

OCR Sudoku Solver

Sudoku Conquerors

11 Décembre 2023

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

Table des matières

| | | |
|----------|--|----------|
| 1 | Notre groupe | 4 |
| 1.1 | Les membres du groupe | 4 |
| 2 | Répartition des tâches | 6 |
| 2.1 | Réseau de neurones | 6 |
| 2.2 | Détection de la grille | 7 |
| 2.3 | Découpage de l'image | 7 |
| 2.4 | Prétraitement | 7 |
| 2.4.1 | Suppressions des couleurs | 7 |
| 2.4.2 | Rotation de l'image | 7 |
| 2.5 | Sudoku Solver | 8 |
| 2.6 | Affichage du résultat | 8 |
| 3 | Etat d'avancement du projet et aspects techniques | 8 |
| 3.1 | Réseau de neurones (Evariste et Yako) | 8 |
| 3.1.1 | Soutenance 1 | 8 |
| 3.1.2 | Soutenance 2 | 11 |
| 3.2 | Prétraitement | 15 |
| 3.2.1 | Suppression des couleurs (Eliott) | 15 |
| 3.2.2 | Rotation (Eliott) | 21 |
| 3.2.3 | Détection de la grille (Robin) | 25 |
| 3.2.4 | Découpage de l'image (Robin) | 32 |
| 3.2.5 | Soutenance 2 | 32 |

| | | |
|----------|--|-----------|
| 3.3 | Interface utilisateur (Thomas) | 34 |
| 3.3.1 | Site Web | 34 |
| 3.3.2 | Application | 35 |
| 4 | Problèmes rencontrés | 42 |
| 4.1 | Réseau de neurones | 42 |
| 4.2 | Suppression des couleurs | 43 |
| 4.3 | Détection de la grille | 47 |
| 5 | Conclusion | 48 |

1 Notre groupe

Notre groupe s'est constitué assez vite suite à l'annonce du projet. Nous nous entendons tous très bien. Nous sommes un groupe de 5 élèves et nous sommes enthousiastes à l'idée de réaliser ce projet.

Contrairement à l'année dernière, ce projet est imposé à tous les groupes et le sujet n'est plus libre. Cela nous plaît beaucoup car ce projet implique de nouvelles choses à apprendre telle que la reconnaissance de chiffre avec un réseau de neurones, ou encore la détection de grille sur une image relevant d'un défi encore jamais réalisé par aucun des membres du groupe.

Ainsi, notre groupe aspire à faire le meilleur projet possible avec les outils qu'il a en main.

1.1 Les membres du groupe

Notre groupe est constitué de 5 personnes :

- Elliott : Mon rôle au sein de cette équipe dynamique englobe la transformation en niveaux de gris et la rotation des images, deux éléments clés pour garantir la précision de notre reconnaissance de caractères. J'ai choisi de me pencher sur ces aspects spécifiques parce que je crois fermement en leur importance cruciale. De plus, j'ai déjà eu l'occasion de travailler sur l'optimisation de la conversion en niveaux de gris, ce qui nous permettra de partir du bon pied. Travailler en équipe peut représenter un défi, mais je suis convaincu que ce défi est une opportunité de croissance. Ensemble, nous allons explorer en profondeur tous les aspects techniques de ce projet et créer une application novatrice, aboutissement de nos efforts collectifs.

- Evariste : Je vais m'occuper du réseau de neurone avec Yako et du résolveur de sudoku. J'ai choisi de m'occuper du réseau de neurone car l'intelligence artificielle est un domaine qui m'intéresse beaucoup et comprendre le fonctionnement d'un réseau de neurones est essentielle pour se plonger dans ce domaine.
- Robin : "Je suis enthousiaste à l'idée de participer à un projet sur l'intelligence artificielle, où je vais relever le défi d'apprendre le langage C et d'utiliser SDL. Travailler en équipe représente un défi pour moi, mais j'ai hâte de progresser dans ce domaine. J'ai hâte de commencer cette aventure, de surmonter les obstacles et de contribuer au succès du projet, car je crois en notre capacité à repousser nos limites."
- Thomas : "J'ai hâte de commencer à travailler sur ce projet de deuxième année. J'avais déjà entendu que l'on devait réaliser un certain solver de sudoku, mais je ne pensais pas que la réalisation de celui-ci allait être aussi intéressant. J'ai plutôt bien compris les cours que nous avons eu depuis le début d'année en C, et je pense donc pouvoir apporter de mes connaissances au groupe. J'ai vraiment hâte de découvrir les moindres recoins de ce projet, et de pouvoir une nouvelle fois travailler en groupe."
- Yako : "Je suis enthousiaste à l'idée de travailler sur ce projet. J'ai vraiment hâte de travailler sur les différents aspects techniques de ce projet et j'ai hâte qu'on puisse présenter notre application. J'ai de bonnes compétences en programmation et je vais pouvoir les mettre en valeur durant toute la durée ce projet."

2 Répartition des tâches

| Tâches | Thomas | Yako | Evariste | Robin | Eliott |
|---------------------------|--------|------|----------|-------|--------|
| Site Web | X | | | | |
| Réseau de neurones | | X | X | | |
| Détection de la grille | | | | X | |
| Solver | | | X | | |
| Rotation de l'image | | | | | X |
| Suppressions des couleurs | | | | | X |
| Découpage de l'image | | | | X | |
| Sauvegarde du résultat | X | | | | |
| Application GTK | X | | | | |

Ci-dessus : Tableau de la répartition des tâches entre les membres

2.1 Réseau de neurones

Tout d'abord, c'est Evariste et Yako qui auront la lourde tâche de réaliser les deux réseaux de neurones qui sont demandés pour la première et la dernière soutenance.

Ayant déjà été en groupe pour le projet du 2ème semestre de l'année de SUP, ils savent comment bien travailler ensemble et réaliseront cette tâche de la meilleure façon possible.

De plus, ils ont réalisé l'année dernière une présentation sur l'intelligence artificielle et ont ainsi déjà des connaissances dans le domaine du Machine Learning ainsi que dans les réseaux de neurones.

Pour cela, ils vont devoir tout d'abord réaliser un réseau de neurones capable d'apprendre la fonction logique XOR. Puis, pour la soutenance

finale, ils devront réaliser un réseau de neurones capable de reconnaître les différents chiffre de 0 à 9, qui sont écrits à l'ordinateur.

2.2 Détection de la grille

Ensuite, c'est Robin qui aura la tâche de réaliser l'algorithme nécessaire pour la détection de la grille de sudoku. Cette tâche est également lourde et laborieuse mais les autres membres du groupe lui viendront en aide si besoin car c'est avant tout un travail de groupe.

2.3 Découpage de l'image

Robin aura également la tâche d'implémenter le moyen de découper l'image et de sauvegarder les cases de la grille de sudoku, une tâche qui demande aussi des recherches et du temps. Cette étape est nécessaire et importante car c'est ces images découpées qui seront données au réseau de neurones et qui permettra la reconstruction de la grille de sudoku de façon numérique.

2.4 Prétraitement

2.4 Suppressions des couleurs

Elliott s'occupera de réaliser les filtres d'images nécessaires à la suppression des couleurs sur une image pour obtenir une image plus simple à traiter et limiter les possibilités d'erreurs pour le réseau de neurone. Cela permettra aussi à notre application d'accepter en entrée des documents directement issus d'un scanner ou d'une photo.

2.4 Rotation de l'image

Dans cette idée de prétraitement, Elliott s'occupera aussi de l'implémentation d'un algorithme permettant la rotation d'une image d'un certain angle pour faciliter le redressement d'une image manuel et auto-

matique, permettant le fait de pouvoir mettre une image pas complètement parfaite dans notre application.

2.5 Sudoku Solver

Evariste a la tâche de réaliser l'algorithme du Sudoku Solver. Le Sudoku, bien que semblant simple à première vue, présente des défis mathématiques et logiques complexes. Pour résoudre efficacement ces grilles de 9x9 cases, il faut un algorithme capable d'analyser chaque numéro présent, de suivre les règles du jeu et de déterminer la solution de manière précise et rapide.

2.6 Affichage du résultat

Comme nous sommes cinq à réaliser ce projet, Thomas a décidé de réaliser une application finale. Cette application permet de lier chacune des différentes parties entre elle. L'objectif étant d'avoir une représentation concrète et facile à l'utilisation pour une personne qui ne voudrait pas passer par un terminal afin d'exécuter le solver de sudoku.

3 Etat d'avancement du projet et aspects techniques

3.1 Réseau de neurones (Evariste et Yako)

3.1 Soutenance 1

Les réseaux de neurones sont une composante fondamentale de notre projet. En effet, sans un bon réseau de neurones avec les paramètres qui conviennent, il n'est pas possible de résoudre un sudoku car les cases du sudoku ne sont pas correctement reconnus. Ainsi, il est important de bien s'informer sur le fonctionnement des réseaux de neurones ainsi que de leur trouver une implémentation adéquate en C.

Nous avons découvert le travail d'un développeur astucieux qui avait

créé une structure de données Matrix et une classe NeuralNetwork en langage C. Cette ressource s'est avérée être une pierre angulaire pour notre tâche, car elle a facilité considérablement notre travail. La structure Matrix contenait toutes les opérations nécessaires tel que la multiplication matricielle, la transposition, l'addition et la soustraction, toutes ces opérations sont nécessaires au bon déroulé du réseau de neurones. Grâce à cette base solide, on a pu se concentrer sur la compréhension des principes fondamentaux de l'apprentissage profond.

```

typedef struct {
    int input; //Number of input neurons
    int output; //Number of output neurons
    double learning_rate; //Learning rate
    int hidden; //Number of hidden layers
    int *hiddens; //Number of neurons per hidden layer
    Matrix** hidden_weights; //List of all the weights in the hidden layers
    Matrix** hidden_bias; //List of all the bias in the hidden layers
    Matrix* output_bias; //Bias in the output layer
    Matrix* output_weights; //Weight in the output layer
} NeuralNetwork;

```

```

12
13
14
15     typedef struct {
16         double** entries;
17         int rows;
18         int cols;
19     } Matrix;
20
21
22
23

```

FIGURE 1 – Structures utilisées

On a pris le temps d'explorer et de décortiquer la logique derrière chaque ligne de code, pour qu'on puisse bien se l'approprier et ainsi cela nous a permis d'acquérir une compréhension approfondie des mécanismes sous-jacents à un réseau de neurones. On a donc adapté cette structure à notre réseau de neurone visant à apprendre XOR en modifiant les paramètres, les fonctions d'activation et les hyperparamètres.

