

Second Rapport

MGJV

December 2023

Contents

1	Le groupe	4
1.1	Victor AGAHI	4
1.2	Mehdi AZOUZ	4
1.3	Jules MAGNAN	4
1.4	Gaspard TORTERAT SLANDA	5
2	Ressources humaines	5
3	Prétraitement	6
3.1	Filtre grayscale	6
3.2	Flou de Gauss	6
3.3	Noir et Blanc	7
4	Traitement	8
4.1	Canny Edge	8
4.1.1	Calcul des gradients	8
4.1.2	Double Treshold	8
4.1.3	Hysteris	9
4.2	Redressement de l'image	9
4.2.1	Rotation de l'image	9
4.2.2	Redressement automatique	9
4.3	Transformée de Hough	10
4.3.1	Accumulateur	10
4.3.2	Maximums locaux	10
5	Découpage de la grille	12
5.1	Flood-fill	12
5.2	Itératif	12
5.3	Extraction	12
5.4	Separation	13
5.4.1	Régularisation	13
5.4.2	Découpage	13

6	Epuration	13
6.1	Colonnes	13
6.2	Reconnaissance du chiffre	14
6.2.1	Traitement de la case	14
6.2.2	Detection	14
6.2.3	Utilité autre	14
6.3	D'image à tableau	15
6.4	Preparation de la grille	15
7	Résolution du sudoku	16
7.1	Lecture de fichier	16
7.2	Solvabilité	16
7.3	Résolution	16
7.4	Ecriture	17
7.5	Technique	17
7.6	Affichage	17
8	Réseau de neurones	18
8.1	XOR	19
8.1.1	Propagation avant	19
8.1.2	Rétropopagation	19
8.1.3	Présentation	19
8.1.4	Entraînement	20
8.1.5	Affichage	20
8.2	OCR	21
8.2.1	Propagation avant	21
8.2.2	Rétropopagation	21
8.2.3	Présentation	21
8.2.4	Neurones d'entrée	23
8.2.5	Couche caché	24
8.2.6	Neurones de sortie	25
8.2.7	Entraînement	26
8.2.8	Base de donnée	27
8.2.9	Affichage	28
9	Application	29
9.1	Introduction	29
9.2	Expérience utilisateur intuitive	29
9.3	Résolution automatisée de sudokus	29
9.4	Fonctionnalités avancées	29
9.5	Manipulation des grilles de sudoku	29
9.6	Design visuel cohérent	29
10	Technologies Utilisées	30
10.1	Utilisation de GTK	30
10.2	Contrôle total sur la conception	30
10.3	Personnalisation avancée	30
10.4	Adaptation aux besoins spécifiques du projet	30
10.5	Contrôle de la performance	31

11 Outils Utilisés	31
11.1 GTK (GIMP Toolkit)	31
11.1.1 Widgets Riches	31
11.1.2 Événements et Signaux	31
11.1.3 Personnalisation Visuelle	31
11.2 GTK Grid et GTK Box	32
11.2.1 GTK Grid	32
11.2.2 GTK Box	32
12 Interface Graphique	32
12.1 Boutons et Fonctionnalités	32
12.1.1 Noir et Blanc	32
12.1.2 Hough	32
12.1.3 Rotation	33
12.1.4 Grid Display	33
12.1.5 Load Image	33
12.1.6 Back	33
12.1.7 Draw	33
12.1.8 Solve	33
13 Convivialité et Accessibilité	34
13.1 Convivialité	34
13.1.1 Contrôles Intuitifs	34
13.1.2 Messages d'Erreur Informatifs	34
13.1.3 Structure Visuelle Cohérente	34
13.2 Accessibilité	34
13.2.1 Contraste et Taille du Texte	34
13.2.2 Navigation au Clavier	34
13.2.3 Compatibilité avec les Outils d'Assistance	35
14 CSS dans le Design	35
14.1 Personnalisation Visuelle	35
14.1.1 Contrôle des Propriétés d'Apparence	35
14.1.2 Thèmes Visuels	35
14.2 Maintien de la Cohérence	35
14.2.1 Réutilisabilité du Code	35
14.2.2 Facilitation des Modifications	35
14.3 Responsivité de l'Interface	36
14.3.1 Adaptabilité à Différentes Tailles d'Écran	36
14.3.2 Media Queries	36
15 Conclusion	36
15.1 Minutie dans la Conception	36
15.2 Puissance et Flexibilité	36
15.3 Expérience Utilisateur Enrichissante	36

1 Le groupe

1.1 Victor AGAHI

Bonjour, je suis Victor Agahi, étudiant à l'EPITA. Attiré par l'univers de l'informatique dès mes premières années, je considère que notre projet actuel constitue un défi stimulant. Il nous offre l'opportunité d'approfondir nos compétences techniques en langage C, tout en nous permettant d'explorer de nouvelles thématiques passionnantes, notamment dans le domaine de l'intelligence artificielle. Ce projet, axé sur un logiciel OCR, renforce ma conviction quant à l'importance de l'informatique dans notre société, et j'aborde cette initiative avec sérieux et je suis prêt à relever les défis techniques en équipe.



Figure 1: Victor Agahi

1.2 Mehdi AZOUZ

Bonjour, je m'appelle Mehdi, étudiant en deuxième année à EPITA. Passionné d'informatique depuis mon plus jeune âge, je trouve ce projet très intéressant car il nous permet d'apprendre de nouvelles thématiques telles que l'intelligence artificielle.

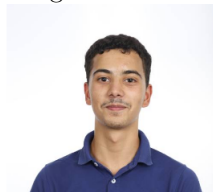


Figure 2: Mehdi AZOUZ

1.3 Jules MAGNAN

Bonjour, étudiant à l'Epita je me passionne pour les défis informatiques. Ce projet dans lequel nous nous lançons est très formateur il nous pousse à mener des défis techniques en C et nous forme à créer des projets en équipe.



Figure 3: Jules MAGNAN

1.4 Gaspard TORTERAT SLANDA

Bonjour, je suis Gaspard, un étudiant d'EPITA. Je suis convaincu que l'informatique est un domaine important de notre société qui regorge de défi. C'est avec sérieux et minutie que je compte réaliser ce projet centré sur un logiciel OCR.



Figure 4: Gaspard TORTERAT SLANDA

2 Ressources humaines

Ci-dessous la répartition des tâches pour la première période de travail.

Membre \ Tâche	Gaspard	Jules	Mehdi	Victor
Traitement de l'image				
Prétraitement	Principal			
Rotation	Principal			
Détection de la grille	Principal			
Découpage de la grille		Principal		
Traitement des cases		Principal		
Transforme de Hough				Principal
OCR				
Réseau de neurones		Second	Principal	
Banque d'images		Second	Principal	
Autre				
Sudoku		Principal		
UI				Principal

Table 1: Répartition des tâches

3 Prétraitement

Le prétraitement est l'étape qui doit nettoyer les impuretés de l'image pour que le réseau de neurones puisse performer au mieux.

$$\int_1^2 \sum_{\alpha}^8 1$$

3.1 Filtre grayscale

Une image se découpe en pixels. Un pixel se décompose en trois composantes : le rouge, le vert et le bleu. Le but du filtre grayscale est de diminuer la quantité d'information. Nous appliquons la formule suivante sur chaque pixel de l'image :

$$\forall (x, y) \in \llbracket 0, height \rrbracket \times \llbracket 0, width \rrbracket$$

$$IMAGE(x, y) = 0.3 * R + 0.59 * V + 0.11 * B$$

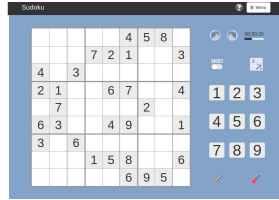


Figure 5: Avant filtre grayscale

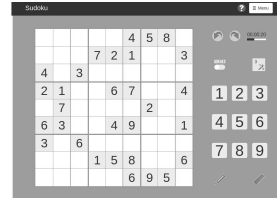


Figure 6: Après filtre grayscale

3.2 Flou de Gauss

Une image peut être bruitée. Le bruit se traduit par l'apparition de pixels noirs ou de fort contrastes dans certaines zones de l'image. Pour pallier à ce premier problème, bien que peu courant en pratique, le filtre médian est idéal.

