EPITA Rennes

Rapport de soutenance 1

Raiders Sudoku

42 Seconds To Epita

Alex DREAU
Arthur BARREAU
Kévin LUBERT
Joris BELY

9 Novembre 2023

Table des matières

1 Introduction

Ce document constitue le rapport de soutenance 1 du projet Raiders Sudoku. Cette première étape marque un jalon significatif dans le développement de notre application visant à résoudre des grilles de Sudoku à partir d'images. Au cours de cette période précédant la première soutenance, notre équipe a fait des avancées majeures dans divers domaines du projet, surmonté des défis techniques et organisé notre travail de manière à optimiser notre progression.

L'objectif principal de notre projet est de créer un système capable de résoudre des grilles de Sudoku à partir d'images prises en entrée. Cette tâche complexe nécessite une combinaison de plusieurs domaines de compétence, notamment le traitement d'image, le développement d'un réseau de neurones pour la reconnaissance des chiffres, la résolution effective du Sudoku et la création d'une interface utilisateur conviviale.

Nous avons divisé le projet en différentes tâches, chaque membre de l'équipe étant responsable d'un domaine spécifique. Le traitement de l'image a été notre point de départ, où nous avons développé des algorithmes pour nettoyer, détecter, et extraire la grille de Sudoku des images. En parallèle, nous avons travaillé sur la mise en place d'un réseau de neurones pour la reconnaissance des chiffres, en utilisant une banque de données que nous avons générée.

La résolution du Sudoku est un autre aspect crucial, et nous avons mis en place un algorithme de backtracking efficace, tout en veillant à vérifier la solubilité des grilles. Enfin, pour rendre le projet accessible aux utilisateurs, nous avons conçu une application conviviale grâce à la librairie GTK et à l'outil Glade. Cette application permettra aux utilisateurs d'importer des images, de résoudre des grilles de Sudoku et de visualiser les résultats.

Dans ce rapport, nous présenterons en détail les avancées de chaque domaine, le rôle de chaque membre de l'équipe, et les prochaines étapes de développement. Nous sommes fiers de nos réalisations à ce stade, et nous sommes impatients de poursuivre notre travail sur Raiders Sudoku pour la prochaine soutenance.

2 Equipe et répartition des tâches

Les tâches qui composent le projet ont été séparées en quatre principales catégories qui sont, le traitement de l'image, le réseau de neurones, la résolution du sudoku et la création de l'application. La majorité des domaines peuvent ensuite être séparés en sous-domaines représentant des plus petites tâches. Nous présenterons ici les taches dans leurs ordres d'application logique.

Le traitement de l'image est le premier domaine qui intervient dans la suite d'algorithmes permettant de passer d'une image à une grille de sudoku résolue. Dans ce domaine, on va appliquer une suite de traitement pour nettoyer l'image et la rendre plus lisible. Il faudra ensuite détecter les lignes pour pouvoir trouver le positon exact de la grille sur l'image. Une fois la grille détectée, il faudra extraire les cases contenant un chiffre. Cette partie représente une grosse charge de travail et nécessite l'implémentation de nombreuses méthodes de traitement et d'analyse de l'image.

Une fois les cases de l'image isolée, il faut détecter quel chiffre est présent dans chaque case. Pour ce faire, un second domaine est la réalisation d'un réseau de neurones qui prendra en entrer une image au format 28x28 pixels et qui donnera en sortie le chiffre présent sur la case. Pour ce faire, il faudra récolter des images dont l'on connaît déjà le résultat attendu, ces images seront utilisées pour entraîner le réseau de neurones.

Maintenant que nous avons traité l'image et que nous savons quel chiffre contient chaque case du sudoku, il est nécessaire de compléter les cases qui n'ont rien d'inscrit. On va donc dans ce domaine résoudre une grille de sudoku et sauvegarder le résultat dans un nouveau fichier. Il faudra prendre en compte la possibilité pour une grille d'être totalement vide ou de contenir des cas impossible, par exemple si une erreur est présente sur la grille et qu'un même chiffre est présent deux fois sur la même ligne.

Avec les trois domaines précédemment décrit nous avons effectuer tout le processus de détection et de résolution du Sudoku, nous ne pouvons cependant pas demander à l'utilisateur d'exécuter lui-même une suite de commandes pour obtenir le résultat final. La partie de l'application permettra donc de faire le lien entre toutes les parties et de proposer à l'utilisateur une interface visuelle lui permettant de choisir quelle image il souhaite traiter et le résultat de son traitement.

Chaque membre du groupe est responsable d'un des domaines, les assignations sont visibles dans le tableau 1. Cependant certains domaines sont moins conséquent que d'autres comme par exemple la résolution de la grille de Sudoku. Les responsables de ces domaines nécessitants moins de travail ont donc aider pour le traitement de l'image, une partie avec une grosse charge de travail.

