Rapport de soutenance I

Projet : Kalytera

Hafid HOUSNI

David FARGETTON

Marie LEGAY

Octobre 2021

Table des matières

1 Introduction

La reconnaissance optique de caractère (OCR : Optical Character Recognition en anglais) permet d'extraire des données d'une image en pour les rendre utilisable par l'ordinateur. C'est sur ce principe que notre projet est basé, nous demandant ainsi de créer un programme capable d'analyser une image de sudoku, d'en extraire les données pour ensuite résoudre le sudoku. Ce projet peut être découper en 3 grandes parties, la premières étant le pré traitement de l'image permettant de réduire les impuretés de l'image pouvant nuire au bon fonctionnement du programme. Ensuite nous avons le réseau de neurone qui permet de reconnaître les caractères et finalement la résolution de sudoku, le tout avec une interface graphique claire et intuitive. Ce projet doit être réaliser en langage C et s'étend sur un semestre entier.

Pour la réalisation de ce projet nous avons formé un groupe de trois élèves qui a pour nom Kalytera. Ce nom de groupe vient du grec "kalýtera" qui signifie "meilleur" car nous sommes confiant en nos capacités a réalisé ce projet. Notre groupe est composé de 3 membres : Hafid HOUSNI, Marie LEGAY et David FARGETTON, venant tous de la classe C2 promotion 2025.

Dans ce premier rapport de soutenance, on vous présentera tout d'abord la répartition des tâches ainsi que les avancements sur ce projet, pour ensuite détailler les parties réalisés, avec nos difficultés éventuelles lors de ces étapes, et on terminera par une brève conclusion.

2 Présentations personnelles

Hafid HOUSNI

Pour présenter mon parcours, je viens d'un baccalauréat S-SVT spécialité mathématiques. Je suis arrivé à EPITA avec la promo 2023, puis ayant raté mon S1, j'ai fait ma Spé en tant que 2024# puis 2024, mais n'ayant pas tout validé du S3, je dois refaire certains modules dont celui d'IP.

L'informatique est un domaine dont je suis passioné, tout comme les jeux vidéos ou les nouvelles technologies en général.

Mon objectif evec ce projet et d'acquérir de nouvelles connaissances et d'enrichier mes compétences en programmation à travers le développement d'un programme OCR.

Le projet est une excellente opportunité qui me permettra de m'améliorer et me perfectionner en tant qu'étudiant à EPITA avant d'entrer dans le monde professionnel.

David FARGETTON

Pour présenter rapidement mon parcours scolaire, je viens d'un BAC S SVT spécialité ISN (Informatique et Science du Numérique), je suis ensuite parti à l'ECE une école d'ingénieur en électronique où j'ai raté mon année, suite à quoi je me suis réoriente vers l'EPITA car l'informatique de passionnait. L'année passé mon groupe et moi avons réalisé un projet nommée Bingy qui était un réseau social sur le thème cinématographique. J'ai quelque lacunes en math qui n'ont pas facilité mon travail sur ce projet mais je fais mon possible. Partir sur un projet plus technique m'a permis de me concentrer non pas sur l'originalité du projet, mais plus sur le plan technique et mathématique qui sera probablement très présent dans mes projets futurs. Malgré un début difficile avec le langage C, je compte bien m'investir dans ce projet.

Marie LEGAY

Je viens d'un bac S-SVT avec la spécialité ISN, Infomatique et Science du Numérique. L'informatique est un sujet auquel j'ai commencé à m'intéresser durant mon adolescence, particulièrement dû à ma passion pour les jeux vidéos. Même si au fil du temps, cet intérêt pour la création des jeux vidéo c'est estompé, ma curiosité pour les autres domaines liés à l'informatique est resté et, au moment de choisir la suite de mes études, je me suis tournée vers l'EPITA.
Ce projet est une opportunité de découvrir un nouvel aspect de l'informatique : la reconnaissance de caractères.