Pour le second, nous utilisons un flou de Gauss avec un kernel de 5x5. Cette étape uniformise la distribution des valeurs des pixels. Elle vise à améliorer les chances de détection des contours de la grille. Ainsi nous appliquons la convolution suivante à chaque pixel de l'image :

$$\forall (x, y) \in \llbracket 0, height \rrbracket \times \llbracket 0, width \rrbracket$$

$$\text{Soit } P(x, y) = IMAGE(x, y)$$

$$\text{Soit } M(x, y) = \begin{pmatrix} P(x-2, y-2) & P(x-2, y-1) & P(x-2, y) & P(x-2, y+1) & P(x-2, y+2) \\ P(x-1, y-2) & P(x-1, y-1) & P(x-1, y) & P(x-1, y+1) & P(x-1, y+2) \\ P(x, y-2) & P(x, y-1) & P(x, y) & P(x, y+1) & P(x, y+2) \\ P(x+1, y-2) & P(x+1, y-1) & P(x+1, y) & P(x+1, y+1) & P(x+1, y+2) \\ P(x+2, y-2) & P(x+2, y-1) & P(x+2, y) & P(x+2, y+1) & P(x+2, y+2) \end{pmatrix}$$

$$\text{Alors } IMAGE(x, y) = \frac{1}{273} \times \begin{pmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{pmatrix} * M(x, y)$$

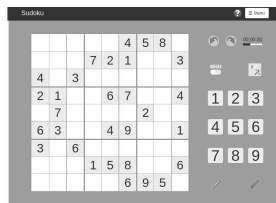


Figure 7: Avant Flou de Gaus

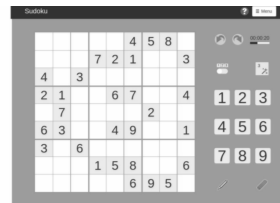


Figure 8: Après flou de Gauss

3.3 Noir et Blanc

Pour convertir l'image en noir et blanc, nous avons utilisé un seuil adaptatif. Il s'agit ici, pour chaque pixel, de calculer la moyenne des valeurs des 441 pixels voisins. On note cette moyenne T . Si la valeur du pixel de l'image aux coordonnées (x, y) est supérieure à T , alors le pixel de la nouvelle image aux coordonnées (x, y) sera blanc. Sinon, il sera noir.

Cela dit, un tel traitement peuvent s'avérer trop long pour s'incorporer dans une application temps réel. Pour parer à ce problème, nous avons utilisé un prétraitement appelé *Image intégrale*.



Figure 9: Avant filtre noir et blanc

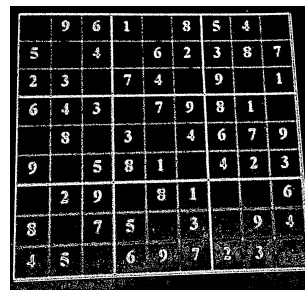


Figure 10: Après filtre noir et blanc

4 Traitement

4.1 Canny Edge

Pour détecter la grille de sudoku, nous devons dans un premier temps extraire les contours de l'image. Pour cela nous utilisons l'algorithme Canny Edge. Son déroulement s'effectue en trois étapes : Un calcul du gradient de chaque pixel, puis une sélection des pixels selon la direction du gradient, et enfin une autre sélection parmi les pixels restants.

4.1.1 Calcul des gradients

Pour la première étape nous utilisons deux informations du gradient : sa norme et sa direction. Nous utilisons pour le calcul du gradient le masque de Sobel, ci-dessous définit.

$$\text{Soient } G_X = \begin{pmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{pmatrix} * M(x, y) \text{ et } G_Y = \begin{pmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} * M(x, y)$$

Alors on définit la magnitude G du gradient comme la norme de \vec{G}

$$G(x, y) = \sqrt{G_X^2 + G_Y^2}$$

De plus on définit la direction du gradient que l'on note Θ par

$$\Theta(x, y) = \arctan\left(\frac{G_Y}{G_X}\right)$$

Ensuite nous regroupons les angles des pixels selon quatre intervalles :

$$\left[0, \frac{\pi}{4}\right[\left[\frac{\pi}{4}, \frac{\pi}{2}\right[\left[\frac{\pi}{2}, \frac{3\pi}{4}\right[\left[\frac{3\pi}{4}, \pi\right[$$

Pour gardons alors les pixels qui sont des maximums locaux selon la direction de leur gradient.

4.1.2 Double Threshold

Cette étape consiste à trier les pixels selon la magnitude de leur gradient. La méthode s'aide de deux seuils arbitraires $T1$ et $T2$ tels que $T1 < T2$.

1. $G(x, y) < T1$
2. $T1 \leq G(x, y) < T2$
3. $T2 \leq G(x, y)$

Les pixels dont la magnitude est inférieure au premier seuil sont supprimés de l'image. Les pixels dont la magnitude est comprise entre le premier seuil et le deuxième seuil sont marqués en tant que pixels faibles, tandis que les autres pixels sont marqués pixels forts.

4.1.3 Hysteris

Enfin, l'algorithme vérifie que pixel faible est voisin immédiat d'un pixel fort.
Si la recherche est négative, le pixel est supprimé de l'image. Sinon, le pixel devient un pixel fort.



Figure 11: Avant Canny Edge

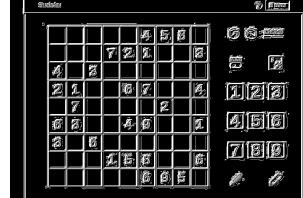


Figure 12: Après Canny Edge

4.2 Redressement de l'image

Conformément au cahier des charges, nous implémentons la rotation de l'image de sudoku. Nous tournons l'image en appliquant une matrice de rotation à deux dimensions.

4.2.1 Rotation de l'image

Nous appliquons les trois matrices de rotation suivantes aux coordonnées de chaque pixel : Soit θ l'angle de rotation et $IMG(x, y)$ un pixel de coordonnées (x, y) .

$$\text{Soient } P = \begin{pmatrix} 0 & -\tan(\frac{\theta}{2}) \\ 1 & 0 \end{pmatrix} \quad Q = \begin{pmatrix} 1 & 0 \\ \sin(\theta) & 1 \end{pmatrix} \quad \text{et } R = \begin{pmatrix} 1 & -\tan(\frac{\theta}{2}) \\ 0 & 1 \end{pmatrix}$$

Alors les nouvelles coordonnées (x', y') du pixel sont données par:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \times P \times Q \times R$$

4.2.2 Redressement automatique

Nous avons implémenté la rotation automatique de l'image. Nous utilisons dans un premier temps la transformée de Hough, puis nous détectons le plus mince angle formé par une droite de l'image et l'axe horizontal.