Table 1 – Répartition des tâches

Tâches	Joris	Alex	Kevin	Arthur
Prétraitement		Responsable		Suppléant
Rotation	Responsable			
Segmentation		Responsable		
Transformation de Hough		Responsable		
Détection des carrés		Responsable		
Réseau de neurones			Responsable	
Banque d'images			Responsable	
Solveur de Sudoku	Responsable			
Parsing input utilisateur				Responsable
Application				Responsable

L'équipe derrière Raiders Sudoku est une évolution de l'équipe ayant effectuer le projet S2 Raiders Tactics. Suite à la perte malheureuse de Romain Seznec, un nouveau membre a été introduit dans l'équipe : Joris Belly.

2.1 Alex DREAU

Depuis mon enfance, ma curiosité envers le monde qui m'entourait a toujours été présente. En grandissant, j'ai pris conscience du rôle essentiel de l'informatique dans notre univers. C'est ainsi que j'ai pris la décision de m'intéresser de plus près à ce domaine et de m'engager sérieusement dans la programmation en rejoignant EPITA.

Lors du projet S2, j'ai principalement axé mon travail sur des aspects liés à la 3D au détriment de la programmation, ce qui a parfois suscité ma frustration. L'OCR, quant à lui, m'a ouvert les portes de l'imagerie, une discipline de l'informatique qui me semblait intimidante, car je n'avais jamais eu l'occasion d'implémenter des algorithmes complexes, comme ceux nécessaires à la détection des bords d'une image. Au cours de ce projet, j'ai eu l'occasion de visionner des vidéos illustrant la détection d'objets en mouvement, ce qui m'a semblé extraordinaire, étant donné le travail requis simplement pour détecter la grille d'un sudoku. Ce projet va me permettre de perfectionner mes compétences en réseaux de neurones et en imagerie.

Ma passion pour la photographie m'a conduit à me concentrer sur le traitement d'image, dans le but de mieux comprendre les algorithmes utilisés par mes logiciels de retouche photo favoris.

2.2 Arthur BARREAU

Depuis son enfance, Arthur Barreau a nourri une passion pour les jeux vidéo, grâce à laquelle il a découvert l'univers du PC, suscitant en lui un vif intérêt pour l'informatique. Il a donc commencé à bidouiller toutes sortes de petits programmes pour l'assister dans ces jeux. Ceci a fait en sorte qu'il était réellement intéressé par le développement informatique ainsi que la cybersécurité.

Le choix d'Arthur de rejoindre EPITA pour ses études supérieures était pour lui une évidence, puisque c'était l'établissement parfait pour développer davantage ses compétences dans ce domaine. Initialement, il était un peu inquiet à l'idée de se lancer dans un projet aussi complexe, en utilisant un langage qu'il ne maîtrisait pas complètement. Cependant, il est entouré de collègues motivés, ce qui facilite considérablement la tâche. Il est fier de ce qu'ils ont accompli au cours de la première phase de ce projet, et il a hâte de poursuivre sur cette lancée.

2.3 Kevin LUBERT

Kévin est bénévole en tant que développeur java senior dans l'association *Ekalia* sous le pseudonyme de Popop12. Il a déjà participé à l'élaboration d'événements rassemblant plusieurs centaines de personnes. Il a donc été nommé chef de projet et s'occupera de la partie réseau de neurones de l'application.

2.4 Joris BELY

Joris, ma passion pour la programmation a débuté dès mon entrée à EPITA. Il a particulièrement apprécié le projet S2, notamment dans son aspect graphique. Le défi passionnant d'implémenter le projet OCR en langage C m'anime, et il prend beaucoup de plaisir à travailler sur ce projet. Il est très satisfait de ce que ils ont accompli au cours de cette première étape de travail, et il est convaincu qu'ils vont redoubler d'efforts pour la deuxième phase dans l'objectif d'atteindre le Saint 20/20.

3 Avancement du projet et aspects techniques

3.1 Solveur de sudoku

Pour résoudre le sudoku, nous allons travailler de manière récursive. Nous avous utilise L'algorithme de backtracking qui peut être implémenté récursivement, ce qui permet de parcourir toutes les combinaisons possibles de manière systématique. Cependant, il est essentiel de mettre en place des mécanismes pour éviter de revisiter des solutions déjà explorées (technique de "pruning") afin d'optimiser les performances de l'algorithme.

3.1.1 Solubilité du sudoku

Avant de chercher la solution du sudoku, on vérifie s'il n'y a pas d'incohérence qui rendrait le sudoku insoluble, comme deux chiffres identiques sur la même ligne, colonne ou encore dans la section de 3x3. Si le sudoku est impossible, il renvoie une erreur.

Autrement, lors de la tentative de résolution, s'il parcourt toutes les possibilités et qu'il ne trouve aucune solution, alors il renvoie une erreur.

