Rapport de projet final
OCR 2023

Yannick BIA (Chef de Projet)
Raphaël MARINO
Robin PERGAUD
Alexandre MARTIN

Sommaire

1 Introduction : Présentation du projet

1.1 OCR Sudoku Solver

L'objectif de ce projet consiste à développer un logiciel de type OCR (Reconnaissance Optique de Caractères) conçu pour résoudre des grilles de sudoku. En d'autres termes, notre application est capable de prendre en entrée une image représentant une grille de sudoku, et en retour, elle affichera la grille résolue. Dans sa version actuelle notre application offre une interface graphique permettant à l'utilisateur de charger une image au format standard, de la visualiser, de corriger certains défauts éventuels, et enfin d'afficher la grille entièrement remplie et résolue. De nombreux procédés de traitement d'images sont utilisés dans cette application et nous vous laissons les découvrir à travers cette présentation générale de son élaboration. De plus, cette grille résolue peut être sauvegardée pour consultation ultérieure.

Un autre aspect clé de notre application est le réseau neurones qui permet la reconnaissance automatique des différents chiffres dans la grille.

1.2 Les objectifs

Les étapes que nous devons accomplir pour ce projet sont les suivantes :

Objectifs Globaux :

- Chargement d'une image
- Suppression des couleurs (conversion en niveaux de gris, puis en noir et blanc)
- Prétraitement de l'image
- Détection de la grille de sudoku
- Identification des cases de la grille
- Extraction des chiffres présents dans chaque case
- Reconnaissance des caractères (en l'occurrence, les chiffres)
- Reconstruction de la grille
- Résolution de la grille
- Affichage de la grille résolue
- Sauvegarde de la grille résolue

Objectifs pour la Première Soutenance :

- Chargement d'une image et conversion en niveaux de gris, puis en noir et blanc
- Possibilité de rotation manuelle de l'image

- Détection de la grille de sudoku et localisation des cases
- Découpage de l'image et sauvegarde de chaque case sous forme d'image individuelle
- Implémentation de l'algorithme de résolution de sudoku dans le programme en ligne de commande "solver"
- Réalisation d'une preuve de concept pour un réseau de neurones en implémentant un mini-réseau capable d'apprendre la fonction OU EXCLUSIF
- Sauvegarde de la grille résolue

Objectifs pour la Soutenance Finale :

- Le prétraitement complet
- Le réseau de neurones complet et fonctionnel
- Apprentissage
- Reconnaissance des chiffres de la grille
- La reconstruction de la grille
- La résolution de la grille
- L'affichage de la grille
- La sauvegarde du résultat
- Une interface graphique permettant d'utiliser tous ces éléments

Ainsi, ces objectifs marquent les étapes clés que nous avons du accomplir pour arriver au résultat actuel. Il est également important de noter que certaines étapes essentielles ne sont pas listées mais seront présentées dans ce rapport, tel que la suppression des lignes noires dans les images générées par le découpage de la grille par exemple. De plus, nous avons utilisé les bibliothèques SDL et SDL2, étant présentent sur les machines de l'école.

1.3 Présentation des membres du groupe et répartition des tâches

Yannick : Depuis toujours j'ai eu un certain intérêt pour l'informatique. D'abord j'ai comme la plupart d'entre nous commencé par regardé les grands bidouillés sur les jeux vidéos. Le modding sur les jeux a, pour moi, été la porte d'entré vers le monde de l'informatique. Puis en grandissant j'ai appris à construire mes propres PC. J'ai aussi expérimenté les limites du matériel. En effet plus jeune je me souviens avoir eu un ordinateur que j'ai adoré. Et je l'ai adoré pas parce que j'ai pu avoir des sessions de jeux passionnantes. Non, je l'ai aimé parce que pour pouvoir joué je devais passer autant de temps à essayer de l'optimiser dans l'optique du jeu vidéo qu'à effectivement joué. Je finissais carrément par mieux connaître chacun des recoins de mon ordinateur que les jeux que j'étais censé explorer et apprécier pendant mes sessions de jeu. C'est pour ça qu'aujourd'hui de nouveaux projets sont toujours une occasion de plus d'apprendre et de me lancé de nouveaux défis.

Raphaël : Je suis de nature une personne persévérante, sérieuse et qui a de l'humour lorsqu'il en faut. Un de mes points forts est que je n'ai pas de problème pour parler en public en partie grâce à mes 7 ans de théâtre qui m'ont permis de vaincre ma timidité. De plus, le codage me plaît énormément depuis que j'ai réaliser mon dernier projet Catcho's avec la Pespi Korp ! Concernant le projet, le réaliser est d'une grande difficulté mais cela me permet de me lancer dans l'inconnu et de découvrir des outils que je n'avais jamais utilisé.

Robin : Depuis ma plus tendre enfance, je me lance des défis qui peuvent paraître autant inutiles que complexes, ainsi l'accomplissement de ces projets me permet en vain d'éprouver une certaine satisfaction. En triomphant de la célèbre redstone de Minecraft et du terrifiant chat de Scratch, me voilà enfin près à me lancer dans un nouveau projet à la fois ambitieux et palpitant ! Grâce à l'expérience que j'ai acquis lors du projet S2, je suis conscient que ce projet demande une charge de travaille non négligeable. C'est pourquoi je suis prêt à redoubler d'effort afin que notre projet puisse voir le jour !

Alexandre : Je m'appelle Alexandre Martin, et j'ai toujours été passionné par l'informatique depuis mon plus jeune âge. Cette fascination a été grandement nourrie par mon père, qui travaille dans le domaine de l'informatique, et qui m'a toujours encouragé à explorer cet univers captivant. Le projet avec mes camarades se déroule à merveille, j'ai retrouvé la même motivation que pour le projet de l'année dernière , et je trouve le projet particulièrement intéressant. C'est une expérience enrichissante qui me permet d'appliquer mes compétences informatiques tout en développant de nouvelles compétences pour résoudre des problèmes complexes. Je suis impatient de voir où ce projet nous mènera.

Concernant la répartition des tâches, nous avons décidé de créer deux sous-groupes. Le premier sous-groupe, composé de Raphaël et Robin, s'occupera des trois parties suivantes : la rotation manuelle de l'image, la détection de la grille de sudoku et des cases qui la compose, du découpage de l'image et de sauvegarder chaque case sous forme d'image individuelle. Le second groupe, composé de Yannick et Alexandre, s'occupera des trois parties restantes : le chargement d'une image et la conversion en gris puis en noir et blanc, l'implémentation de l'algorithme de résolution du sudoku (le solver) et de la réalisation d'une preuve de concept pour un réseau de neurones capable d'apprendre la fonction ou-exclusif.

Cette séparation en deux groupes permet de travailler de manière mieux organiser et de diviser les tâches équitablement entre les membres du groupe. Ainsi, nous pouvons travailler en duo sur une tâche ou en solo, bien entendu, si un groupe à fini, il peut aider l'autre groupe.

2 Les approches adoptées pour la première soutenance

2.1 Les parties de Raphaël et Robin

2.1.1 Rotation manuelle de l'image

Le développement de la fonction de rotation manuelle d'une image comporte plusieurs difficultés et aspects à prendre en compte pour garantir son bon fonctionnement et son efficacité dans le cadre du projet de reconnaissance de grille de sudoku. Voici ce qu'il faut prévoir :

- Choix de l'Angle de Rotation : L'utilisateur devrait pouvoir spécifier l'angle de rotation souhaité. Il est essentiel de prendre en compte la plage d'angles autorisés et de gérer les erreurs potentielles si l'utilisateur entre un angle invalide.
- Qualité de l'Image : La qualité de l'image peut affecter la précision de la rotation. L'image pourrait être floue, pixelisée ou contenir du bruit. Des techniques de prétraitement, telles que la réduction du bruit ou l'amélioration de la netteté, peuvent être nécessaires avant la rotation. Mais ces potentielles problèmes seront gérés pour la soutenance finale et non pas pour cette première soutenance.
- Méthode de Rotation : Il existe différentes méthodes de rotation, notamment la rotation affine ou la rotation par interpolation. Le choix de la méthode dépendra de la complexité de l'image et de la précision requise.
- Perte de Données : La rotation d'une image peut entraîner une perte de données si une partie de l'image est coupée. Il convient de gérer cette perte potentielle, par exemple en agrandissant l'image avant la rotation pour s'assurer que rien d'important n'est coupé ou par diverses autres moyens.
- Performance et Temps de Calcul : La rotation d'une image peut être une opération coûteuse en termes de temps de calcul, en particulier pour des images de haute résolution. Il faudra optimiser les performances pour garantir une rotation rapide, surtout si l'application doit fonctionner en temps réel.
- Tests et Validation : La fonction de rotation doit être soigneusement testée pour s'assurer qu'elle produit des résultats précis et fiables dans une variété de cas d'utilisation. Des tests avec des images de sudoku de différentes orientations et qualités doivent être effectués.
- Gestion des Erreurs : Il est essentiel de mettre en place un mécanisme de gestion des erreurs pour traiter les éventuelles défaillances de la rotation, telles que des angles incorrects par exemple.

En résumé, la fonction de rotation manuelle d'une image est une étape cruciale du processus de reconnaissance de grille de sudoku, mais elle comporte des défis techniques et ergonomiques importants. Une planification soignée, une conception robuste et des tests rigoureux pour garantir le succès de cette fonctionnalité.

Maintenant que nous avons retracé les nombreuses difficultés que nous avons rencontrées ou que nous rencontrerons, nous allons expliquer le chemin que nous avons emprunté pour réussir une rotation manuelle. Nous avons utilisé les bibliothèques SDL et SDLimage pour gérer les étapes graphiques. L'idée principale de notre rotation pour éviter des pertes de données est la suivante : créer un carré derrière l'image originale et de créer une fenêtre SDL de la taille de ce carré.