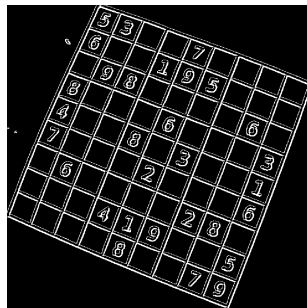


Figure 13: Avant rotation

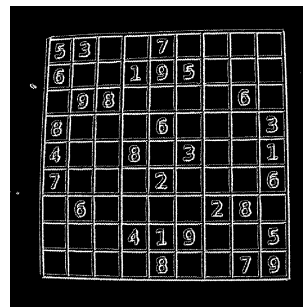


Figure 14: Après rotation

4.3 Transformée de Hough

Pour pouvoir détecter des formes dans une image, nous utilisons un algorithme répandu : la transformée de Hough. Nous centrons cette méthode sur la détection de lignes dans l'image. Elle utilise un accumulateur dans l'espace polaire qui entrepose les droites. Cette méthode s'utilise sur une image traitée au préalable par Canny Edge ou autre un algorithme dont le but est semblable.

4.3.1 Accumulateur

Pour chaque pixel $M(x,y)$, il faut calculer la distance ρ à l'origine, donnée en fonction de l'angle θ par :

$$\forall \theta \in [0, 180] \quad \rho = x * \cos(\theta) + y * \sin(\theta)$$

Ensuite il faut incrémenter la position $[\rho] [\theta]$ de l'accumulateur.

Puis, il est possible d'enregistrer l'accumulateur sous la forme d'une image ci-dessous, en veillant à normaliser les valeurs pour qu'elles se situent entre 0 et 255. Pour visualiser les contours identifiés, nous parcourons notre accumulateur et sélectionnons chaque point de coordonnées (ρ, θ) situé au-dessus d'un seuil préalablement défini. Nous récupérons ensuite les coordonnées polaires et en déduisons les coordonnées cartésiennes (x_0, y_0) à l'aide des formules suivantes :

$$x_0 = \rho \cdot \cos(\theta) \quad \text{et} \quad y_0 = \rho \cdot \sin(\theta)$$

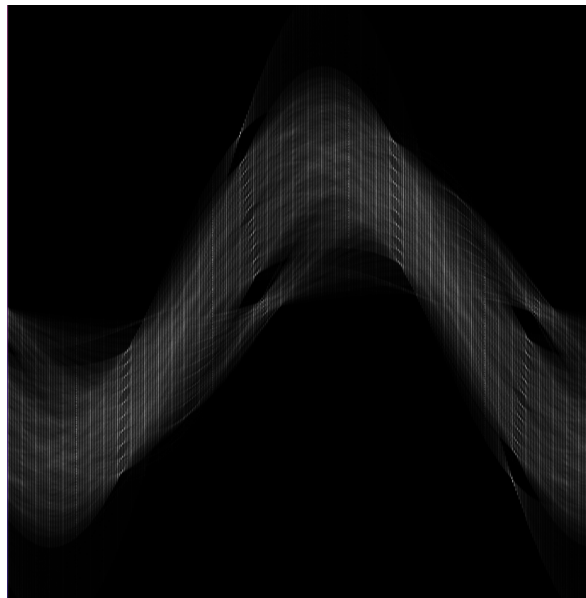


Figure 15: Accumulateur normalisé

4.3.2 Maximums locaux

Dans cette étape, nous récupérons les pics de l'accumulateur et supprimons leurs valeurs voisines dans un cercle d'une taille prédéfinie. Nous sélectionnons ces pics s'ils sont supérieurs à un seuil défini au préalable.

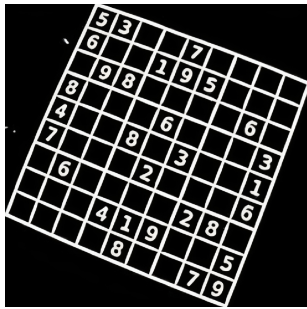


Figure 16: Avant Hough

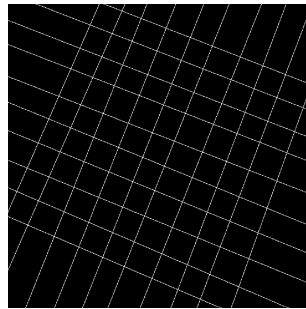


Figure 17: Après Hough

5 Découpage de la grille

Dans cette étape, nous nous intéressons à découper la grille du sudoku. Nous utilisons un algorithme de flood-fill.

5.1 Flood-fill

Pour trouver la grille, nous sommes partis du postulat suivant :

La grille est logiquement l'ensemble de pixels connectés le plus grand dans la grande majorité des images de sudoku.

Nous définissons un pixel "connecté" comme étant un pixel lui-même blanc et dont les quatre voisins cardinaux (Nord/Sud/Est/Ouest) sont blancs. A partir de cette définition et de notre logique précédente, nous utilisons un algorithme de remplissage sur notre image. Ensuite, nous découpons la grille en utilisant la librairie SDL.

5.2 Itératif

Comme certaines images dépassent les 2000 pixels, il est important de laisser de côté la récursion.

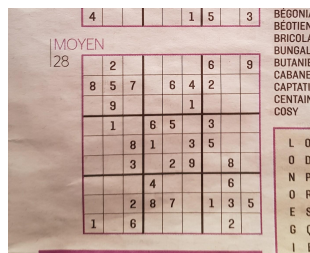


Figure 18: Avant Flood

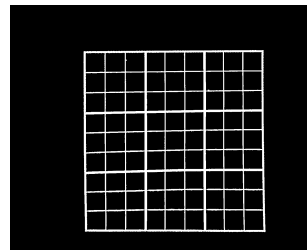


Figure 19: Après Flood

5.3 Extraction

Une fois le flood-fill effectué, nous extrayons la grille du sudoku.



Figure 20: Avant extraction

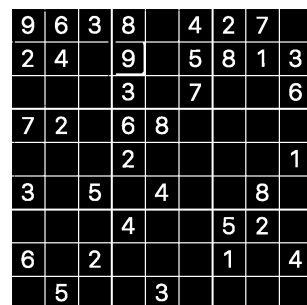


Figure 21: Après extraction

5.4 Separation

5.4.1 Régularisation

La séparation de la grille est un élément essentiel de l'OCR. L'algorithme doit permettre de découper chaque case à la perfection afin de pouvoir ensuite récupérer les chiffres. Lors du prétraitement, il peut arriver que certaines images ne soient pas bien centrées. Dans notre cas, il peut arriver que certaines images soient décalées vers le bas, et donc qu'elles aient une bordure en haut mais pas sur les côtés. C'est pourquoi nous avons implémenté un algorithme de régularisation de la grille.

Cet algorithme fonctionne en deux étapes. Tout d'abord, il vérifie si la grille est décalée dans un sens particulier. En fonction de cette détection, l'algorithme va recentrer la grille vers le milieu. Ensuite, pour obtenir un ajustement parfait, un autre algorithme va vérifier l'écart entre la bordure de la grille par rapport à l'image et réajuster la grille comme un zoom.

Cette approche permet d'assurer une précision maximale lors de la séparation de la grille, garantissant ainsi une meilleure performance dans le processus de résolution du Sudoku. En corrigeant les éventuels décalages et en ajustant la position de la grille, notre logiciel optimise la qualité de l'extraction des chiffres, facilitant ainsi la résolution ultérieure du Sudoku.

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

Figure 22: Avant Regularisation

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

Figure 23: Après Regularisation

5.4.2 Découpage

Maintenant que la grille a été régularisée, il suffit simplement de découper l'image en 81 images correspondant chacune à une case que nous allons épurer par la suite.

6 Epuration

6.1 Colonnes

L'épuration des cases est nécessaire pour qu'elles puissent être lues efficacement par notre réseau de neurones. Pour éliminer les lignes, l'algorithme commence par tracer automatiquement le contour de chaque case de manière à ce que le plus gros morceau restant soit le chiffre.

6.2 Reconnaissance du chiffre

6.2.1 Traitement de la case

Après l'effacement des colonnes, un algorithme se déplace sur l'image pour identifier l'amas de pixels le plus important, qui correspond nécessairement au chiffre, étant donné que nous avons préalablement supprimé les bordures. Une fois cette étape accomplie, nous pouvons redessiner les pixels du chiffre sur une nouvelle image que nous enregistrons.