Après avoir essayé différents paramètres, nous avons réussi à préciser le pourcentage de réussite de notre réseau de neurone.

On a réalisé un réseau de neurone capable de s'occuper du cas XOR. Le plus dur a été de bien comprendre le fonctionnement de la backpropa-

gation et l'influence de chacun des paramètres sur la capacité d'apprentissage du réseau de neurones comme le nombre de couches, le nombre de neurones par couche et la learning rate.

Pour le réseau XOR, on l'a paramétré de la manière suivante : il est constitué de 2 neurones d'entrées (l'opérateur XOR a besoin de seulement 2 entrées (0 ou 1), il possède une couche cachée contenant 2 neurones et 1 seul neurone de sortie qui a une valeur proche de 0 ou proche de 1, cela nous permet d'avoir un résultat qui convient. On utilise alors la fonction `round()` de `<math.h>` pour obtenir la valeur entière la plus proche qui est soit 0 soit 1.

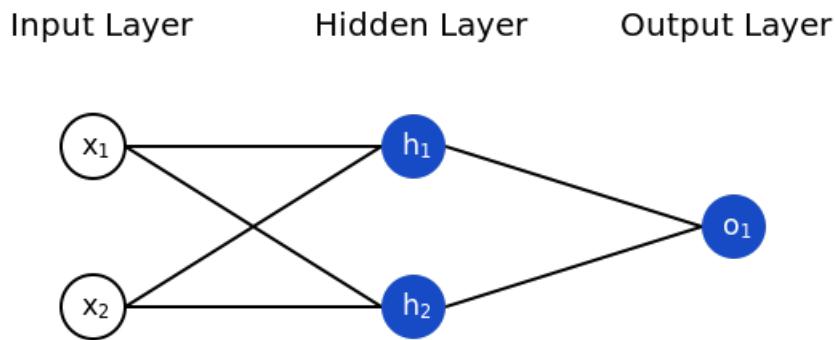


FIGURE 2 – Architecture du réseau de neurone XOR

Il est arrivé que l'entraînement du réseau XOR tourne indéfiniment et que le score stagne (par exemple, au bout de 3000 répétitions d'entraînements, le score était de 76% et au bout de 4000 aussi de 76%, comme si il n'y avait pas d'amélioration), dans ce cas, on réinitialise notre réseau pour laisser faire l'aléatoire et on arrive à obtenir dans tous les cas 100% de précision.

3.1 Soutenance 2

Pendant la phase finale de notre projet, nous avons consacré nos efforts à la conception et à l'implémentation d'un réseau de neurones capable de reconnaître les chiffres de 0 à 9, spécifiquement ceux écrits à l'ordinateur.

Cette tâche s'est avérée à la fois stimulante et complexe, nécessitant l'utilisation des structures Matrix et NeuralNetwork, ainsi que le guide détaillé fourni dans le cahier des charges du projet.

Dans un premier temps, nous avons réussi à mettre en œuvre avec succès les algorithmes de feed forward et de backpropagation indispensables au bon fonctionnement du réseau. Les étapes cruciales de l'entraînement et des tests du réseau ont également été développées avec succès. Pour les fonctions d'activation, nous avons premièrement opté pour la sigmoïde et sa dérivée, en combinant cela avec une descente de gradient et un learning rate. La fonction sigmoïde et sa dérivée sont :

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$$

L'évaluation des performances du réseau nous a conduits à explorer différentes fonctions d'activation, en privilégiant finalement la fonction Leaky ReLU et la dérivée de ReLU basique :

$$\text{Leaky ReLU}(x) = \begin{cases} 0.01x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

$$\text{ReLU}'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

Ces fonctions ont donné des résultats plus prometteurs que la sigmoïde donc nous avons optés pour celles-ci.

En ce qui concerne les jeux de données d'entraînement, nous avons découvert un ensemble sur Kaggle comprenant environ 6000 images que nous avons judicieusement divisé en ensembles de train et de test pour l'entraînement du réseau.

Cet ensemble de données est au format CSV, et il est d'une importance cruciale pour l'entraînement de notre réseau de neurones. Ce fichier CSV est structuré de manière spécifique, où la première colonne contient les étiquettes (labels) correspondant aux chiffres représentés par les images. Chaque ligne du fichier CSV représente une image de chiffre, et les pixels de cette image sont inclus dans les colonnes suivantes.

FIGURE 3 – Exemple du format dans le fichier .csv

Afin de tirer parti de ce jeu de données dans le contexte de notre réseau de neurones, nous avons utilisé une fonction nommée "csv_to_imgs". Cette fonction joue un rôle essentiel dans la conversion de chaque ligne du CSV en un objet en langage C de type "Img". L'objet "Img" est une structure spécialement conçue pour notre réseau de neurones et encapsule les informations nécessaires pour traiter chaque image de manière appropriée.

```
typedef struct {
    Matrix* img_data;
    int label;
} Img;
```

FIGURE 4 – Structure "Img"

Cette procédure de conversion nous a permis de transformer efficacement les données brutes du CSV en un format exploitable par notre réseau de neurones. Une fois cette étape accomplie, nous avons pu utiliser chaque image ainsi convertie dans le dataset pour entraîner notre modèle. Cette méthodologie a été cruciale pour adapter les données d'entrée à la structure de notre réseau, facilitant ainsi le processus d'entraînement et permettant à notre modèle de reconnaître de manière précise les chiffres de 0 à 9 présents dans les images informatiques.

Une partie substantielle de notre travail a été dédiée aux essais et aux ajustements, en particulier pour déterminer la learning rate optimale et le nombre de neurones nécessaires dans la seule couche cachée de notre réseau.

Suite à de nombreux tests, nos résultats les plus probants ont émergé lorsque nous avons utilisé une learning rate de 0.08 avec une configuration de 200 neurones dans la couche cachée, atteignant ainsi une réussite de 96% sur le dataset d'entraînement.

De manière particulièrement gratifiante, notre réseau a également réussi à reconnaître les cinq images de sudoku incluses dans le test, avec un taux de réussite impressionnant de 100%. Ces résultats soulignent la robustesse de notre approche et la capacité de notre réseau de neurones à généraliser efficacement à de nouvelles données.

Ainsi, l'architecture de notre réseau de neurones est la suivante :

- Input : 784 neurones
 - Hidden layer (1) : 200 neurones
 - Output layer : 10 neurones
-

- Learning rate : 0.08
- Fonctions d'activations : Leaky ReLU et la dérivée de ReLU

3.2 Prétraitement

3.2 Suppression des couleurs (Eliott)

La suppression des couleurs est une étape fondamentale de notre projet OCR de résolution de grilles de sudoku. Elle permet de préparer l'image pour la détection de la grille, la reconnaissance des caractères, et la résolution du sudoku. Nous aborderons cette étape avec une attention particulière pour garantir des résultats de haute qualité, même face à des images d'entrée complexes. L'une des étapes cruciales de notre projet consiste en la suppression des couleurs d'une image contenant une grille de sudoku, pour ensuite travailler en niveaux de gris puis en noir et blanc. Cette phase est essentielle pour optimiser la détection des caractères et des cases de la grille, ainsi que pour préparer l'image à la reconnaissance de caractères.

Conversion en niveaux de gris : Pour commencer, nous convertissons l'image initiale en niveaux de gris. Cette opération permet de réduire la complexité de l'image en éliminant l'information colorée, tout en conservant les nuances de gris qui sont essentielles pour la détection des caractères.

Binarisation (conversion en noir et blanc) : Après la conversion en niveaux de gris, l'étape suivante est la binarisation de l'image. Cette étape consiste à transformer l'image en une image binaire, où les pixels sont soit noirs (représentant l'encre) soit blancs (représentant le papier de fond). La binarisation est cruciale pour simplifier la détection des traits et des caractères. Plusieurs méthodes de binarisation peuvent être utilisées, telles que la méthode d'Otsu, l'adaptative thresholding, etc. Le choix de la méthode dépendra de l'image d'entrée et de sa qualité.

Prétraitement : Après la binarisation, un certain prétraitement peut être appliqué pour améliorer la qualité de l'image. Cela peut inclure des opérations telles que la réduction du bruit, la suppression des artefacts indésirables, et la mise en évidence des traits et des contours des caractères. Les opérations de prétraitement visent à simplifier l'analyse ultérieure de l'image.

Résultats attendus : À la fin de cette étape, nous devrions obtenir une image en noir et blanc de haute qualité, où les caractères du sudoku et les lignes de la grille sont clairement visibles. Cette image préparée sera la base pour la détection de la grille, des cases, la reconnaissance des caractères et la résolution du sudoku.

Défis potentiels : La suppression des couleurs et la binarisation peuvent être sensibles à la qualité de l'image d'entrée, aux ombres, aux reflets, et à d'autres anomalies. Des techniques avancées de traitement d'image, telles que la morphologie mathématique, la détection de contours, et la suppression du bruit, seront nécessaires pour surmonter ces défis.

Prétraitement des Images :

Le prétraitement des images représente la première étape cruciale de notre algorithme, visant à optimiser la qualité des images en entrée. Cette phase se révèle essentielle pour garantir la performance et la précision ultérieures du processus de résolution de sudoku. Avant de procéder à la conversion en niveaux de gris, nous devons anticiper et corriger d'éventuelles distorsions introduites par des filtres ou une prédominance de couleur dans les images.

