premières fonctions Un filtre médian, le filtre médian consiste à prendre pour chaque pixel de l'image que l'on traite les huit pixels composant son entourage, récupérer leur valeurs R uniquement car à ce niveau de traitement les trois composantes ont la même valeur et ensuite les classer en ordre croissant pour trouver la valeur médiane et enfin remplacer la valeur du pixel qui était traité par la valeur médiane. Ce filtre est à été choisi principalement car il est très utile pour réduire les bruits sels et poivres d'une image en nuance de gris. Elle permet aussi sur une image en noir et blanc d'adoucir les bords et de combler les pixels manquants parfois.



Explication technique de la procédure de traitement

Pour le traitement nous avons du faire de nombreux ajustements, dans un premier temps nous avons développé chacune des fonction présentées ci dessus. Dans un deuxième temps nous avons tenté de nombreux ordres de traitements pour voir lequel donner le meilleur résultat. Au final nous avons choisi de commencer par l'application d'un traitement grayscale puis un filtre médian pour réduire les bruits sel et poivre qui pouvait être présent, par la suite nous continuons une augmentation du contraste avec la fonction Gamma puis Blackscale pour binariser l'image et faire en sorte qu'elle soit facilement exploitable pour les prochaines étapes du programme, pour finir nous rappiquons un filtre médian. Une fois ces traitements fait nous utilisons la transformée de Hough pour situer la grille de sudoku et son inclinaison, une fois redressée et isolée nous affichons à nouveau dans la fenêtre le résultat du pré-traitement.

Puis la segmentation

3.2 Détection de la grille

Pour la première soutenance, Hugo s'est chargée de la partie segmentation de l'image. Celle-ci c'est faite en plusieurs étapes, la recherche de la grille à travers l'image et ensuite la découpe de l'image.

Pour la recherche de la grille, nous avons décidé de faire un algorithme "searchgrille".

Cette fonction vérifie pour chaque pixel de l'image au coordonnées (x,y) si la grille commence à ces coordonnées. Pour ce faire nous appelons une autre fonction "goodresearch" qui vérifie plusieurs conditions pour savoir si la grille commence à ses coordonnées (c'est-à-dire aux coordonnées x et y). Si la fonction renvoie 1 au coordonnées x et y.

L'algorithme searchgrille retourne les valeurs de x et y.

La fonction goodresearch prend comme paramètre l'image, la coordonnée x, la coordonnée y

Nous allons donc devoir vérifier :

- si le pixel(x,y) de l'image est noir
- si le pixel(x+1, y) de l'image est noir (soit le pixel à droite de x et y) et si le pixel(x,y+1) de l'image est noir (soit le pixel en dessous de x et y).

Si ces dernières conditions sont prouvées alors :

- on vérifie si le pixel(x+1,y+1) de l'image est blanc (le pixel en bas à droite de x et y).
- on crée alors une variable $y1 = y$, on fait alors une boucle qui augmente $y1$ de 1 à chaque fois qu'on rentre dans la boucle tant que pixel(x+1,y1) de l'image est blanc.

Une fois qu'on sort de la boucle on crée une variable l qui est égal $y1 - y$.

- On crée une nouvelle variable $newy = y$, on vérifie alors si le pixel(x,newy) de l'image est noir pour tout newy inférieur à $y + L$.

- On crée une autre variable $newx = x$, on vérifie alors si le pixel(newx,y) de l'image est noir pour tout newx inférieur à $x + L$.

Si toutes les conditions sont établies, on vérifie alors en dernier si le pixel($x + L/3, y + L/3$), le pixel($x + 2L/3, y + 2L/3$) et ($x + L, y + L$) de l'image sont noirs (soit les coordonnées finales des pixels des moyens carrés qui sont dans la diagonale de (x,y) à (x+L,y+L)).

Cette fonction renvoie 1 si tous les paramètres sont justifiés. 0 si un des paramètres est rejetées.

En ce qui concerne le découpe de l'image, Hugo a réalisé une fonction qui parcours toutes les coordonnées de la grille de l'image. Pour cela, nous avons fait deux boucles :

- une boucle qui parcours toutes les coordonnées sur y tant que y est inférieur à la longueur de la grille.
 - une boucle qui parcours toutes les coordonnées sur x tant que x est inférieur à la longueur de la grille.
- Ainsi, cette fonction appelle une fonction qui découpe l'image aux coordonnées x et y.

Cette fonction prend une image, une coordonnée x, une coordonnée y et la longueur L en paramètres. Elle découpe l'image aux coordonnées (x,y) de longueur L/9 en hauteur et L/9 en largeur. Cette image est sauvegardée avec pour nom casexy. Ici, x et y sont les coordonnées mise en paramètre.

Pour la deuxième soutenance, beaucoup de changements sont apparus sur la découpe de l'image car l'image que j'avais à traiter a beaucoup changé. Dorénavant, suite aux traitements entièrement effectué la détection de l'image a changé même si certains principes sont les mêmes.

En effet, on teste toujours pour chaque pixel x et y dans les limites de l'image si la grille commence à cette grille. Pour ceci on appelle toujours "goodresearch" prenant comme paramètre toujours une image un x et un y. La fonction "goodresearch" renvoie 0 si la grille ne commence pas aux coordonnées x et y.

La goodresearch fonctionne alors de cette façon :

- on teste toujours si le pixel est noir sinon on renvoie 0.
- On calcule la longueur LX qui correspond à la longueur tant que le pixel sur l'axe X est noir,

- puis on calcule la longueur LY qui correspond à la longueur tant que le pixel sur l'axe Y est noir,
- si la longueur LX est égal à la longueur LY on renvoie 1 sinon on renvoie 0.
- la longueur LX étant enregistré alors dans la fonction principale en L afin de découper l'image.

On cadre alors l'image sur la grille grâce à une fonction Zoom qui change l'image avec pour coordonnées de départ x et y et comme hauteur et largeur L.

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

Comme on peut le voir, ici l'image 5 de base a été modifié de façon que l'image soit composé seulement de la grille.

On fait ensuite le découpage qui fonctionnait déjà à la première soutenance. Celui si découpe chaque carré de l'image avec comme taille $L/9$ en hauteur et en largeur. Les images sont enregistrés en format .bmp avec comme taille 28*28, afin de favoriser l'apprentissage.



Les images sont alors découpés et enregistré de cette façon, à droite le carré en haut à gauche qui comme sur l'image d'au dessus avec la grille entière et vie. A gauche, l'image correspond à l'avant dernier carré sur l'axe x tout en bas de l'image de la grille entière. De plus, l'image supprime quasiment toute représentation des cadres de la grille afin de faciliter le travail de l'image sur le réseau neuronal.

Ainsi, par rapport à la première soutenance, la détection de la grille ainsi que le découpage sont beaucoup plus précis permettant une meilleure résolution du Sudoku.

Ensuite le réseau neuronal

3.3 Réseau Neuronal

3.3.1 X-OR

Pour la première soutenance, il nous était demandé d'implémenter un réseau neuronal apprenant la porte logique "X-OR".

Le principe de cette porte logique est simple, lorsque les entrées sont différentes, la sortie sera 1, la sortie sera 0 sinon. On peut se référer à la table de vérité pour plus de détails. Il a donc fallu implémenter un réseau neuronal qui, en ayant deux entrées composées, soit de 0 ou 1, puisse sortir la bonne valeur de la porte logique.

Un réseau neuronal est un concept algorithmique qui va simuler la réflexion d'un cerveau, mais ici, lors de ce projet, avec un approche bien plus simple.

Le réseau neuronal est composé d'une "Input layer" qui est un couche où l'on va insérer les valeurs d'entrée, il a ici deux cellules pour la Input layer. Il y a ensuite une "Hidden layer" qui sera une couche intermédiaire de valeur, et enfin la couche "Output layer" qui sera la couche des valeurs de sortie, qui ici possède une seule valeur de sortie.