6.2.2 Detection

Après avoir épuré chaque case, il est maintenant nécessaire de déterminer celles qui contiennent des chiffres pour les envoyer à notre réseau de neurones. Certaines cases présentent cependant des amas de pixels plus ou moins importants, pouvant être des chiffres. De plus, dans les cases qui ne devraient pas contenir de chiffres, des amas de pixels peuvent subsister après le prétraitement. Il est donc essentiel que, sur n'importe quelle image, nous soyons en mesure de prédire si la case contient ou non un chiffre.

Pour ce faire, nous parcourons les images épurées du Sudoku afin de calculer la moyenne de pixels par image. Ensuite, nous considérons que toutes les images ayant un nombre de pixels de chiffres supérieur à la moyenne sont des cases contenant des chiffres. À l'origine, nous avons simplement défini un seuil, de sorte que si une image avait un nombre de pixels de chiffres dépassant un certain seuil, elle était forcément considérée comme contenant un chiffre. Cependant, nous avons constaté que dans certains Sudokus, les chiffres peuvent être écrits très petit, et sur certaines grilles, les amas non chiffrés peuvent être plus importants que ces petits chiffres. C'est pourquoi nous avons plutôt opté pour l'utilisation de la moyenne de pixels de chiffres par case dans le Sudoku.

6.2.3 Utilité autre

Ce système d'épuration présente plusieurs avantages. Tout d'abord, il est capable, dans une très grande majorité des cas, d'extraire le chiffre d'une case, en le centrant vers le milieu grâce aux traitements effectués précédemment. Cela s'avère également avantageux lorsque nous avons tenté de générer notre propre base de données en créant un script complet, comprenant tous les traitements précédents. Nous étions ainsi capables, à partir d'un Sudoku, d'extraire tous les chiffres, puis de les étiqueter, nous permettant ainsi de constituer notre propre base de données. Cette approche a contribué à améliorer le réseau de neurones en lui fournissant une diversité de types de chiffres pour un entraînement plus robuste.

6.3 D'image à tableau

Maintenant que nous sommes en mesure d'obtenir chaque case indépendamment, il est nécessaire de les retranscrire en tableaux de chiffres. Pour ce faire, nous récupérons la matrice de pixels de l'image grâce à SDL, puis nous examinons chaque pixel pour déterminer s'il est noir ou blanc. Ensuite, nous inscrivons la valeur du pixel dans une matrice, préparant ainsi les données pour être lues par le réseau de neurones.

6.4 Preparation de la grille

Pour préparer la grille de sudoku nous créer une liste de int. Nous allons parcourir chaque cases du sudoku, si elle reconnu comme un chiffre alors nous appellons la fonction de prediction du reseau de neuronne pour mettre le chiffre dans le tableau. Si la case n'est pas reconnue comme un chiffre, alors nous mettons un 0 dans le tableau, ce systeme est efficace et permet de préparer la grille pour entre n'avoir plus qu'a l'envoyer dans l'algorithme de resolution.

7 Résolution du sudoku

7.1 Lecture de fichier

Pour garantir que la résolution du Sudoku soit conforme, nous avons mis en place un algorithme de lecture de fichier. Le fichier du Sudoku initial contient les caractères déjà présents dans le Sudoku, représentant les cases vides par des points. Lors de la lecture du fichier, les caractères sont ajoutés à une matrice, et les points sont remplacés par des zéros pour la résolution. Pour simplifier l'utilisation de notre tableau, il sera alloué à l'aide de la fonction 'calloc'.

$$\begin{bmatrix} 8 & 3 & . & . & 7 & . & . & . & . \\ 6 & . & . & 1 & 9 & 5 & . & . & . \\ . & 9 & 8 & . & . & . & . & 6 & . \\ 8 & . & . & . & 6 & . & . & . & 3 \\ 4 & . & . & 8 & . & 3 & . & . & 1 \\ 7 & . & . & . & 2 & . & . & . & 6 \\ . & 6 & . & . & . & . & 2 & 8 & . \\ . & . & . & 4 & 1 & 9 & . & . & 5 \\ . & . & . & . & 8 & . & . & 7 & 9 \end{bmatrix}$$

Figure 24: Exemple de grille non resolution en format original

7.2 Solvabilité

Avant de commencer la résolution du Sudoku, par souci de performance et d'efficacité, il est indispensable de vérifier sa solubilité. Lorsque l'exécution de l'algorithme de résolution est appelée, nous effectuons d'abord la vérification de chaque ligne, colonne et carré de la grille pour garantir que le Sudoku peut être résolu.

7.3 Résolution

Ensuite, l'algorithme de résolution est assez intuitif. À la première occurrence d'un zéro dans le tableau, nous le remplaçons par un 1, puis vérifions si cette substitution entraîne une erreur dans le Sudoku (nous vérifions si le Sudoku reste correct). Si le remplacement ne pose pas de problème, nous passons aux cases suivantes. Sinon, nous remplaçons le 1 par un 2, et ainsi de suite, jusqu'à trouver une combinaison qui fonctionne. Bien entendu, si après avoir atteint 9, la résolution ne fonctionne pas, l'algorithme recommence depuis le début en modifiant la configuration de départ jusqu'à trouver une combinaison fonctionnelle.

7.4 Ecriture

Lorsque le Sudoku est résolu, il ne nous reste plus qu'à réécrire la matrice dans un fichier, dans le même format que le fichier initial, avec l'extension ".result". Ce fichier contient ainsi le résultat final du Sudoku.

$$\begin{bmatrix} 1 & 2 & 7 & 6 & 3 & 4 & 5 & 8 & 9 \\ 5 & 8 & 9 & 7 & 2 & 1 & 6 & 4 & 3 \\ 4 & 6 & 3 & 9 & 8 & 5 & 1 & 2 & 7 \\ 2 & 1 & 8 & 5 & 6 & 7 & 3 & 9 & 4 \\ 9 & 7 & 4 & 8 & 1 & 3 & 2 & 6 & 5 \\ 6 & 3 & 5 & 2 & 4 & 9 & 8 & 7 & 1 \\ 3 & 5 & 6 & 4 & 9 & 2 & 7 & 1 & 8 \\ 7 & 9 & 2 & 1 & 5 & 8 & 4 & 3 & 6 \\ 8 & 4 & 1 & 3 & 7 & 6 & 9 & 5 & 2 \end{bmatrix}$$

Figure 25: Exemple de grille non resolution en format ".result"

7.5 Technique

Pour assurer la conformité de la résolution du Sudoku, le dossier est pourvu d'un Makefile permettant la création d'un exécutable. Ensuite, en utilisant cet exécutable et en passant en paramètre le fichier contenant le Sudoku, le résultat sera généré dans le même dossier avec l'extension ".result".

7.6 Affichage

Pour que l'utilisateur puisse voir le sudoku résolu, nous avons disposé sur une image blanche les chiffres découverts en vert.

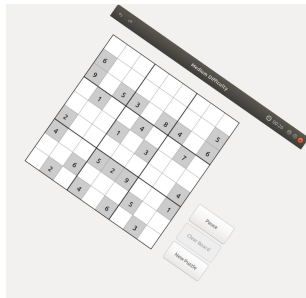


Figure 26: Avant Résolution

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

Figure 27: Affichage

8 Réseau de neurones

Un réseau de neurones est un modèle informatique organisé en plusieurs parties. La première partie, appelée **propagation avant (ou front propagation)**, consiste à faire prédire au réseau de neurones une valeur de sortie en utilisant les données d'entrée. Cette prédiction est basée sur les connexions pondérées entre les neurones et les fonctions d'activation appliquées à ces connexions.