Notre approche novatrice consiste à identifier les couleurs les plus présentes dans l'image, puis à les convertir en blanc. Cette stratégie permet d'accentuer le contraste de l'image, améliorant ainsi la visibilité des dé-

tails cruciaux. L'utilisation de cette technique avant la conversion en niveaux de gris est fondamentale, car elle garantit une adaptabilité accrue face à des variations d'éclairage et à des altérations chromatiques.

En résumé, le prétraitement des images ne se limite pas à une simple conversion de couleur, mais constitue une véritable correction anticipée des défauts potentiels. Cette approche proactive renforce la robustesse de notre algorithme face à une diversité d'images d'entrée, assurant ainsi des résultats fiables et précis lors de la résolution du sudoku.

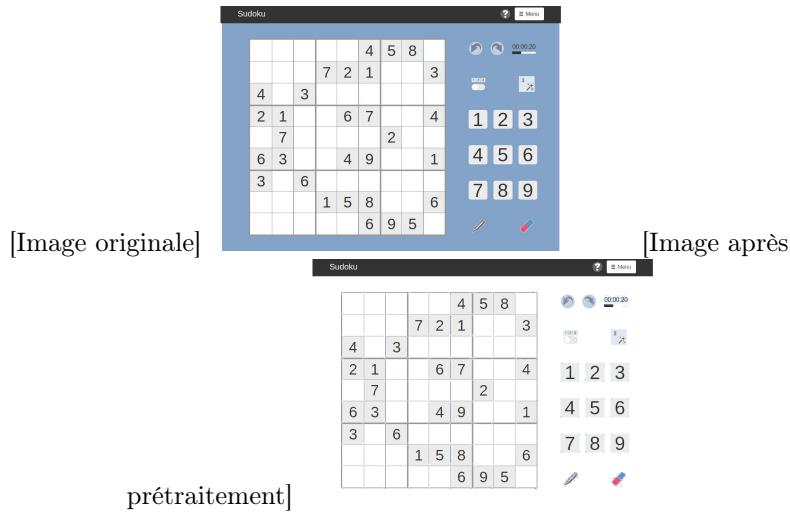


FIGURE 5 – Avant/Après prétraitement

L’étape de conversion en niveaux de gris est d’une importance cruciale dans le processus de traitement d’images pour la résolution de grilles de sudoku via OCR. La conversion en niveaux de gris vise à transformer une image couleur en une version monochrome qui conserve les nuances de gris, tout en éliminant les informations de couleur inutiles. Il existe plusieurs algorithmes couramment utilisés pour effectuer cette conversion, et le choix de l’algorithme dépend de divers facteurs tels que la qualité de l’image, les performances requises et les besoins spécifiques de l’application.

Le choix de l’algorithme de conversion en niveaux de gris dépendra de la nature de l’image d’entrée et de la complexité de la tâche de reconnaissance de caractères. Il peut être judicieux d’expérimenter avec différents algorithmes pour déterminer celui qui offre les meilleurs résultats pour notre application de résolution de grilles de sudoku. Parallèlement, l’optimisation de ces algorithmes est cruciale pour garantir des performances rapides dans le traitement des images, car la vitesse de traitement est essentielle pour une expérience utilisateur fluide.

Voici les deux algorithmes de conversion en niveaux de gris qui ont été pris en considération : Moyenne pondérée (Luminance) : Cet algorithme est largement utilisé pour la conversion en niveaux de gris, car il tient compte de la perception humaine de la luminosité. Il attribue des poids différents aux composantes de couleur (rouge, vert et bleu) en fonction de leur importance dans la perception de la luminance. La formule typique est :

$$Gris = 0.22 \times R + 0.65 \times G + 0.11 \times B$$

Cette formule favorise le canal vert (G) car il est plus sensible à la perception de la luminance par l'œil humain.

Cependant, l'algorithme dont nous avons besoin sera, par la suite, exploité par un ordinateur. Il n'est donc pas utile de surcharger le canal vert car le programme qui traitera l'image par la suite ne prendra pas en compte le biais de l'œil humain. Pour cela, nous avons utilisé un autre algorithme dont voici la formule :

$$Gris = 0.299 \times R + 0.587 \times G + 0.114 \times B$$

Décomposition en teinte-saturation-luminance (HSL) : En utilisant l'espace colorimétrique HSL, vous pouvez extraire la composante de luminance (L) pour obtenir une image en niveaux de gris. Cela permet une plus grande flexibilité dans le choix de la composante de couleur à privilégier. Ainsi, nous avons pu choisir un nombre teintes prédéfinis (20) et l'adapter à notre luminance. L'image finale se détermine en fonction de ce nombre de teinte, ce qui nous permet d'obtenir une image fluide mais également segmentée en un nombre fixe et simple de niveaux de gris.

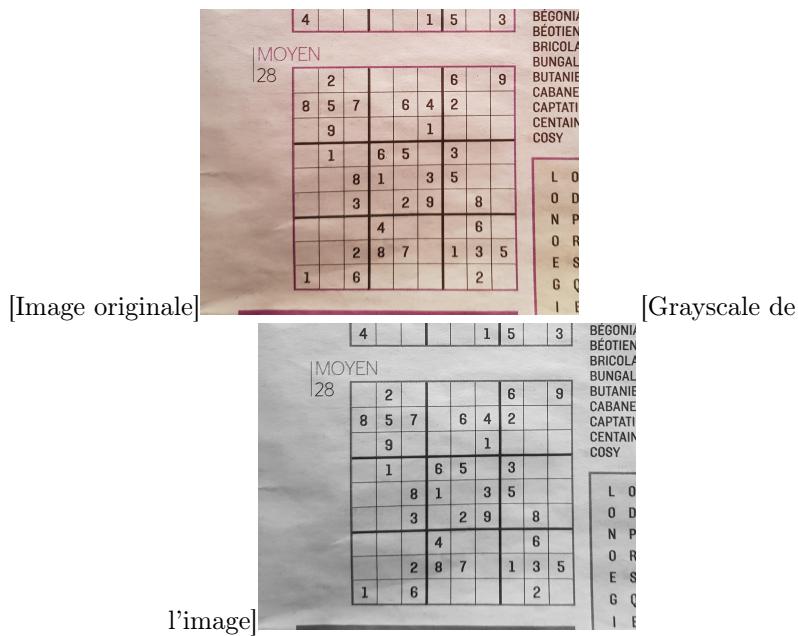


FIGURE 6 – Avant/Après filtre Grayscale

Binarisation Adaptative :

Après avoir perfectionné le contraste des images et les avoir converties en niveaux de gris, la phase cruciale de binarisation intervient. Cette étape revêt une importance majeure dans le processus global de résolution du sudoku, puisqu'elle transforme l'image en noir et blanc, préparant ainsi le terrain pour les analyses ultérieures.

Notre approche de binarisation est basée sur un seuil adaptatif, conçu pour s'adapter de manière dynamique aux nuances de gris spécifiques à chaque image. Contrairement à une approche statique, notre algorithme analyse une liste d'occurrences des 256 nuances de gris présentes dans l'image. Ce processus implique un parcours méticuleux de l'image en niveaux de gris pour enregistrer la fréquence de chaque nuance.

Le seuil est ensuite calculé en identifiant la valeur à partir de laquelle

au moins 10% des pixels appartiennent à l'image. Cette approche adaptative garantit une binarisation précise, ajustée aux caractéristiques intrinsèques de chaque image, renforçant ainsi la robustesse de l'algorithme dans des conditions variables.

En conclusion, la binarisation adaptative représente une avancée significative dans la qualité du traitement d'image, assurant une conversion efficace en noir et blanc, prélude essentiel à l'étape suivante de résolution du sudoku.

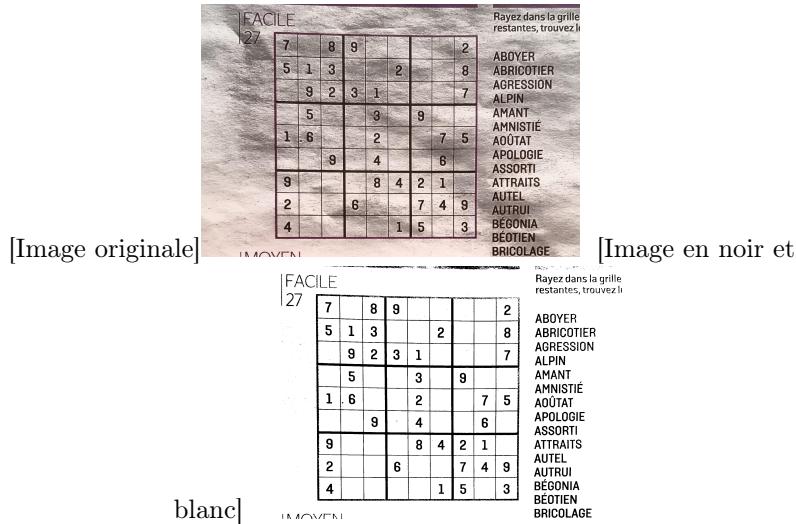


FIGURE 7 – Avant/Après Noir et Blanc

3.2 Rotation (Elliott)

Rotation manuelle

La rotation de l'image est une étape essentielle de notre projet de résolution de grilles de sudoku par OCR. Elle repose sur la détection de l'angle de rotation, le redressement de l'image, l'interpolation, et des méthodes d'optimisation pour assurer des performances efficaces. Cette

phase est essentielle pour garantir la précision de la détection des cases et la reconnaissance des caractères, contribuant ainsi à la réussite globale de notre application.

Cette étape vise à compenser les éventuelles inclinaisons ou rotations de l'image de la grille de sudoku, de manière à la redresser et à la rendre parfaitement alignée. La rotation de l'image est cruciale pour garantir la précision de la détection de la grille et la reconnaissance des caractères.

