THE SUDO PACMAN project

Premier rapport de projet - Projet de S3

2023

Chef de projet
Lucas ESTRADE.

Membres
Johan BOURDAIS, Arthur FOURCART, Ayoub HAJJI.

Table des matières.

1 *Introduction*

Dans le cadre de notre seconde années préparatoire à l'école d'Epita, il nous est proposé un projet de trois mois, par groupe de quatre élèves, mettant en pratique nos connaissances et nos compétences acquises lors de nos premières années. La finalité de ce projet est de nous faire réaliser un logiciel de type OCR (Optical Character Recognition ou reconnaissance optique de caractères) ayant pour but de résoudre des grilles de sudoku. Pour y parvenir, le projet a été découper en quatre catégories. La première catégorie est le pré-traitement d'image, cette partie va nous permettent d'obtenir une image utilisable par notre logiciel à partir d'une image de basse qualité. La seconde catégorie est le traitement de l'image, dans cette catégories le logiciel va détecter les limites de la grille présente dans notre image et va la découper en plusieurs cases comportantes soit un chiffre soit rien. La troisièmes catégorie est le réseau de neurones, cette partie va nous permettent de faire le lien entre notre image et notre algorithme de résolution de sudoku. Le réseau va lire les cases donner par notre traitement et va récupérer l'élément présent à l'intérieur. La dernière partie est le résolveur de sodoku qui est comme son nom l'indique l'algorithme qui résoudra nos sudokus.

Pour cette première soutenance, nous débuterons ce rapport par une présentation de notre équipe SUDO PACMAN. Ensuite, nous aborderons plus en détails les différentes catégories évoquées précédemment. Nous continuerons ensuite par rappeler les moyens que nous avons à notre dispositions pour réaliser ce projet. Pour finir, nous parlerons de l'avancement de notre projet pour cette soutenance.

2 *Présentation de l'équipe SUDO PACMAN*

Dans cette partie, nous allons vous présenter le groupe SUDO PACMAN, les membres qui le composent et leurs charges dans le projet.

2.1 Présentation de l'équipe

Notre équipe s'intitule SUDO PACMAN et à pour objectif de réaliser un logiciel résolvant des sudokus. Ce groupe est composé de Johan Bourdais, Lucas Estrade, Arthur Fourcart, Ayoub Hajji, quatre élèves provenant tous de la classe C1.

2.2 Présentation des membres

Johan BOURDAIS :

Depuis tout petit, j'ai toujours été curieux du monde qui m'entourait. En grandissant, je me suis rendu compte que l'informatique prenait beaucoup de place dans ce dernier. Je m'y suis donc intéressé et je me suis réellement lancé dans la programmation avec EPITA. J'appréhende beaucoup ce projet car je suis en charge du réseau de neurone et je n'ai aucune connaissance en terme de machine learning. Mais cela ne me démotive pas car c'est un domaine qui m'intéresse beaucoup.

Lucas ESTRADE :

J'ai commencé la programmation en apprenant le langage HTLM et CSS en seconde, puis j'ai pratiqué un petit peu de python dans ma spécialité de première et terminale, Sciences de l'Ingénieur. Ces pratiques du code m'ont amené à rechercher une école d'informatique pour mes études supérieures, et c'est ainsi que je suis tombé sur EPITA durant un salon étudiant. J'ai can- didaté pour l'école, et j'ai été fort-heureusement été admis. Je suis globalement quelqu'un de très passionné et j'aime beaucoup m'investir à fond dans ce qui m'intéresse, la pro- grammation en fait partie, certes, mais les bus, les métros, ou encore le tennis, sont parmi mes plus grandes passions. Cela fait donc quelques années que j'ai un penchant pour l'informatique, et, ce projet, qui représente quelque chose de très nouveau

pour moi comparé à tout ce que j'ai pu faire, m'excite et m'intrigue tout particulièrement.

Arthur FOURCART :

J'ai toujours voulu construire tout ce qui me passait par la tête. J'ai très vite compris que pour réaliser mes projets les plus amusants, j'allais devoir passer par de la programmation. C'est pour cette raison que j'ai décider de rejoindre EPITA sans aucune expérience en programmation. Dans ce début de projet de résolution de sudoku, j'ai eu quelques problèmes de mauvaise gestion de mon temps mais j'ai quand même pu aider sur le réseau de neurones. Je ne me démotive pas pour autant la partie du réseau de neurones m'intéresse fortement et je compte bien avancer dessus pour la prochaine soutenance.