3 Avancement du projet

3.1 Répartition des tâches

Dans le tableau ci-dessous, vous pourrez voir la répartition des charges de travail :

Tâche	David	Marie	Hafid
Chargement d'une image			X
Suppression des couleurs			X
Prétraitement			X
Détection de la grille	X		X
Détection des cases de la grille		X	
Récupération des chiffres présents dans les cases		X	
Reconnaissance de caractères		X	
Reconstruction de la grille			
Résolution de la grille	X		
Affichage de la grille résolue	X		
Sauvegarde de la grille résolue	X		

3.2 Avancement général

Voir ci-dessous l'état estimé et actuel d'avancement du projet.

Tâche	Avancement prévu	Avancement réel
Chargement d'une image	100%	100%
Suppression des couleurs	100%	100%
Pré-traitement		60%
Détection de la grille	100%	20%
Détection des cases de la grille		
Récupération des chiffres présents dans les cases	5%	5%
Reconnaissance de caractères	10%	10%
Reconstruction de la grille	0%	0%
Résolution de la grille	100%	100%
Affichage de la grille résolue	0%	0%
Sauvegarde de la grille résolue	0%	0%

4 Chargement de l'image

La reconnaissance optique de caractère (OCR) doit commencer par le chargement d'une image de type bitmap afin de pouvoir l'exploiter et aussi de l'afficher.

Afin d'exploiter l'image en entrée du programme OCR, il nous a paru idéal d'utiliser les bibliothèques SDL2 et SDL2_image pour prendre en charge plusieurs formats de fichiers de fichiers en plus du .BMP, nativement supporté par SDL2.

La bibiothèque SDL2 nous permet d'utiliser le type "SDL_Surface *" qui est un pointeur vers l'image sous forme de matrice longueur*largeur.

5 Suppression des couleurs

La suppression des couleurs de l'image s'effectue en deux étapes : le passage en niveaux de gris, puis la binarisation (passage en noir et blanc).

5.1 Niveaux de gris

Une image contient des pixels. Chaque pixel est la combinaison de 3 couleurs : le rouge, le vert et le bleu. Chaque couleur forme un octet donc la valeur décimale est comprise entre 0 et 255.

Pour appliquer un niveau de gris à l'image, la méthode la plus évidente est de calculer la luminance, c'est à dire la somme des couleurs divisé par 3, puis de le réinjecter à chaque couleurs.

Néanmoins, la luminance telle quelle ne permets pas toujours de distinguer le bon niveau de gris. Ainsi, on utilise une formule mathématique beaucoup plus appropriée :

$$luminance = 0.212671 * rouge + 0.715160 * vert + 0.072169f * bleu$$

Image originale

image en niveau de gris

5.2 Binarisation (noir et blanc)

Après avoir passé notre image en niveau de gris, nous allons le passer en noir et blanc. Nous avons implémenté deux méthodes différentes pour cela.

5.2.1 Méthode d'Otsu

La méthode d'Otsu est sûrement la plus intuitive.

A partir d'un seuil obtenu par l'algorithme d'Otsu, si la luminance est inférieure au seuil, le pixel sera blanc sinon il sera noir.

Afin de déterminer ce seuil, l'algorithme d'Otsu va utiliser un histogramme balayant tous les niveaux d'intensité (de 0 à 255).

Ensuite, on parcourt tous les seuils possibles et on effectue des calculs afin de déterminer une variance pour chaque classe d'intensité.

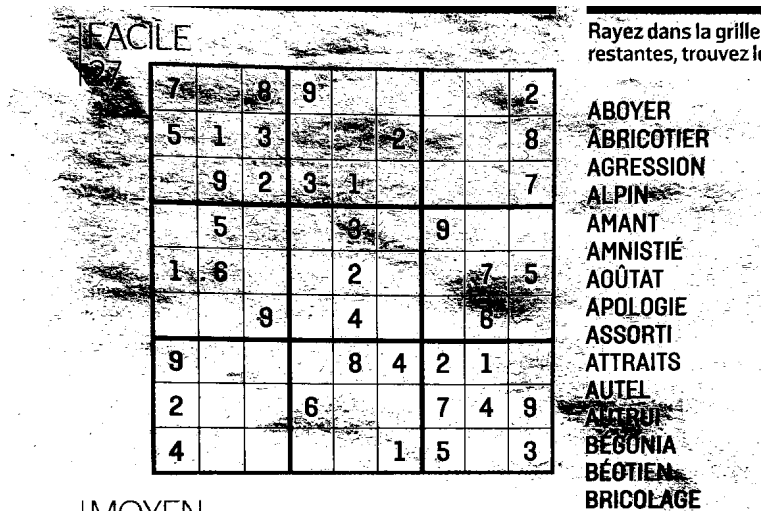
La variance interclasse est donnée par :

$$\sigma_b^2(t) = w_1(t)w_2(t)[\mu_1(t) - \mu_2(t)]^2 \text{ où } \mu_i \text{ est la moyenne de la classe i}$$