Afin d'avoir une représentation visuelle, vous pouvez vous référer au schéma ci-dessous. On peut y voir des liens qui lient ces layers, sont appelés "weight" (poids). Chaque cellule possède un biais.

Le principe de l'algorithme est de répéter l'apprentissage sur un nombre d'époque défini, de préférence un grand nombre pour avoir de meilleurs résultats (>5000).

Plusieurs variables sont initialisées au début de l'algorithme :

- Les tailles des layers (nombres de cellules)
- Les tableaux de caractère d'inputs (les différents patterns d'apprentissage)
- Les tableaux "Target" qui sont les résultats attendus afin de compares avec les résultats "Output".

On va par la suite initialiser les poids de chaque layers, il n'est pas recommandé de les initialiser à 0, donc on les initialisent à l'aide d'un fonction Random() qui comme son nom l'indique va donner des valeur aléatoires comprises entre -1 et 1.

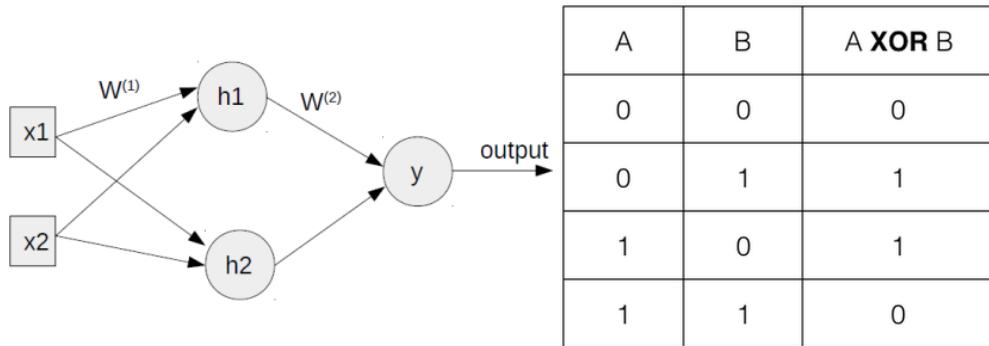
Notre algorithme va parcourir le réseau neuronal depuis l'input layer vers l'output layer un certain nombre de fois, en mettant à jour les poids de chaque layer, mais afin que l'apprentissage se fasse, il ne faut pas que les erreurs se répètent aussi : il y a un système de "back propagation", un renvoi d'information qui traverse le réseau dans l'autre sens, depuis l'output layer, vers les hidden layers, s'il y en a plusieurs.

A chaque époque, l'erreur "Error" est mise à 0 au début de la boucle. Elle est mise à jour en réalisant une comparaison : résultat attendu soustrait au résultat obtenu, au carré, tout ça produit de un demi.

Un tableau comparatif des résultats est affiche sur la console lorsque l'algorithme a terminé ses calculs. Il est composé de deux partie, l'une montrant l'évolution de la variable "Error" en fonction des époques, l'autre montre les différences entre les résultats obtenus et les résultats attendus.

On voit dans le premier tableau, un affichage toutes les 500 époques, variables qui est notamment modifiable. La deuxième partie montre les résultats des 4 patterns présents dans la table de vérité, avec la colonne "Target1" les résultats attendus et la colonne "Output1" les résultats obtenus.

La fonction d'activation des cellules des neurones est la fonction mathématique sigmoïde, elle est présente sur toutes les couches dans l'algorithme, mais dans notre réseau de neurone final, la fonction d'activation de l'output layer sera la fonction "Softmax" car elle garantira de meilleur résultats.



```

Epoch 0      : Error = 0.529350
Epoch 500     : Error = 0.003013
Epoch 1000    : Error = 0.000780
Epoch 1500    : Error = 0.000442
Epoch 2000    : Error = 0.000307
Epoch 2500    : Error = 0.000234
Epoch 3000    : Error = 0.000189
Epoch 3500    : Error = 0.000159
Epoch 4000    : Error = 0.000137
Epoch 4500    : Error = 0.000120
Epoch 5000    : Error = 0.000107

NETWORK DATA - EPOCH 5306
Pat      Input1        Input2        Target1      Output1
1       0.000000      0.000000     0.000000     0.006461
2       1.000000      0.000000     1.000000     0.993283
3       0.000000      1.000000     1.000000     0.993283
4       1.000000      1.000000     0.000000     0.008245

```

3.3.2 Apprentissage

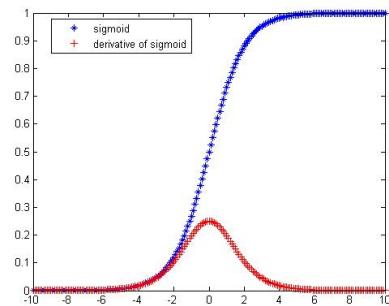
Premièrement, le réseau neuronal reconnaissant la fonction XOR, mais il a fallu mettre en place des "Struct" afin de mieux gérer les différentes parties du réseau et d'en améliorer ses performances. Le réseau neuronal a donc été découpé en plusieurs parties, une partie où l'on initialise le réseau "net". Puis, à l'instar de ce qui a pu être fait pour le XOR, les biais sont initialisés aléatoirement, à l'aide d'une fonction "Random" qui génère des "double" compris entre -1 et 1.

Vient après la fonction "Forward Pass" et "BackWard Pass" qui correspondent au parcours dans le réseau de neurones des biais composants notre image, de la couche Input vers la couche Output, et dans l'autre sens, afin de mettre à jour l'erreur dans le réseau de neurones.

A chaque fois, les poids et les biais sont mis à jour dans leur fonction respective, "UpdateBiases" et "UpdateWeights". Les valeurs des poids et des biais sont donc mis à jour dans les tableaux et réutilisées dans les prochaines itérations sur les prochaines images. Enfin la fonction "NeuralNetworkOCR" va exécuter les différentes étapes décrites précédemment afin d'entraîner le réseau sur toutes les images "exemples" disponibles. A chaque biais va être appliquée une fonction d'activation afin que la valeur reparte dans le réseau de neurones. Nous avons utilisé la fonction "sigmoid", et lors de la propagation retour (Backward-Pass), on va utiliser la dérivée de cette fonction.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\sigma'(x) = \frac{d}{dx} \sigma(x) = \sigma(x)(1 - \sigma(x))$$



Afin de reconnaître des caractère, le réseau neuronal a besoin d'avoir appris. C'est à dire qu'il doit avoir travaillé avec des images avant de pouvoir reconnaître des images inconnues. La première partie est donc l'apprentissage de caractère et la sauvegarde de fichiers contenant les poids et les biais du réseau post-apprentissage.

Le fichier "Neural Network" contient plusieurs fichier et sous-dossiers. Premièrement, le fichier "dagit" contient tout le nécessaire à la base de donnée nécessaire. Chaque chiffre possèdent un fichier Premièrement, il a fallu récupérer une base d'image, neuf image correspondant aux neuf chiffres de 1 à 9, avec plusieurs écritures différentes.

Afin de transcrire ces images dans le réseau de neuronal, chaque image est passée dans une fonction qui va premièrement faire en sorte qu'on obtienne un fichier de 784 caractères, 28 lignes par 28 colonnes de l'image. Lorsque que le pixel teste est blanc, on rempli le tableau avec un 0, et si le pixel est noir on rempli le tableau avec un 1 à la même place. Nous avons maintenant des fichiers texte comprenant chacun 784 caractères pour chaque images.