Ayoub HAJJI :

Je suis un passionné d'apprentissage, toujours avide d'acquérir de nouvelles compétences et expériences. Mon intérêt pour le travail en équipe m'a conduit à rejoindre l'EPITA, une école qui valorise la collaboration. Dans ce projet Sudoku, j'ai joué un rôle essentiel dans le prétraitement d'images, où j'ai acquis des compétences précieuses liées à la manipulation d'images et à l'optimisation des performances.Mon rôle consistait à préparer l'image d'entrée de manière à ce qu'elle puisse être traitée efficacement par les étapes ultérieures de notre algorithme de résolution

2.3 Répartition des charges

Personnes / Tâche	Lucas	Johan	Arthur	Ayoub
Chargement d'image				X
suppression des couleurs				X
Rotation de l'image	X			X
Détection de la grille et des cases	X			
Découpage de l'image	X			
Algorithme de résolution		X		
Réseau de neurones		X	X	

3 *Pré-traitement de l'image*

Dans le cadre de notre projet visant à développer une application en langage C capable de résoudre des grilles de sudoku à partir d'images, une phase se démarque des autres,

pour son importance au bon fonctionnement du projet. Cette phase est la phase de pré-traitement de l'image. Cette phase consiste à préparer l'image d'entrée de manière à ce qu'elle puisse être traitée efficacement par les étapes ultérieures de notre algorithme de résolution. Cette étape est essentielle pour garantir la précision et la rapidité de notre application.

3.1 Chargement d'image

3.2 Suppression des couleurs

a *Conversion en nsolveur d'équationiveau de gris*

La première étape de pré-traitement a consisté à convertir l'image couleur en une image en niveaux de gris. Cette opération simplifie considérablement le traitement ultérieur en réduisant la complexité des données tout en préservant les informations clés nécessaires à la résolution du sudoku. Nous avons utilisé un filtre de conversion en niveaux de gris pour obtenir une image monochrome, qui servirait de base pour nos étapes suivantes

b *Binarisation de l'image*

Une fois l'image convertie en niveaux de gris, nous avons procédé à la binarisation de l'image. Pour cela, nous avons calculé la valeur moyenne des pics de l'histogramme des luminances des pixels de l'image. En fixant un seuil approprié, j'ai transformé l'image en une image binaire, où les pixels ayant une luminance supérieure au seuil étaient considérés comme blancs, tandis que les pixels ayant une luminance inférieure au seuil étaient considérés comme noirs. Cette étape a permis de segmenter efficacement les chiffres de la grille du reste de l'image

3.3 Rotation de l'image

Une étape cruciale de notre processus de prétraitement de l'image consistait à garantir que l'image de la grille de sudoku soit correctement orientée pour faciliter la détection des cellules de la grille et la reconnaissance des chiffres. À cette fin, nous avons développé un

programme dédié permettant aux utilisateurs de faire pivoter manuellement l'image au besoin. Le programme de gestion de la rotation manuelle offre une solution interactive pour les cas où l'image d'origine pourrait ne pas être parfaitement alignée. Les utilisateurs peuvent ajuster la rotation de l'image jusqu'à ce qu'elle soit correctement orientée, ce qui est essentiel pour garantir la précision de la reconnaissance des chiffres et la résolution de la grille. Cette fonctionnalité offre une grande flexibilité et permet à notre application de s'adapter à diverses orientations d'images d'entrée.