Le seuil à appliquer correspond au maximum des $\sigma_b^2(t)$

L'inconvénient de cette méthode est qu'elle utilise un seuillage global. Ainsi, elle n'est pas adapté pour les images contenant du bruit.

Pour ces dernières, on préférera utiliser une méthode de seuillage local.



Exemple de résultat non adapté

5.2.2 Méthode de l'Adaptive Thresholding

Si l'image utilisée en entrée du programme OCR contient du bruit, par exemple, une tâche de café, l'algorithme d'Otsu montrera ses limites.

Après plusieurs heures de recherches, et plusieurs méthodes testées, nous obtenons de bons résultats avec l'Adaptative Thresholding (une adaptation de la méthode de Wellner).

La méthode est démontrée sur ce lien : `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.420.7883&rep=rep1&type=pdf`

Cette méthode utilise une matrice d'image intégrale définie par cette formule :

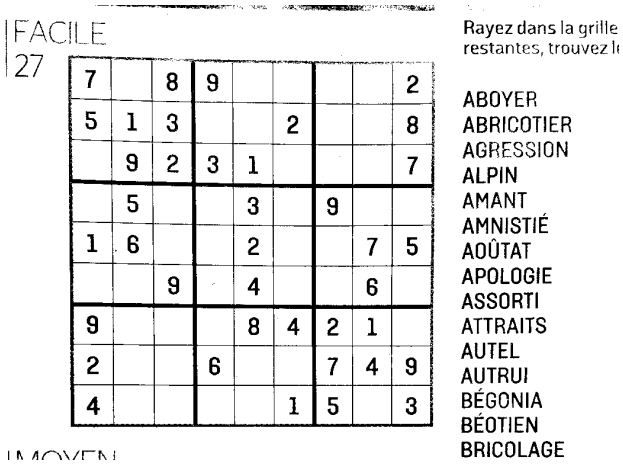
$$I(x, y) = f(x, y) + I(x - 1, y) + I(x, y - 1) - I(x - 1, y - 1)$$

où I(x, y) est la somme des termes f(x, y) à gauche et au dessus du pixel (x, y)

Ensuite, on compare le pixel (x,y) à la moyenne des pixels qui l'entourent. Si la valeur du pixel est à $t\%$ inférieur à la moyenne, il deviendra noir, sinon il deviendra blanc.

La valeur de t est importante, si elle est trop faible, la binarisation laissera du bruit sous forme de tâches noires, et si elle est trop élevée, des pixels pourraient être blanchis par erreurs.

Après plusieurs tests, on a déterminé que la valeur t à 60% était idéale pour les images ayant du bruit.



Exemple de résultat

6 Pré-traitement

On désigne par prétraitement l'ensemble des opérations à effectuée sur l'image.

6.1 Suppression des bruits

Une fois la suppression des couleurs effectuée, il peut subsister du bruit.

Afin de supprimer ce bruit, on peut utiliser un filtre médian.

On utilise une fenêtre 3*3 avec le pixel à modifier au centre et les pixels aux alentours pour les autres cases de la fenêtre.

Ensuite on trie les pixels par ordre croissant.

Le pixel à réinjecter correspond à l'élément médian de la fenêtre

6.2 Redressement de l'image

Pour l'instant, le redressement de l'image consiste en une rotation de l'image par un angle donné par l'utilisateur.

La rotation s'effectue en appliquant les formules de trigonométrie.

Pour la prochaine soutenance nous prévoyons d'implémenter une rotation automatique, sans avoir à entrer un angle.