Nous avons donc rendu l'image de fond plus grande que l'image originale pour ainsi éviter que l'image soit couper dans ses coins même à 45° (45° étant la rotation où l'image s'étend sur la surface la plus vaste). Le carré mis en fond peut donc être considéré comme une bordure supplémentaire sur l'image originale et qui permet donc que l'image soit entièrement présente quelque soit la rotation. Nous avons crée une nouvelle surface vide pour l'image résultante de la rotation.

Ainsi, pour la rotation, nous avons utilisé une rotation simple qui calcule les nouvelles coordonnées de l'image une fois l'angle appliqué. Cette rotation est dite affine, c'est une transformation linéaire qui conserve les proportions entre les points et les distances relatives, en utilisant les formules trigonométriques pour calculer les nouvelles coordonnées des pixels après la rotation par rapport à l'angle renseigné.

Nous avons géré les angles négatifs et à virgules, nous avons également libéré la mémoire lorsque c'était nécessaire et laissé la possibilité de fermer la fenêtre SDL. De plus, nous avons crée une fonction permettant de charger les images de type .jpeg. Etant donné que les images de nos tests sont de cette extension, il est essentiel pour nous de pouvoir les lires. De nombreuses fonctions internes à SDL nous ont permis de faciliter la création de cette fonction.

En résumé, notre programme charge une image, ajoute une bordure rouge (ou transparente), la fait tourner d'un angle spécifié par l'utilisateur, et affiche le résultat dans une fenêtre SDL.

Voici un exemple de son exécution avec une bordure rouge, mais cette bordure sera blanche :

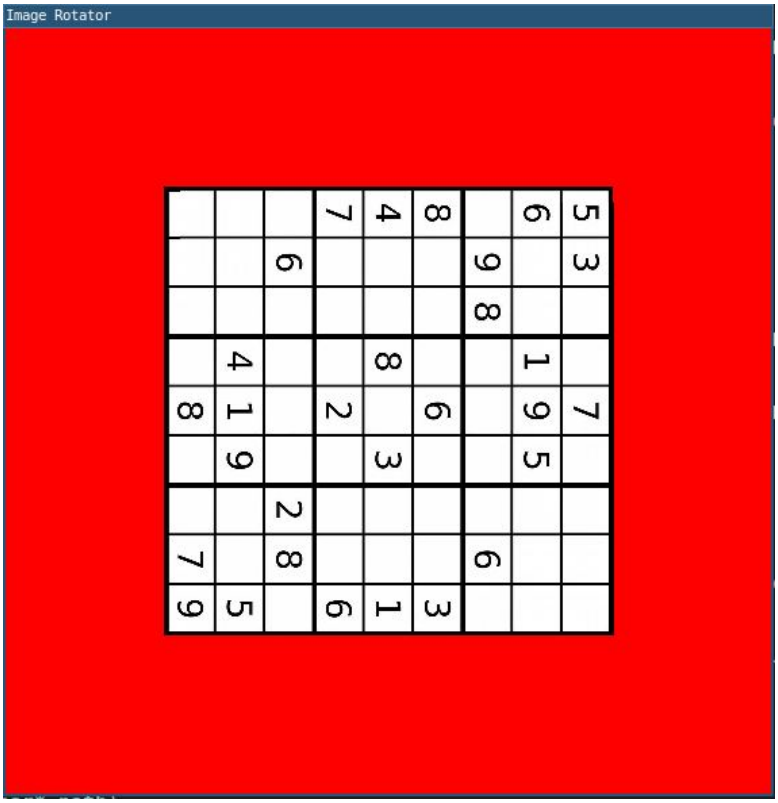

Figure 1: -90°

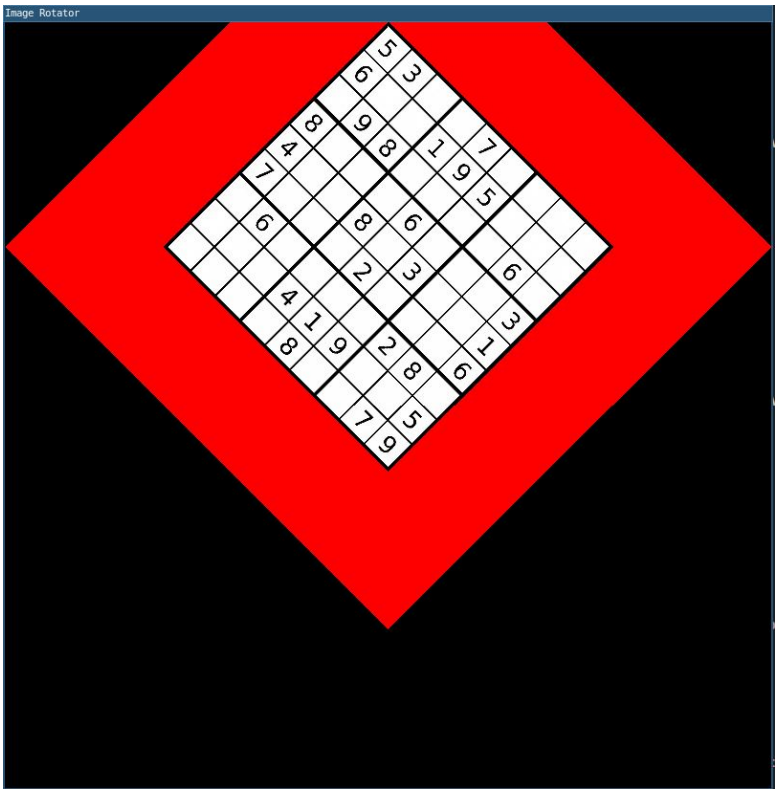

Figure 2: 45°

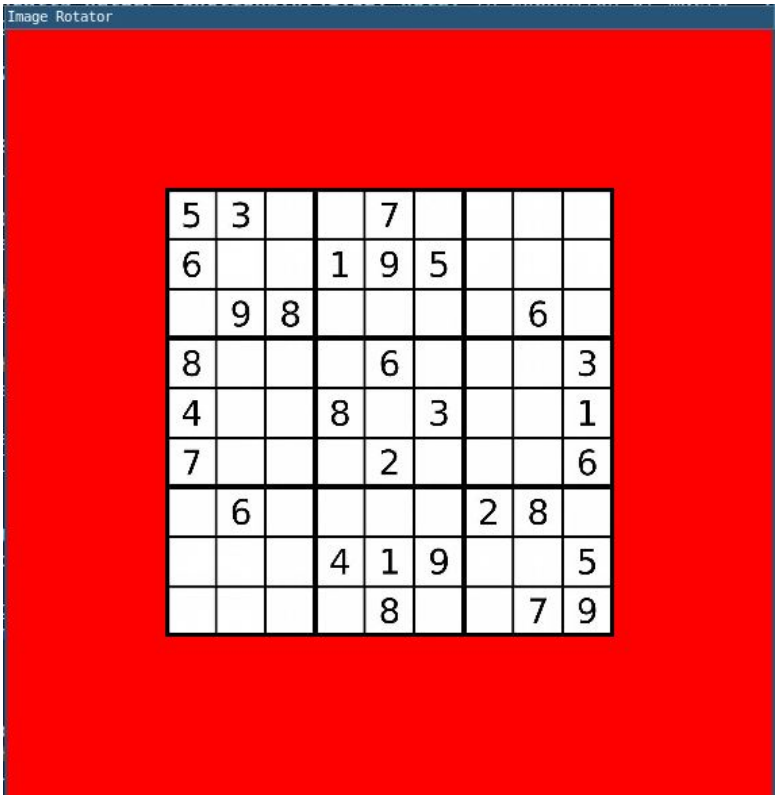

Figure 3: 0°

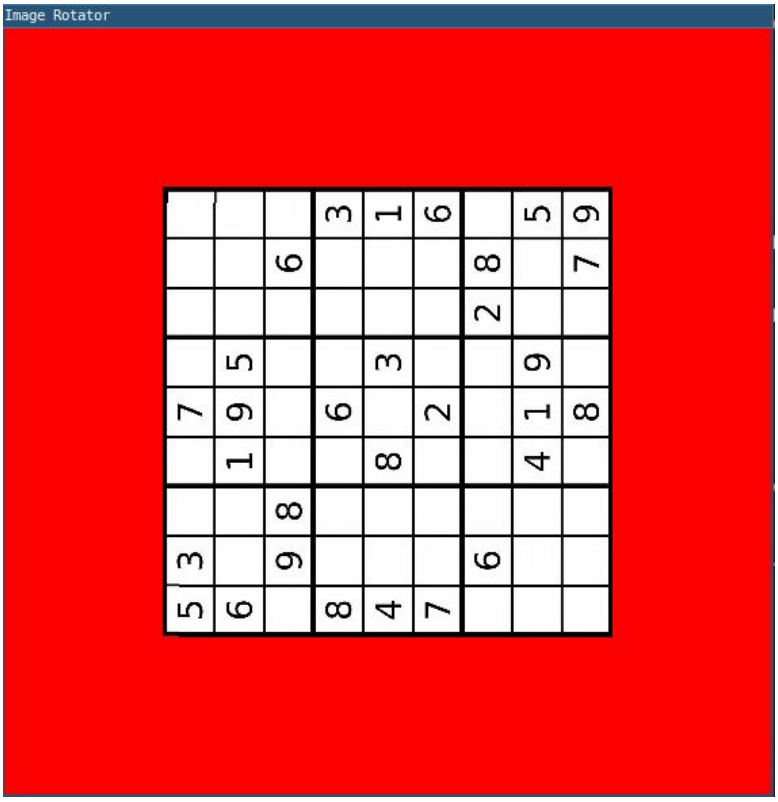

Figure 4: 90°

2.1.2 Détection de la grille et de la position des cases

La détection de la grille de sudoku et des cases est une étape cruciale dans le processus de reconnaissance de grille de sudoku. Voici les difficultés et les éléments à prendre en compte lors du développement de cette fonctionnalité :

Détection de la Grille de Sudoku :

- Variabilité des Grilles : Les grilles de sudoku peuvent varier en termes de taille, d'orientation, de couleur et de qualité de l'image. Il est essentiel de développer un algorithme de détection robuste capable de gérer cette variabilité.
- Prétraitement d'Image : L'image peut contenir du bruit, des artefacts ou des distorsions. Un prétraitement de l'image, tel que la réduction du bruit et l'amélioration de la netteté, est nécessaire pour améliorer la qualité de l'image. Mais cette difficulté ne sera peut-être pas optimale dès la première soutenance.
- Méthode de Détection : Il existe différentes méthodes de détection de grille de sudoku, telles que la détection de lignes et de coins, la détection de formes ou l'utilisation de réseaux de neurones. Le choix de la méthode dépendra de la complexité de l'image et de la précision requise.
- Gestion des Erreurs : Il peut y avoir des cas où la détection de la grille échoue ou produit des résultats incorrects. Il est nécessaire de mettre en place un mécanisme de gestion des erreurs pour gérer ces situations.