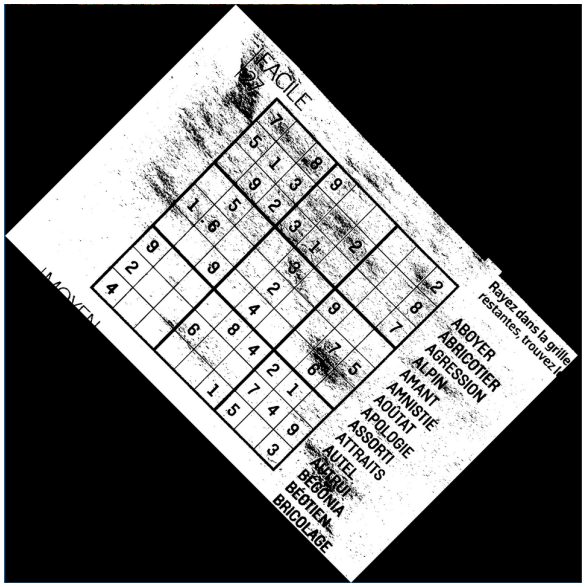


Figure 1: Grille testée avant rotation

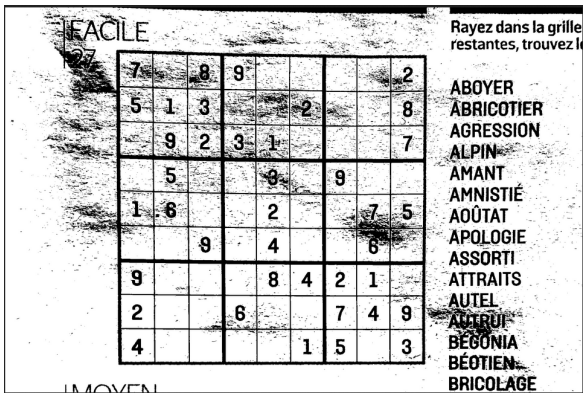


Figure 2: Grille tester après rotation

4 *Traitement de l'image*

Tout l'algorithme est présent dans le fichier "carving.c".

4.1 Détection de la grille et de la positions des cases

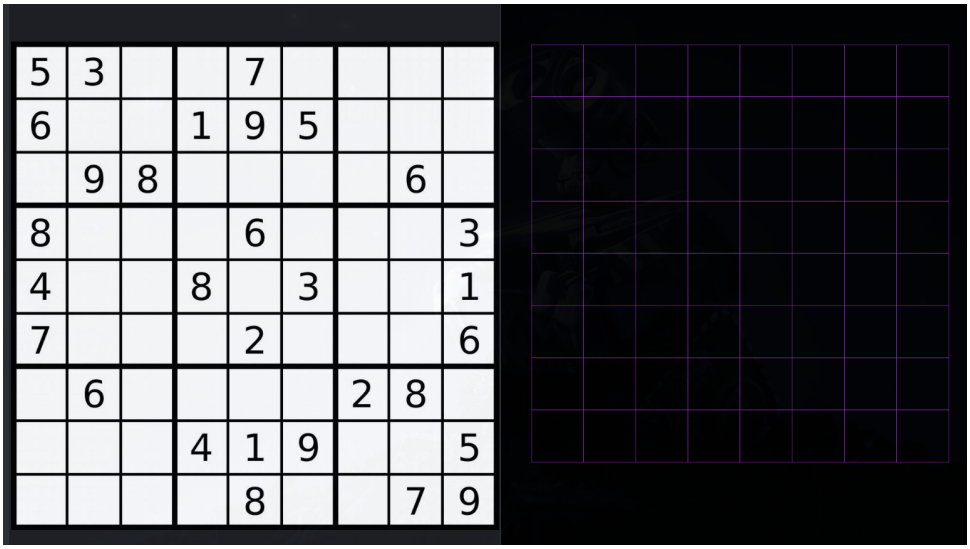
Pour la détection de la grille, nous avons utilisé une méthode simple. Le problème de celle-ci est qu'elle nécessite un prétraitement parfait, car l'algorithme ne fonctionne pas si il existe une imperfection sur l'image ou si la grille est trop petite dans un coin de l'image. Cependant cet algorithme est capable de détecter n'importe quelle forme géométrique à 4 côtés, et d'en extraire un quadrillage de 9 cases par 9. Ce qui est donc très pratique pour les grilles de sudoku. Il fonctionne de la manière suivante. Nous partons du milieu gauche de l'image, et nous projetons le plus loin possible jusqu'à croiser un pixel noir. Une fois que ce dernier est trouvé, nous partons du même endroit 10 pixels plus haut et nous répétons la même fonction. En théorie, si l'image est bien nettoyée, nous obtenons donc avec certitude 2 points distincts du côté gauche de la grille. Grâce à ces deux points on peut alors déterminer si on a affaire à une pente montante ou descendante, ou une ligne droite, mais ce cas est inclus dans un des deux autres. Le but ici est d'accéder aux 2 extrémités du côté gauche. Si nous avons affaire à une pente montante, alors, pour le bord supérieur gauche, nous parcourons la matrice de pixel à partir du milieu gauche (donc y = *hauteur / 2* et x = *0*). Et à chaque ligne de pixel, si nous croisons un pixel noir, et si la coordonée en x de ce pixel se trouve être plus élevée que le pixel avec la coordonnée en x la plus élévée précédent, alors nous le remplaçons, et ainsi de suite jusqu'à arriver en haut de l'image. En revanche pour bord inférieur gauche, nous partons encore une fois du milieu gauche, sauf que cette fois ci l'objectif est juste de récupérer le premier pixel noir sur la dernière où il y en a. Grâce à ces 2 bords que nous récupérons, nous pouvons tracer le côté gauche de la grille. Le côté droit fonctionne de la même façon, il est inutile de le détailler à nouveau. Ainsi, tous les côtés de la grille sont reliés. Ensuite chaque côté est divisé en 9 sous-segments, et les points séparant les sous-segments sont reliés à leurs opposés, qui se trouvent, vous l'aurez deviné, sur le côté opposé. Un segment ab se divise