Voici comment nous allons aborder cette phase :

- Redressement de l'image : Une fois l'angle de rotation détecté, nous pouvons procéder au redressement de l'image. Cette opération consiste à appliquer une transformation affine à l'image, qui la fera pivoter de manière à ce que la grille de sudoku soit parfaitement alignée avec les bords de l'image. Cette étape est cruciale pour simplifier la détection de la grille et des cases, ainsi que pour assurer une reconnaissance de caractères précise.
- Interpolation et ajustement : Lors de la rotation de l'image, des pixels supplémentaires peuvent être introduits ou des pixels existants peuvent être supprimés. Pour éviter toute perte d'information ou toute déformation, une interpolation peut être utilisée pour ajuster l'image redressée. Les méthodes d'interpolation courantes incluent l'interpolation bilinéaire, bicubique, ou d'autres techniques avancées, en fonction des besoins de l'application.
- Optimisation de la vitesse de traitement : Il est important de noter que le redressement de l'image peut causer des opérations coûteuses en termes de temps de traitement. Par conséquent, l'optimisation de ces opérations est cruciale pour garantir des performances rapides de l'application. Des techniques telles que la mise en cache, la parallélisation, et l'utilisation de bibliothèques d'optimisation matérielle peuvent être envisagées pour améliorer l'efficacité.
- Tests et ajustements : Comme l'angle de rotation initial peut varier

en fonction des images d'entrée, il est important de mettre en place des tests de qualité pour évaluer l'efficacité de la rotation et apporter des ajustements si nécessaire. Des mécanismes de rétroaction et de contrôle de la précision seront mis en place pour garantir des résultats fiables.

Le redressement de l'image sera effectué en utilisant une transformation affine. Cette transformation affine est déterminée par l'angle de rotation détecté lors d'une rotation automatique ou celui donné lors d'une rotation manuelle. La transformation affine est un modèle mathématique qui peut être utilisé pour effectuer une rotation, une translation, une mise à l'échelle, et une cisaillement de l'image. Dans notre cas, nous nous concentrerons principalement sur la rotation. La transformation affine appliquée à l'image lui permettra d'être pivotée de manière à ce que la grille de sudoku soit parfaitement alignée avec les côtés de l'image.

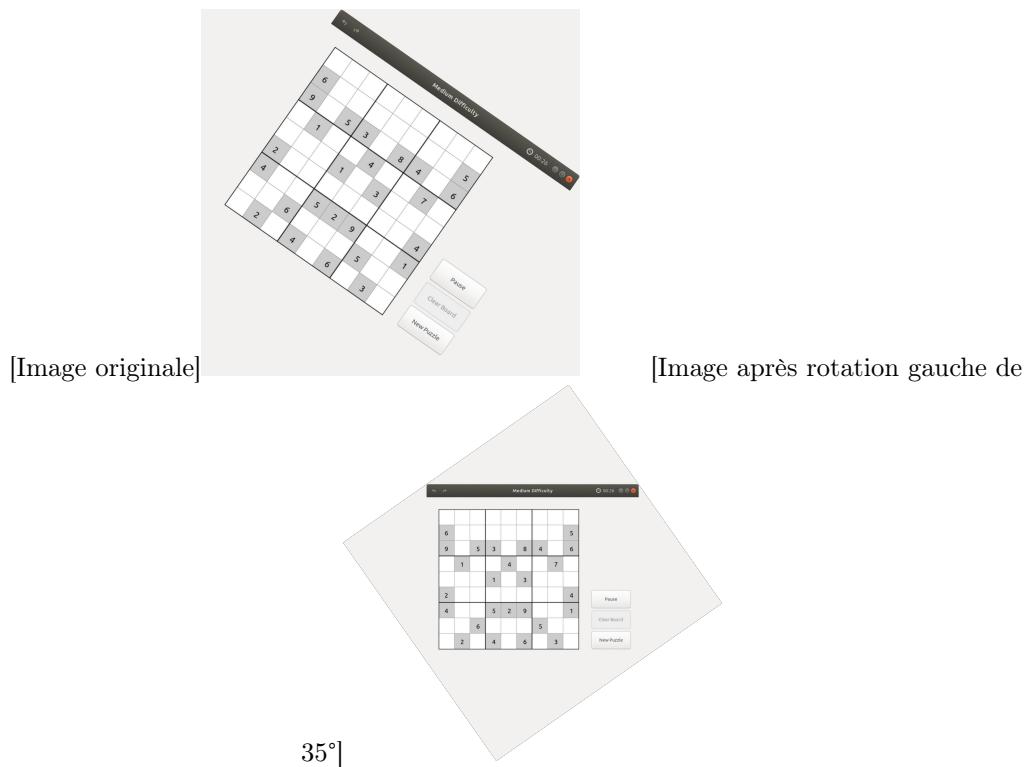


FIGURE 8 – Avant/Après rotation

Rotation Automatique

L’automatisation de la rotation des images constitue un pilier essentiel de notre projet, visant à garantir une orientation optimale pour l’analyse du sudoku. Notre équipe a mis en œuvre une fonction sophistiquée dédiée à la détection automatique de l’angle de rotation nécessaire.

Pour instaurer cette rotation automatique, nous avons opté pour une approche en plusieurs étapes. Tout d’abord, l’image est soumise à un filtre de Canny, une technique de détection d’arêtes, qui constitue la première étape dans l’identification des éléments structuraux clés. Ensuite, une détection préliminaire de la grille est effectuée, permettant d’extraire

les lignes principales de la grille de sudoku.

L'angle de rotation dominant est déterminé en analysant ces lignes, assurant ainsi une orientation cohérente pour l'ensemble de la grille. Cette détection repose sur la fréquence des angles présents dans la grille, avec une attention particulière portée à l'angle le plus prédominant.

Une fois l'angle déterminé, il est converti de radians à degrés, puis inversé pour obtenir une rotation dans le sens trigonométrique. Enfin, la fonction de rotation manuelle est invoquée avec l'angle calculé, garantissant ainsi que l'image est correctement alignée pour les étapes ultérieures du processus de résolution du sudoku.

En somme, la rotation automatique représente une innovation clé dans notre projet, améliorant la précision de l'analyse en garantissant une orientation optimale des images, même dans des conditions initiales variables.

3.2 Détection de la grille (Robin)

Soutenance 1

Pour effectuer la détection des contours de la grille, nous avons choisi d'utiliser l'approche basée sur les gradients, en utilisant le filtre de Roberts. Cette méthode est relativement simple à mettre en place, bien qu'elle soit moins efficace pour gérer le bruit d'image que d'autres filtres, tels que le filtre de Prewitt avec l'approche Laplacien.

Le choix de détecter les contours dans l'image est motivé par son utilité. En effet, la détection des contours (edge detection) dans une image nous permet d'atteindre plusieurs objectifs : reconnaître les objets présents dans une scène, réaliser une segmentation de l'image, permettant de différencier les différentes zones de l'image et pour finir, extraire des

informations pertinentes.

L'approche gradient est une technique fondamentale dans le domaine de la détection de contours en traitement d'image. Pour comprendre cette méthode, il est essentiel de considérer une image comme une fonction f à deux variables, où $f(x, y)$ représente l'intensité lumineuse d'un pixel à la position (x, y) dans l'image.

| | | | | | | |
|----|----|-----|-----|-----|-----|-----|
| 15 | 20 | 100 | 120 | 130 | 130 | 135 |
| 25 | 20 | 25 | 120 | 120 | 140 | 140 |
| 20 | 15 | 20 | 15 | 130 | 135 | 135 |
| 20 | 20 | 15 | 20 | 150 | 130 | 135 |
| 15 | 35 | 30 | 20 | 100 | 110 | 110 |
| 25 | 30 | 25 | 30 | 100 | 110 | 120 |
| 20 | 25 | 30 | 20 | 25 | 110 | 120 |

FIGURE 9 – Exemple image avec le niveau RGB de chaque pixel

Dans ce contexte, on peut définir deux dérivées partielles, une par rapport à x (les lignes) et une par rapport à y (les colonnes). Ces dérivées partielles permettent de mesurer les variations d'intensité lumineuse le long des axes x et y , révélant ainsi les contours présents dans l'image.

Afin de réaliser la détection de contours à l'aide de l'approche gradient nous avons suivi les étapes suivantes :

1. Calcul des dérivées partielles : À partir de l'image I, on calcule deux images Gh (gradient horizontal) et Gv (gradient vertical). Ces images sont obtenues en filtrant l'image I avec deux masques différents, h et v, qui sont conçus pour détecter les variations d'intensité lumineuse horizontales et verticales, respectivement. Ces masques sont généralement des opérateurs de convolution qui capturent les transitions d'intensité.

| GH | | | | | | | Gv | | | | | | |
|-----------|-----|-----|-----|-----|------|------|-----------|-----|------|------|------|------|-----|
| 5 | 5 | 20 | 0 | 10 | 10 | -135 | 5 | -80 | -95 | -10 | -10 | 5 | 140 |
| -10 | 0 | -10 | 10 | 15 | -5 | -140 | 0 | -10 | -100 | -105 | -10 | -5 | 135 |
| 0 | 0 | 0 | 135 | 0 | 0 | -135 | 5 | 0 | 0 | -110 | 15 | -5 | 135 |
| 15 | 10 | 5 | 80 | -40 | -20 | -135 | -5 | 20 | 10 | -130 | -30 | -25 | 110 |
| 15 | -10 | 0 | 90 | 10 | 10 | -110 | -10 | 0 | 5 | -70 | 0 | 0 | 120 |
| 0 | 0 | -5 | -5 | 0 | 10 | -120 | -10 | 0 | 0 | -90 | -85 | -10 | 120 |
| -20 | -25 | -30 | -20 | -25 | -110 | -120 | -25 | -30 | -20 | -25 | -110 | -120 | 0 |

FIGURE 10 – Exemple GHorizontal et GVertical

2. Combinaison des dérivées : Les images Gh et Gv contiennent respectivement les informations sur les contours horizontaux et verticaux dans l'image. Pour obtenir une mesure globale du gradient en chaque pixel, on calcule la norme du gradient, notée G, en utilisant la formule :

$$G = \sqrt{GH^2 + GV^2}$$

| | | | | | | |
|----|----|-----|-----|-----|-----|-----|
| 7 | 80 | 97 | 10 | 14 | 11 | 194 |
| 10 | 10 | 100 | 105 | 18 | 7 | 194 |
| 5 | 0 | 0 | 174 | 15 | 5 | 191 |
| 16 | 22 | 11 | 153 | 50 | 32 | 174 |
| 18 | 10 | 5 | 114 | 10 | 10 | 163 |
| 10 | 0 | 5 | 90 | 85 | 14 | 170 |
| 32 | 39 | 36 | 32 | 113 | 163 | 120 |

FIGURE 11 – $G(x, y) = \sqrt{Gh(x, y)^2 + Gv(x, y)^2}$

Cette opération combine les informations de gradient horizontal et vertical pour évaluer la force du gradient en chaque pixel.