Détection des Cases de la Grille de Sudoku :

- Localisation des Cases : Une fois la grille de sudoku détectée, il est nécessaire de localiser les positions exactes des cases. Cela peut être complexe si les cases sont inclinées ou déformées.
- Délimitation des Cases : Il faut délimiter les cases de manière précise, en identifiant leurs coins ou leurs limites. Des techniques de détection de contours et de segmentation sont couramment utilisées. Ainsi, cette difficulté n'est pas importante maintenant et sera gérée dans la partie suivante.
- Gestion des Erreurs de Découpage : Il peut y avoir des cas où la détection des cases échoue ou produit des résultats incorrects. Un mécanisme de gestion des erreurs est nécessaire.
- Orientation des Cases : Si la grille de sudoku est inclinée, il est important de prendre en compte cette orientation lors de la détection des cases pour garantir la précision de la reconnaissance des chiffres.
- Tests Rigoureux : Des tests doivent être effectués pour s'assurer que la détection de la grille et des cases est précise et fiable. Les tests doivent couvrir une variété de cas d'utilisation, y compris des images de sudoku avec des défauts.

En résumé, la détection de la grille de sudoku et des cases est une étape critique du processus de reconnaissance de grille de sudoku. Pour assurer le succès de cette fonctionnalité, il est essentiel de prendre en compte les défis techniques, d'optimiser les méthodes de détection, de garantir une détection précise et de réaliser des tests approfondis pour valider le processus.

De plus, il est important de prendre en compte le fait que certaines fonctionnalités mentionnées au-dessus seront implémentées dans la partie suivante et que la détection de la grille se fera quelle que soit son orientation, ainsi la tâche est plus ardue que prévu...

Maintenant que nous avons mentionné les nombreuses difficultés que nous avons rencontrées lors de la création de cet outil de détection, voici son implémentation: Nous avons utilisé la méthode de Hough. La méthode de Hough est une technique de traitement d'images qui est principalement utilisée pour détecter des formes géométriques, telles que des lignes droites, des cercles, des ellipses, etc., dans une image. La méthode de Hough repose sur le principe suivant : au lieu de détecter des objets en recherchant directement leurs caractéristiques dans l'image, elle transforme l'image en un espace de paramètres où les caractéristiques souhaitées sont plus facilement identifiables. Les étapes de base de la méthode de Hough sont les suivantes :

- Transformation de l'image : La première étape consiste à convertir l'image originale en une forme d'image transformée, généralement appelée "espace de Hough". Pour la détection de lignes droites, cela implique généralement d'identifier les points d'intersection de l'image avec les lignes possibles. Chaque point d'intersection contribue à une valeur dans l'espace de Hough qui représente les paramètres de la ligne (par exemple, l'inclinaison et l'interception).

- Accumulation : Dans l'espace de Hough, chaque point d'intersection contribue à une courbe ou à une cellule dans l'espace de paramètres correspondant aux paramètres de la ligne qui le traverse. Plusieurs points d'intersection sur la même ligne génèrent des courbes ou des cellules fortement accumulées.

- Détection des caractéristiques : Une fois que l'accumulation est terminée, il est possible de rechercher des pics dans l'espace de Hough. Ces pics correspondent aux lignes détectées dans l'image d'origine. Les paramètres de ces pics (inclinaison et interception) décrivent les caractéristiques des lignes détectées. Ainsi, la méthode de Hough offre une approche robuste pour la détection de formes même en présence de bruit ou d'occlusions dans l'image.

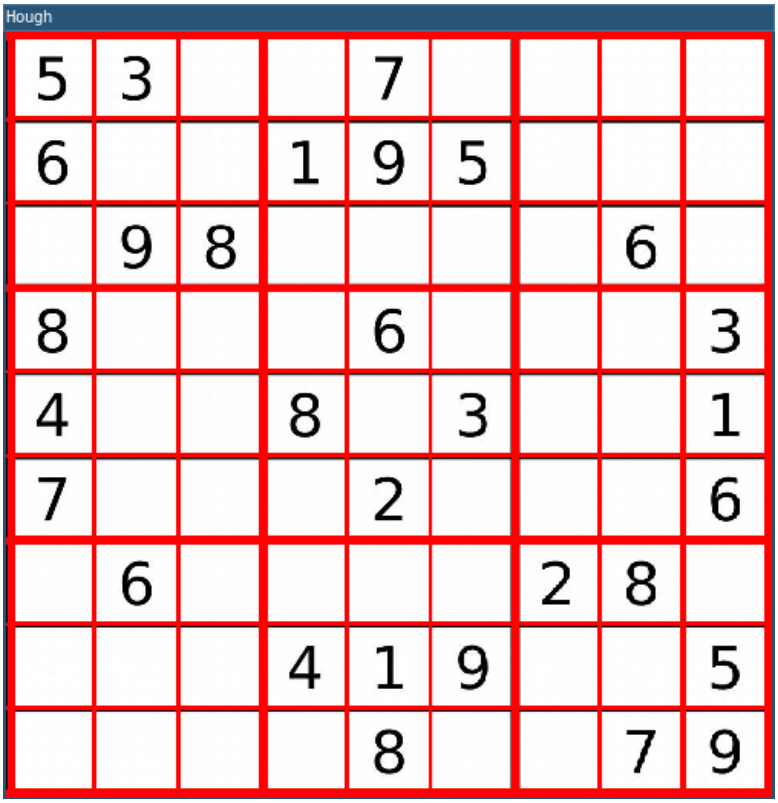

Figure 5: Détection de la grille par le méthode de Hough

2.1.3 Découpage de l'image (sauvegarde de chaque case sous la forme d'une image)

Le découpage de l'image et la sauvegarde de chaque case sous forme d'image individuelle sont des tâches cruciales dans le processus de reconnaissance de la grille de sudoku. Les difficultés et les éléments à prendre en compte lors du développement de cette fonctionnalité sont les suivantes :

- Détection des Cases : La première étape consiste à détecter les limites de chaque case de la grille de sudoku dans l'image. Cela peut être compliqué, en particulier si l'image n'est pas parfaitement alignée, si les cases ne sont pas de taille uniforme ou si la qualité de l'image est médiocre. Ainsi, cette étape sera réalisée uniquement si le sudoku est droit, si il ne l'est pas, il peut être mis à l'endroit avec l'outil de rotation que nous avons développé.

- Précision du Découpage : Le découpage des cases doit être précis pour éviter de couper des chiffres ou d'inclure des parties indésirables de l'image. Des techniques de traitement d'image, telles que la détection de contours ou la segmentation, sont nécessaires pour obtenir un découpage précis.

- Gestion de l'Orientation : Si l'image a été rodée manuellement à l'étape précédente, il est essentiel de prendre en compte cette rotation lors du découpage des cases pour garantir que les chiffres soient correctement orientés. Ainsi, comme mentionné dans la détection des cases l'outil rotation pourra être utilisé pour remettre l'image dans le bon sens.

- Gestion des Erreurs de Découpage : Il peut y avoir des cas où la détection des cases échoue ou produit des résultats incorrects. Il est nécessaire de mettre en place un mécanisme de gestion des erreurs pour gérer ces situations.

- Nommage et Stockage des Images : Chaque case découpée doit être correctement nommée et stockée de manière organisée. Il peut être judicieux de suivre une convention de nommage logique, telle que "case1.jpg", "case2.jpg", etc.

- Échelle et Résolution : Il faut prendre en compte l'échelle et la résolution des images découpées. Elles doivent être cohérentes et adaptées à la résolution d'origine de l'image.

- Gestion de la Mémoire : Le découpage et la sauvegarde de chaque case peuvent consommer de la mémoire, en particulier si l'image est de grande taille. Une gestion efficace de la mémoire est nécessaire pour éviter les problèmes de performances.

- Tests Rigoureux : Des tests doivent être effectués pour s'assurer que le découpage est précis et que les cases sont correctement sauvegardées. Les tests doivent couvrir une variété de cas d'utilisation, y compris des images de sudoku avec des défauts. Par exemple, certaines des images générées seront encore encadré par les lignes de la grille.

- Gestion des Fichiers : Il convient de gérer correctement la création, la suppression et la sauvegarde des fichiers image découpés pour éviter d'encombrer le système de fichiers. Ne pas créer trop d'images.

En résumé, le découpage de l'image en cases individuelles et leur sauvegarde est une étape importante du processus de reconnaissance de grille de sudoku. Pour assurer le succès de cette fonctionnalité, il est essentiel de prendre en compte les défis techniques, de garantir un découpage précis et une gestion efficace des fichiers, et de réaliser des tests approfondis pour valider le processus.

Maintenant que nous avons passé en revue les multiples difficultés que nous pouvons rencontrer, voici notre implémentation.