2

```
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
0000000000111000000000000000
0000000000111111100000000000
000000000011000001110000000000
000000000011000001110000000000
00000000001100000000000000
00000000001100000000000000
00000000001100000000000000
00000000001100000000000000
00000000001100000000000000
00000000001100000000000000
00000000001100000000000000
00000000001100000000000000
00000000001100000000000000
00000000001100000000000000
00000000001100000000000000
00000000001100000000000000
00000000001100000000000000
00000000001100000000000000
00000000001100000000000000
```

Il faut maintenant que le réseau apprenne à reconnaître les chiffres, mais il lui faut un but pour pouvoir comparer ce qu'il apprend et apprendre de ses erreurs. Un tableau "Goal" de 10 par 10 est donc créé afin. Chaque ligne est attribuée à un chiffre, la première a "1", la deuxième a "2" etc. Chaque ligne est composée de 9 entiers, chaque entier peut être soit 1 soit 0, l'entier est positionné à un au niveau du chiffre auquel il correspond.

1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1

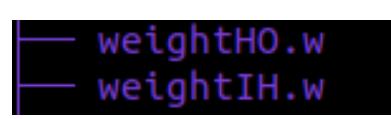
Le réseau de neurone va donc s'exécuter lorsque l'utilisateur choisi de l'entraîner. Lors de l'apprentissage, l'évolution est affichée sur la console avec comme information :

- L'erreur d'apprentissage
- Les chiffres appris et/ou non appris

Lorsque l'apprentissage est terminé une fonction "Save Data" intervient. Cette fonction permet d'extraire les poids et les biais du réseau de neurones à la fin de l'apprentissage dans 4 fichiers distincts :

- "weightHO.w" qui sauvegarde les poids de la couche Hidden à la couche Output
- "weightIH.w" qui sauvegarde les poids de la couche Input à la couche Hidden
- "biasO.b" qui sauvegarde les biais de la couche Output
- "biasH.b" qui sauvegarde les biais de la couche Hidden

Ces fichiers seront la clé du fonctionnement de la reconnaissance des chiffres.



3.3.3 Reconnaissance de caractère

La reconnaissance va être similaire à l'entraînement de notre réseau. Premièrement, lors de la récupération des 81 cases de la grille de Sudoku il faut vérifier si l'on va traiter la case ou non : il est inutile de traiter une case blanche. Une fonction va donc vérifier si la case est blanche ou non en calculant la valeur des pixels de l'image, si tous les pixels ont une valeur RGB égale à 255, alors la case est blanche. Nous allons traiter les images qui ne sont pas considérées comme blanches. L'image va être après convertie en un tableau de 784 caractères, 28 multipliées par 28 colonnes, comme expliqué précédemment.

Le réseau de neurone sera celui extrait des données enregistrées dans les fichiers ".w" et ".b". Cette image va désormais parcourir le réseau de neurone, c'est la fonction "Convert" qui intervient, l'image rentrée en paramètre va renvoyer un caractère correspondant au chiffre reconnu. On va donc construire une grille de résolution en fichier texte afin de résoudre le Sudoku dans la partie "Résolution de la Grille".

Afin d'afficher l'évolution de l'apprentissage, une fonction "PrintState" intervient pour afficher la mise à jour des différents paramètres à travers les époques.

Chaque ligne correspond à un chiffre, au fil du temps on voit l'évolution des chiffres qui sont reconnus ou non en fonction de leur parcours dans le réseau neuronal.

```

0.357773 | Char recognized: 6 | ErrorRate: 0.000075
-0.385049 | Char entered: 7 | Char recognized: 7 | ErrorRate: 0.000092
0.536367 | Char entered: 8 | Char recognized: 8 | ErrorRate: 0.000112
-0.489289 | Char entered: 9 | Char recognized: 9 | ErrorRate: 0.000141
-0.046794 | Epoch 3300 | MaxErrorRate = 0.008149
0.85967 | Char entered: 1 | Char recognized: 1 | ErrorRate: 0.000166
0.876882 | Char entered: 2 | Char recognized: 2 | ErrorRate: 0.000167
0.075305 | Char entered: 3 | Char recognized: 3 | ErrorRate: 0.000141
-0.768807 | Char entered: 4 | Char recognized: 4 | ErrorRate: 0.000176
-0.150705 | Char entered: 5 | Char recognized: 5 | ErrorRate: 0.000196
-0.618497 | Char entered: 6 | Char recognized: 6 | ErrorRate: 0.000172
0.543703 | Char entered: 7 | Char recognized: 7 | ErrorRate: 0.000199
0.457331 | Char entered: 8 | Char recognized: 8 | ErrorRate: 0.000109
Position Found = 7 Expected 1 OK
Char entered: 1 | Char recognized: 7 | ErrorRate: 2.032282
Position Found = 9 Expected 2 OK
Char entered: 2 | Char recognized: 9 | ErrorRate: 1.287116
Position Found = 9 Expected 3 OK
Char entered: 3 | Char recognized: 9 | ErrorRate: 1.430537
Position Found = 2 Expected 4 OK
Char entered: 4 | Char recognized: 2 | ErrorRate: 0.729672
Position Found = 1 Expected 5 OK
Char entered: 5 | Char recognized: 6 | ErrorRate: 0.511236
Position Found = 2 Expected 6 OK
Char entered: 6 | Char recognized: 2 | ErrorRate: 0.520281
Position Found = 5 Expected 7 OK
Char entered: 7 | Char recognized: 5 | ErrorRate: 0.415125
Position Found = 6 Expected 8 OK
Char entered: 8 | Char recognized: 6 | ErrorRate: 0.479304
Position Found = 1 Expected 9 OK
Char entered: 9 | Char recognized: 1 | ErrorRate: 0.535542
Epoch 3400 | MaxErrorRate = 2.032282
Position Found = 1 Expected 1 OK
Char entered: 1 | Char recognized: 1 | ErrorRate: 0.004091
Position Found = 2 Expected 2 OK
Char entered: 2 | Char recognized: 2 | ErrorRate: 0.003969
Position Found = 3 Expected 3 OK
Char entered: 3 | Char recognized: 3 | ErrorRate: 0.003709
Position Found = 4 Expected 4 OK
Char entered: 4 | Char recognized: 4 | ErrorRate: 0.003509
Position Found = 5 Expected 5 OK
Char entered: 5 | Char recognized: 5 | ErrorRate: 0.006075
Position Found = 6 Expected 6 OK
Char entered: 6 | Char recognized: 6 | ErrorRate: 0.003077
Position Found = 7 Expected 7 OK
Char entered: 7 | Char recognized: 7 | ErrorRate: 0.003578
Position Found = 8 Expected 8 OK
Char entered: 8 | Char recognized: 8 | ErrorRate: 0.005950
Position Found = 9 Expected 9 OK
Char entered: 9 | Char recognized: 9 | ErrorRate: 0.004215
Epoch 100 | MaxErrorRate = 0.006709
Position Found = 1 Expected 1 OK
Char entered: 1 | Char recognized: 1 | ErrorRate: 0.000085
Char entered: 2 | Char recognized: 2 | ErrorRate: 0.000136
Char entered: 3 | Char recognized: 3 | ErrorRate: 0.000132
Char entered: 4 | Char recognized: 4 | ErrorRate: 0.000105
Char entered: 5 | Char recognized: 5 | ErrorRate: 0.000093
Char entered: 6 | Char recognized: 6 | ErrorRate: 0.000070
Char entered: 7 | Char recognized: 7 | ErrorRate: 0.000087
Char entered: 8 | Char recognized: 8 | ErrorRate: 0.000085
Char entered: 9 | Char recognized: 9 | ErrorRate: 0.000087
Epoch 3400 | MaxErrorRate = 0.000141
Position Found = 1 Expected 1 OK
Char entered: 1 | Char recognized: 1 | ErrorRate: 0.000102
Position Found = 2 Expected 2 OK
Char entered: 2 | Char recognized: 2 | ErrorRate: 0.000082
Char entered: 3 | Char recognized: 3 | ErrorRate: 0.000132
Char entered: 4 | Char recognized: 4 | ErrorRate: 0.000074
Char entered: 5 | Char recognized: 5 | ErrorRate: 0.000093
Char entered: 6 | Char recognized: 6 | ErrorRate: 0.000070
Char entered: 7 | Char recognized: 7 | ErrorRate: 0.000087
Char entered: 8 | Char recognized: 8 | ErrorRate: 0.000085
Char entered: 9 | Char recognized: 9 | ErrorRate: 0.000085
Epoch 3500 | MaxErrorRate = 0.000136
Position Found = 1 Expected 1 OK
Char entered: 1 | Char recognized: 1 | ErrorRate: 0.000100
Char entered: 2 | Char recognized: 2 | ErrorRate: 0.000082
Char entered: 3 | Char recognized: 3 | ErrorRate: 0.000132
Char entered: 4 | Char recognized: 4 | ErrorRate: 0.000074
Char entered: 5 | Char recognized: 5 | ErrorRate: 0.000093
Char entered: 6 | Char recognized: 6 | ErrorRate: 0.000070
Char entered: 7 | Char recognized: 7 | ErrorRate: 0.000087
Char entered: 8 | Char recognized: 8 | ErrorRate: 0.000085
Char entered: 9 | Char recognized: 9 | ErrorRate: 0.000085
    
```