3.1.2 Sauvegarder des résultats

Comme vous avez pu le voir dans la figure précédente, nous avons choisi de sauvegarder la grille de sudoku résolue sous forme d'image, en faisant ressortir les chiffres ajoutés à la grille. De plus, la grille est également sauvegardée sous forme de fichier, de la même manière que l'entrée de l'exécutable solver.

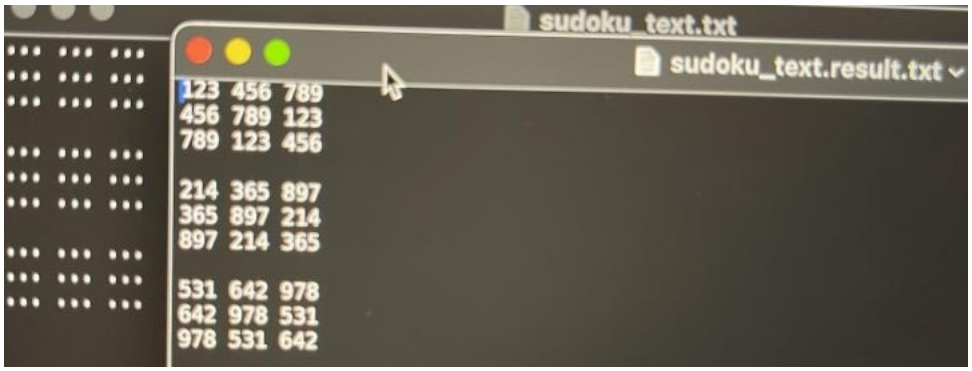

FIGURE 1 – Grille binarisée

3.2 Traitement de l'image

3.2.1 Prétraitement de l'image

Pour passer d'une image de base à une image traitable, il faut d'abord effectuer un pré-traitement de cette image, pour cela nous opérons un "grayscale" suivi d'une binarisation, nous avons choisi d'implémenter l'algorithme de Otsu qui permet de trouver un seuil (threshold) adaptatif selon la répartition statistique des différents niveaux de gris dans une image. De surcroît, nous avons implémenter un algorithme qui augmente les contrastes de gris pour normaliser la luminosité de l'image. De la même manière qu'otsu nous utilisons pour ces deux fonctionnalités des histogrammes de fréquences des différents niveaux de gris.

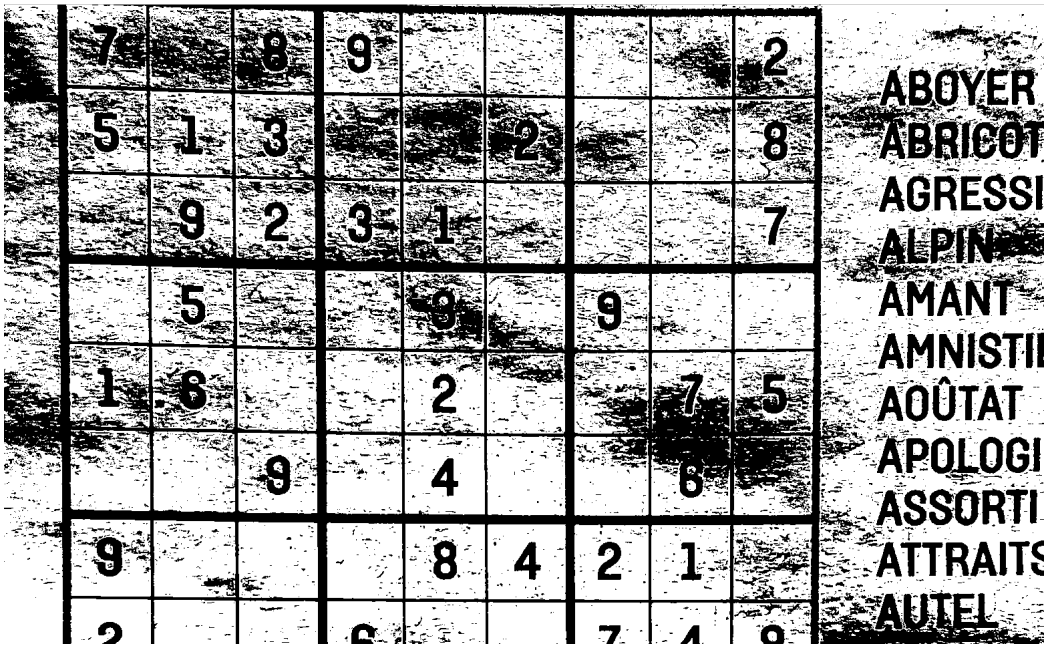

FIGURE 2 – Grille binarisée

3.2.2 Filtre et flou : Traitement de l'image

Pour régler les problèmes de bruits ou de déformations sur notre image nous utilisons plusieurs algorithmes de flou et de filtre pour que cela soit traitable ensuite. Pour l'instant nous avons implémenter le filtre de sobel, les opérateurs de Canny ainsi que les flou gaussien et de Blox.