Tout d'abord, notre technique va se baser sur une segmentation d'une image en petites images en se basant sur la détection de lignes horizontales et verticales. Nous avons utilisé les bibliothèques SDL pour faciliter la manipulation de l'image. Ainsi, nous avons crée de nombreuses fonctions permettant de manipuler les pixels des images, par exemple getpixel et putpixel mais également setlines qui, comme son nom l'indique, créer des lignes sur l'image aux coordonnées renseignées.

Avec ces fonctions annexes que nous avons créées, la réalisation de la fonction segmentation est bien plus simple, nous traçons des lignes rouges à la position des lignes noires de la grille et nous sauvegardons des images de chaque case du sudoku qui sont comprises entre ces mêmes lignes rouges créées auparavant. Chaque image créée est donc sauvegardée sous le nom de son numéro d'ajout, la dernière case sera la 81 par exemple. De plus, nous avons géré le cas des images qui possèdent encore le contour de la grille noire. Nous avons décidé de créer une bordure blanche qui couvre cette partie. Ainsi, chaque image possède ou non un chiffre et n'est pas constituée de pixels noirs parasites.

Cependant, il est important de préciser que le sudoku prit en paramètre doit être droit et s'il ne l'est pas, il peut être mis à l'endroit grâce à notre outil de rotation développer précédemment. Pour les tests, nous avons décidé d'appliquer notre outil sur l'image 01 présentes dans le rapport de projet OCR.

Nous avons donc les résultats suivants (les lignes rouges du premier sudoku sont très peu visible):

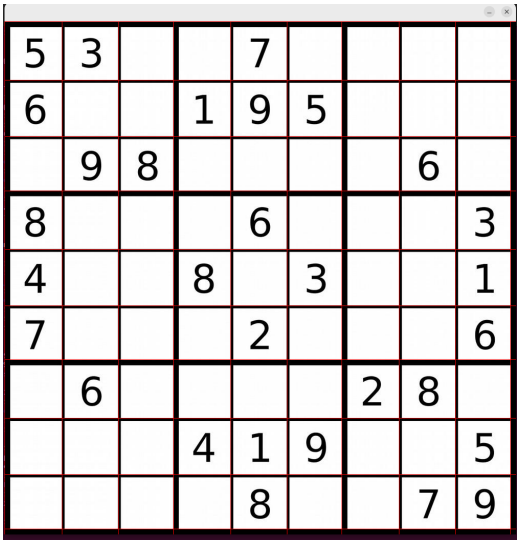

Figure 6: Sudoku avec les lignes rouges tracées permettant de détecter les cases

Figure 7: Quelques images sauvegardées

Ainsi, nous pouvons conclure sur la partie dans laquelle nous avons réalisé la rotation manuelle d'une image, la détection de la grille de sudoku et des cases et la segmentation permettant la sauvegarde de chaque case du sudoku.

La partie suivante liste les outils réalisés par Yannick et Alexandre, soit plus précisément le chargement d'une image et la suppression des couleurs, l'implémentation de l'algorithme de résolution du sudoku (le solver) et le réseau de neurones (ou-exclusif).

2.2 Les parties de Yannick et Alexandre

2.2.1 Chargement d'une image et suppression des couleurs

Le chargement d'une image et la suppression des couleurs sont les premières étapes cruciales dans le processus de traitement d'image pour la reconnaissance de grille de sudoku.

Voici les difficultés et les éléments à prendre en compte lors du développement de ces fonctionnalités :

- Chargement d'une Image :
- Formats d'Image : Les images peuvent être au format JPEG, PNG, BMP, etc. L'application doit être capable de charger une variété de formats d'image courants. La gestion des formats non pris en charge peut être complexe.
- Taille de l'Image : Les images peuvent varier en taille, de petites vignettes à des images de haute résolution. L'application doit être capable de gérer des images de différentes tailles de manière efficace.
- Contrôle de la Qualité : Il est essentiel de vérifier la qualité de l'image lors du chargement pour s'assurer qu'elle est lisible et appropriée pour le traitement. Les images floues, sous-exposées ou comportant d'autres problèmes de qualité doivent être identifiées.
- Gestion des Erreurs : Il peut y avoir des cas où le chargement de l'image échoue en raison de problèmes de format ou de corruption de fichier. Un mécanisme de gestion des erreurs est nécessaire.

Suppression des Couleurs :

- Conversion en Niveaux de Gris : La conversion de l'image en niveaux de gris peut sembler simple, mais elle peut nécessiter une gestion appropriée de la luminance et du contraste pour éviter la perte d'informations.
- Conversion en Noir et Blanc : La conversion en noir et blanc, également connue sous le nom de binarisation, peut être complexe. Le choix de la méthode de seuillage (par exemple, seuillage global ou adaptatif) doit être optimisé pour chaque image.
- Gestion des Couleurs Saturées : Certaines images peuvent contenir des couleurs saturées ou des reflets, ce qui peut rendre la suppression des couleurs plus difficile. Des techniques de prétraitement peuvent être nécessaires.

- Contrôle de la Qualité : Après la suppression des couleurs, il est essentiel de s'assurer que l'image résultante conserve une qualité suffisante pour la détection et la reconnaissance des caractères.
- Gestion des Erreurs : La suppression des couleurs peut échouer ou produire des résultats inattendus sur certaines images. La gestion des erreurs est nécessaire pour gérer ces situations.
- Tests Rigoureux : Des tests doivent être effectués pour s'assurer que la suppression des couleurs est réalisée avec précision et que l'image est prête pour les étapes suivantes du traitement.

En résumé, le chargement d'une image et la suppression des couleurs sont des étapes essentielles dans le processus de traitement d'image pour la reconnaissance de grille de sudoku. Pour garantir le succès de ces fonctionnalités, il est nécessaire de prendre en compte les défis techniques, d'optimiser les méthodes de conversion et de réaliser des tests approfondis pour valider le processus.

Maintenant que vous connaissez les difficultés que nous avons pu rencontrer, voici des détails sur notre implémentation.

Le chargement d'une image peut prendre en compte les quatres formats suivants: JPG, JPEG, PNG et BPM. De plus, les images prises en compte ne seront pas affectées par une baisse de qualité. Ainsi, l'outil peut lire n'importe quel type d'image, il reste à faire confiance à l'utilisateur et son utilisation sur des images de sudoku et non pas sur des images différentes. La partie de suppression de couleurs quant à elles est très complexe, la conversion en gris autrement appelé grayscale est simple, chaque couleur est transformée en variance de gris. Cependant la seconde partie est bien plus ardue, elle consiste à convertir l'image en noir et blanc selon si le gris est plus foncé ou plus clair. Certaines images comportent des tâches ou bruits parasites et elles peuvent entraver le bon fonctionnement de cet outil.

Ainsi, il est important de bien traiter les bruits parasites en appliquant de nombreux filtres ou en utilisant une technique annexe. La qualité de l'image peut être affectée par les nombreux filtres appliqués sur celle-ci, un traitement de la qualité est donc essentiel si les filtres sont puissants. Dans notre outil, ce traitement n'est pour l'instant pas nécessaire.

Voici un exemple de suppression des couleurs sur un sudoku :

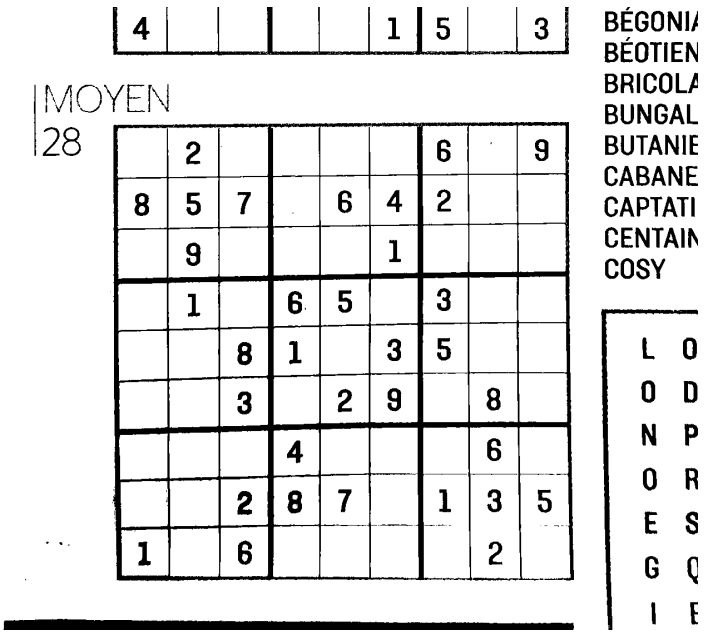

Figure 8: Exemple de suppression des couleurs sur un sudoku lambda

2.2.2 Implémentation de l'algorithme de résolution d'un sudoku (le solver)

L'implémentation de l'algorithme de résolution d'un sudoku (le solver) est une étape cruciale dans le processus de reconnaissance de grille de sudoku. Voici les difficultés et les éléments à prendre en compte lors du développement de cette fonctionnalité :

- Différentes Complexités : Les grilles de sudoku peuvent varier en termes de complexité, allant des grilles simples aux grilles très difficiles. L'algorithme doit être capable de résoudre efficacement une large gamme de grilles.
- Algorithmes de Résolution : Il existe divers algorithmes de résolution de sudoku, notamment l'algorithme de recherche arrière (backtracking), les techniques de propagation de contraintes (constraint propagation), et d'autres méthodes. Le choix de l'algorithme dépend de la complexité de la grille et des exigences de performance.
- Optimisation de l'Algorithme : Il peut être nécessaire d'optimiser l'algorithme pour garantir qu'il résout rapidement les grilles. Cela peut inclure l'implémentation de techniques de branch and bound ou d'autres stratégies d'optimisation.
- Communication avec le Programme Principal : L'algorithme de résolution doit être intégré de manière transparente dans le programme principal de reconnaissance de grille de sudoku, de sorte qu'il puisse recevoir les données de la grille à résoudre et renvoyer la grille résolue.