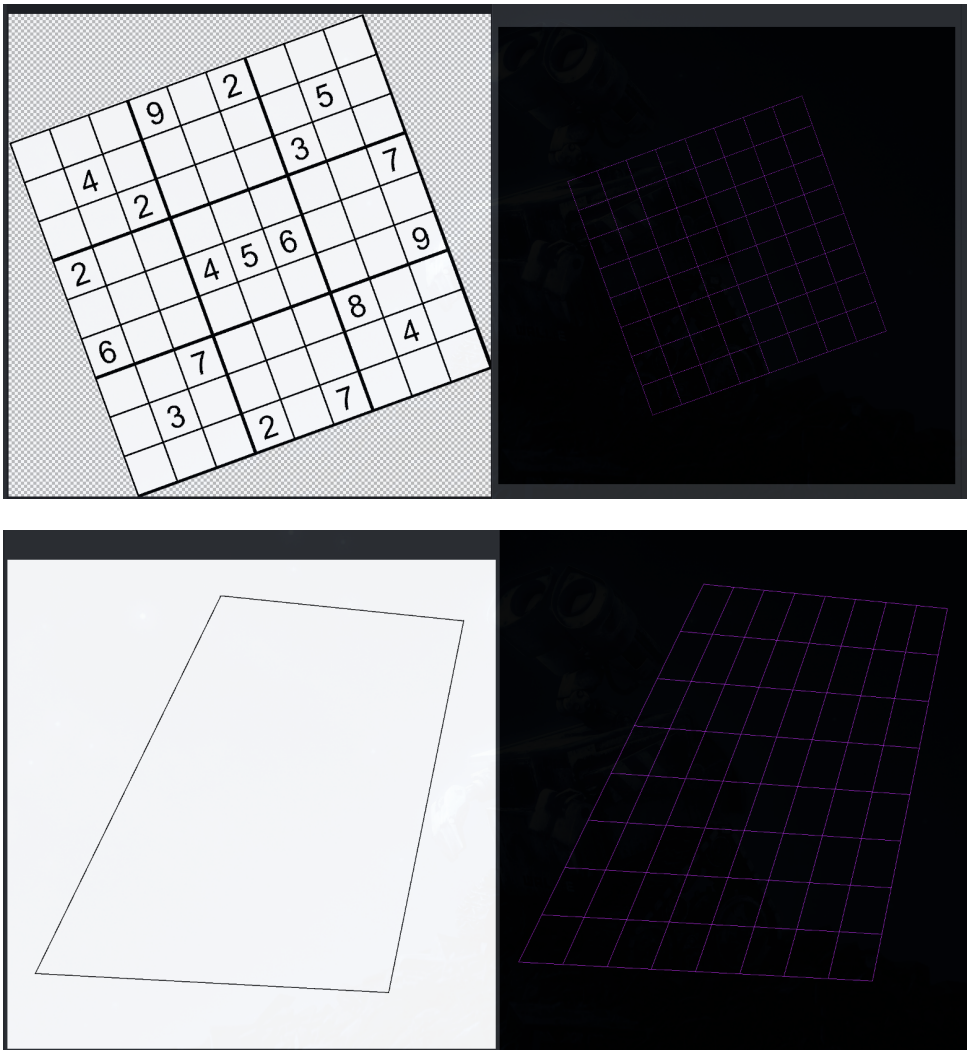
en 9 avec la formule suivante : pour la coordonnée x *a + (i / 9) * (b - a)*, pour i allant de 0 à 9. La coordonnée y est calculée identiquement. Ensuite nous calculons toutes les intersections du quadrillage créé. Ces dernières sont calculées sur le même principe. Elles seront très utiles pour le découpage.