Ensuite, il y a la **rétropropagation (ou backpropagation)**, qui intervient après la prédiction. Son rôle est de corriger d'éventuelles erreurs faites par le réseau. Pour ce faire, elle utilise un algorithme qui ajuste les poids des connexions entre les neurones en fonction de la différence entre la prédiction du réseau et la valeur attendue. Cette rétropropagation permet au réseau de s'améliorer progressivement au fil de l'apprentissage, en minimisant les erreurs entre les prédictions et les valeurs réelles.

8.1 XOR

8.1.1 Propagation avant

Nous avons décidé d'utiliser la fonction sigmoïde pour notre fonction d'activation elle permet de faire prédire à notre réseau de neurones une valeur de sortie.

Fonction sigmoïde: $f(x) = \frac{1}{1 + e^{-x}}$

8.1.2 Rétropopagation

Nous avons décidé d'utiliser la fonction dérivée de sigmoïde afin de corriger les éventuelles erreurs du réseau de neurones. Cette fonction d'apprentissage permet au réseau de neurones de comprendre ses erreurs. Fonction dérivée de sigmoïde: $f'(x) = x \cdot (1 - x)$

8.1.3 Présentation

Notre réseau de neurones est organisé selon la structure suivante : il comprend 2 entrées, chacune pouvant prendre des valeurs comprises entre 0 et 1. Le réseau est également doté d'une couche cachée composée de 3 neurones. En outre, notre réseau de neurones possède une sortie qui produit une valeur comprise entre 0 et 1, visant à se rapprocher le plus possible de 0 ou de 1.

Voici ci-dessous un schéma représentant notre réseau de neurones:

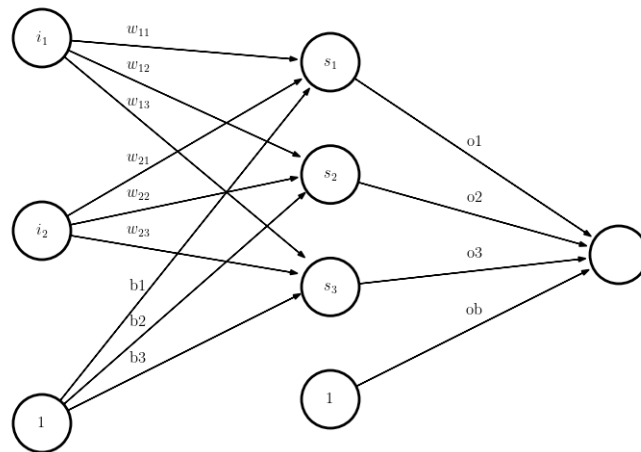


Figure 28: Schéma réseau de neurones XOR

8.1.4 Entraînement

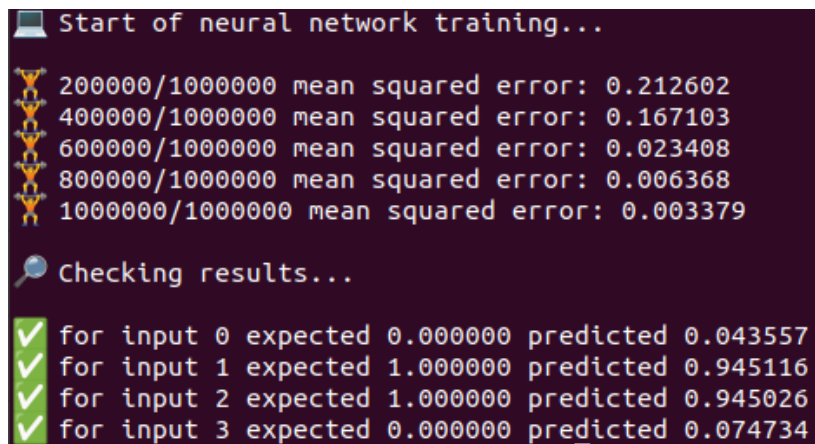
Pour entraîner notre réseau de neurones, nous testons à de nombreuses reprises les 4 combinaisons possibles des entrées XOR. Cela permet au réseau de neurones d'ajuster les biais et les poids de manière à minimiser les erreurs au fil des tests. Nous avons décidé de réaliser 1 000 000 de tests pour réduire au maximum la marge d'erreur lors des évaluations.

8.1.5 Affichage

Nous avons choisi d'implémenter un affichage sur le terminal lors de l'exécution du programme afin de faciliter la compréhension du fonctionnement de chaque composant du réseau de neurones. Cela nous permet de visualiser la progression du réseau au fil de son apprentissage.

Lorsque vous lancez l'exécutable, les informations affichées dans le terminal vous donnent un aperçu détaillé des opérations effectuées par chaque partie du réseau. Cette fonctionnalité offre une meilleure transparence sur les calculs effectués par les neurones, la manière dont les poids et les biais sont ajustés, ainsi que la manière dont le réseau se rapproche des valeurs attendues.

Ce suivi en temps réel dans le terminal offre une opportunité précieuse pour analyser le comportement du réseau de neurones et ajuster les paramètres si nécessaire, permettant ainsi d'optimiser les performances du modèle. Cette visibilité accrue contribue à une meilleure compréhension du processus d'apprentissage et facilite le débogage en cas d'erreurs ou d'incohérences dans les résultats obtenus.



```
Start of neural network training...  
200000/1000000 mean squared error: 0.212602  
400000/1000000 mean squared error: 0.167103  
600000/1000000 mean squared error: 0.023408  
800000/1000000 mean squared error: 0.006368  
1000000/1000000 mean squared error: 0.003379  
Checking results...  
✓ for input 0 expected 0.000000 predicted 0.043557  
✓ for input 1 expected 1.000000 predicted 0.945116  
✓ for input 2 expected 1.000000 predicted 0.945026  
✓ for input 3 expected 0.000000 predicted 0.074734
```

Figure 29: Affichage du terminal XOR

8.2 OCR

8.2.1 Propagation avant

Nous avons décidé d'utiliser la fonction softmax pour notre fonction d'activation elle permet de faire prédire à notre réseau de neurones une valeur de sortie.

$$\text{Softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}} \quad (1)$$

8.2.2 Rétropopagation

Nous avons décidé d'utiliser la fonction dérivée de sigmoïde afin de corriger les éventuelles erreurs du réseau de neurones. Cette fonction d'apprentissage permet au réseau de neurones de comprendre ses erreurs. Fonction dérivée de sigmoïde: $f'(x) = x \cdot (1 - x)$

8.2.3 Présentation

Notre réseau de neurones est organisé selon la structure suivante : il comprend 784 entrées, correspondant aux 784 pixels de l'image de taille 28x28. Chaque entrée peut prendre des valeurs comprises entre 0 et 1, reflétant l'intensité lumineuse du pixel. Si le pixel est noir, la valeur en entrée du neurone sera 1, sinon elle sera 0, créant ainsi une représentation binaire de l'image.

Le réseau est enrichi d'une couche cachée composée de 30 neurones. Ces neurones, agissant comme des encodeurs de caractéristiques, captent des motifs et des relations complexes présents dans l'image. Cette couche cachée permet au réseau d'apprendre des représentations abstraites, améliorant ainsi sa capacité à discriminer entre différentes images.

Par ailleurs, notre réseau de neurones possède 10 sorties, chacune produisant une valeur comprise entre 0 et 1. Ces sorties représentent les probabilités que l'image en entrée corresponde à chacun des chiffres de 0 à 9. L'utilisation de la fonction softmax garantit que la somme de ces probabilités est égale à 1. Ainsi, le réseau attribue une probabilité à chaque classe, et la classe avec la probabilité la plus élevée est considérée comme la prédiction finale.