- Grilles Non Solubles : Certaines grilles peuvent être non solubles ou avoir plusieurs solutions. L'algorithme doit être capable de gérer ces situations et renvoyer des résultats appropriés.
- Gestion des Cas Limites : Des cas limites, tels que des grilles partiellement remplies ou des grilles incorrectes, doivent être gérés de manière à éviter les erreurs.
- Validation des Résultats : L'algorithme doit être soumis à des tests rigoureux pour s'assurer qu'il résout correctement une variété de grilles de sudoku, y compris des grilles de différents niveaux de complexité.
- Tests de Performance : Des tests de performance doivent être réalisés pour s'assurer que l'algorithme résout rapidement les grilles, en particulier celles de haute complexité.

En résumé, l'implémentation de l'algorithme de résolution d'un sudoku (le solver) est une étape essentielle du processus de reconnaissance de grille de sudoku. Pour garantir le succès de cette fonctionnalité, il est nécessaire de prendre en compte les défis techniques, d'optimiser l'algorithme, de réaliser des tests approfondis et de fournir une documentation adéquate.

Maintenant que nous avons retracé les difficultés que nous avons pu rencontrer, nous avons décidé d'implémenter le solver de la manière suivante :

Tout d'abord, nous avons utilisé une matrice bidimensionnelle pour représenter la grille de Sudoku, où chaque cellule peut contenir un chiffre (de 1 à 9) ou être vide (0 ou un autre indicateur). De plus, nous avons vérifiez que la grille initiale est valide, c'est-à-dire qu'elle ne viole pas les règles du Sudoku. Aucune ligne, colonne ou région (les régions sont les sous-grilles de 3x3) ne contient de doublons de chiffres.

Plusieurs algorithmes peuvent être utilisés pour résoudre le Sudoku. L'un des plus couramment utilisés est l'algorithme de rétrogradation (backtracking). Voici comment cela fonctionne :

On commence par parcourir chaque cellule vide de la grille. Pour chaque cellule, on essaye de placer un chiffre valide (de 1 à 9). Si on place un chiffre valide, on passe à la cellule suivante. Si on atteind une impasse (aucun chiffre valide ne peut être placé), on revient en arrière (d'où le terme "backtracking") pour réviser les choix précédents. Il faut répéter ces étapes jusqu'à ce que la grille soit complètement remplie ou que toutes les possibilités soient parcourues. Une fois que le solveur a rempli toute la grille, il ne nous reste qu'à vérifier que la grille résultante est correcte.

Voici un exemple de l'utilisation du solver :

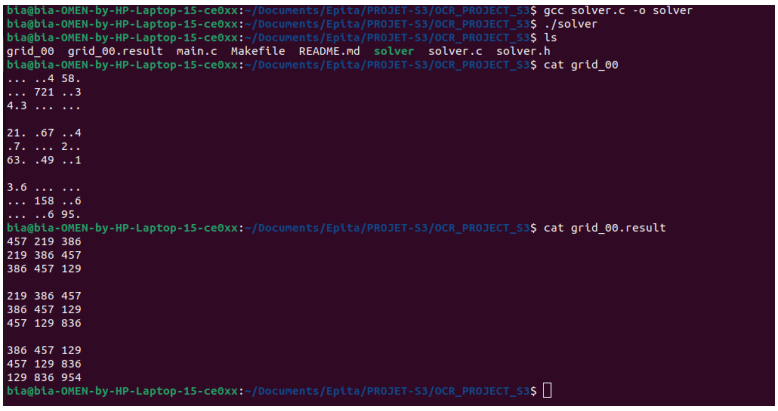

Figure 9: Utilisation du solver sur une grille de sudoku

2.2.3 La preuve de concept du réseau de neurones)

Le développement de la preuve de concept d'un réseau de neurones pour résoudre un problème de Sudoku à partir d'une image présente plusieurs défis et considérations importantes :

- Collecte de données d'entraînement : Il est important de rassembler un ensemble de données contenant des images de grilles de Sudoku et leurs solutions correspondantes pour l'entraînement du réseau de neurones. Si les résultats sont incorrectes, il faut indiquer d'avantages de ressources et ne pas hésiter à guider le réseau de neurones vers un résultat convenable.

- Prétraitement des données : Les images d'entrée doivent subir un prétraitement pour les adapter au format requis par le réseau de neurones. Cela peut impliquer la redimension, la normalisation, la conversion en niveaux de gris, etc.

- Architecture du réseau de neurones : Une architecture de réseau de neurones appropriée au problème doit être choisie. Pour la résolution de Sudoku, une architecture de réseau de neurones convolutif (CNN) ou une combinaison de CNN et de réseaux de neurones récurrents (RNN) peut être envisagée.

- Validation et test : L'évaluation de la précision du modèle doit être effectuée en utilisant des données de validation et de test distinctes pour garantir que le réseau est capable de généraliser à de nouvelles images de Sudoku.

- Gestion des erreurs : La résolution de Sudoku à partir d'images peut comporter des erreurs. Il est essentiel de gérer les cas où le réseau de neurones ne parvient pas à résoudre correctement la grille.

- Optimisation : L'optimisation du modèle, en termes de précision et d'efficacité, est cruciale. L'architecture du réseau et les hyperparamètres doivent peut-être être ajustés pour améliorer les résultats.

- Sauvegarde et rechargement du modèle : Des fonctionnalités pour sauvegarder le modèle après l'entraînement et le recharger ultérieurement peuvent être implémentées pour une utilisation ultérieure dans l'application principale.

En résumé, le développement de la preuve de concept d'un réseau de neurones pour résoudre des Sudoku à partir d'images est un projet complexe qui nécessite une collecte de données soignée, une conception d'architecture appropriée, une formation efficace, une validation rigoureuse et une gestion des erreurs.

C'est une découverte pour nous de manipuler un réseau pouvant apprendre par lui-même. Maintenant que les défis et difficultés potentielles ont été abordés, voici quelques explications sur notre implémentation du réseau de neurones :

Notre implémentation utilise la rétropropagation du gradient pour apprendre une tâche spécifique, en l'occurrence la résolution du problème XOR.

Le programme effectue une boucle d'apprentissage pour un certain nombre d'époques. Pour chaque exemple d'entraînement, il effectue la propagation avant pour calculer la sortie prédite du réseau. Ensuite, il effectue la rétropropagation pour mettre à jour les poids des connexions et minimiser l'erreur. Les erreurs des couches de sortie sont calculées à l'aide de la dérivée de la fonction sigmoïde et sont utilisées pour ajuster les poids.

Après l'apprentissage, le réseau est testé avec les exemples d'entrée d'origine pour voir comment il se comporte. Les entrées sont propagées à travers le réseau pour obtenir des prédictions, et les résultats sont comparés aux sorties attendues. Les résultats de ces tests sont affichés, montrant les entrées, les sorties prédites, et les sorties attendues. Cette partie semble très énigmatique mais il faut tester cet outil par sois même pour comprendre son fonctionnement.

2.3 L'avancement du projet et les ajouts

2.3.1 Les améliorations et changements

Nous sommes plutôt satisfaits du parcours que nous avons effectué pour l'instant. Nous avons rencontré de nombreux obstacles et difficultés mais notre capacité à s'adapter aux défis qui nous font face nous a permis de compléter la plupart des attentes pour la première soutenance même si il reste des choses à améliorer.

Concernant les améliorations et les changements :

- Améliorations des filtres noir et blanc pour qu'aucune tâche ou bruit parasite le bon fonctionnement de la transformation.
- Veiller au bon fonctionnement de la création des images des cases du sudoku, qu'aucune case ne se retrouve en double par exemple.
- Une détection de la grille plus simple et plus optimisée.

2.3.2 Les ajouts

Voici la liste complète des ajouts demandé pour la dernière soutenance :

- Le prétraitement complet
- Le réseau de neurones complet et fonctionnel
- Apprentissage
- Reconnaissance des chiffres de la grille.
- La reconstruction de la grille
- La résolution de la grille
- L'affichage de la grille
- La sauvegarde du résultat
- Une interface graphique permettant d'utiliser tous ces éléments.

Bien entendu, la plupart des éléments ont déjà été créer pour la première soutenance, mais ils ne sont pas encore finalisés. Ainsi, nous aimerions ajouter d'autres éléments tels qu'un site internet pour le projet mais il sera crée uniquement si les éléments cités auparavant ont été implémentés et sont pleinement fonctionnels.

Ainsi s'achève la partie du travail réalisé pour la première soutenance. Nous avions rencontrées de nombreuses difficultés mais nous avions su réaliser les outils nécessaire pour la confection de notre application. Nous pouvons donc entamer les explications sur la partie du travail réalisée pour la soutenance finale.

3 Les approches adoptées pour la soutenance finale

Dans cette partie vous retrouverez toutes les modifications et ajouts que nous avons effectués sur notre application pour que les objectifs de la soutenance finale cités dans la partie précédente soient respectés. Nous avons le regret d'annoncer que par manque de temps nous n'avons pas pu créer un site web pour notre application mais les autres améliorations ont bien été réalisés par exemple :

- L'amélioration des filtres noir et blanc pour qu'aucune tâche ou bruit parasite le bon fonctionnement de la transformation.

- Veiller au bon fonctionnement de la création des images des cases du sudoku, qu'aucune case ne se retrouve en double par exemple.

- La détection de la grille plus simple et plus optimisée.

De plus, nous espérons que notre interface utilisateur vous plaira, elle est simple d'utilisation et permet à l'utilisateur d'être guidé tout le long du processus.

3.1 Les parties de Raphaël et Robin

3.1.1 L'interface utilisateur

Une interface utilisateur claire et facile d'accès est essentielle pour une application. C'est pourquoi nous avons décidés d'utiliser la bibliothèque Gtk qui nous permet de réaliser une interface de manière simple et optimisée.