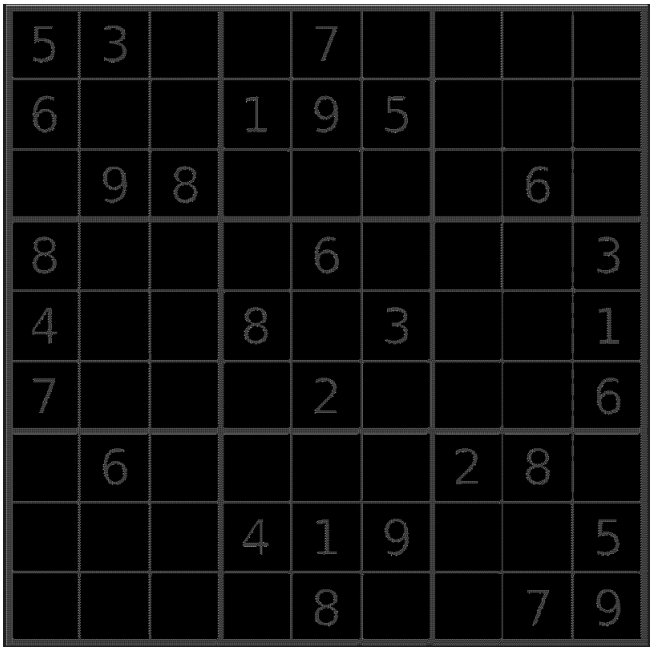

FIGURE 3 – Filtre de Sobel

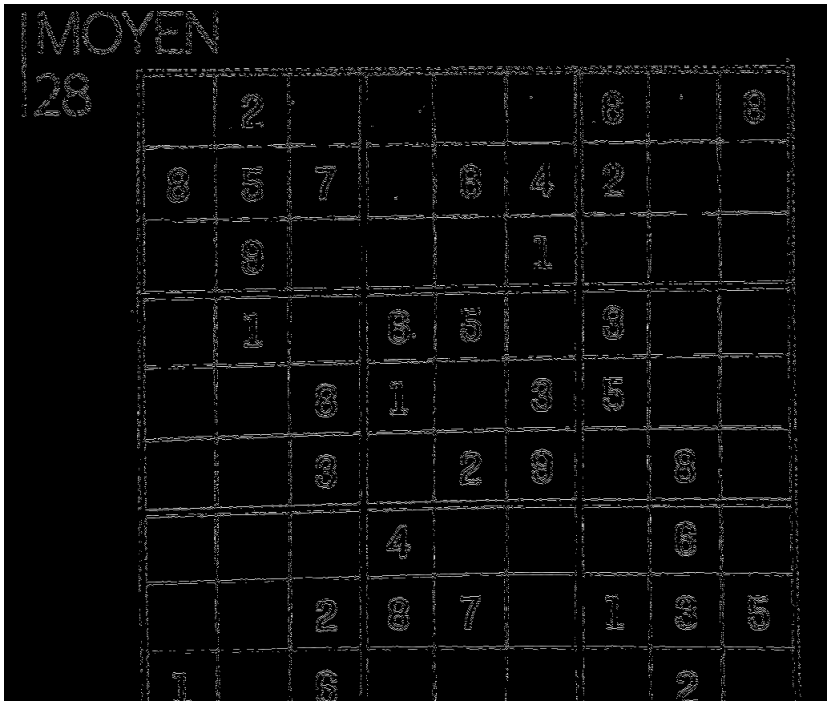

FIGURE 4 – Filtre de Canny

3.2.3 Transformée de Hough

Nous avons utilisé la transformée de Hough pour pouvoir détecter les lignes de notre grille de sudoku.

C'est un procédé qui fonctionne en utilisant des coordonnées polaire, et une matrice d'accumulation.

A travers tous les pixels blancs d'une image dont les bords ont été mis en évidence par un algorithme type "Edge-Detector" du genre Canny ou Sobel. On incrémente notre matrice d'accumulations aux coordonnées polaire de notre accumulateur.

Ensuite en utilisant un passage des coordonnées polaires à cartésiennes nous traitons les pics locaux de notre accumulateur (supérieur à ses 8 voisins) des valeurs supérieur à un seuil.

Ce traitement est le traçage d'une ligne traversant l'image suivant la ligne reconnue.