La phase d'entraînement implique l'ajustement itératif des poids et des biais du réseau à l'aide de l'algorithme de rétropropagation du gradient. Cet ajustement vise à minimiser l'écart entre les prédictions du réseau et les étiquettes réelles associées à chaque image d'entraînement. Cette étape permet au réseau d'apprendre à extraire des caractéristiques pertinentes des images, améliorant ainsi sa capacité à généraliser sur de nouvelles données.

En phase d'inférence, l'image est présentée au réseau, et les activations des neurones sont calculées à travers les différentes couches jusqu'à l'obtention de la sortie. La classe prédite est déterminée en fonction des probabilités attribuées à chaque classe par la couche de sortie. L'évaluation du modèle sur un ensemble de validation indépendant garantit sa capacité à généraliser correctement, évitant ainsi le surajustement aux données d'entraînement.

Voici ci-dessous un schéma représentant notre réseau de neurones:

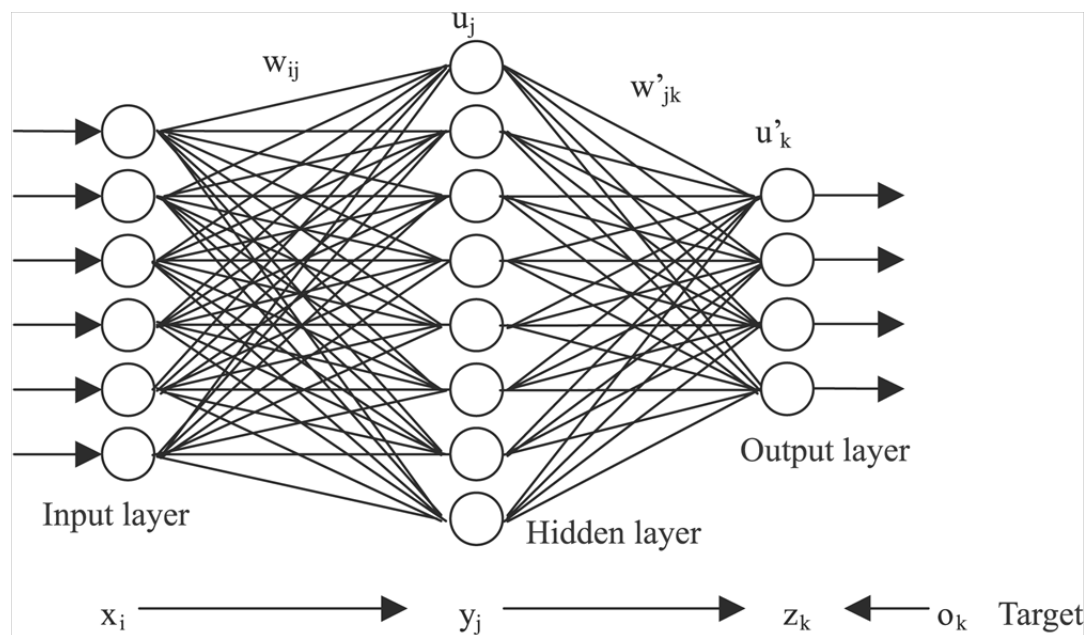


Figure 30: Schéma réseau de neurones OCR

8.2.4 Neurones d'entrée

Les entrées de notre réseau de neurones constituent la première étape cruciale dans le processus de traitement de l'information. Structuré pour traiter des images de taille 28x28 pixels, notre réseau compte 784 entrées, chacune correspondant à la valeur d'un pixel spécifique dans l'image. Ces valeurs peuvent varier de 0 à 1, représentant l'intensité lumineuse du pixel respectif.

Chaque entrée de notre réseau agit comme un point de données, reflétant la présence ou l'absence d'information lumineuse à une position spécifique de l'image. En d'autres termes, si le pixel est noir, la valeur d'entrée correspondante sera 1, et si le pixel est blanc, la valeur sera 0. Cette représentation binaire crée une structure de données adaptée au traitement par les neurones du réseau.

Cette approche binaire permet au réseau de capturer efficacement les caractéristiques importantes des images, en mettant l'accent sur la présence ou l'absence de certaines informations visuelles. Le processus d'ajustement des poids et des biais du réseau au cours de l'entraînement vise à optimiser la capacité du réseau à interpréter ces valeurs d'entrée et à effectuer des prédictions précises.

Il est intéressant de noter que la qualité des informations en entrée, ainsi que la diversité des données d'entraînement, jouent un rôle crucial dans la performance globale du réseau. Des entrées bien choisies permettent au réseau d'apprendre des motifs significatifs, améliorant ainsi sa capacité à généraliser sur de nouvelles données.

En résumé, les entrées de notre réseau de neurones, représentant les pixels d'une image, fournissent la matière première à partir de laquelle le réseau extrait des caractéristiques pertinentes lors de l'apprentissage. La qualité et la structure de ces entrées sont des éléments clés pour garantir le bon fonctionnement et la performance du réseau dans la tâche spécifique qu'il entreprend.

8.2.5 Couche caché

Les couches cachées de notre réseau de neurones jouent un rôle fondamental dans le processus d'apprentissage et de représentation des caractéristiques. Dans notre architecture, la couche cachée, composée de 30 neurones, agit comme un espace d'encodage où des motifs et des relations complexes présents dans l'image sont captés et appris.

Chaque neurone de la couche cachée agit comme un détecteur de caractéristiques, cherchant à identifier des motifs spécifiques ou des combinaisons de pixels qui sont significatifs pour la tâche à accomplir. En ajustant les poids associés à chaque connexion neuronale, le réseau apprend à reconnaître des motifs hiérarchiques, permettant une représentation plus abstraite de l'information présente dans les données d'entrée.

La capacité des neurones de la couche cachée à extraire des informations pertinentes contribue directement à l'amélioration de la capacité du réseau à discriminer entre différentes classes. Ces caractéristiques apprises dans la couche cachée servent ainsi de base pour la prise de décision ultime dans la couche de sortie, où les probabilités associées à chaque classe sont calculées.

Il est important de souligner que l'entraînement du réseau implique l'ajustement des poids de manière itérative à travers les différentes couches, grâce à l'algorithme de rétropropagation du gradient. Ce processus permet d'optimiser la représentation apprise dans la couche cachée, améliorant ainsi la capacité du réseau à généraliser et à faire des prédictions précises sur de nouvelles données.

En résumé, les couches cachées de notre réseau de neurones agissent comme des extracteurs de caractéristiques, transformant l'information brute des pixels en représentations plus abstraites et discriminantes. Leur rôle central dans le processus d'apprentissage contribue de manière significative à la performance globale du réseau dans la tâche spécifique pour laquelle il a été conçu.

8.2.6 Neurones de sortie

Les sorties de notre réseau de neurones représentent la phase finale du processus de traitement, où les informations apprises et encodées dans les couches cachées sont converties en résultats significatifs. Dans notre architecture, le réseau est configuré pour avoir 10 sorties, chacune produisant une valeur entre 0 et 1. Ces sorties sont interprétées comme les probabilités que l'image en entrée corresponde à l'une des dix classes, représentant les chiffres de 0 à 9.

La mise en œuvre de la fonction softmax au niveau des sorties assure que la somme de toutes les probabilités est égale à 1. Cela crée une distribution de probabilités parmi les différentes classes, permettant au réseau de prendre des décisions probabilistes et d'attribuer une certaine confiance à chaque prédiction.

Lors de la phase d'inférence, l'image est propagée à travers le réseau, et les activations des neurones de la couche de sortie sont calculées. La classe prédite est alors déterminée en sélectionnant celle qui correspond à la plus haute probabilité. Ainsi, les sorties du réseau fournissent une réponse quant à la classe probable de l'image d'entrée.