Enfin la résolution de la grille

3.4 Résolution de la grille

3.4.1 Première soutenance

Au moyen de cette partie, je décris la méthode de résolution de grille et la manière d'implémentation de cette résolution. Il est très peu probable que cette partie change pour la dernière soutenance. L'algorithme de résolution est efficace et ne prend pas trop de mémoire sachant que seul le rendu visuel divergera de l'affichage actuelle dans la console. Il existe deux types de méthode :

- Celles solubles directement par un schéma visuel ou par la lecture des candidats.
- Celles solubles exclusivement par la lecture des candidats.

On désigne les candidats d'une case les chiffres possibles que cette case peut admettre. L'ensemble des candidats d'un énoncé de Sudoku sont toutes les possibilités de toutes les cellules de la grille. Il est conseillé, pour faciliter la tâche, de commencer à résoudre visuellement le maximum de cases. Dans le but de terminer avec le minimum de candidats à écrire il est fondamentale de compléter ou de se conformer à l'une des méthodes prescrites.

J'ai préféré implémenter la première qui me permet de gérer plusieurs cas de candidats impossible à la fois. Voici donc comment le code fonctionne.

Une grille Sudoku complète est un tableau de 9 cases sur 9, subdivisé en 9 carrés de 3 cases de côté. Chaque case contient un seul chiffre allant de 1 à 9. Chaque ligne, colonne, et carré de 3 X 3 incluent obligatoirement ces 9 chiffres. Par conséquent, pris isolément, une ligne, une colonne ou un carré de 3 sur 3 ne peuvent contenir plusieurs fois une même valeur.

Un énoncé Sudoku est une grille incomplète n'acceptant qu'une solution unique.

On définira indifféremment une grille, étant un énoncé en cours de résolution ou la solution est la résolution complète de l'énoncé. Il a donc fallut prendre l'énoncé et tester des candidats sur la grille. Chaque candidat sera vérifier par rapport à ligne et la colonne ou il sera tester ainsi que sur le carre de 3x3 auquel il appartient. Des qu'une erreur se manifestera dans le retour de la fonction qui vérifie la validité de la grille alors cela impliquera que l'énoncé n'est pas solvable et donc le programme renverra une erreur.

Tout ceci était la partie concernant la résolution de l'énoncé. Maintenant il faut expliquer comment récupérer la l'énoncé. En effet l'énoncé se trouve dans un fichier texte, ce qui implique que les caractères sont sous la forme de char alors que nous les voulons sous forme de int. Il a donc fallu prendre en compte ceci et utiliser le fait que le char '0' (zéro) est égal à 48 en int. Sachant ça une simple soustraction suivie d'un cast remplace bien le char en int. Le problème résolu il a suffit de parcourir le fichier et de prendre deux compteurs représentant les lignes et les colonnes pour placer le nombre au bon endroit. On appelle alors l'algorithme de résolution de la l'énoncé et on créer un fichier contenant la solution de l'énoncé qui est la fin de l'exécution successive de chaque partie du programme. L'affichage de ce fichier rendu sera une des améliorations de notre programme.

```
[nix-shell:~/adam.rili/src/grid_solver]$ time ./solver grid_00
real    0m0.001s
user    0m0.001s
sys     0m0.000s

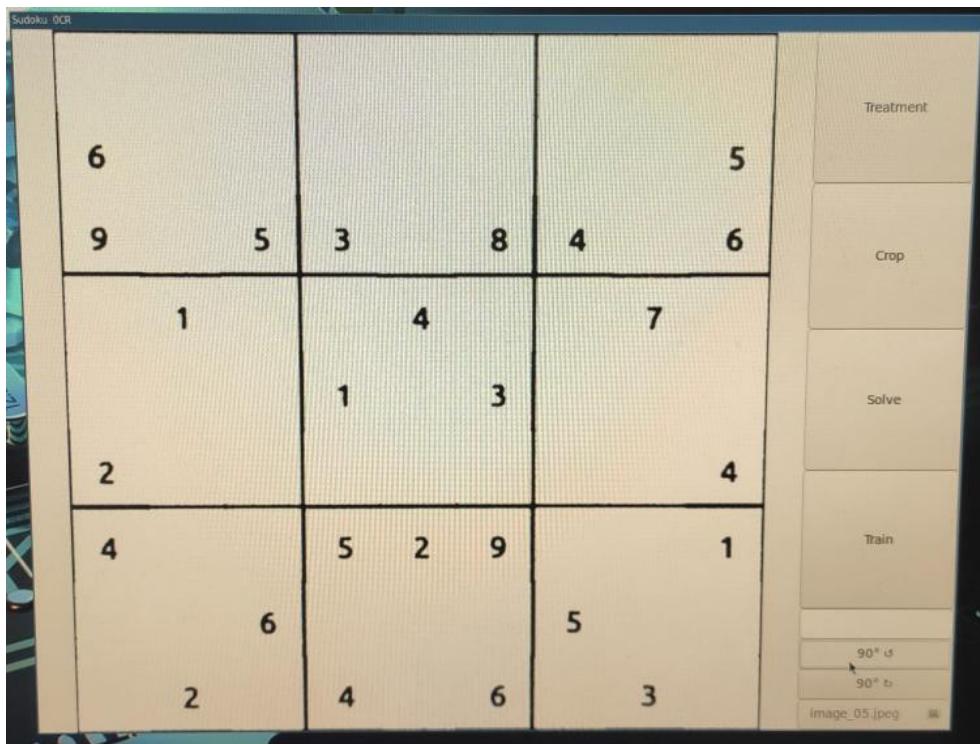
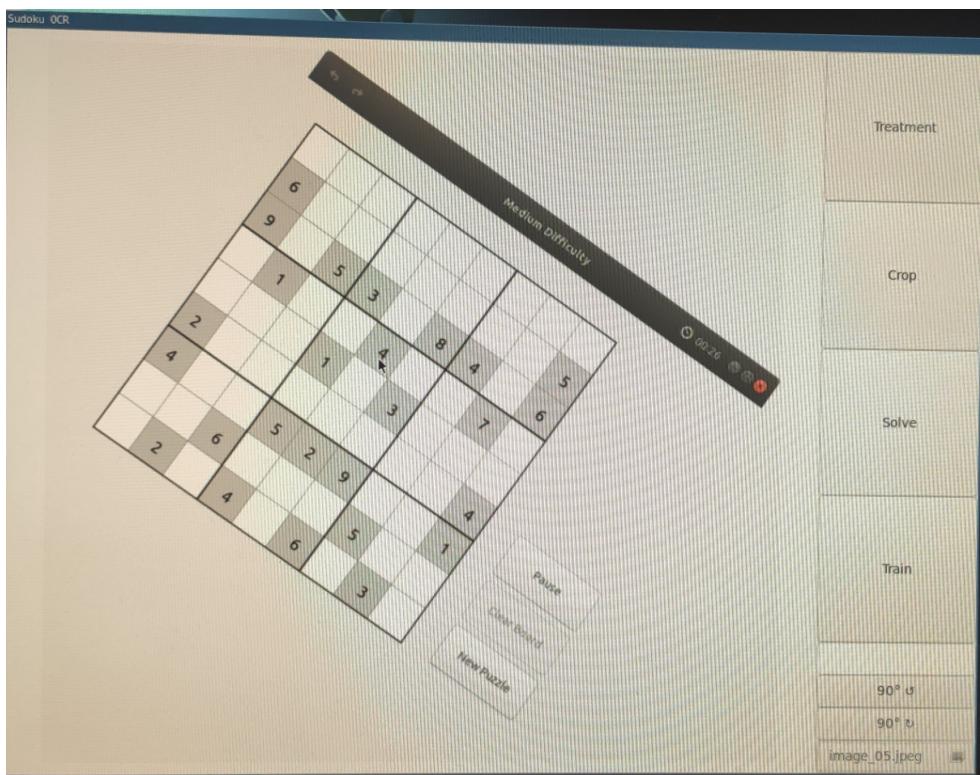
[nix-shell:~/adam.rili/src/grid_solver]$ time ./solver grid_01
real    0m0.003s
user    0m0.002s
sys     0m0.001s
```