G après seuillage S=60 (moyenne de G)

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |

FIGURE 12 – application du seuillage sur G

3. Image binaire : L'image résultante G est généralement convertie en une image binaire en appliquant un seuil. Les pixels dont la valeur de G dépasse le seuil sont considérés comme des contours, tandis que les autres pixels sont considérés comme le fond. Cette étape permet de segmenter l'image en zones de contour et de fond, facilitant ainsi l'identification des contours dans l'image.

En combinant ces informations, on peut obtenir une image binaire mettant en évidence les contours de l'image, ce qui s'avère essentiel pour la détection des cases de la grille.

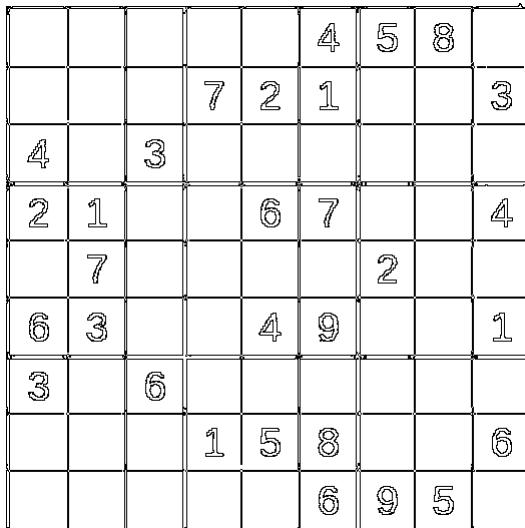


FIGURE 13 – application du seuillage sur G

Soutenance 2

La détection des cases s'est révélée être la partie la plus complexe de mon projet. Au départ, nous avons tenté de créer une fonction qui parcourrait les pixels noirs (représentant les contours) en cherchant à former un carré en explorant les pixels à droite, en bas, à gauche et à droite. Cependant, cette approche fonctionnait bien pour des contours en forme de carrés parfaits, mais échouait lorsque les contours n'étaient pas réguliers.

Nous avons ensuite revu notre stratégie et avons décidé d'aborder une approche utilisant les histogrammes des lignes, ce qui s'est avéré être plus adapté pour détecter les formes géométriques de la grille. Nous utilisons une méthode basée sur des histogrammes pour localiser les bords supérieurs, inférieurs, gauches et droits de la grille d'un sudoku dans une image. Tout d'abord, nous initialisons des histogrammes pour les

bords horizontaux et verticaux, respectivement. Ces histogrammes sont utilisés pour accumuler le nombre de pixels noirs (rouge == 0) le long de certaines colonnes ou lignes de l'image.

En procédant à la recherche des pixels noirs à intervalles réguliers le long de ces colonnes ou lignes, nous construisons des représentations des bords de la grille. Chaque point de l'histogramme enregistre le nombre de pixels noirs rencontrés à cet endroit spécifique. Après avoir parcouru l'image de cette manière, nous identifions le point le plus saillant dans chaque histogramme, indiquant l'emplacement des coins supérieur, inférieur, gauche et droit de la grille.

En procédant à la recherche des pixels noirs à intervalles réguliers le long de ces colonnes ou lignes, nous construisons des représentations des bords de la grille. Chaque point de l'histogramme enregistre le nombre de pixels noirs rencontrés à cet endroit spécifique. Après avoir parcouru l'image de cette manière, nous identifions le point le plus saillant dans chaque histogramme, indiquant l'emplacement des coins supérieur, inférieur, gauche et droit de la grille.

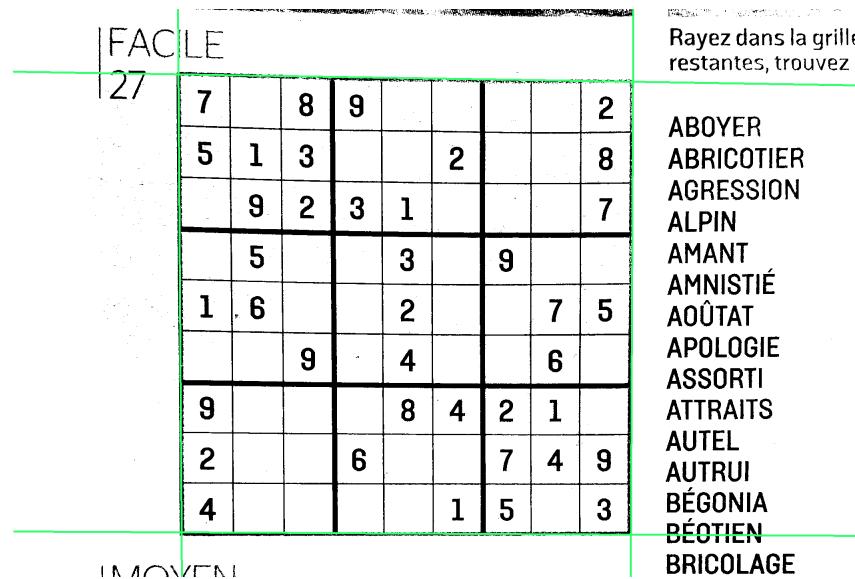


FIGURE 14 – ligne trouver après filtrage des histogrammes

Il est important de noter que bien que notre approche ne repose pas sur l’algorithme de Harris, elle se révèle efficace pour notre objectif spécifique de détecter la structure de la grille dans le contexte d’un sudoku. En ajustant les paramètres et les intervalles de recherche, nous avons pu obtenir des résultats robustes pour la localisation précise des coins de la grille.

3.2 Découpage de l'image (Robin)

3.2 Soutenance 2

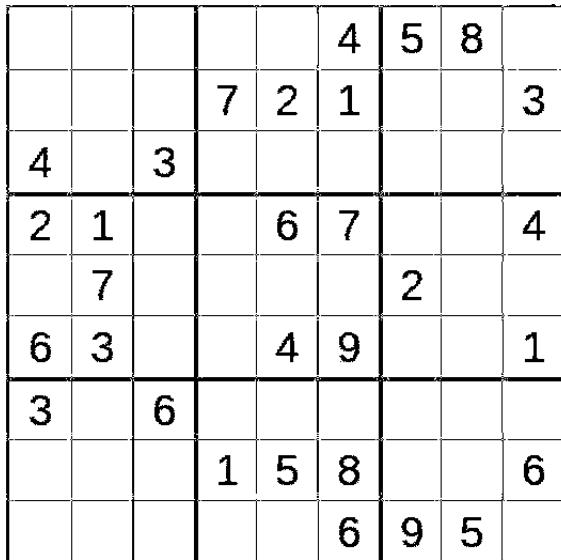


FIGURE 15 – Découpage de l'image selon les coins trouvés

Nous avons dorénavant une image carré avec la grille de sudoku remplissant entièrement l'image. La première étape du processus de découpage consiste à redimensionner l'image d'origine à l'aide de la fonction 'ResizeImage'. Cette fonction crée une nouvelle surface avec une taille spécifiée (900 pixels) et utilise 'SDL BlitScaled' pour effectuer la mise à l'échelle de l'image d'origine vers la nouvelle surface.

Le redimensionnement permet de normaliser la taille de l'image d'entrée. Les images provenant de différentes sources peuvent avoir des résolutions variées, et en redimensionnant l'image à une taille spécifiée (dans ce cas, 900 pixels), nous assurons une cohérence dans le traitement ultérieur. Cela simplifie également le processus d'analyse, car toutes les images traitées auront la même taille, facilitant ainsi la manipulation et la comparaison des données.

La fonction 'RemoveBorderLines' prend ensuite en charge la suppression des lignes de bord des cellules dans le tableau 2D. Elle marque récursivement les cellules connectées aux bords comme visitées, garantissant ainsi l'élimination des lignes de bord indésirables.

La fonction 'CreateSurfaceFrom2DArray' transforme ensuite le tableau binaire en une nouvelle surface SDL. Les pixels de la surface sont définis en noir ou blanc en fonction des valeurs du tableau binaire.

Enfin, la fonction 'SplitImageIntoCells' utilise l'image traitée, la redimensionne, supprime les lignes de bord, puis divise l'image en 81 cellules individuelles. Chaque cellule est sauvegardée en tant qu'image PNG distincte dans le répertoire "cells". Cette fonctionnalité parcourt les 81 cellules de la grille, crée des surfaces pour chaque cellule, et sauvegarde les cellules individuelles en tant que fichiers PNG numérotés.

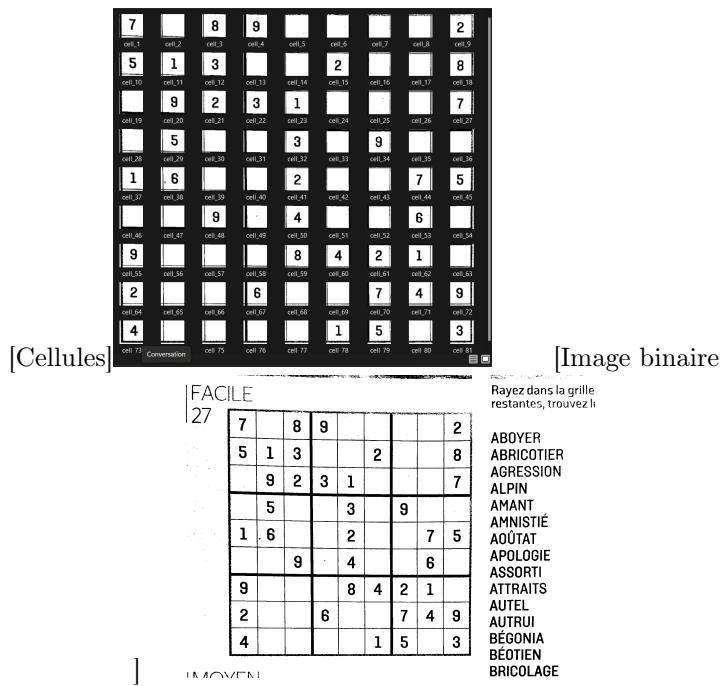


FIGURE 16 – Découpage de la grille de Sudoku en 81 cellules individuelles

3.3 Interface utilisateur (Thomas)

3.3 Site Web

Le site web a plusieurs utilité dans ce projet, dans un premier temps nous arrivons sur une page d'accueil présentant notre projet.