4.2 Découpage de l'image

Comme nous possédons chaque intersection du quadrillage obtenu à la fin de la détection, le découpage se fait en parcourant le tableau à 2 dimensions des intersections. Ensuite nous récupérons les coordonnées x et y maximums de ces 4 points pour ne pas couper la case si celle-ci n'est pas droite, on crée une nouvelle surface grâce à ces nouvelles coordonnées, puis nous copions le contenu de l'image source entre les x et y maximums dans la nouvelle surface. Cette nouvelle surface correspond alors à une case, et elle est sauvegardée avec ses coordonnées x et y dans son nom comme suit : "tile_xy.png" en coordonnées matricielles. Cette étape est répétée pour toutes les cases et elles sont rangées dans le dossier grill. Cette méthode permet d'avoir des résultats satisfaisants même si la grille n'est pas droite du tout, (tester avec "utile.png"). Cependant les résidus de bordure de grille sont inévitables.

Quelques exemples :





5 Réseau de neurones

5.1 Structure globale

Pour le réseau de neurones, nous avons choisi d'implémenter une structure classique d'un réseau de neurones. Découpé en différente partie, une partie propagation avant, qui va avoir pour but de prédire une valeur de sortie, ensuite une propagation arrière qui va servir à corriger les éventuelles erreurs du réseau en ajustant les poids et biais de chaque couche du réseau.

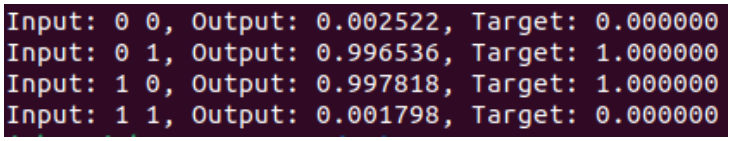


Figure 3: Résultat du XOR

5.2 Propagation avant

Pour la propagation avant, nous avons choisi d'utiliser la fonction sigmoïde comme ci-contre : f(x) = $\frac{1}{1+e^{-}x}$.

5.3 Propagation arrière

Pour la propagation arrière, nous utilisons simplement la dérivée de la fonction sigmoïde ci-contre : f'(x) = x * (1 - x).

5.4 XOR

Pour réaliser le XOR, nous avons utilisé un réseau composé de 3 couches. La première couche composée de 2 neurones qui vont contenir les deux inputs du XOR. La couche cachée elle est composée de 3 neurones et la dernière couche est composée d'un seul neurone qui nous donnera l'output du XOR. Les poids et biais de chaque couche sont initialisés en choisissant aléatoirement un nombre entre -1 et 1. Pour entraîner notre réseau, nous répétons 100000 fois la fonction de propagation arrière sur les 4 inputs différents du XOR ((0,0),(0,1),(1,0),(1,1)) avec un taux d'apprentissage de 1.

6 Résolveur de sodoku

Le solveur a été implémenté via l'utilisation du "backtracking", c'est une méthode dite de "brute force" qui va tester toutes les possibilités par récursion et ainsi trouver une solution.