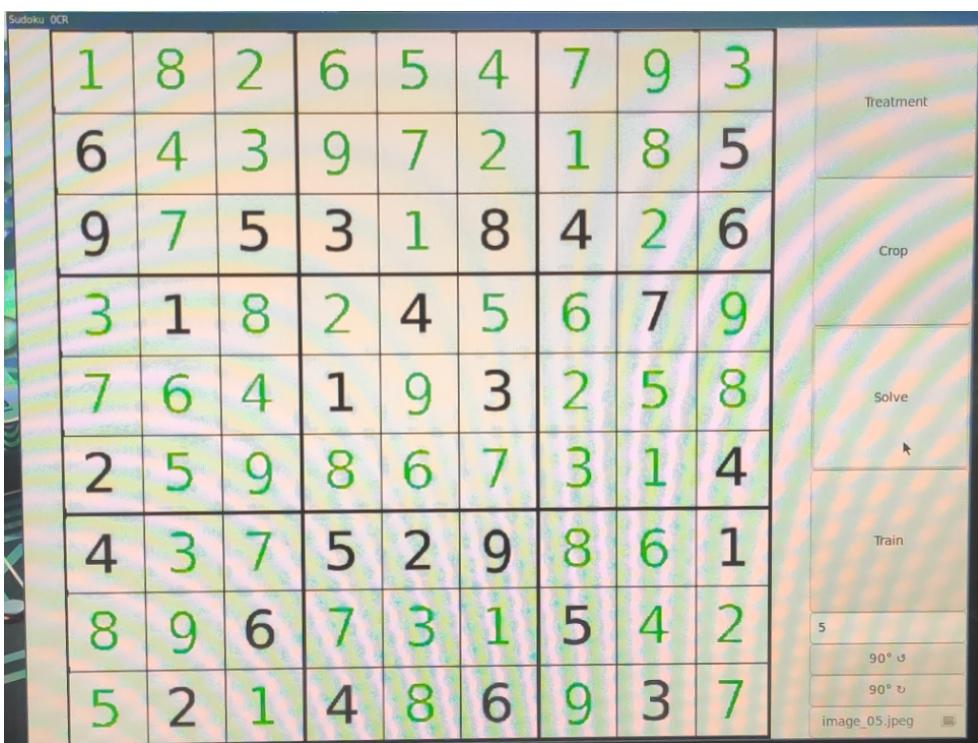
3.4.2 Deuxième soutenance

Comme présenter plus haut l'algorithme de résolution de la grille n'a pas été changer. Ses performances sont satisfaisant et sont exécution correspond a nos attente. Néanmoins, le renvoi de la grille a ete lui modifier. Maintenant, une grille s'affichera avec en noir les chiffres present sur la grille et en vert les chiffre rajouter par le programme. Pour ce faire un tableau est utiliser pour sauvegarder l'information de la présence ou non du chiffre dans l'image de base. On pourrait assimiler cela aux algorithmes de parcours de graph qui utilise un vecteur de marque pour vérifier si un noeud a été visite ou non. Pour afficher les chiffres sur la grille nous avons un dossier "digits_on_grid" qui contient les images en couleur noir et blanc et un dossier "solve_digit" qui lui a les chiffres en blanc et vert. En fonction de la place dans le vecteur de marque le chiffre va etre copier au bonne coordonner pixels par pixels.

3.5 Interface Graphique

Pour ce qui est de l'interface graphique du programme nous avons décidés d'utiliser GTK et glade, une librairie et un logiciel que nous avions utilisé une semaine plus tot dans un TP de programmation. Nous avons opté pour un agencement simple et sobre mais efficace. Une zone a gauche réservée a l'affichage de l'image contenant la grille a chaque étape du programme et un zone légèrement plus petite a droite contenant différents boutons avec une fonction différente pour chacun. Le premier a pour but de choisir l'image parmi les fichier de la machine, l'image choisi sera directement afficher dans l'interface. Un deuxième autre pour lancer le pré-traitement sur cette image, donc tout ce qui vas permettre la découpe et détection de la grille. Un troisième pour résoudre la grille de l'image découper en utilisant une grille trouver par le réseau neuronal. Enfin, un dernier bouton pour lancer l'entraînement du réseau neuronal afin de sauvegarder les poids et les biais pour la résolution de la grille. Cette partie du travail a aussi comporté une difficulté majeure, celle de devoir tout rassembler en un seul fichier effectuant tout les traitement et actions que nous avions programmées dans le bon ordre. Pour cela il fallait avoir bien compris les intentions de chacun dans leurs fonction et également très bien comprendre la façon donc le projet était compiler. Parfois a cause de manque de communication nous n'avions pas même idée en tête et alors beaucoup de temps a été perdu mais au final nous avons réussi a nous mettre d'accord pour avoir une interface permettant de bien utiliser le programme.





3.6 Organisation du projet

3.6.1 Première soutenance

Afin d'avancer de manière efficace dans le projet et ne pas rencontrer de problème de compréhension en mélangeant les tâches de chaque personne il a été décidé de, pour l'instant, découper le projet en 4 grandes parties. Qui sont le Traitement de l'Image, la Segmentation, la Résolution de la Grille et le Réseau Neuronal.

Tableau de répartition jusqu'à la première soutenance

	Adam	Mathis	Hugo	Léo
Traitement Images				
Noir/Blanc		X		
Rotation	X	X		
Lissage		X		
Segmentation				
Reconnaissance grille			X	
Découpage de l'image		X	X	
Résolution Grille				
Traitement du fichier input	X			
Résolution du Sudoku	X			
Traitement du fichier output	X			
Réseau Neuronal				
XOR				X
Apprentissage				X

3.6.2 Soutenance final

Tableau de répartition pour la deuxième soutenance

	Adam	Mathis	Hugo	Léo
Traitement Images				
Sobel		X		
Hough	X	X		
Reduction de bruit		X		
Segmentation				
Reconnaissance grille			X	
Decoupage de l'image		X	X	
Resolution Grille				
Traitement du fichier input	X			
Resolution du Sudoku	X			
Traitement du fichier output	X			
Réseau Neuronal				
Apprentissage				X
Réseau neuronal				X

4 Site Internet

->Lien du site web de notre projet :
https://dragonkill25.github.io/OCR_S3/

Il contient des images et toutes l'avancer de notre projet. Ainsi que des parties personnelles plus développer que sur le rapport.