Cette configuration de sortie permet une interprétation claire des prédictions du réseau, facilitant la compréhension de la confiance accordée à chaque classe. Ces probabilités peuvent être utilisées pour évaluer la fiabilité des prédictions du modèle, notamment dans les cas où une réponse incertaine pourrait être cruciale.

En résumé, les sorties de notre réseau de neurones, guidées par la fonction softmax, offrent une représentation probabiliste des prédictions du modèle. Ces sorties sont le résultat d'un processus complexe d'apprentissage où le réseau a assimilé des informations pour prendre des décisions éclairées sur la classe à laquelle appartient l'image en entrée.

8.2.7 Entraînement

Pour l'entraînement de notre réseau de neurones, la décision a été prise d'exploiter notre propre base de données. L'origine de ce choix réside dans notre ambition initiale d'utiliser la renommée base de données MNIST. Malheureusement, lors des premières itérations, il est devenu évident que notre réseau, dans sa configuration initiale, ne présentait pas les performances requises pour traiter efficacement nos images spécifiques.

Face à ce constat, nous avons entrepris une analyse approfondie de notre architecture de réseau de neurones. Des ajustements et des améliorations ont été minutieusement appliqués afin de mieux adapter le modèle à la nature particulière de nos données. Ce processus itératif de raffinement a été essentiel pour surmonter les obstacles initiaux et maximiser le potentiel de notre réseau.

Au cours de cette phase d'optimisation, des techniques avancées d'apprentissage profond ont été intégrées, renforçant ainsi la capacité du réseau à extraire des caractéristiques pertinentes de nos images. Les résultats ont été significatifs, démontrant une amélioration substantielle des performances. Cette approche de personnalisation de l'architecture a non seulement résolu les défis spécifiques posés par nos données, mais elle a également enrichi notre compréhension des nuances inhérentes à notre domaine d'application.

En définitive, cette démarche proactive a non seulement hissé notre réseau à des niveaux de performance satisfaisants, mais elle a également élargi notre expertise dans le domaine de l'apprentissage profond, renforçant ainsi notre capacité à résoudre des problèmes complexes liés à des jeux de données spécifiques.



Figure 31: Exemple image d'entraînement réseau de neurones

8.2.8 Base de donnée

Nous avons choisi d'utiliser à la fois la base de données MNIST et EMNIST, en complément de nos propres données, afin d'optimiser la capacité de reconnaissance de notre réseau neuronal. Cette approche stratégique vise à doter notre modèle d'une robustesse accrue en lui exposant à une diversité étendue d'images de chiffres manuscrits.

La base de données MNIST offre un ensemble standardisé de chiffres manuscrits de 0 à 9, tandis que la base de données EMNIST élargit cette variété en incluant des styles d'écriture diversifiés provenant de différentes sources. Cette combinaison intentionnelle de ressources nous permet de former notre réseau sur un ensemble de données étendu, favorisant ainsi une meilleure généralisation et une adaptabilité aux variations stylistiques propres à l'écriture manuscrite.

En parallèle, l'intégration de nos propres données contribue à personnaliser davantage l'entraînement du réseau en l'adaptant spécifiquement aux caractéristiques uniques des images que nous souhaitons reconnaître. Cela permet d'adresser des nuances particulières qui peuvent être cruciales dans le contexte de notre application spécifique.

Cette approche hybride, combinant des bases de données bien établies avec des données internes, vise à maximiser la performance du réseau en lui fournissant une variété suffisante pour affronter divers scénarios du monde réel. À travers cette démarche, nous cherchons à obtenir une capacité de reconnaissance optimale pour les images que nous lui soumettons, élevant ainsi la qualité et la fiabilité des prédictions de notre modèle.

9 Application

9.1 Introduction

L'essence de cette interface est orientée vers la création d'une expérience utilisateur à la fois intuitive et complète, mettant l'accent sur la résolution automatisée de sudokus à partir d'images. L'objectif est d'offrir aux utilisateurs une plateforme conviviale et fonctionnelle tout en intégrant des fonctionnalités avancées pour la manipulation et le prétraitement des grilles de sudoku.

9.2 Expérience utilisateur intuitive

L'interface graphique vise à offrir une expérience utilisateur transparente et accessible. Les éléments visuels et interactifs sont soigneusement conçus pour être intuitifs, permettant aux utilisateurs, même novices, de naviguer aisément à travers les différentes fonctionnalités proposées par l'application.

9.3 Résolution automatisée de sudokus

L'objectif principal de l'interface est de permettre aux utilisateurs de résoudre automatiquement des sudokus à partir d'images. Pour ce faire, l'interface propose des fonctionnalités OCR (Reconnaissance Optique de Caractères) avancées pour extraire les informations pertinentes à partir des images des grilles de sudoku. Les algorithmes intégrés analysent les données extraites pour résoudre de manière automatisée les sudokus présentés.

9.4 Fonctionnalités avancées

En plus de la résolution automatisée, l'interface offre des fonctionnalités avancées pour la manipulation et le prétraitement des grilles de sudoku. Les utilisateurs peuvent, par exemple, importer des images de sudokus à partir de diverses sources, ajuster la qualité de l'image, ou même effectuer des rotations et des redimensionnements pour optimiser la reconnaissance des caractères.

9.5 Manipulation des grilles de sudoku

L'interface permet également aux utilisateurs de manipuler les grilles de sudoku de manière interactive. Ils peuvent remplir manuellement les cases, effacer des valeurs, ou même sélectionner des stratégies de résolution spécifiques. Cette approche hybride, combinant à la fois l'automatisation et l'interaction manuelle, offre une flexibilité accrue dans la manière dont les utilisateurs choisissent de résoudre leurs sudokus.

9.6 Design visuel cohérent

L'ensemble de l'interface graphique est conçu avec un souci particulier pour la cohérence visuelle. L'utilisation de couleurs, de symboles et de mises en page est pensée de manière à fournir une expérience homogène et agréable. L'objectif est de rendre l'utilisation de l'application aussi agréable que possible, tout en facilitant la compréhension des fonctionnalités disponibles.

En somme, l'interface graphique développée pour le projet OCR Sudoku Solver s'efforce de réaliser un équilibre entre l'automatisation sophistiquée de la résolution des sudokus à partir d'images et la fourniture d'outils avancés et intuitifs pour la manipulation des grilles. L'accent mis sur une expérience utilisateur conviviale vise à rendre l'application accessible à un large éventail d'utilisateurs, des débutants aux utilisateurs expérimentés, tout en répondant aux besoins variés liés à la résolution de sudokus.

10 Technologies Utilisées

L'implémentation de l'interface graphique pour le projet OCR Sudoku Solver repose sur l'utilisation de la bibliothèque GTK (GIMP Toolkit), une bibliothèque logicielle open-source largement utilisée pour le développement d'interfaces utilisateur graphiques. Cependant, une particularité importante dans ce projet est la décision de ne pas utiliser l'outil Glade, habituellement associé à GTK pour faciliter la conception visuelle des interfaces. Au lieu de cela, l'interface a été entièrement conçue et implémentée manuellement, garantissant un contrôle total sur chaque aspect du processus de conception et d'implémentation.

10.1 Utilisation de GTK

GTK est choisi comme base pour l'interface en raison de sa popularité, de sa flexibilité et de sa compatibilité avec le langage de programmation utilisé (probablement le langage C, comme indiqué précédemment). GTK offre un ensemble complet de widgets (éléments d'interface utilisateur) et de fonctionnalités pour la création d'interfaces graphiques interactives.