7 Détection de la grille et de la position des cases

7.1 Filtre de Sobel

Le filtre de Sobel est une méthode permettant de détecter les contours dans une image.

Le filtre de Sobel calcule le gradiant de l'intensité pour chaque pixel définie par cette formule :

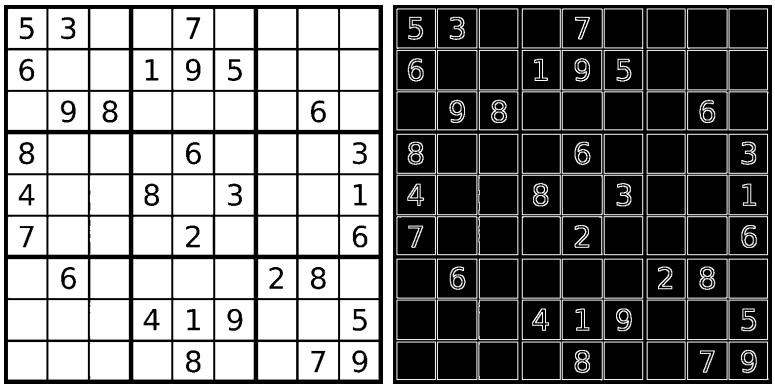
$$G_{x,y} = \sqrt{Gx^2 + Gy^2}$$

Si le gradient est ensuite reinjecté dans le pixel.

Pour calculer ce gradient on utilise deux matrices pour obtenir les valeurs de Gx et Gy sur une image A :

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \mathbf{A} \quad \text{et} \quad \mathbf{G_y} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * \mathbf{A}$$

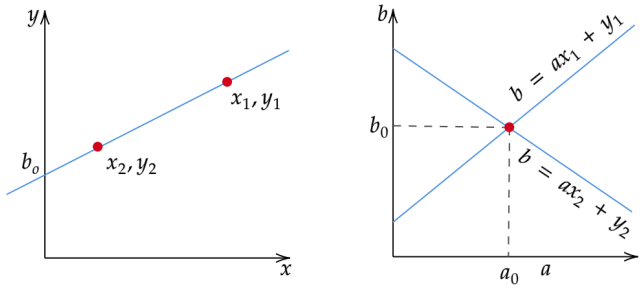
Voici ci-dessus un exemple du filtrage de Sobel (à gauche, l'image binarisée, à droite le résultat par le filtrage de Sobel)



7.2 Transformation de Hough

Afin de découper la grille il est nécessaire de détecter et récupérer les coordonnées des lignes présentes, pour cela nous utilisons l'algorithme de Hough. Nous partirons comme image de base, la "edge map" donné par le filtre de Sobel.
Pour comprendre cette transformation il faut expliquer ce qu'est l'espace de Hough. C'est un array à deux dimensions qui sur l'axe horizontal représente la pente et l'axe

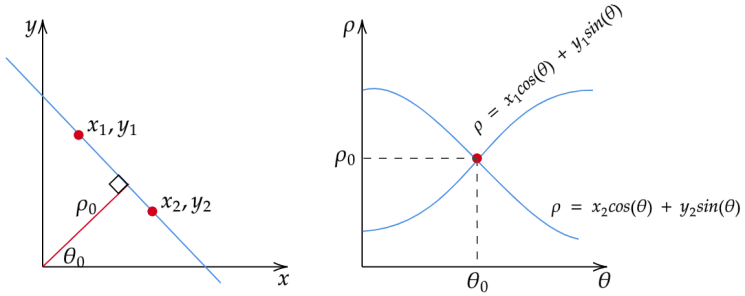
vertical l'intersection d'une droite. Chaque edge line (ligne de point blanc sur la edge map) définit par deux point $(x1, y1)$ et $(x2, y2)$ vas être représenter sur l'espace de Hough par un point de coordonnée horizontal égal à la pente de la droite et de coordonnée vertical égal à $b = a * x1 + y1$ et $b = a * x2 + y2$, le point d'intersection.