4.1 Structure du site

-> Accueil :

Nous y présentons notre projet en lui-même : son principe, son but et surtout son intérêt. Plus bas nous avons la possibilité de télécharger le projet.

Il y a aussi des images de notre application.

-> Avancement du projet :

Dans cette section vous pouvez y retrouver une vidéo d'avancement de notre jeu grâce à l'application "gource", qui, lancer dans un terminal et dans un dossier git permet de retracer l'historique des commits depuis la création du projet. Cette vidéo est assez intéressante car elle montre bien l'avancement des modifications et des choix pris durant tout le projet. Mais pas seulement, on peut voir une activité régulière dans ce projet ce qui montre un investissement de toutes les personnes du groupe dans celui-ci.

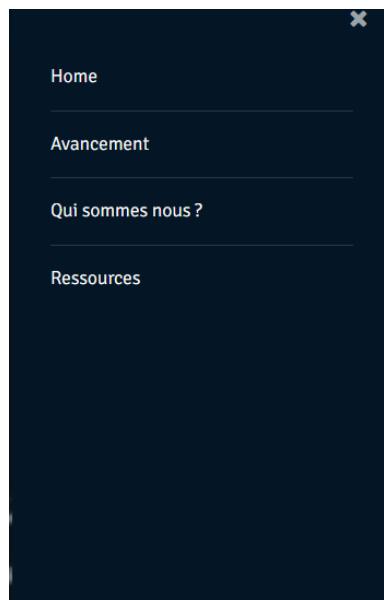
Enfin, nos rapports de soutenance ainsi que notre cahier des charges sont disponible à la visualisation par tous ceux intéressé et qui auront des questions sur la réalisation de ce projet.

-> **Qui sommes nous ? :**

C'est dans cette partie où nous expliquons qui nous sommes, de qui est compose notre groupe.

-> **Document :**

Dans cette nouvelle partie créée pour cette soutenance finale, nous pouvons y retrouver toutes les images et résolution de la grille possible selon l'image donnée.



Avancement de notre projet durant l'année



Vidéo des changements dans notre dossier Git

5 Difficulté rencontrées

5.1 Pré-traitement de l'image

5.1.1 Soutenance 1

Identification du problème :

Lors du développement de ma partie j'ai eu un problème de mémoire et j'ai du recommencer à zéro. J'ai pu identifier le problème et j'ai remarqué qu'il venait de ma façon d'effectuer la rotation. J'ai aussi eu de nombreux problèmes pour compiler mon code sur les machines de l'école car je ne connaissais pas trop la syntaxe des Makefiles.

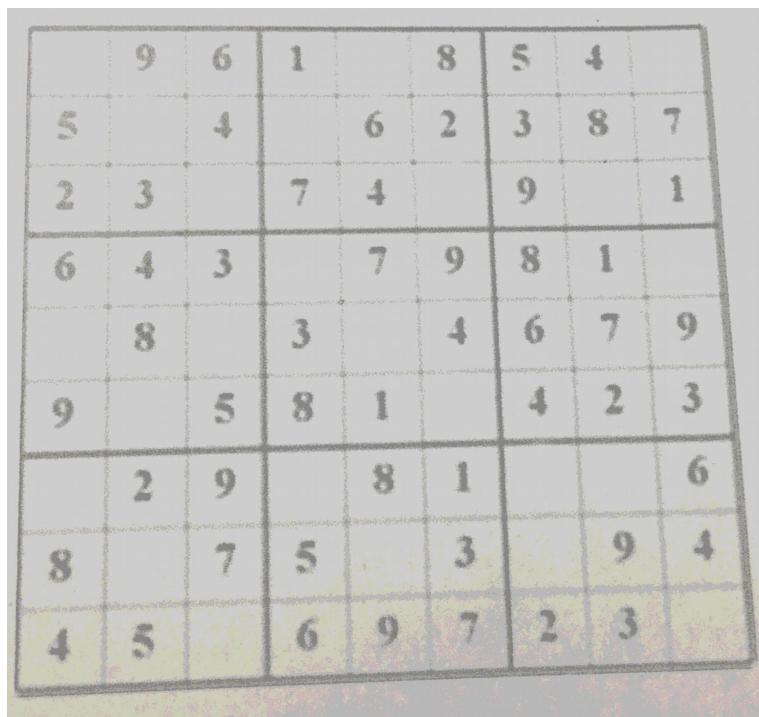
Mise en place de la solution :

Après avoir trouver l'origine du problème j'ai momentanément remplacé ma méthode de rotation par une méthode plus simple mais légèrement moins efficace c'est à dire le "Rotozoom" de la bibliothèque "SDL GFX". Pour régler les problèmes de compilation j'ai juste du me renseigner sur la syntaxe et le fonctionnement des Makefiles.

5.1.2 Soutenance 2

Identification du problème 1 :

L'un des plus gros problèmes à cette soutenance a été la persistance de certains pixels gris après l'application du blackscale. Cela était extrêmement gênant pour les fonctions qui suivaient le pré-traitement comme la détection de la grille ou le réseau neuronal. Ce genre d'image peut poser énormément de problèmes par exemple, car l'on perd beaucoup d'information lors du traitement pour la rendre utilisable. Les lignes trop fines sont perdues et les chiffres très peu lisibles.



Mise en place de la solution 1 :

Le problème venait simplement de la fonction "Sobel" qui créait de nouveaux pixels gris, nous avons simplement passé le "BlackScale" après le "Sobel" pour régler le problème.

Identification du problème 2 :

Encore un fois nous avons eu de nombreux problèmes de segmentation, nous avons du faire beaucoup de test pour chercher d'où les problèmes pouvait parvenir et nous en avons conclu que certain venait des strings utilisées pour maintenir à jour l'image affichée par gtk et la Surface SDL.

Mise en place de la solution 2 :

Nous avons revu la façon dont nous utilisions ces strings pour une manière plus sécurisé, avec plus de tests et les problèmes de segmentation sont disparus un par un.

5.2 Découpage de l'image

5.2.1 Soutenance 1

Identification du problème 1 :

Étant chargé du découpage, j'ai rencontré énormément de problèmes notamment sur la détection de la grille. J'ai refait de nombreuses fois la mise en place de mon code, c'est à dire de ses conditions et ses paramètres. Le problème de la non résolution du problème étant que si nous n'avions pas les coordonnées de la grille, nous ne pouvions pas découper l'image en 81 images de tailles respectifs représentant les cases de notre grille. En effet, la recherche de la grille m'a pris beaucoup de temps, de réflexion et de rage, à cause de différents bugs. Cela a représenté un frein majeur au découpage qui lui fonctionnait.

Identification du problème 2 :

Alors que j'étais dos au mur, les autres membres m'ont aidé à déboguer ma fonction. Quand je compilais cette dernière un "segmentation fault" apparaissait. Cette erreur était due à un Makefile défectueux.

Mise en place de la solution :

Pour régler le premier problème, j'ai du faire la détection de la grille afin de détecter les coordonnées x, y ainsi que la longueur de la hauteur (respectivement largeur) de la grille. Une fois cette détection réalisé, j'ai effectué un ZOOM de l'image sur la grille. Une fois le zoom réalisé, j'ai créé une nouvelle surface vide qui contient le zoom de l'image avec pour hauteur et largeur, la hauteur de la grille divisé par 9. Une fois le zoom effectué, j'ai enregistré l'image dans un fichier.

En ce qui concerne le deuxième problème, Adam, chef du projet m'a aidé à réalisé un Makefile de meilleur qualité afin de faire fonctionner la compilation.

5.2.2 Soutenance 2

Identification du problème :

Nous avons rencontré peu de problèmes direct concernant la détection de l'image et le découpage de la grille. En effet, le découpage étant déjà opérationnel durant la première soutenance, nous avons du juste changé de nombreuses fois, la fonction de détection au fur et à mesure que le traitement était changé.

