OCR (Optical Character Recognition)
by Obstacle Course Racing

Rapport De Soutenance 1

Participants :
Adam RILI
Mathis RABOUILLE
Hugo RATTE
Léo DEVIN

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

28 Octobre 2021

Sommaire

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. 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 a dire un chiffre. Il faut donc passer par la détection de la grille sur l'image puis par la découpe a la bonne taille de chaque petit carré pour les stocker respectivement dans une nouvelle image.

–> **Résolution de la Grille :**
Dans cette troisième partie gère la résolution de la grille et l'affichage de cette dernière. L'algorithme proposer montrera son efficacité face a des grille de complexité variante du plus simple au plus exigent.

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

2 Obstacle Course Racing

2.1 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.

2.1.1 Adam

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.

2.1.2 Mathis

Lors de mon travail j'ai eu quelques problèmes au début car je n'était 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'implémentions. 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.

2.1.3 Hugo

Lors de la réalisation du projet, nous avons rapidement fixé le fait que je fasse la partie découpage des images. 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 ING 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.

2.1.4 Léo

J'ai pu beaucoup apprendre lors de mes précédents projets notamment lors du projet s2 ou nous devions forme un groupe soude et productif afin de finaliser 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.

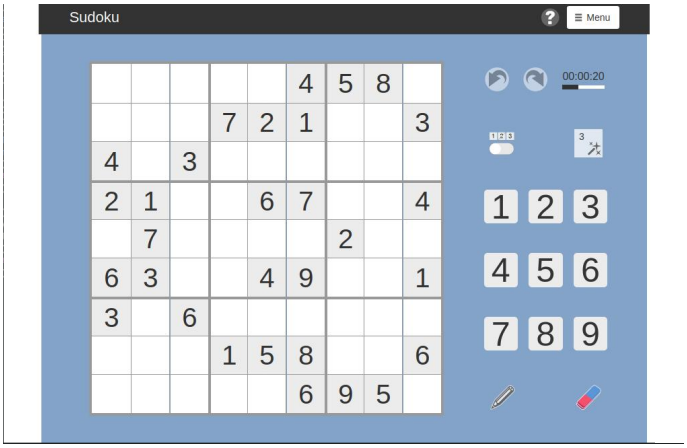
2.2 Répartition des taches

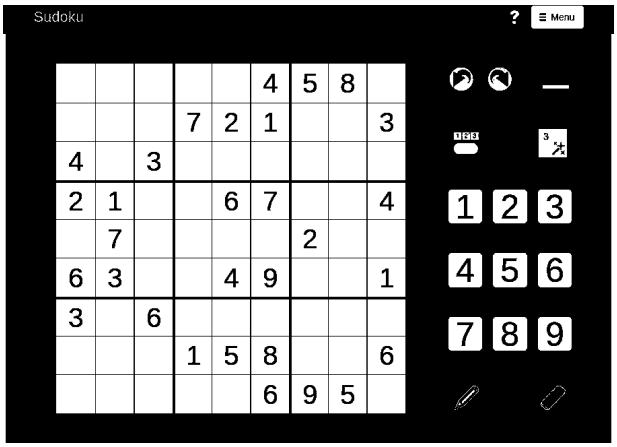
Voici maintenant l'explication de fonctionnement précis de notre programme dans l'ordre logique de son exécution.

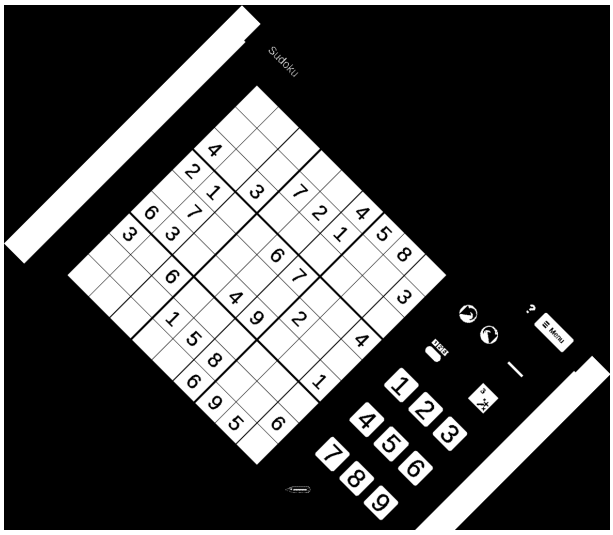
D'abord le pré-traitement

2.2.1 Mathis

Lors du pré-traitement le programme va s'occuper de récupérer l'image dans l'endroit ou on l'a enregistrer a partir d'un argument donné à l'exécution du programme. On vas superposer plusieurs filtre afin de supprimer un maximum d'impuretés et permettre un visualisation de l'image la plus efficace pour le traitement qui vas lui s'occuper du découpage de l'image. 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 tout 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. Ensuite on augmente le "Gamma" à un contraste le plus efficace pour la fonction qui suit qui est un "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. L'origine d'une image se trouve dans le coin supérieur gauche et les pixels sont stockés dans un tableau a une dimension de taille fixe, pour accéder à un pixel nous utilisons donc la formule : pixels[y * w + x] avec w la largeur en pixels de l'image. puis pour terminer tourner dans le sens trigonométrique l'image d'un angle donné au préalable à l'aide de la fonction Rotozoom venant de la bibliothèque SDL GFX, pour le moment nous utilisons cette fonction mais nous comptons implémenter un une fonction qui effectuera la rotation d'une manière plus efficace. Pour la suite nous comptons ajouter une réduction de bruit en rajoutant des traitements à l'image tels qu'un flou gaussien par exemple. Une très grande partie de mon travail lors des prochaines semaines consistera à travailler en coopération avec Léo.