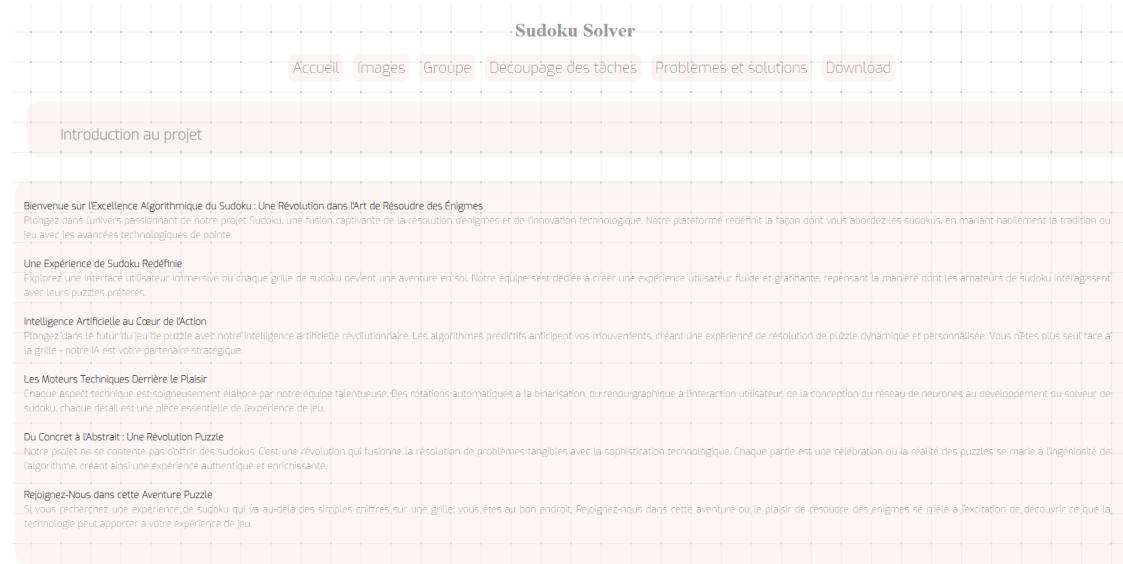


FIGURE 17 – Accueil

Suite à cela, l'utilisateur a le choix entre différentes séctions : Images, Groupe, Découpage des tâches, Download.

Dans le dossier image nous pouvons retrouver une panoplie de différentes images, qui représente différentes parties du projet, afin d'illustrer dans au maximum notre projet.

Dans la partie Decoupage des tâches, nous retrouvons le découpage des tâches

L'onglet Problèmes et solutions représente explique les différents problèmes que nous aurions pu rencontrer dans ce projet

Enfin le dossier download, permet tout simplement de pouvoir télécharger notre fameux Solver de Sudoku

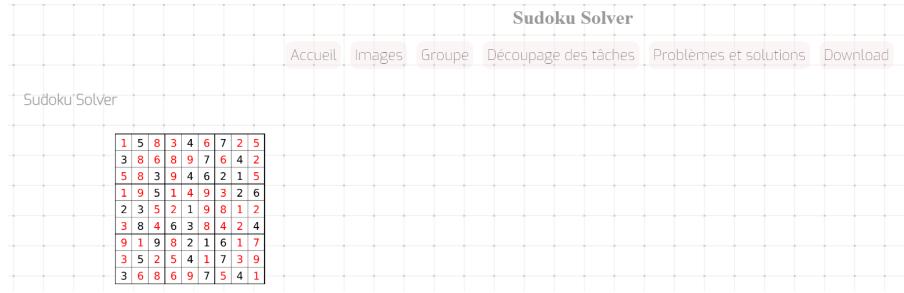


FIGURE 18 – Download

3.3 Application

Dans un premier temps, j'ai entrepris de développer une application en utilisant la bibliothèque SDL. Mes premières approches consistaient à générer les chiffres du Sudoku en les dessinant pixel par pixel. Cependant, au cours de cette phase, j'ai rencontré des problèmes de conflits de pixels, entraînant une apparence peu lisible avec des cases totalement grisées. Pour remédier à cela, j'ai opté pour une approche plus claire en dessinant chaque chiffre individuellement, visant ainsi à obtenir le meilleur rendu possible.

Par la suite, j'ai mis en œuvre une simple itération à travers la matrice du Sudoku. Pour chaque chiffre rencontré, j'ai appelé la fonction correspondante chargée de dessiner ce chiffre spécifique. Un aspect crucial était d'effectuer une vérification pour déterminer si le chiffre était déjà présent dans la grille initiale du Sudoku. Dans le cas où le chiffre était nouveau, je veillais à l'afficher avec une couleur distincte pour différencier les chiffres originaux des nouveaux.

Malheureusement, malgré mes efforts, cette approche n'a pas abouti comme prévu. Par conséquent, j'ai décidé de migrer vers la bibliothèque GTK, ce qui a considérablement simplifié le processus de développement et a permis d'obtenir des résultats plus satisfaisants. Avec GTK, j'ai pu créer une interface graphique fluide et fonctionnelle pour mon application Sudoku, marquant ainsi une étape importante dans le développement du projet.

Lorsque l'application est ouverte, plusieurs choix s'offrent à l'utilisateur. Il peut soit résoudre le Sudoku en utilisant le bouton "Solve", soit opter pour une approche plus détaillée en visualisant chaque étape du processus de résolution.

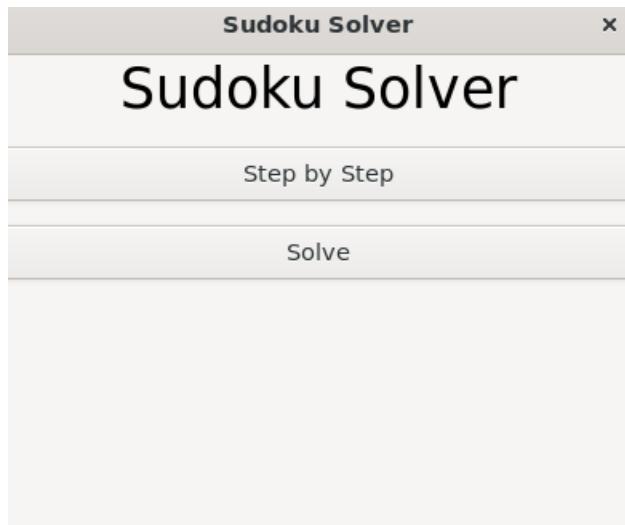


FIGURE 19 – Présentations des différents choix

Si l'utilisateur choisit l'option "Step by Step", une nouvelle fenêtre s'affiche, offrant une vue détaillée de chaque étape du processus de résolution.

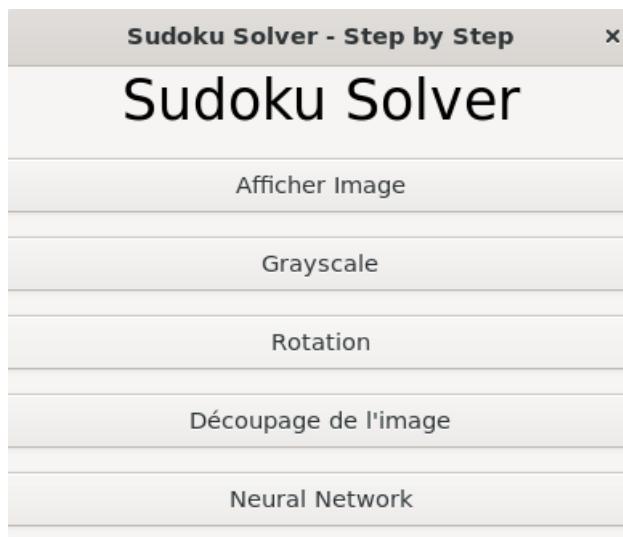


FIGURE 20 – Step by Step

Dans cette fenêtre, toutes les différentes étapes nécessaires pour obtenir un Sudoku résolu sont présentées. Chaque bouton conduit à un exemple illustrant en détail chaque implémentation, permettant ainsi une compréhension approfondie de chaque étape du processus.

Revenons à la fenêtre principale, celle qui propose les deux options "Solve" et "Step by Step". En se concentrant sur le bouton "Solve", c'est à ce moment que la magie opère.

Ma contribution majeure a été de créer une interconnexion harmonieuse entre chaque étape du processus de résolution. J'ai veillé à ce que chaque partie communique efficacement avec les autres, garantissant ainsi une coordination sans accroc. Chaque composant du processus interagit de manière fluide, et chaque étape renvoie le résultat attendu sans compromis. Cette synchronisation minutieuse permet à l'utilisateur de résoudre le Sudoku de manière transparente, avec la certitude que chaque étape contribue de manière cohérente à l'ensemble du processus de résolution.

Après avoir sélectionné l'option "Solve", l'interface ouvre un navigateur de dossiers, permettant à l'utilisateur de choisir un fichier contenant une image du Sudoku à résoudre. Ce choix a été fait pour offrir une flexibilité à l'utilisateur, lui permettant de travailler avec différentes images de Sudokus provenant de diverses sources.