Représentation d'une droite (gauche) en un point sur l'espace de Hough (droite) en coordonnées cartésiennes

Représenter des droite en coordonnées cartésiennes a ses limites surtout quand les lignes sont vertical, il est donc nécessaire de passer sur un repère polaire. Ainsi chaque point de l'espace de Hough auras comme coordonnées (ρ, θ) avec :

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



Représentation d'une droite (gauche) en un point sur l'espace de Hough (droite) en coordonnées polaires

Les edge point sont stockés sur l'espace de Hough par un principe de vote, pour chaque ligne sur la edge map, on incrémente de 1 sa valeur à la position correspondante sur l'espace de Hough. Donc plus un point de l'espace de Hough a une

valeur élevé plus il y a de ligne de la edge image qui l'intercepte. Ainsi en prenant tous les points de la edge map ayant une valeur dépassant un certain seuil, on peut approximé la position des ligne de la grille de sudoku.

8 Réseau Neuronal

Le réseau neuronal servira à reconnaître les caractères, manuscrits ou non, présents dans la grille, qui aura été pré-traitée et découpée auparavant. Sa structure comprend 3 couches de neurones : une couche qui s'occupe des entrées (Input Layer) , une couche "cachée" qui sert au traitement des informations (Hidden Layer), et une dernière couche qui gère la sortie (Output Layer).

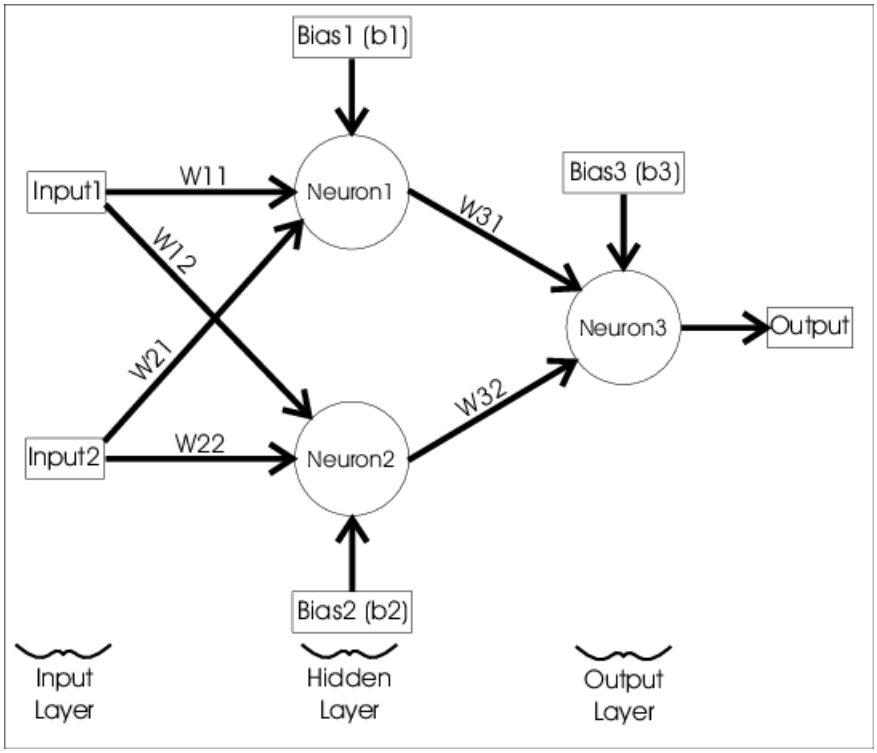


Schéma simple d'un réseau neuronal reconnaissant une porte Ou Exclusif

Le réseau sera d'abord entraîner avec des images, contenant les caractères que l'on souhaite qu'il reconnaisse.
Une fois que le réseau aura analysé et renvoyé ce qu'il détecte sur les images, la marge d'erreur, définie de façon à ce que le programme renvoi le bon résultat dans la majorité des cas dans un court laps de temps, servira à corriger son apprentissage, suivant le principe de rétropropagation (back-propagation). Cette opération met à jour les poids des neurones, ainsi que leurs biais, pour affiner le résultat et tendre vers ce qui est attendu. Suite à ce traitement, les neurones ayant créer le plus d'erreurs auront été modifiés afin d'améliorer leur résultats et qu'ils passent sous la marge d'erreur. Cette modification dépends de leur poids, mis à jour à chaque itérations, et du biais, la différence entre la sortie envoyée et le résultat attendu.
Cette phase d'apprentissage terminée, le réseau pourra reconnaître les caractères sur des images avec lesquelles il ne s'est pas entraîné et permettra alors de passer à la résolution de la grille.