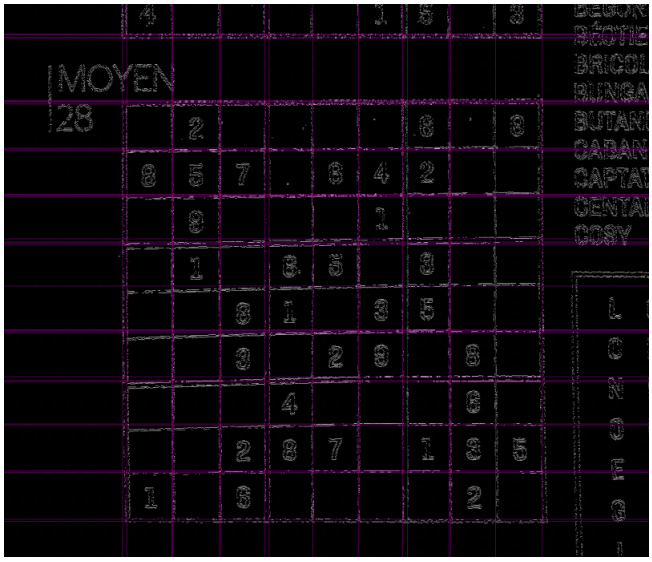

FIGURE 5 – Détection de ligne de hough

De surcroît les fonctions dessinant les lignes reconnues par Hough colorient les intersections entre ces dernières en rouge, nous avons utilisé cela pour faire un histograme de fréquence des écarts entre chaque pixel rouge de notre image, ce qui nous permet sachant qu'une grille de sudoku comprends 180 côtés égaux, en renvoyant l'indice du maximum de notre histograme d'analyser et de découper les cases sur l'image directement : Une fois les cases isolées nous chercons le plus grand carrés et les sous-carrés en éliminant toutes les lignes portées sur plus d'un axe (diagonale des carrés), le plus grand carré reconnu étant tout le temps la grille, nous colorions son périmètre en bleu avant de cropper l'image autour.

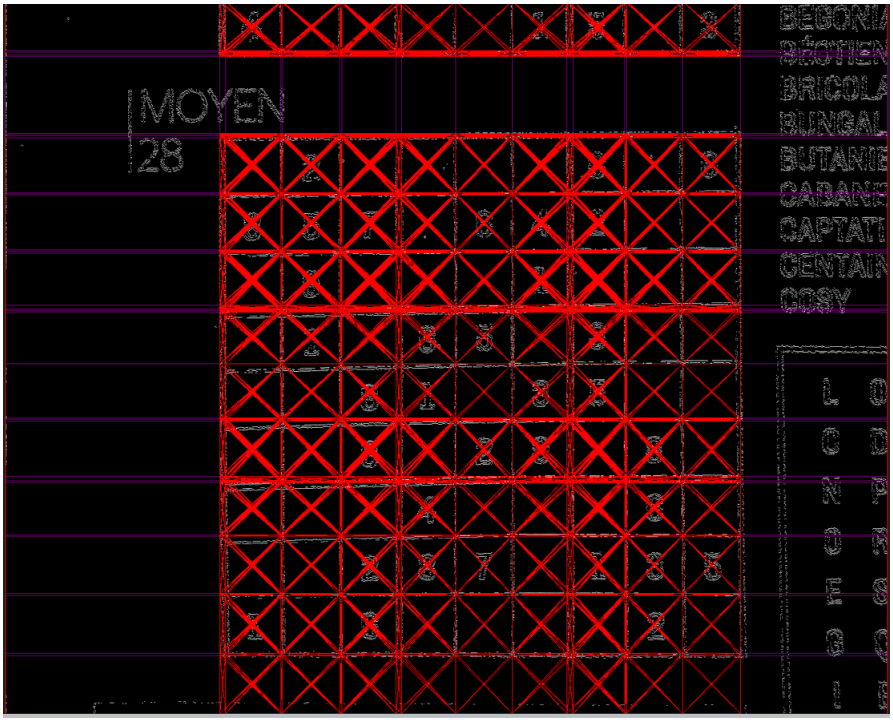

FIGURE 6 – Cases détectées par l'histograme

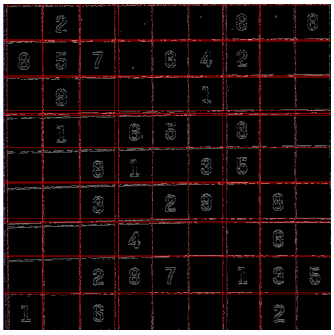

FIGURE 7 – Grille découpée

Une fois que l'on a récupéré sur l'image filtrée par Canny ou Sobel les coordonnées de la grille, et de ses cases, il suffit de reprendre l'image binarisée avant application des filtres pour y appliquer les segmentation et récupérer les cases une à une

FIGURE 8 – Case découpée

3.3 Rotation

Pour effectuer la rotation d'une image, voici l'algorithme que nous avons mis en œuvre :

— Dupliquer l'image actuelle.
— Calculer la position du nouveau pixel en utilisant la formule suivante :

$$x2 = \cos(\theta) \cdot (x1 - cx) + \sin(\theta) \cdot (y1 - cy)$$
$$y2 = \cos(\theta) \cdot (x1 - cx) - \sin(\theta) \cdot (y1 - cy)$$

Ici, θ représente l'angle de rotation en degrés, $(x1, y1)$ correspond aux anciennes coordonnées du pixel, et $(x2, y2)$ aux nouvelles coordonnées. Cependant, pour éviter d'obtenir une image avec des pixels noirs alternés, il est essentiel d'appliquer l'interpolation bilinéaire telle qu'expliquée dans la section 5.3.1.