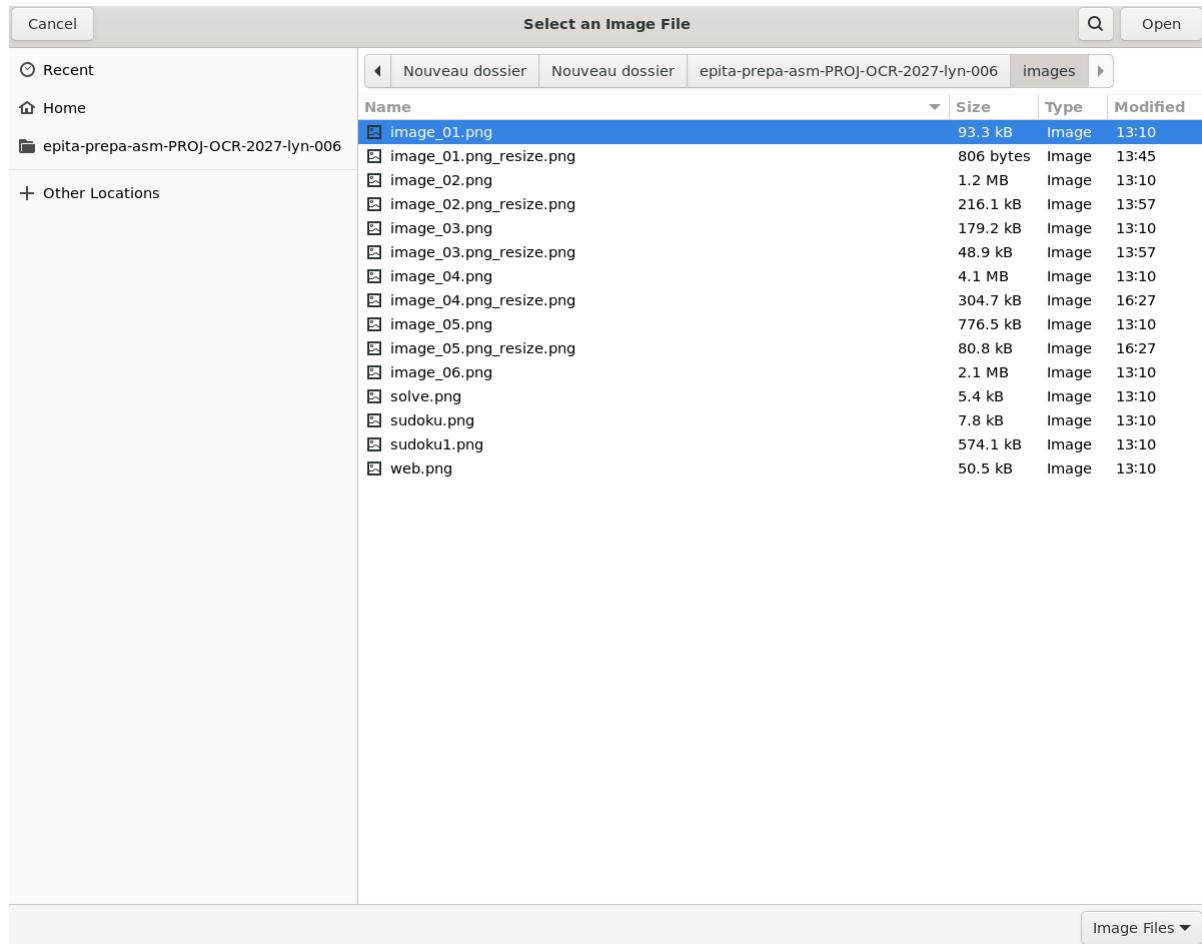


FIGURE 21 – Navigateur de Dossier

Une fois l'image du Sudoku sélectionnée, le programme commence par effectuer un traitement d'image. Ce traitement peut comprendre plusieurs étapes, telles que la détection des contours, la segmentation des cellules du Sudoku, et la conversion des chiffres écrits à la main en données numériques exploitablees.

Ensuite, l'image prétraitée du Sudoku est soumise au réseau de neurones, qui a été entraîné préalablement pour reconnaître les chiffres. Chaque chiffre est identifié individuellement, et le réseau de neurones attribue des probabilités à chaque identification. Ces probabilités peuvent être interprétées comme la confiance du réseau dans la justesse de sa détection.

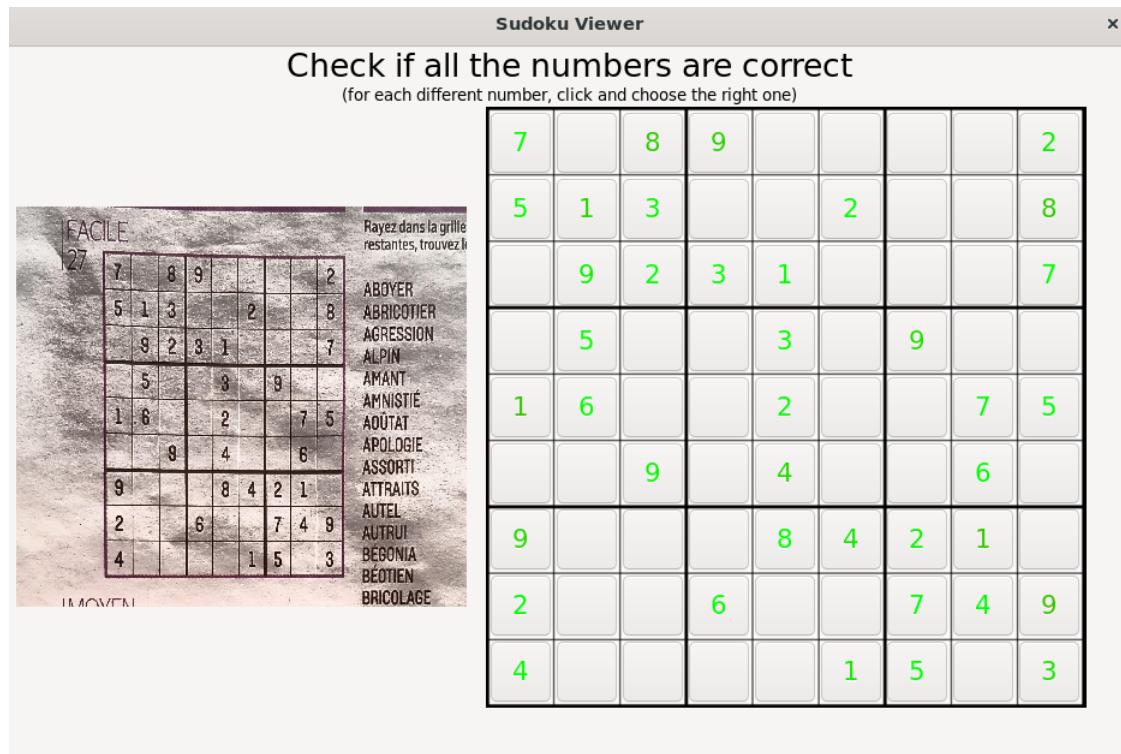


FIGURE 22 – Correction

Une nouvelle fenêtre s'ouvre pour permettre à l'utilisateur de vérifier les résultats de la détection. Chaque case du Sudoku est affichée avec le chiffre identifié par le réseau de neurones et la probabilité associée. Les chiffres avec une confiance modérée apparaissent en nuances de rouge, tandis que ceux avec une détection plus sûre sont affichés en vert.

Si le réseau de neurones fait une erreur de détection, l'utilisateur peut interagir avec l'interface en sélectionnant la case incorrecte et en la corrigeant manuellement. Cette étape interactive offre à l'utilisateur un contrôle direct sur le processus de résolution et garantit l'exactitude du résultat final.

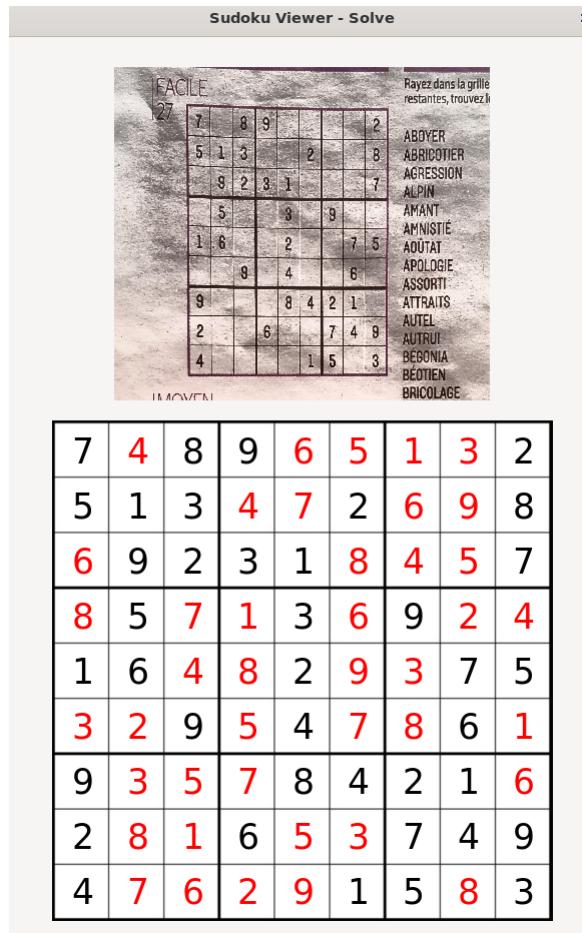


FIGURE 23 – Résultat

Une fois que tous les chiffres ont été validés et que l'utilisateur est satisfait de la détection, il peut fermer la fenêtre. À ce stade, l'interface affiche le Sudoku résolu, représentant le résultat final du processus de résolution.

Cette approche combinant traitement d'image, reconnaissance par réseau de neurones, et interaction utilisateur offre une solution robuste pour résoudre des Sudokus à partir d'images, tout en assurant une transparence et un contrôle sur le processus.