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

Puis la segmentation

2.2.2 Hugo

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 carrées 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 sauvergardé avec pour nom casexy. Ici, x et y sont les coordonnées mise en paramètre.

Ensuite le réseau neuronal

2.2.3 Léo

Pour la première soutenance, il nous était demande 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 a 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 compose d'une "Input layer" qui est un couche ou 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 biai.

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 recommande de les initialiser a 0, donc on les initialisent a l'aide d'un fonction rando() qui comme son nom l'indique va donner des valeur
aléatoires.

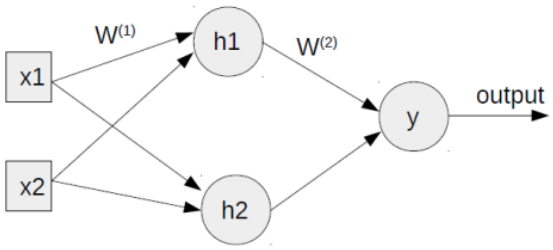
Notre algorithme va parcourir le réseau neuronal depuis l'input layer vers l'output layer un certain nombre de fois, en mettant a jour les poids de chaque layer, mais afin que l'apprentissage se fasse, il ne faut pas que les erreurs se répètent aussi : il y a un système de "back propagation", un renvoi 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 a 0 au début de la boucle. Elle est mise a jour en réalisant une comparaison : résultat attendu soustrait au résultat obtenu, au carre, tout ça produit de un demi.

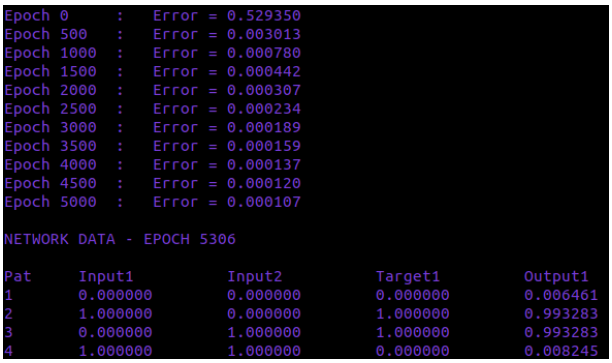
Un tableau comparatif des résultats est affiche sur la console lorsque l'algorithme a termine ses calculs. Il est compose de deux partie, l'une montrant l'évolution de la variable "Error" en fonction des époques, l'autre montre les différences entres les résultats obtenus et les résultats attendus.

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.



A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0



Enfin la résolution de la grille

2.2.4 Adam

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 a l'une des méthodes prescrites.
J'ai préféré implémenter la première qui me permet de gérer plusieurs cas de candidats impossible a 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 a 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 a 48 en int. Sachant ca une simple soustraction suivi d'un cast remplace bien le char en int. Le problème résolu il a suffit de parcourir le fichier et de prendre deux conteurs 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 amélioration 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
```

2.3 Répartition jusqu'à la première soutenance

Afin d'avancer de manière efficace dans le projet et ne pas rencontrer de problème de compréhension en mélangent les taches de chaque personne il a été décidé de, pour l'instant, découper le projet en 4 grandes partie. 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	
Decoupage de l'image		X	X	
Resolution Grille				
Traitement du fichier input	X			
Resolution du Sudoku	X			
Traitement du fichier output	X			
Réseau Neuronal				
XOR				X
Apprentissage				X

3 Difficulté rencontrées

3.1 Pré-traitement de l'image

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'ecole 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.

3.2 Découpage de l'image

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 bcp de temps, de réflexion et de rage, à cause de bug. 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 du 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 crée 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.

3.3 Réseau neuronal

Identification du problème :

Afin d'implémenter un réseau neuronal, il faut déjà comprendre le concept, ce qui n'est pas

une mince a 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 changement comme par exemple l'utilisation et la manipulation de listes. J'ai rencontre des problèmes sur l'implémentation.

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 prédéfinis mais ayant utilise des liste en C auparavant j'ai préféré rester dans mon confort en initialisant des tailles. Il faudra par la suite réussir a utiliser cette implémentation avec les données que mes camarades de projets auront récupéré dans leurs algorithmes.

3.4 Résolution de la grille

Identification du problème :

Personnellement, j'ai fait face a un seul problème majeur résidant lors de la lecture de chaque caractère présent dans le fichier étant un char*, c'est a 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 estimer 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.

4 Prevision

Pour la prochaine et dernière soutenance, on améliorera et ajoutera des parties a 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 ou 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 prévision pour la soutenance final.

4.1 Tableau d'avancement

	Soutenance 1	Soutenance 2
Rotation	90	100
Traitement couleur	75	100
Reconnaissance de la grille	90	100
Decoupage de l'image	90	100
Resolution de la grille	100	100
Affichage de la grille resolu	25	100
XOR	100	100
Reseau Neuronal	35	100

Tableau d'avancement des soutenances (en %)

5 Annexes

5.1 Bibliographie et références

Pour XOR

–> 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/