Pour ce début de réseau, nous avons configuré un mini-réseau apprenant la fonction ou exclusif. Ses données d'entrées sont cette fois ci des couples x, qui vaut 0 ou 1, et y , qui vaut lui aussi 0 ou 1, indépendants. Les sortis sont 0, si le couple (x,y) vaut (0,0) ou (1,1), ou 1, dans les autres cas.
Le poids des neurones est mis à jour après chaque itération d'analyse d'un tableau de couples, généré aléatoirement au début. A chaque fois qu'un tableau est fini d'être évalué par le réseau, la marge d'erreur est calculée. Si elle est suffisamment basse, l'algorithme peut sortir de sa phase d'apprentissage. Sinon, les poids des neurones cachés sont mis a jour et une nouvelle itération se lance.
La fonction sigmoïde se sert des entrées et du poids des neurones pour calculer le renvoi de chaque neurone et, par conséquent, les sorties du réseau, tandis que la rétropropagation se fait de la couche de sortie vers la couche dite cachée.

9 Résolution sudoku

Pour produire un projet complet, il est nécessaire d'avoir un bon algorithme de résolution de sudoku. Après recherche une méthode est ressortie, le backtracking. C'est une méthode qui consiste à parcourir le sudoku case par case, choisir la première valeur valide pour cette case puis passer à la case suivante. Lorsque le programme rencontre une case où il ne peut pas trouver de valeur valide, le programme vas retourner en arrière et prendre la valeur suivante et ainsi de suite.
Elle se nomme "backtracking" car c'est une méthode qui revient sur ses pas lors d'un blocage pour trouver le chemin approprié.

La première étape pour réaliser ce programme était de construire les fonctions "load" et "save" permettant respectivement de récupérer sous la forme d'un array une grille dans un fichier texte et sauvegarder une grille sur un fichier texte.
Pour la fonction load je lis le fichier caractère par caractère, si c'est un chiffre je l'ajoute dans mon array, si c'est un point j'ajoute un 0 et finalement si c'est un espace ou retour à la ligne, je l'ignore et continue. La fonction save suit le même principe dans le sens inverse.

Avant de me lancer sur l'algorithme principal il me faut quelque fonction complémentaire. La première transforme un carré de 3 cases par 3 cases en un array à 1 dimension. La deuxième fonction permet de vérifier si un nombre à une position donnée dans la grille est valide.
Pour cela le programme parcourt la ligne, la colonne et le carré où il est et vérifie si le chiffre en question est dedans. Pour finir il me fallait un fonction qui vérifie si la grille est résolue, pour cela le programme parcours la grille entière et vérifie si il existe un 0 dedans. Si il y a un 0 alors la grille n'est pas résolue.

L'algorithme principal parcours chaque case de la grille, si un nombre est déjà dedans alors il passe à la case suivante sinon il prend le premier chiffre valide pouvant rentrer dedans et relance l'algorithme avec la grille contenant la nouvelle valeur. Si aucun chiffre n'est valide alors l'algorithme sort de la boucle, reviens à l'appel

précédent qui vas tester la valeur suivante pour la case et recommencer le processus. Après chaque appel récursif l'algorithme vérifie si la grille est compléter, si oui l'algorithme s'arrête et retourne la grille complété.

10 Conclusion

Ce projet est mené à bien par un groupe de trois étudiants en informatique motivés et prêts a y consacrer le temps qu'il faut. Le projet se décompose en trois grands parties : le traitement d'image, l'apprentissage par un réseau de neurones et l'interface utilisateur. Nous avons déjà réaliser le chargement de l'image ainsi le pré-traitement avec entre autre la binarisation et le filtre de sobel, le découpage de l'image est commencé mais n'est pas assez avancé pour un faire une démonstration et finalement le réseau neuronal a fait ses preuves en apprenant la porte logique XOR.

Le projet est sur la bonne voie, les bases de la suite sont déjà présentes et permettront de continuer sans encombre.