Mise en place de la solution :

Ainsi, pour résoudre le problème de découpage de l'image, nous avons choisi de travailler sur l'image ayant la grille la plus dur à déchiffrer. Au final, après des heures à travailler dessus, nous avons réussi à résoudre le problème

Identification du problème 2 :

Un deuxième problème rencontré a été de trouver la longueur L, puisque si celle-ci était mauvaise, la détection l'était également, donc le découpage aussi.

Mise en place de la solution 2 :

Ainsi, Hugo a du changer de fonctionnement de détermination de la longueur L par rapport à la première soutenance. Il a du étudier plusieurs possibilités avant de choisir la meilleure.

5.3 Réseau neuronal

5.3.1 Soutenance 1

Identification du problème :

Afin d'implémenter un réseau neuronal, il faut déjà comprendre le concept, ce qui n'est pas une mince à faire. De plus, le C étant un langage plus proche du langage machine que les autres langages que nous avons pu voir, il a fallu s'adapter aux changements comme par exemple l'utilisation et la manipulation de listes. J'ai rencontré des problèmes sur l'implémentations.

Mise en place de la solution :

Après avoir eu les cours magistraux de programmation j'ai pu en apprendre plus sur la manipulation de liste et l'utilisation de pointeurs. Le C est pratique pour initialiser des listes sans tailles prédéfinies mais ayant utilisé des listes en C auparavant j'ai préféré rester dans mon confort en initialisant des tailles. Il faudra par la suite réussir à utiliser cette implémentations avec les données que mes camarades de projets auront récupérées dans leurs algorithmes.

5.3.2 Soutenance 2

Identification du problème :

Lors de cette deuxième partie il a fallu adapter le réseau de neurones pour qu'il reconnaisse des chiffres écrits et non pas une simple porte logique "XOR". En effet l'algorithme a du voir des changements dans son fonctionnement. Un autre problème a été aussi rencontré lorsque l'on voulait étudier les images récupérées après les étapes précédentes, les images n'étaient pas de la bonne taille. De plus, afin d'améliorer l'efficacité du réseau, il fallait passer par des "struct", qui sont des nouvelles notions apprises et travailler en cours d'apprentissage. Et nous avons aussi de passer des simples tableaux (ex : Input[taille]) à l'utilisation de pointeurs. Malheureusement, l'entraînement n'ayant pas pu être efficace sur toutes les images de notre maigre base de données, l'apprentissage n'était efficace que sur certains chiffres.

Mise en place de la solution :

Afin de faire en sorte que le réseau du "XOR" soit adapté il a fallu faire plusieurs choses. Premièrement il a fallu dire qu'on veut reconnaître 9 caractères, il a fallu agrandir la couche "Hidden Layer", elle possède désormais 20 cellules. Afin d'avoir une image qui puisse entrer dans le réseau de neurone, il a fallu ajuster la taille de l'image pour qu'elle mesure 28 pixels en colonnes par 28 pixels en colonne. L'utilisation de l'espace mémoire ("calloc()" et "malloc()") était assez compliquée au début étant donné que nous n'avions pas beaucoup de connaissances, mais grâce aux cours notamment nous avons pu réussir à les utiliser.

5.4 Résolution de la grille

5.4.1 Soutenance 1

Identification du problème :

Personnellement, j'ai fait face à un seul problème majeur résidant lors de la lecture de chaque caractère présent dans le fichier étant un char*, c'est à dire un pointeur vers un caractère. Il a donc fallut effectuer un choix entre travailler avec des pointeurs ou alors en dynamique mais sans pointeur.

Mise en place de la solution :

J'ai estimé devoir travailler en dynamique afin de m'assurer une gestion stable. A contrario, j'ai considéré travailler avec des pointeurs comme étant une source d'incertitude. J'ai donc créé une liste temporaire afin de passer des char* vers un int dans ma liste principale.

5.4.2 Soutenance 2

Identification du problème :

Pour la dernière soutenance un des soucis de la partie résolution de la grille a été de trouver une méthode efficace pour renvoyer une image propre et qui soit fidèle à celle de base. Après être passé par construire une image de rien et ajouter des images de chiffres au bons endroits. Néanmoins, du fait de créer une surface noir et de coller des images à fond blanc au bonne coordonner revient à avoir des lignes noir prépondérante et pas très esthétique sur l'image.

Un autre problème a été le choix de comment faire en sorte que la couleur des chiffres rajouter ne soit pas la même que ceux de base présent sur la grille.

Mise en place de la solution :

Afin de palier à ce problème j'ai créé de toutes pièces une grille correspondante et adapter à mes images afin qu'elle colle parfaitement à mes besoins. J'ai donc ensuite itérer sur chaque image et copier chaque pixel de l'image correspondante au chiffre au bon endroit. Nous avons donc une grille propre et qui est assez rapide.

6 Évolution du projet

6.1 Première Soutenance

6.1.1 Prévisions données lors de la première soutenance

Pour la prochaine et dernière soutenance, on améliorera et ajoutera des parties à notre projet. Tout d'abord lors de la détection de la grille on implémentera la transformation de Hough qui ne permettra de récupérer de manière optimale qu'importe l'image. Concernant le pré-traitement il faudra effectuer une réduction de bruit pour bien éliminer toutes les impuretés de l'image. Une interface graphique sera aussi présente qui permettra d'afficher la grille avec une image et où les couleurs des chiffres rajouter seront d'une couleur différente. Enfin le réseau neuronal permettra une apprentissage avec de la back propagation.

Ci-dessous ce trouve un tableau d'avancement pour cette soutenance et les prévisions pour la soutenance final.

6.1.2 Tableau d'avancement

	Soutenance 1	Soutenance 2
Rotation	90	100
Traitement couleur	75	100
Reconnaissance de la grille	90	100
Découpage de l'image	90	100
Résolution de la grille	100	100
Affichage de la grille résolue	25	100
XOR	100	100
Réseau Neuronal	35	100

Tableau d'avancement des soutenances (en %)

6.1.3 Avancement final

Au terme de ce long et intéressant projet nous en venons à un résultat collectif assez bon. Nous passe par plusieurs étapes. Nous avons du mettre en place des méthodologies de résolution afin d'être aussi bien en équipe que seul. Le projet touche maintenant à sa fin. Nous sommes tous très satisfaits et très fiers d'avoir réussi à produire une application fonctionnelle, un OCR auquel nous avons apporté nos touches personnelles. Le développement d'une application requiert des compétences extrêmement variées, allant des connaissances très techniques aux compétences algorithmique et numérique notamment pour la partie du traitement d'image. Pour chaque membre du groupe, ce fut une bonne expérience et pensons que nous avons collaboré de notre mieux pour ce projet, la collaboration étant une dimension importante de l'informatique et de la vie en entreprise en général. Nous sommes très enthousiastes de monter notre projet et d'en recevoir des retours et espérons que celui-ci arrivera à être utiliser plus tard.

Le seul élément non totalement fini est le réseau de neurone qui n'est complètement fonctionnelle. En effet même si il a 4 chance sur 5 de trouver le bon chiffre cela reste trop peu pour un projet comme celui-ci. Afin de l'améliorer, il aurait donc fallu largement agrandir la base de données, comme celle du MNIST qui regroupe plus de 60000 images d'apprentissages.

Hors ceci Obstacle Course Racing est donc fini !

6.1.4 Recap des éléments présents dans l'application

Notre application comporte plusieurs éléments :

- Une interface permettant de charger une image
- Plusieurs algorithme servant au traitement de l'image
- Un algorithme de découpage de l'image
- Un réseau neuronal permettant la reconnaissance partielle de chiffre
- Un algorithme de résolution de Sudoku rapide et concis à partir de n'importe quelle grille.