Voici quelques-uns des avantages d'utiliser la bibliothèque GTK en C :

- Portabilité : GTK est conçu pour être portable sur différentes plates-formes, ce qui signifie que les applications développées avec GTK peuvent être exécutées sur plusieurs systèmes d'exploitation tels que Linux, Windows et macOS sans nécessiter de modifications majeures.
- Open Source : GTK est distribué sous une licence open source (LGPL), ce qui signifie que les développeurs ont accès au code source, peuvent le modifier et le distribuer librement. Cela favorise la collaboration et la création de logiciels libres.
- Widgets riches : GTK propose une large gamme de widgets (éléments d'interface utilisateur) prêts à l'emploi, tels que des boutons, des boîtes de dialogue, des barres de défilement, etc. Cela facilite le développement d'interfaces utilisateur riches et interactives.
- Thèmes et styles : GTK prend en charge les thèmes, ce qui signifie que l'apparence de l'interface utilisateur peut être facilement modifiée en appliquant différents thèmes. Cela offre aux développeurs la flexibilité nécessaire pour personnaliser l'apparence de leurs applications.

- Internationalisation et localisation : GTK intègre des fonctionnalités facilitant l'internationalisation des applications. Les applications GTK peuvent être traduites dans différentes langues, ce qui est important pour atteindre un public mondial.
- Documentation complète : GTK est accompagné d'une documentation complète et bien entretenue. Cela facilite l'apprentissage pour les développeurs et fournit des informations détaillées sur l'utilisation de chaque composant de la bibliothèque.
- Utilisé par des applications renommées : GTK est utilisé par de nombreuses applications bien connues, telles que le gestionnaire de fenêtres GNOME, l'environnement de bureau XFCE, GIMP (GNU Image Manipulation Program), etc. Cela témoigne de la fiabilité et de la robustesse de la bibliothèque.

En résumé, GTK offre aux développeurs C une plate-forme solide et flexible pour la création d'interfaces graphiques multiplateformes. Cela nous permet donc de réaliser une application simple avec une interface compréhensible. Cependant, bien que GTK présente de nombreux avantages, il existe également des inconvénients associés à son utilisation. Par exemple :

- Taille de la bibliothèque : GTK a une empreinte relativement importante, ce qui peut entraîner des applications plus volumineuses en termes de taille du fichier exécutable. Cela peut être un inconvénient pour les applications où la taille du programme est critique.
- Moins populaire sur certaines plates-formes : Bien que GTK soit largement utilisé sur Linux, il peut être moins populaire sur d'autres plates-formes comme Windows et macOS. Cela peut entraîner un soutien communautaire moindre sur ces systèmes d'exploitation.

Ce dernier point nous a effectivement apporté des difficultés comme des résultats différents selon l'OS présent sur l'ordinateur de l'utilisateur. Mais nous avons su régler ces problèmes en adaptant les boutons et les images présentes sur l'interface par rapport à la taille de la fenêtre. Ainsi, l'interface de l'application s'adaptera quelque soit la dimension de la fenêtre.

Concernant les boutons et le lancement de l'application, nous avons décidé de créer un bouton excessivement grand Load Image qui comme son nom l'indique permet de charger une image que l'utilisateur sélectionne sur son propre répertoire. N'importe quel format d'image est accepté par notre application et c'est une grande force de celle-ci.

Une fois l'image choisi, l'utilisateur à la possibilité de choisir une nouvelle image sur le même bouton que précédemment mais rétrécie et positionner en haut à gauche. De plus, l'image choisi précédemment est donc chargée et positionnée au milieu en bas de l'interface de dimension assez grande pour que l'utilisateur puisse voir les différentes étapes que nous allons effectuer tout le long du processus de résolution du sudoku. Il est également important de mentionner que l'image chargé peut être de n'importe quelle dimensions, elle sera adapté selon les dimensions de la fenêtre.

L'utilisateur peut maintenant commencer le processus complet. Il a la possibilité d'appuyer sur le bouton de l'étape suivant situé en haut à gauche et qui permet de faire le Preprocess autrement dit le conversion des couleurs de l'image en gris avec un grayscale puis en noir et blanc avec les nombreux filtres anti-bruits et grains parasites. Une fois le bouton pressé, l'image a été modifiée par les filtres précédemment listés. L'utilisateur possède alors encore la possibilité de changer d'image pour recommencer le traitement sur une nouvelle image ou il a la possibilité de continuer le traitement avec le bouton de la prochaine étape: Rotation. Comme son nom l'indique, lorsque l'utilisateur appuie sur ce bouton, l'image effectue une rotation pour remettre l'image dans le bon sens si la grille de sudoku n'est pas droite ou est tournée d'un certain degrés. Ne vous inquiétez pas si vous voyez que l'image est entourée d'un fond d'une autre couleur, c'est que la rotation a bien été effectuée !

Après avoir réalisé cette étape, l'utilisateur possède alors encore la possibilité de changer d'image pour recommencer le traitement sur une nouvelle image ou il a la possibilité de continuer le traitement avec le bouton de la prochaine étape: Detection.

Cette étape de détection des lignes et du carré le plus grand permet donc de détecter la grille de sudoku et de recentrer l'image sur celle-ci. C'est une étape très compliqué qui nous a demandé énormément de travail. Une image où l'on peut voir des lignes qui permettent le repérage de la grille est donc affiché. Une fenêtre pop up apparait pour passer à l'étape suivant, il suffit de la fermer pour que le carré le plus grand (et donc la grille) soit détecté. L'image est redimensionnée de manière à avoir la grille pleinement visible, ainsi le reste de l'image à disparu. Puis une nouvelle fenêtre pop up apparaît pour passer à l'étape de segmentation. Cette étape consiste à créer des images de 28x28 des cellules présentent dans le sudoku. L'image visible sur l'écran est donc l'image redimensionnée avec les lignes de la grille retracées pour permettre à la segmentation de s'effectuer.

Enfin, le bouton solver est accessible ce qui permet de résoudre le sudoku en appellant le réseau neuronnes, le solver et la construction d'une nouvelle grille sur la grille et les cases extraites de celle-ci. Le résultat s'affiche et l'utilisateur à la possibilité de sauvegarder l'image ou de recommencer le processus.

Voici donc l'interface, l'application n'est peut-être pas très jolie à première vu mais elle est comme mentionné précédemment simpliste et facile d'utilisation peu importe votre âge tout le monde peut utiliser cette application et c'est une force non négligable:

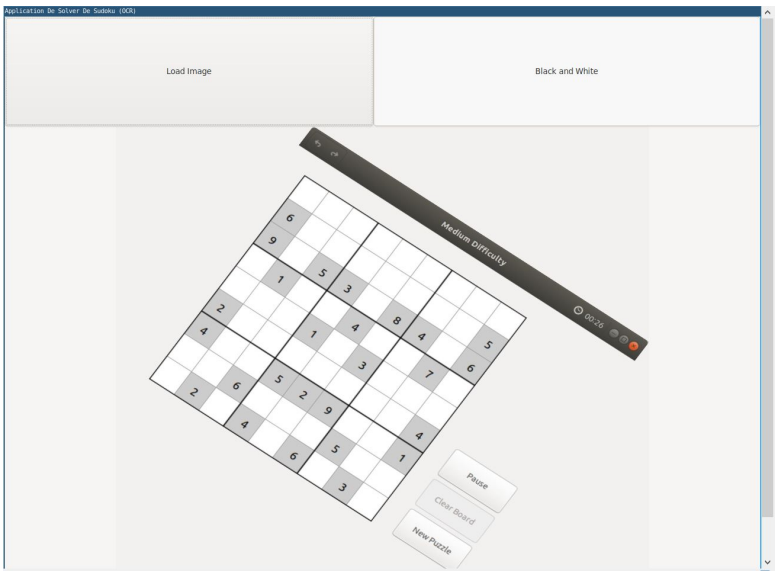

Figure 10: L'image a été chargée

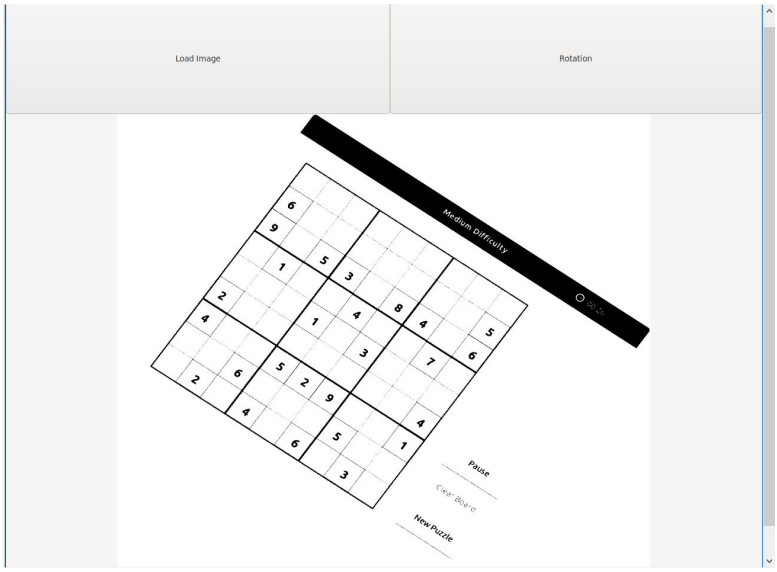

Figure 11: Le Preprocess

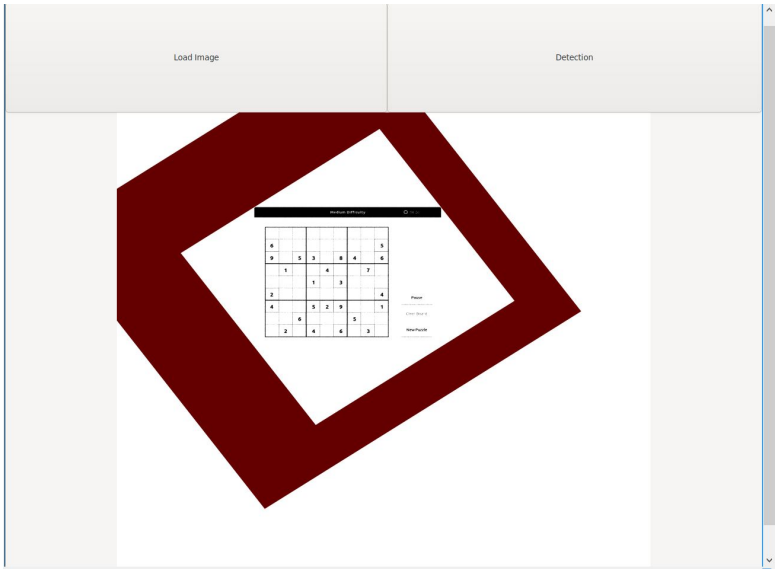

Figure 12: La Rotation

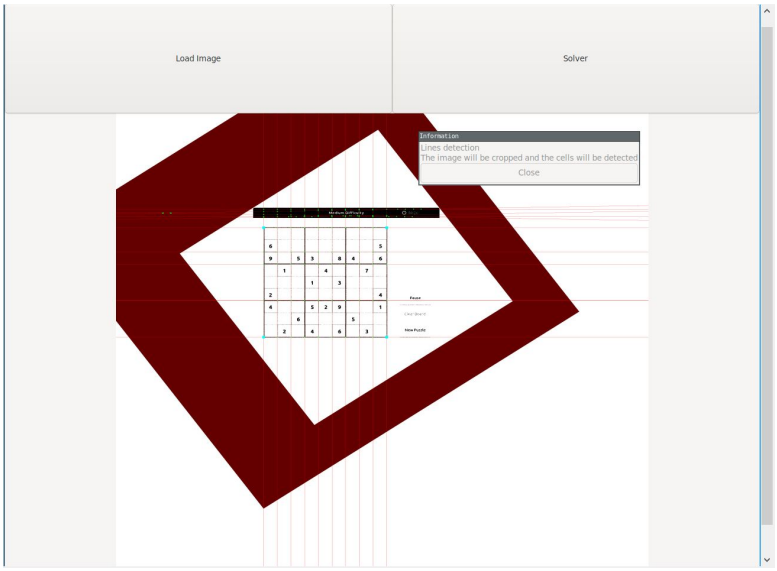

Figure 13: La détection de la grille

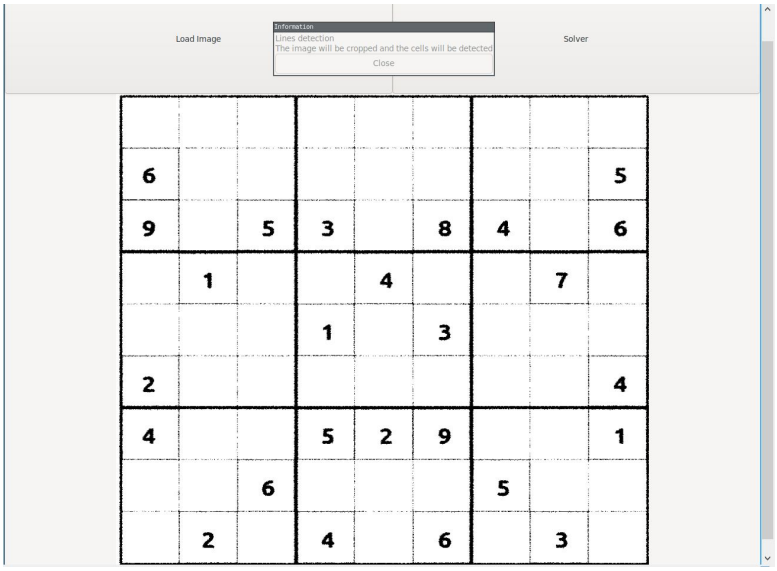

Figure 14: La détection du plus grand carré

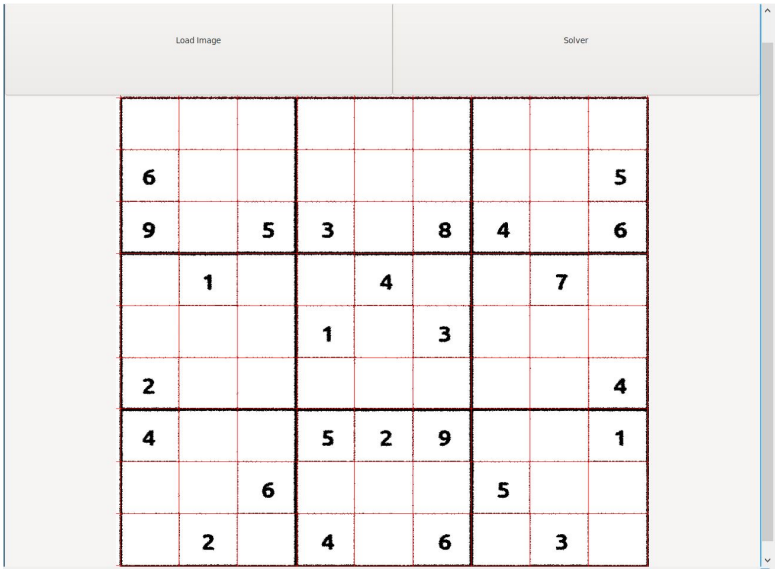

Figure 15: La segmentation

Figure 16: Le résultat

3.1.2 Rotation automatique de l'image

Détecter automatiquement l'angle de rotation d'une image peut être un défi intéressant. Etant donné que nous avons déjà codé la rotation manuelle, cela réduit grandement la difficulté de la tache. Cependant, il reste quelques difficultés :

- Présence d'objets : Si l'image contient d'autres objets, structures ou motifs, la détection de l'angle peut être perturbée. La présence d'éléments complexes peut interférer avec les algorithmes de détection d'angle.
- Angles multiples : Une image peut être inclinée selon plusieurs axes. La détection de l'angle doit être capable de distinguer ces différents angles possibles.
- Complexité algorithmique : Le choix de l'algorithme de détection d'angle peut également poser des défis. Des méthodes basées sur des caractéristiques, des réseaux de neurones, ou d'autres techniques peuvent avoir des performances variables selon le type d'image.

Il est important de considérer ces défis lors du développement de votre fonction de détection automatique de l'angle.

Mais alors quelle technique adopter ?

Nous avons décidé d'adopter un système d'intersections. En utilisant la détection de Hough, nous avons pu repérer les intersections entre les lignes du sudoku et nous avons modifié la fonction de rotation manuelle pour qu'elle puisse, à partir de coordonnées,

calculer l'angle de rotation nécessaire pour mettre l'image droite et donc d'effectuer une rotation parfaite.

Les angles positifs comme négatifs sont toujours gérés et notre rotation est très précise grâce à l'utilisation de float et double.

La rotation est désormais pleinement automatique !

3.1.3 Améliorations de la détection de la grille et de la position des cases

De nombreuses modifications ont été réalisés dans cette partie à causes de certaines difficultés pour notre programme précédent à repérer des grilles.

Notre programme avait des difficultés pour repérer les grilles des images comportants de informations supplémentaires autour de celle-ci. De plus, de nombreuses lignes étant de trop pouvaient fausser la détection.

Ainsi, nous avons pris les précautions nécessaires, nous avons supprimer les lignes proches ce qui permet d'éviter de ce retrouver avec plusieurs lignes sur un même coté de la grille.

Nous avons également détecter les intersections des lignes entre elles ce qui permet à la rotation automatique de fonctionner parfaitement grâce aux multiples coordonnées de ces intersections.

Mais cela permet également d'augmenter la précision de la détection de Hough ce qui nous servira également dans la partie de la segmentation.

Cette détection est essentielle pour la suite et nous sommes très fiers de son évolution, notre programme est maintenant capable de détecter une grille de sudoku dans une image quel que soit son orientation ou les informations parasites autour de la grille.

3.1.4 Améliorations de la segmentation de l'image

Concernant la segmentation et son objectif de générer une image de chacune des cellules du sudoku, nous avons su gérer le cas des images parasites supplémentaires et des images en doubles.

De plus, certaines images comportaient du bruits ou des contours noirs des lignes du sudoku, nous avons gérés ces deux cas en effectuant un traitement d'image pour que le bruit et les contours soient considérés comme des pixels blancs.

Ce changement permet de garantir des images clairs et nettes car elles seront envoyées au réseau de neurones et que la précision de celui-ci peut être perturbé par ces grains parasitent.

La détection du plus grand carré est optimale, et nous avons gérer des cas supplémentaires en cas de mauvaise détection pour que le programme puisse tout de même détecter un carré un peu plus grand mais toujours valide pour la segmentation. Etant donné que notre segmentation marchait correctement depuis la première soutenance, nous n'avons pas eu à faire d'autres modifications majeures. La segmentation surtout dépendante de la détection de la grille est donc pleinement fonctionnelle.

3.2 Les parties de Yannick et Alexandre

3.2.1 Le réseau de neurones

La réalisation d'un réseau de neurones en langage C pour la résolution du Sudoku nous a suscité plusieurs difficultés :

- Structuration des données : Organiser les données d'entrée de manière à ce qu'elles soient compatibles avec le réseau de neurones peut être délicat.
- Choix de l'architecture du réseau : Sélectionner une architecture de réseau de neurones adaptée à la résolution du Sudoku peut être complexe. Cela dépendra de la complexité du problème et de la quantité de données disponibles pour l'entraînement.
- Entraînement du réseau : L'entraînement d'un réseau de neurones peut nécessiter une grande quantité de données annotées. Obtenir un ensemble de données de Sudoku étiqueté peut être difficile, et l'entraînement du réseau peut être coûteux en termes de temps de calcul.
- Détection des cases vides : Identifier les cases vides dans une grille de Sudoku à partir d'une image peut être un défi. Vous devrez peut-être utiliser des techniques de traitement d'image avancées ou d'autres méthodes pour localiser les différentes cases.
- Optimisation des performances : Implémenter un réseau de neurones efficace en C peut être complexe, car vous devrez optimiser les performances du code pour garantir une exécution rapide, en particulier lors de la prédiction sur de grandes quantités de données.
- Gestion de la mémoire : La gestion manuelle de la mémoire en langage C peut être sujette à des erreurs. Assurer une gestion correcte de la mémoire est crucial pour éviter les fuites de mémoire et les erreurs d'exécution.
- Intégration avec d'autres composants : Comme notre réseau de neurones doit être combiné dans une application, l'intégration avec d'autres fonctionnalités peut introduire des complexités supplémentaires.