4 Problèmes rencontrés

4.1 Réseau de neurones

Durant la création du réseau de neurones, de nombreuses erreurs ont été rencontrées. En effet, le réseau de neurones utilisé durant la première soutenance ne suffisait pas et n'était pas du tout modulable. C'est pour cela que nous avons recréé à partir de 0 le code du réseau de neurones et cela fut une tâche de longue haleine, mais qui nous a permis de rendre le réseau de neurones entièrement modulable. En effet, désormais, nous pouvons créer un réseau de neurones avec autant de couches que désiré, une learning rate maîtrisé, une fonction d'activation que nous choisissons et nous avons même le contrôle sur le nombre de neurones par couches.

Une fois le code du réseau fait nous pensions que le plus dur était derrière nous. C'était une erreur... En effet, nous avons pris énormément de temps à bidouiller les paramètres. Nous avons dû faire des centaines de tests à chaque fois en changeant le nombre d'epochs, la learning rate ou même l'architecture du réseau de neurones. Malgré cela, nous n'étions pas satisfaits des résultats. C'est pour cela que nous avons testé de nombreux datasets. Nous avons commencé par le dataset MNIST qui est un dataset de plusieurs dizaines de milliers de chiffres écrits à la main. Malheureusement les résultats avec ce dataset ne nous convenait pas nous avons donc utilisé un dataset dédié à la reconnaissance de chiffre dans un sudoku que nous avons trouvé sur Kaggle.



FIGURE 24 – Examples of MNIST digits

La dernière problématique à laquelle nous avons fait face était l’optimisation. En effet, le réseau de neurones que l’on utilisait était trop gros et prenait donc trop de temps à être entraîné. C’est pour cela que nous avons passé la dernière semaine à essayer de minimiser le réseau de neurones sans pour autant perdre en performance afin de pouvoir faire un exemple d’entraînement du réseau de neurones pendant la soutenance.

4.2 Suppression des couleurs

Suppression des Couleurs : Lorsque nous avons entrepris la suppression des couleurs d’une image, nous avons été confrontés à trois défis majeurs, notamment dans le processus d’augmentation des contrastes.

- Gestion des Contrastes :

- 1 Augmentation des Contrastes : Le prétraitement des images a commencé par une augmentation du contraste, une étape cruciale pour optimiser la qualité de l'analyse. Initialement, notre approche consistait à supprimer la couleur la plus présente dans l'image et la remplacer par du blanc. Cependant, nous avons rapidement rencontré un obstacle significatif. Dans la plupart des images, la couleur prédominante était le blanc lui-même, rendant notre méthode inefficace. Pour remédier à cela, nous avons ajusté notre stratégie en éliminant la couleur la plus présente uniquement si elle n'était pas le blanc. Cependant, ce changement a créé un nouveau problème avec les nuances faibles de blanc-gris.
- 2 Gestion des Nuances de Blanc-Gris : Pour résoudre le dilemme des nuances faibles, nous avons pris la décision de supprimer toutes les couleurs les plus présentes qui n'étaient pas déjà des nuances de gris. Cette approche a montré des résultats prometteurs pour les images avec une couleur principale autre que le gris. Cependant, elle a généré des résultats médiocres pour d'autres images.
- 3 Choix de la Suppression des Deux Couleurs les Plus Présentes : Après plusieurs ajustements, nous avons finalement adopté une solution plus robuste en supprimant les deux couleurs les plus présentes. Cette approche a produit des résultats satisfaisants, offrant un prétraitement efficace tant pour les images principalement en nuances de gris que pour celles avec une couleur domi-

nante. Cette stratégie a permis de surmonter les défis antérieurs en assurant une flexibilité dans le traitement des images, renforçant ainsi la robustesse globale de notre solution de prétraitement.

- Méthode de Binarisation et Gestion des Listes d'Occurrences :

Notre approche initiale pour régler le seuil de binarisation consistait à utiliser une liste d'occurrences classique, comptant le nombre d'occurrences de chaque couleur de pixel. Bien que cette méthode ait démontré son efficacité en termes de qualité de traitement, elle présentait des inconvénients majeurs en termes d'efficacité temporelle, surtout lors du traitement d'images de grande taille.

1 Liste d'Occurrences Adaptative : L'idée de départ était d'ajuster dynamiquement la taille de la liste en fonction des variations de couleurs rencontrées au cours de la recherche du seuil de binarisation. Cela s'est avéré efficace pour améliorer la qualité du traitement, mais le temps nécessaire pour ajuster la taille de la liste de manière adaptative était prohibitif, en particulier lors du traitement d'images de grande taille. Cette inefficacité constituait un obstacle majeur dans notre quête de performances optimales.

2 Transition vers une Liste de Taille Fixe : Conscients de la nécessité d'optimiser notre méthode, nous avons pris la décision stratégique de passer à une liste de taille fixe comprenant 256 éléments. Cette transition a considérablement amélioré l'efficacité temporelle du processus de binarisation, tout en garantissant des résultats de traitement satisfaisants. En optant pour une liste de taille fixe, nous avons pu résoudre le problème de l'adaptation constante de la liste, permettant ainsi un traitement plus rapide et une gestion plus efficace des images, même celles de grande taille.

- Réglage du Pourcentage de Binarisation :

Le réglage du pourcentage de seuil dans le processus de binarisation a été une étape cruciale pour obtenir des résultats optimaux. Initialement, nous avons expérimenté avec un seuil fixe de 20%, qui a produit des résultats satisfaisants pour certaines images, mais n'était pas universellement efficace. Cette constatation nous a conduit à entreprendre des tests approfondis en ajustant le pourcentage pour chaque image individuellement, à la recherche d'une valeur de seuil adaptée.

- 1 Tests Itératifs pour le Pourcentage Optimal : Un processus itératif a été mis en place, impliquant des essais sur l'algorithme avec différents pourcentages de seuil. Bien que le seuil initial de 20% ait bien fonctionné pour certaines images, il ne convenait pas à toutes. Nous avons donc effectué des tests sur l'ensemble des images, modifiant le pourcentage de seuil pour trouver la meilleure valeur pour chaque case.
- 2 Choix du Seuil de 10% : Après des tests approfondis, nous avons retenu un seuil de 10% comme étant une valeur raisonnable pour la majorité des images. Cependant, ce seuil fixe ne garantissait pas une qualité optimale pour toutes les images, notamment en ce qui concerne la détection de la grille.
- 3 Adaptation Dynamique du Pourcentage : Dans la version finale de l'algorithme, nous avons introduit une fonction d'adaptation dynamique du pourcentage. Lors du premier parcours de l'image, nous évaluons le niveau de contraste global et la puissance de gris. Ces informations nous permettent d'ajuster dynamiquement le pourcentage de seuil, avec une variation maximale de 0.05%. Cette approche a considérablement amélioré la flexibilité de l'algorithme, assurant une binarisation efficace et cohérente pour une

variété d'images, quelle que soit leur complexité.

4.3 Détection de la grille

L'utilisation du filtre de Roberts s'est révélée être une étape cruciale dans notre processus de détection des contours pour localiser la grille d'un sudoku dans une image. Cependant, nous avons rencontré un défi significatif lié à la nature carrée du filtre, qui fonctionne idéalement sur des images carrées. Pour surmonter cette limitation, nous avons pris la décision stratégique d'effectuer un zoom sur l'image, transformant ainsi sa géométrie pour la rendre carrée.

Ce choix était essentiel pour garantir que le filtre de Roberts puisse être appliqué de manière efficace, assurant ainsi la précision de la détection des contours même sur des images non carrées. Ce processus de zoom a permis d'adapter l'image de manière optimale au fonctionnement du filtre, assurant ainsi une représentation correcte des variations d'intensité nécessaires à la détection des contours.

Par la suite, lors de l'application de la fonction de détection des histogrammes pour identifier les lignes, nous avons rencontré un défi supplémentaire. Les histogrammes générés ont inclus des lignes qui n'appartenaient pas nécessairement à la grille du sudoku, introduisant ainsi des interférences indésirables.

Pour résoudre ce problème, nous avons mis en place une approche consistant à calculer la distance entre les lignes détectées. En évaluant la distance entre chaque ligne détectée, nous avons pu distinguer les lignes appartenant au sudoku de celles qui ne le faisaient pas. Afin de simplifier le processus, nous avons choisi de conserver uniquement les dix lignes

avec des distances équivalentes entre elles.

Cette stratégie a permis de filtrer les lignes indésirables et de ne conserver que celles qui étaient véritablement liées à la structure de la grille du sudoku.

5 Conclusion

En conclusion, notre groupe a accompli avec brio les premières étapes de notre projet ambitieux visant à résoudre des grilles de sudoku par le biais de la technologie OCR. Lors de la formation de notre équipe, nous avons rapidement instauré une dynamique solide, prête à affronter les défis techniques inhérents à cette entreprise stimulante. À la différence de l'année précédente, où nous avions davantage de liberté dans le choix de nos projets, cette année a représenté une opportunité unique de plonger dans des domaines novateurs, notamment la reconnaissance de chiffres à travers l'utilisation de réseaux de neurones et la détection précise de grilles sur des images. Chacun des membres de notre équipe a assumé des responsabilités clairement définies, allant de la conception d'un réseau de neurones spécialisé dans la reconnaissance des chiffres, à la résolution effective du sudoku. Les avancées significatives que nous avons accomplies jusqu'à présent dans la mise en œuvre de ces composantes confirment notre progression vers la création d'une application complète, capable de résoudre des grilles de sudoku à partir d'images. Notre collaboration étroite, fondée sur le partage continu de connaissances et d'expertise, s'est avérée cruciale pour notre succès. À l'approche de la deuxième soutenance, nous anticipons avec confiance l'achèvement de nos réseaux

de neurones et la finalisation de l'application. Ces développements permettront aux utilisateurs de résoudre des sudokus de manière fluide et intuitive. Animés par notre enthousiasme, nous sommes impatients de poursuivre ce parcours stimulant et de présenter, lors de la dernière soutenance, un produit exceptionnel attestant de notre engagement envers l'excellence et l'innovation.