3.3.1 Interpolation bilinéaire

Une grille régulière sert de base à l'interpolation. Les quatre points noirs correspondent aux points de données existants, tandis que le point rouge est la position pour laquelle nous souhaitons effectuer une interpolation. Cette interpolation est représentée par une fonction quadratique de la forme suivante :

$$f(x, y) = ax + by + cxy + d$$

La valeur $f(x, y)$ est l'estimation réalisée au point de coordonnées (x, y), et les constantes a, b, c, et d sont déterminées à partir des valeurs des quatre voisins $(x1, y1)$, $(x1, y2)$, $(x2, y1)$, et $(x2, y2)$ du point (x, y) pour lequel nous effectuons l'interpolation. En connaissant les valeurs en ces points, un système de quatre équations à quatre inconnues peut être établi :

$$f(x1, y1) = ax1 + by1 + cx1y1 + d$$
$$f(x2, y1) = ax2 + by1 + cx2y1 + d$$
$$f(x1, y2) = ax1 + by2 + cx1y2 + d$$
$$f(x2, y2) = ax2 + by2 + cx2y2 + d$$

L'interpolation bilinéaire peut être interprétée comme une série de deux interpolations linéaires, une dans chaque direction.

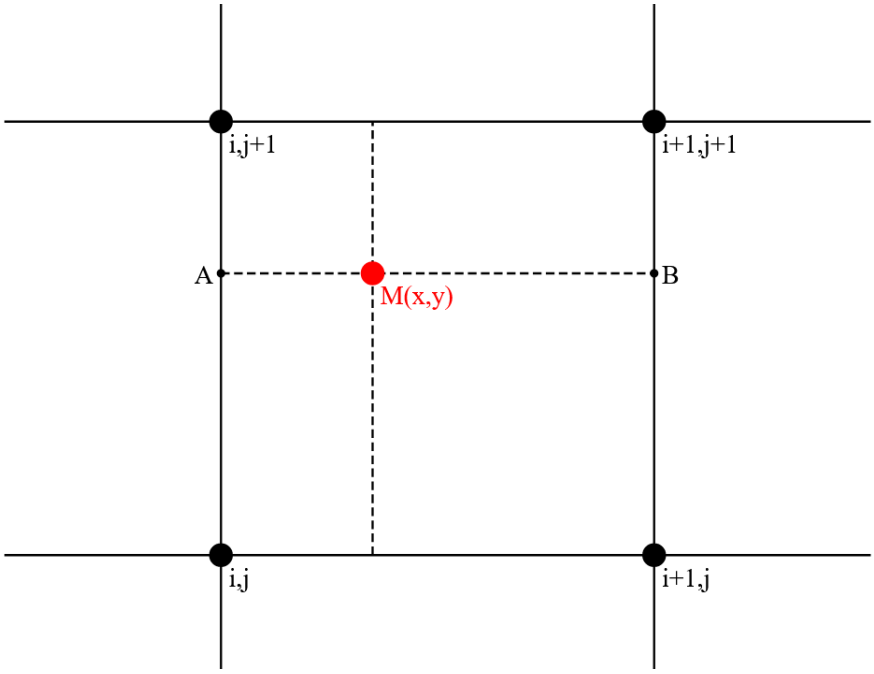

FIGURE 9 – interpolation lineaire d'un pixel

3.4 Réseau de neurones

3.4.1 Banque d'images

Pour les données permettant d'entraîner notre réseau de neurones, nous avons initialement utilisé MNIST pour *Modified National Institute of Standards and Technology*, en référence à l'Institut national des normes et de la technologie des États-Unis. Cet ensemble de données a été créé à la fin des années 1990 par deux chercheurs, Yann LeCun, Corinna Cortes, et Christopher J.C. Burges, dans le but de promouvoir la recherche en reconnaissance de caractères manuscrits et d'encourager le développement de techniques de traitement d'image.

MNIST contient un ensemble de 70 000 images en noir et blanc, chacune étant une représentation d'un chiffre manuscrit de zéro à neuf. Ces images ont été extraites de formulaires et de documents du Bureau de recensement des États-Unis et prétraitées pour obtenir une taille uniforme de 28x28 pixels.