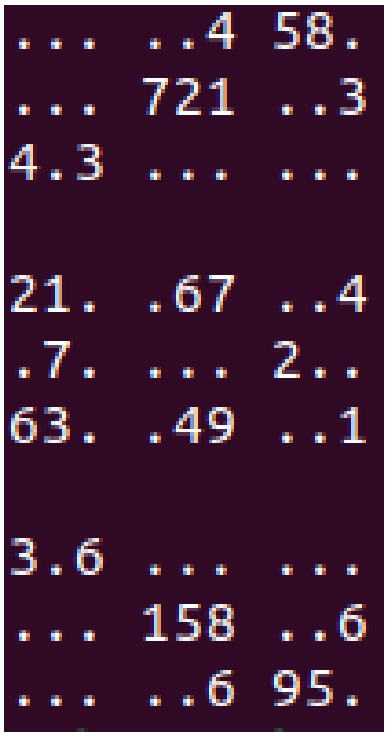


Figure 4: Grille testée

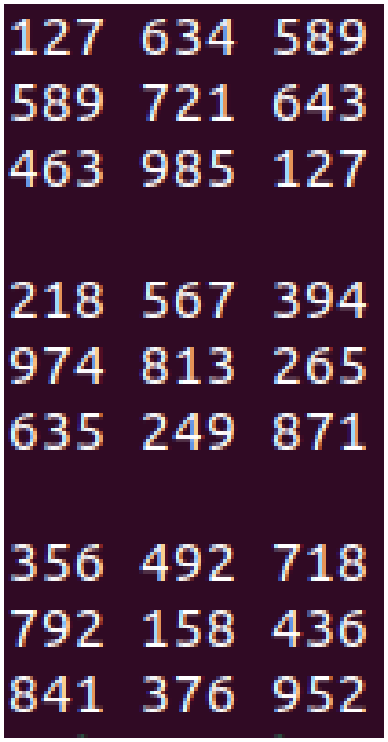


Figure 5: Résultat du solver

7 *Nos contraintes et nos moyens*

7.1 Nos contraintes

a *Langage C*

Il est impératif que tout notre projet soit exclusivement écrit en langage C.

b *Machine de l'école*

Les machines de l'école sont utilisées comme machine de référence pour évaluer notre travail. Il est donc nécessaire que notre projet compile et s'exécute sur l'environnement de travail fournit par l'école (ici Linux). L'utilisation de tous les logiciels installés sur les machines est donc autorisée.

c *Contraintes lier au code*

Les lignes de codes ne devront pas dépasser 80 colonnes et ni contenir d'espaces inutiles enfin de ligne. Les identifiants utilisés et les commentaires devront être impérativement

en anglais.

d *Contraintes de compilation*

Les codes doit pouvoir compiler sans erreurs avec au moins les options : -Wall -Wextra

e *Dépôt GIT*

Pour notre rendu il est obligatoire d'utiliser le dépôts GIT qui nous a été donner en début de projet. Notre dépot GIT ne dois seulement contenir que nos rapport en format pdf, tout le code source de notre projet, un fichier Makefile aidant à compiler notre code, un fichier README expliquant comment utiliser notre logiciel, un fichier AUTHORS contenant les informations des membres et des images qui serviront d'exemple durant nos soutenances.

7.2 Outils & Logiciels

a *GIT*

Nous utilisons GIT pour structurer notre projet, pour partager nos travaux entre les membres et pour éviter les conflits lors de notre rendu de projet.

b *Emacs ou VIM*

Ces logiciels vont nous permettre de rédiger nos programmes en langages C et de nous assister en cas d'erreur de syntaxe.

c *GNU MAKE*

Cet outil nous permet de compiler nos programmes en C plus efficacement et plus rapidement et nous averti des différentes erreurs créées par notre programme.

d *Discord*

Nous utilisons l'application Discord pour communiquer entre nous et organiser nos sessions de travail grâce à des salons structurés sur un serveur créé à cet effet.

8 *État d'avancement*

Soutenances / Tâches	1ère	2ème
Chargement d'une image	100%	100%
Suppressions des couleurs	100%	100%
Pré-traitement	25%	100%
Détection de la grille	70%	100%
Détection des cases de la grille	70%	100%
Récupération des chiffres présents dans les cases	50%	100%
Reconnaissance de caractère	20%	100%
Reconstruction de la grille	0%	100%
Résolution de la grille	100%	100%
Affichage de la grille résolue	20%	100%
Sauvegarde de la grille résolue	100%	100%

9 *Conclusion*

Pour conclure ce rapport, notre avancement pour cette soutenance est assez satisfaisant. Nous avons compléter plusieurs parties majeures de notre projet comme l'algorithme de résolution de sudoku, la détection de la grille ou encore nous avons fait nos premiers pas dans le réseau de neurones. Bien sûr le projet est encore loin d'être terminer mais nous pensons être sur le bon chemin pour le compléter.