6.2 Représentation graphique du projet

TâcheSoutenance	Soutenance 1	Soutenance 2
Rotation	90	100
Traitement couleur	75	100
Reconnaissance de la grille	90	100
Découpage de l'image	90	100
Résolution de la grille	100	100
Affichage de la grille résolue	25	100
XOR	100	100
Reseau Neuronal	35	80

Tableau d'avancement des soutenances (en %)

7 Expérience personnelle

7.0.1 RILI Adam

Je considère ce projet comme une expérience très enrichissante personnellement. Le projet avait une réel intérêt ainsi que des facettes inattendu. Il parait au premier abord simple et facile. Mais que nenni, c'est un projet long et complexe. Ajouter a cela le temps limiter dont nous disposons l'organisation était donc un élément primordiale !

Évidemment, au niveau du travail de groupe ce n'est pas la première fois que j'en fais que ce soit à l'EPITA ou avant au lycée. Ce projet m'a permis de développer mes compétences en programmation mais aussi de gagner en autonomie. En effet l'aspect orienter de ce projet m'a appris à me débrouiller par mes propres moyens afin de résoudre les différents problèmes auxquels j'ai pu être confronter. J'ai pu également partager mes connaissances avec les membres de mon groupe et inversement ce qui à créé un véritable esprit d'équipe dans celui-ci.

Dans un élan personnel je me prononcerai quant à la gestion de ce projet en la caractérisant de optimal pour notre groupe. Le travail que nous fournissions ressort beaucoup de cet esprit de cohésion. Nous nous amusions, entraidons, aidons les uns les autres et par conséquent progressions ensemble.

Mon rôles de chef de projet m'a parmi de voir et parcourir un peu toute les parties de notre projet. J'ai trouvé que c'était intéressant car au final notre application dépend de l'implémentation simultané et dépendante du travail personnel de chacun. Donc cela nous poussaient a ne pas avoir de retard et toujours en faire plus.

Sur ma partie, les problèmes algorithmique et mathématique comme l'implémentassions de Hough ou Sobel fut un défi et le relever a été instructif et professionnellement instructif.

Notre groupe était et est encore motivé par ce projet. Ces derniers mois vu la condition dans le monde ne nous ont pas écarté de nos rôles respectifs et nos devoirs envers les uns les autres. Néanmoins certaines périodes se sont avérées contraignantes notamment par rapport à la gestion du repository Git que nous utilisions.

C'est donc heureux et fier que je termine ce projet durant ce semestre. Et j'espère que les prochains projets durant mon cursus supérieur se passe comme celui-ci.

J'en ressors grandis et avec une autonomie grandissante et florissante pour la suite, ainsi qu'une envie d'apprendre et de comprendre en pleine expansions.

7.0.2 DEVIN Léo

Lors de ce projet j'ai pu beaucoup apprendre, notamment grâce au travail de groupe mais aussi au travail personnel. M'étant penché sur la partie réseau neuronal j'ai pu m'épanouir dans de nouvelles notions très intéressantes.

A l'instar du projet S2, le projet OCR a pu susciter chez moi une participation et un attachement à un groupe. Je ne connaissais pas forcément mes camarades avant, ils sont devenus des personnes avec qui je peux travailler et m'entraider. Ces dernières semaines de travail ont été très intense et on pu montrer une bonne coordination. Nous sommes restés pendant une semaine tous les soirs sur le campus tard afin de travailler en groupe et être efficace.

Je suis fier de ce que nous avons réalisé avec mes camarades même si la reconnaissance de caractère n'est pas totalement opérationnelle, c'est le seul point négatif, car c'était ma partie et je n'ai pas pu aboutir. A travers ce projet j'ai pu aussi découvrir un nouveau langage qui est le C. Ayant adoré le C lors de ma première année, je suis agréablement surpris de retrouver des similarités et d'apprendre de nouvelles notions comme la manipulation de mémoire.

En ce qui concerne le réseau de neurone, j'avais déjà entendu parler de ce type d'algorithme auparavant. Le fait de recréer tout un réseau neuronal qui reconnaît pour la première fois une porte logique "XOR", puis reconnaître des chiffres inconnus était vraiment passionnant même si cela était plutôt quelque chose à mettre en place.

Pour remédier à cela j'ai regardé beaucoup de vidéo sur le sujet, de chercheurs ou bien même d'étudiant. Je me suis aussi beaucoup documenté sur des sites de recherche ou bien d'étudiants. Les documentations du C étant principalement en anglais, j'ai du mettre à profit ma pratique et même je suis sur qu'elle s'est améliorée

7.0.3 RATTE Hugo

L'OCR étant finit, le projet m'a apporté pas mal de notions pour la suite. Ce projet n'est pas mon premier, donc cela a été plus facile de se répartir les tâches. Malgré tout ce fut le premier sur le langage C. N'ayant jamais codé sur ce langage, bien que ressemblant au C, quelques changements et le manque de documentation m'ont quelques fois un petit peu contrarié.

Je me suis réellement épanoui dans ce groupe de travail, bien qu'on ai jamais travaillé ensemble auparavant, nous avons passé beaucoup de moments en salle machine après les cours. Ces moments étaient souvent chaleureux et dans une ambiance saine d'entraide.

Que pensez de la suite ? Eh bien, j'aimerais pourquoi pas refaire des projets dans le langage C, ou dans d'autres projets. J'aime bien les projets, bien que long et périlleux, je trouve que la satisfaction de voir son projet grandir au fur et à mesure n'est vraiment pas descriptible.

De plus, j'aimerais dans le futur, pourquoi pas travailler sur une partie différente, ne touchant pas SDL, notamment pourquoi pas sur la création d'un réseau de neurones, qui pourrait me passionner tout autant voire même peut-être plus.

Pour conclure, le projet m'a fortement plu, il m'a appris à mieux vendre un projet, travailler en équipe donc en autre écouter les autres. De plus, le projet m'a également apporté sur le plan technique. J'ai hâte de faire de nouveaux projets en ING et dans ma vie, professionnelle le plus rapidement possible je l'espère.

7.0.4 RABOUILLE Mathis

Ce long et difficile projet nous a beaucoup apporté et je pense avoir appris de nombreuses choses car je ne me suis pas tellement concentré sur une partie, ma partie étant quasiment finie à la fin de la première soutenance je me suis concentré sur d'autres choses et j'en ai profité pour aider les autres.

Cela m'a permis d'apprendre d'autres choses comme l'utilisation un peu plus poussée de GTK pour créer des interfaces utilisateurs. J'ai également beaucoup travaillé sur la fusion de toutes les parties dans un fichier principal, il a donc fallu que je comprenne parfaitement la manière dont les autres avaient pensé leurs codes et la compilation du projet en général.

Au final je pense vraiment que ce projet, même si il nous a pris beaucoup de temps, nous a permis de nous améliorer globalement en C et en travail d'équipe. De plus il y'a eu une bonne ambiance dans le groupe ce qui a facilité le travail grandement.

Je pense sincèrement que ce projet nous a apporté des choses qui nous serviront après l'école et même avant cela en ING1. Donc ne pas négliger ce projet a été un choix que je ne regretterais pas aussi nous sommes fiers de ce que nous avons produit car nous partions de rien en C et dans la plupart des domaines abordés dans ce projet.

8 Annexes

8.1 Bibliographie et références

Pour le réseau de neurone

-> SoftMax :

<https://codereview.stackexchange.com/questions/180467/implementing-softmax-in-c/>

-> X-OR :

<https://github.com/Frixoe/xor-neural-network/blob/master/XOR-Net-Notebook.ipynb/>

-> Réseau de neurone :

<http://neuralnetworksanddeeplearning.com/chap1.html>

Pour SDL

-> SDL et SDL2 :

<https://www.libsdl.org/>

-> SDL image :

https://www.libsdl.org/projects/SDL_image/

-> SDL gfx :

https://www.ferzkopp.net/wordpress/2016/01/02/sdl_gfx-sdl2_gfx/

Pour Gtk

<https://www.gtk.org/>