Notre logiciel doit cependant résoudre des grilles qui ont été créées par ordinateurs et qui contiennent donc des caractères pouvant différer grandement de caractères manuscrits. Nous avons donc décider de générer notre propos banque d'images à l'aide d'un programme Python qui se récupère de très nombreuses polices et qui génère des images au format 28x28. Le programme nous permet de générer les chiffres de 0 à 9,

FIGURE 10 – Exemples d'images MNIST

mais également les lettres de A à F pour une potentielle utilisation future.

Nous utilisons pour le moment uniquement notre banque d'images, mais réfléchissons à utiliser pour la version finale du projet une combinaison des deux banques de données pour reconnaître à la fois des caractères d'ordinateur et des caractères manuscrit.

FIGURE 11 – Exemple d'image généré

3.4.2 Utilisation des matrices

Pour simplifier la création du réseau de neurones, nous avons décidé d'utiliser la représentation sous forme de matrices, une approche courante dans le domaine de l'apprentissage automatique. Cependant, ce type de données n'était pas disponible nativement en langage C. Pour résoudre cette contrainte, nous avons dû développer une structure de données personnalisée pour représenter ces matrices.

Créer cette structure a nécessité un travail minutieux, car il fallait tenir compte de la complexité des opérations matricielles requises par le réseau de neurones. Étant donné que chaque opération est répété un grand nombre de fois lors de l'entraînement, chaque petite optimisation permet de gagner un temps non-négligeable. Nous avons également implémenté les principaux outils nécessaires à la manipulation des matrices, tels que les opérations de multiplication, d'addition et de transposition.

Les données de la matrice sont stockées dans un espace contigu de la mémoire en suivant l'ordre *row-major*. Une des optimisations appliquées à la transposition a été de ne plus recopier totalement la matrice, mais de simplement inverser le nombre de colonnes et le nombre de ligne, puis d'utiliser l'ordre *column-major*. D'autres optimisations pourraient être effectuées sur les fonctions appliquées aux matrices, nous les effectuerons pour la dernière soutenance si nécessaire.

```
typedef struct {
        int rows;
        int columns;
        int isTransposed;
        double* data;
} Matrix;
```

FIGURE 12 – Structure des matrices

3.4.3 Création du réseau

Pour la composition de notre réseau de neurones, nous avons suivi à la lettre les conseils écrits dans le dossier du projet. Notre réseau est donc composer de 3 couches. Une première couche de 784 neurones servant d'entrée au réseau. 748 correspond au résultat de la multiplication de 28 par 28, c'est-à-dire le nombre de pixels qui compose une image au format normalisé comme définit plus haut. En seconde position, nous avons une couche cachée de 256 neurones utilisant la fonction sigmoïde. Finalement, la troisième couche sert de sortie au réseau et contient 10 neurones correspondant aux 10 chiffres entre 0 et 9. On applique sur cette couche la fonction softmax qui permet d'obtenir une probabilité pour chaque chiffre.

Pour l'entraînement, nous avons initialement utilisé le jeu de donnée MNIST et plus précisément les 60 000 images dédiées à l'entraînement. Toutes les 1000 passes d'entraînement nous testons notre réseau sur les 10 000 images dédié à cela. Avec ce jeu de donnée et notre réseau plutôt simple, nous obtenons un résultat de 96% sur le jeu de test, un résultat plutôt convenable étant donné que nous n'avons pas cherché à obtenir les paramètres les plus optimisé possible. Cependant, lorsque nous avons essayé d'utiliser notre réseau sur des chiffres généré par ordinateur, seul 2 images ont été reconnue correctement sur 6. Lorsque nous sommes passés sur un entraînement à partir d'images généré par ordinateur, les 6 images ont été correctement reconnue. La figure 4 présente la sortie console obtenue lors de l'entraînement de notre réseau avec le jeu de donnée MNIST. On peut voir qu'à partir de la 6000e passe d'entraînement déjà 92% des images du jeu de tests sont correctement reconnue.

Maintenant que nous avons une base fonctionnelle, nous allons pouvoir effectuer des améliorations pour la soutenance finale. Tout d'abord, nous allons vérifier que notre code actuel ne contient pas d'erreurs qui pourraient faire baisser la fiabilité du réseau. Nous essaierons ensuite de mieux configurer notre réseau pour obtenir des paramètres qui soient à la fois optimisés et avec un bon taux de reconnaissance des images. Il faudra également s'adapter à la partie traitement de l'image. En effet, nous utilisons pour le moment des images "parfaites" pour tester le réseau, elles sont générées sur mesure et ne contiennent que le caractère sans aucun défaut. Si toutes les parties précédentes fonctionnent correctement nous pourrons voir pour améliorer le réseau et reconnaître à

```
Epoch : 0, Train accur : 0.187500, Test accur : 0.116300
Epoch : 1000, Train accur : 1.000000, Test accur : 0.856600
Epoch : 2000, Train accur : 1.000000, Test accur : 0.887300
Epoch : 3000, Train accur : 0.937500, Test accur : 0.901500
Epoch : 4000, Train accur : 1.000000, Test accur : 0.914200
Epoch : 5000, Train accur : 0.937500, Test accur : 0.916000
Epoch : 6000, Train accur : 0.875000, Test accur : 0.924500
Epoch : 7000, Train accur : 0.812500, Test accur : 0.921800
Epoch : 8000, Train accur : 0.937500, Test accur : 0.925700
Epoch : 9000, Train accur : 0.875000, Test accur : 0.928700
```