Parlons maintenant de l'implémentation que nous avons décidés de choisir.

Nous avons choisi de représenter la grille de Sudoku sous forme d'une matrice bidimensionnelle en langage C. Chaque élément de la matrice correspond à une case du Sudoku, avec les chiffres connus représentés par leurs valeurs respectives et les cases vides dénotées par une valeur spéciale, telle que zéro.

Avant d'alimenter les données dans le réseau de neurones, nous appliquons des opérations de prétraitement directement sur la matrice. Cette étape permet de normaliser les valeurs et de les ramener dans une plage spécifique, facilitant ainsi le processus d'apprentissage.

Nous utilisons ensuite la matrice de Sudoku comme entrée du réseau, en l'aplatissant

pour obtenir un vecteur unidimensionnel. Chaque élément de ce vecteur représente une case de la grille, simplifiant ainsi la gestion des données en entrée.

La sortie du réseau est également structurée sous forme de matrice, où chaque ligne représente une case du Sudoku et chaque colonne renvoie la probabilité associée à un chiffre particulier. Cette approche offre une interprétation claire des résultats du réseau, facilitant ainsi la détermination des chiffres prédits pour chaque case.

Cette mise en œuvre par matrice s'inscrit dans une démarche efficace et élégante pour résoudre les Sudoku à l'aide de réseaux de neurones, tirant parti de la structure naturelle du problème et facilitant les différentes étapes du processus. L'utilisation de cette technique s'aligne avec notre engagement à trouver des solutions innovantes pour des problèmes complexes tels que la résolution de Sudoku.

De plus, pour garantir l'apprentissage efficace de notre réseau de neurones, nous avons intégré la technique de descente de gradient, une méthode d'optimisation largement utilisée dans le domaine de l'apprentissage automatique. Cette approche vise à ajuster les poids du réseau afin de minimiser la fonction de perte associée à la différence entre les prédictions du modèle et les valeurs réelles.

La descente de gradient opère en itérations successives, calculant les gradients par rapport aux poids du réseau et ajustant ces poids dans la direction opposée au gradient. Nous avons utilisé des bibliothèques et des outils spécifiques en langage C pour implémenter cette technique, optimisant ainsi les paramètres du réseau de manière itérative au fil du temps.

Cette étape très important permet d'affiner les capacités prédictives du réseau de neurones, améliorant progressivement sa performance sur des données de Sudoku variées. La descente de gradient offre ainsi une approche robuste pour l'optimisation des poids du réseau, contribuant à la précision et à l'efficacité globale de notre modèle de résolution de Sudoku.

Il a été très compliqué pour nous de trouvé une base de données assez importante pour que notre réseau de neurones s'entraîne, nos images doivent être blanches avec des chiffres noirs alors que la plupart des images présentent dans les bases de données d'internet sont en noires avec des chiffres blancs.

Ainsi, nous avons réaliser une fonction pour inverser les couleurs des images présentent pour que notre réseau neurones puisse s'entraîner convenablement.

3.2.2 Améliorations du Solver

Voici un rappel de toute les étapes présentent à la réalisation du solver.

Notre code est divisé en plusieurs parties distinctes, chacune ayant un rôle spécifique

dans la résolution du puzzle. Voici une analyse détaillée de ses fonctionnalités :

La fonction isSafe évalue si un chiffre peut être placé en toute sécurité dans une cellule donnée du Sudoku.

Elle effectue trois vérifications distinctes :

- Vérification de la ligne : elle examine si le numéro à placer ne se trouve pas déjà dans la ligne actuelle.
- Vérification de la colonne : similaire à la vérification de la ligne mais pour la colonne actuelle.
- Vérification de la sous-grille 3x3 : contrôle si le nombre peut être placé en toute sécurité dans la sous-grille 3x3 actuelle du Sudoku.

Remplissage du tableau (`fill_board`): Cette fonction remplit récursivement le tableau représentant le Sudoku en utilisant la méthode de backtracking. Elle parcourt chaque élément du tableau, essayant d'insérer des chiffres de 1 à 9 en vérifiant s'ils sont sécuritaires avec `isSafe`. Si un chiffre ne convient pas, elle effectue un *backtracking* pour essayer une autre valeur. Cette approche est utilisée pour résoudre le puzzle de manière exhaustive.

Lecture et écriture de fichiers: La fonction `solver` prend en paramètre le chemin d'un fichier contenant une représentation du Sudoku à résoudre. Elle lit le fichier pour initialiser le tableau de Sudoku, où chaque chiffre représente une cellule du puzzle. Une fois le Sudoku résolu, le résultat est écrit dans un nouveau fichier portant le nom du fichier d'entrée suivi de l'extension `.result`.

Optimisations possibles: Bien que le code fonctionne correctement pour résoudre des Sudokus, des améliorations peuvent être apportées pour gérer les erreurs potentielles dans la lecture des fichiers. Par exemple, la vérification de la disponibilité du fichier d'entrée pourrait être intégrée. De plus, une validation supplémentaire des données en entrée pourrait être ajoutée pour s'assurer que le Sudoku fourni est valide.

Un défi important a été de choisir un format facilitant la lecture et l'écriture des données, notamment lors de la manipulation de fichiers. Le choix du format devait être convivial tout en préservant l'intégrité des données.

Pour relever ces défis, notre code utilise une structure de tableau 2D en C pour stocker la grille de Sudoku. Chaque élément du tableau représente une cellule du Sudoku, permettant une manipulation claire et efficace. Cependant, cette approche peut présenter des limites, notamment en termes d'efficacité pour les opérations de recherche dans les lignes, colonnes et sous-grilles.

Des structures de données alternatives, telles que les bitsets ou les tableaux multidimensionnels différemment organisés, pourraient être explorées pour améliorer l'efficacité des opérations de vérification. Par exemple, l'utilisation de bitsets pourrait réduire la

complexité des opérations de recherche et de comparaison.

En conclusion, bien que fonctionnel, notre code de résolution de Sudoku pourrait bénéficier de petites améliorations pour renforcer sa robustesse et sa fiabilité dans la gestion des données d'entrée et de sortie.

3.2.3 Reconstruction de la grille de Sudoku

Notre programme `plain.c` que nous avons développé sert d'outil de visualisation de grilles de Sudoku en utilisant les bibliothèques GTK et Cairo. Notre programme permet de générer une représentation graphique d'un puzzle de Sudoku ainsi que sa solution, avec les chiffres de la solution mis en évidence en rouge.

L'élaboration de ce programme a été confrontée à plusieurs défis techniques, mettant en lumière des aspects délicats de l'implémentation du dessin de Sudoku avec GTK et Cairo.

La première difficulté résidait dans la création initiale de la grille de Sudoku et l'alignement précis des chiffres avec les cases de la grille. Cela impliquait de calculer dynamiquement les dimensions de chaque case en fonction de la taille totale de la grille, tout en veillant à ce que chaque chiffre soit positionné correctement à l'intérieur de sa case respective.

Une autre problématique majeure était d'assurer une adaptation en temps réel des dimensions de la grille et des chiffres en fonction de la taille de la fenêtre. Cela nécessitait une fonctionnalité dynamique pour ajuster la grille et les chiffres de manière proportionnelle lorsque la taille de la fenêtre graphique était modifiée par l'utilisateur.

Enfin, l'enregistrement de l'image créée dans la fenêtre constituait un défi de plus. Cela impliquait de capturer le contenu de la fenêtre graphique une fois que la représentation visuelle du Sudoku était rendue, puis de sauvegarder cette image dans un format approprié, tel que le format PNG, en intégrant précisément les fonctionnalités de Cairo pour créer et enregistrer l'image finale. La résolution de ces difficultés a nécessité une approche méthodique et une compréhension approfondie des fonctionnalités offertes par les bibliothèques GTK et Cairo, ainsi qu'une maîtrise des calculs de positionnement et de redimensionnement pour assurer la qualité visuelle et la fonctionnalité de l'application de dessin de Sudoku.

Notre programme offre une manière conviviale et visuelle de représenter graphiquement des grilles de Sudoku, permettant aux utilisateurs de visualiser à la fois le puzzle initial et sa solution. Son utilisation de GTK et Cairo en fait une application robuste pour la représentation graphique de ce type de problème logique.

Figure 17: Le résultat (le réseau n'a pas été assez entraîné pour cette exemple, cela place des 0 si le sudoku n'a aucune solution)

Figure 18: Résultat d'un sudoku (avec un réseau assez entraîné

4 Conclusion

Pour conclure, nous sommes très fiers de l'application que nous avons réalisé, elle est fonctionnelle et très simple d'utilisation ce qui permet de toucher un large réseau d'utilisateur.

Cela a été compliqué au départ de planifier les multiples tâches de ce projet ambitieux, mais le résultat parle de lui-même, l'objectif a été atteint.

Il nous est parfois arrivé d'avoir des conflits au sein du groupe mais cela nous a permis de débattre et d'en ressortir plus unis. D'échanger nos opinions, nos idées et de réunir nos connaissances. Chacun à pu exprimer sa créativité au sein du projet et cela peut se ressentir dans notre produit final.

De plus, nous avons rencontré de nombreux obstacles durant son élaboration comme par exemple une différence de résultat entre nos différents systèmes d'exploitation mais nous avons su tenir face à ces problèmes en nous adaptant ou en adaptant les techniques utilisées.

Nous espérons que notre application vous plaira.