FIGURE 13 – Sortie du programme lors de l'entraînement avec MNIST

la fois des caractères manuscrits et informatiques.

3.5 Application

Un visuel est une interface utilisateur nous à sembler être primordiale pour un tel projet. Ainsi nous avons décidé de développer une application en utilisant la librairie GTK ainsi que le logiciel Glade afin d'utiliser un fichier XML pour gérer tous les widgets de l'application en évitant de très longues lignes de code qui sont très lourde, bien souvent redondante et peut lisible.

L'objectif final de cette application est de pouvoir importer facilement une image d'une grille de sudoku puis de la résoudre grâce à toutes les autres fonctions réalisées et finalement d'afficher le résultat final tout en nous permettant d'enregistrer ce résultat dans un fichier contenant une image. Tout ceci dans une interface utilisateur facile à comprendre et à utiliser, ce qui permettra d'éviter de passer par des lignes de commandes qu'une personne lambda aurait du mal à comprendre.

3.5.1 Parsing input utilisateur

La récupération des saisies de l'utilisateur est très importante et ceci sera géré majoritairement via l'application. Pour le moment il est déjà possible de récupérer le fichier image que l'utilisateur veut résoudre ainsi que les clics sur les boutons pour lancer les fonctions de résolution. Dans le futur nous rajouterons certainement d'autres informations utilisateurs à récupérer si ceci nous semble utile.

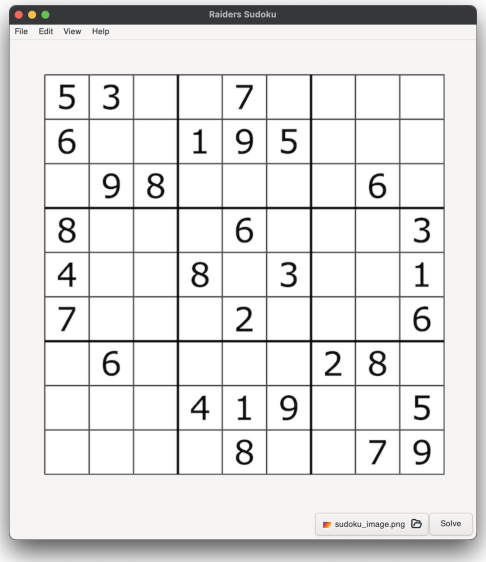

Figure 14 – Rendus de l'application

4 Conclusion

La première soutenance de notre projet a marqué un excellent début de parcours pour Raiders Sudoku. Les différentes étapes de développement sont en bonne voie, avec des points clés déjà bien avancés.

Tout d'abord, notre réseau de neurones s'est révélé prometteur, avec un taux de reconnaissance de 96% sur le jeu de données MNIST. Cependant, nous sommes conscients des défis à venir, notamment l'adaptation à des chiffres générés par ordinateur et la recherche de paramètres plus optimisés. Nous continuerons à améliorer notre réseau pour augmenter sa précision.

La partie de résolution du Sudoku est maintenant achevée. Nous avons développé un algorithme de backtracking efficace pour résoudre les grilles et avons mis en place des vérifications de solubilité pour éviter de perdre du temps sur des grilles insolubles. De plus, nous avons réussi à sauvegarder les résultats sous forme d'images et de fichiers.

L'application que nous développons avec GTK et Glade sera un atout majeur pour notre projet, offrant une interface conviviale pour les utilisateurs finaux. La récupération des saisies de l'utilisateur se déroule de manière fluide, et nous envisageons d'ajouter plus de fonctionnalités à l'avenir pour rendre l'application encore plus polyvalente.

Enfin, malgré quelques retards mineurs, nous sommes confiants quant à notre capacité à rattraper le temps perdu. Il reste du travail à accomplir, mais notre équipe est déterminée à relever les défis qui se présentent.

En conclusion, nous sommes fiers des progrès accomplis jusqu'à présent, et nous sommes impatients de poursuivre notre travail sur Raiders Sudoku pour atteindre nos objectifs pour la prochaine soutenance. Notre équipe est prête à relever les défis à venir et à fournir une solution de haute qualité pour la résolution de grilles de Sudoku.