10.2 Contrôle total sur la conception

La décision de ne pas utiliser Glade indique une préférence pour un contrôle plus fin sur la conception de l'interface. Glade est un outil visuel permettant de créer des interfaces GTK en mode glisser-déposer sans avoir à écrire de code. Cependant, en choisissant de ne pas utiliser Glade, les développeurs ont décidé de gérer manuellement le placement, la disposition et l'interaction des éléments de l'interface.

10.3 Personnalisation avancée

En évitant l'utilisation de Glade, les développeurs peuvent apporter des ajustements et des personnalisations avancées à chaque élément de l'interface. Cela inclut le contrôle précis de la taille, de la position, du style visuel et de l'interaction des widgets. Chaque détail peut être ajusté en fonction des besoins spécifiques du projet, sans être limité par les contraintes imposées par un outil de conception visuelle.

10.4 Adaptation aux besoins spécifiques du projet

En choisissant de coder manuellement l'interface avec GTK, les développeurs peuvent adapter chaque composant visuel pour répondre aux besoins spécifiques du projet OCR Sudoku Solver. Cela inclut l'intégration fluide des fonctionnalités

avancées liées à la résolution de sudokus à partir d'images et à la manipulation des grilles.

10.5 Contrôle de la performance

L'utilisation directe de GTK permet également un contrôle plus précis sur les performances de l'interface. En évitant une surcharge potentielle liée à l'utilisation d'un outil visuel externe, les développeurs peuvent optimiser le code pour garantir une expérience utilisateur fluide et réactive.

En résumé, le choix de ne pas utiliser Glade pour concevoir l'interface graphique avec GTK reflète une volonté de contrôle total sur le processus de conception et d'implémentation. Cela permet une personnalisation approfondie, une adaptation précise aux besoins du projet et un contrôle optimal des performances, contribuant à la création d'une interface robuste et spécifique au contexte du projet OCR Sudoku Solver.

11 Outils Utilisés

L'élaboration de l'interface graphique du Sudoku Solver a bénéficié de l'utilisation d'outils puissants et flexibles tels que GTK (GIMP Toolkit), qui est associé à des concepts comme GTK Grid et GTK Box. Chacun de ces éléments joue un rôle essentiel dans la création d'une interface utilisateur sophistiquée et réactive.

11.1 GTK (GIMP Toolkit)

GTK, ou GIMP Toolkit, est une bibliothèque logicielle open-source largement utilisée dans le développement d'interfaces utilisateur graphiques. Son utilisation dans le cadre du Sudoku Solver offre plusieurs avantages significatifs :

11.1.1 Widgets Riches

GTK fournit une variété de widgets prêts à l'emploi tels que les boutons, les champs de texte, les boîtes, etc. Ces widgets simplifient la création d'éléments interactifs et visuels, contribuant ainsi à la conception globale de l'interface.

11.1.2 Événements et Signaux

GTK permet la gestion efficace des événements utilisateur, comme les clics de souris ou les pressions de touches. Les signaux associés aux widgets facilitent la gestion des interactions utilisateur, permettant ainsi la mise en œuvre de fonctionnalités réactives.

11.1.3 Personnalisation Visuelle

GTK offre une flexibilité considérable en matière de personnalisation visuelle. Les développeurs ont la liberté de définir les propriétés d'apparence des widgets, de choisir des thèmes visuels et d'ajuster la disposition des éléments pour créer une expérience utilisateur esthétiquement cohérente.

11.2 GTK Grid et GTK Box

11.2.1 GTK Grid

La GTK Grid est un conteneur qui organise les widgets en une structure de grille. Dans le contexte du Sudoku Solver, elle est utilisée pour positionner les différents éléments de l'interface de manière ordonnée. Les grilles facilitent la disposition logique des boutons, des champs de texte et d'autres éléments, créant ainsi une structure visuelle claire et ergonomique.

11.2.2 GTK Box

La GTK Box, quant à elle, est un conteneur linéaire qui organise les widgets en une seule ligne ou colonne. Elle est particulièrement utile pour créer des dispositions horizontales ou verticales de manière flexible. Dans l'interface du Sudoku Solver, la GTK Box est utilisée pour regrouper et organiser les éléments de manière logique, améliorant ainsi la lisibilité et la cohérence visuelle.

Ces outils offrent une base solide pour la conception de l'interface graphique, permettant aux développeurs d'implémenter des fonctionnalités avancées tout en maintenant un contrôle fin sur l'apparence et l'interaction des éléments. L'utilisation de GTK, couplée aux concepts de GTK Grid et GTK Box, contribue à la création d'une interface utilisateur robuste, esthétique et fonctionnelle pour le Sudoku Solver.

12 Interface Graphique

La fenêtre principale est agencée de manière à garantir une disposition ergonomique et claire, facilitant ainsi la navigation de l'utilisateur à travers les différentes fonctionnalités du Sudoku Solver. Les éléments visuels tels que les boutons, les champs de texte et les zones d'affichage sont positionnés de manière logique pour créer une hiérarchie visuelle intuitive. L'objectif est de minimiser la confusion et d'optimiser l'accessibilité aux fonctionnalités clés.

12.1 Boutons et Fonctionnalités

12.1.1 Noir et Blanc

Le bouton "Noir et Blanc" incarne une fonctionnalité essentielle en déclenchant l'application d'un filtre noir et blanc à la grille de sudoku. Cette transformation d'image contribue à améliorer la qualité visuelle en éliminant les nuances de couleur. En conséquence, la détection précise des chiffres dans la grille est facilitée, créant ainsi une base solide pour les étapes ultérieures du processus de résolution.

12.1.2 Hough

Le bouton "Hough" exploite la puissance de la transformée de Hough pour détecter les lignes de la grille de sudoku. Cette fonctionnalité s'avère cruciale pour optimiser la détection des cases et des chiffres, améliorant significativement la précision du processus de résolution. La méthode de Hough permet une

identification robuste des structures linéaires dans l'image, garantissant une base fiable pour l'étape de reconnaissance des caractères.

12.1.3 Rotation

Le bouton "Rotation" offre une souplesse considérable en proposant deux modes distincts. La rotation manuelle donne à l'utilisateur la possibilité de régler l'orientation de la grille selon ses préférences. En parallèle, la rotation automatique ajuste automatiquement l'angle, optimisant ainsi la reconnaissance des chiffres. Cette fonctionnalité s'inscrit dans une approche centrée sur l'utilisateur, permettant un contrôle précis et une automatisation pour une expérience optimale.

12.1.4 Grid Display

Le bouton "Grid Display" confère à l'utilisateur le pouvoir d'activer ou de désactiver l'affichage de la grille. Cette fonction offre une flexibilité visuelle, permettant à l'utilisateur de mieux comprendre la structure du sudoku. En masquant ou révélant la grille, cette fonctionnalité facilite la visualisation et l'analyse de la disposition des chiffres, adaptant ainsi l'interface aux préférences individuelles de l'utilisateur.

12.1.5 Load Image

Le bouton "Load Image" propose une interface conviviale pour charger une image de sudoku depuis le navigateur de fichiers. Cette fonctionnalité simplifie le processus d'importation d'images, offrant à l'utilisateur une solution rapide et intuitive pour sélectionner l'image à traiter. Cette facilité d'utilisation contribue à une expérience utilisateur transparente dès le début du processus.

12.1.6 Back

Le bouton "Back" enrichit l'expérience utilisateur en offrant une fonction de retour en arrière. Cette possibilité permet à l'utilisateur de revisiter les étapes précédentes et de réajuster les paramètres au besoin. La flexibilité de revenir en arrière garantit une exploration sans stress du processus, permettant à l'utilisateur de perfectionner les réglages à chaque étape.

12.1.7 Draw

Le bouton "Draw" déclenche un mode interactif permettant à l'utilisateur de dessiner manuellement des chiffres sur la grille. Cette fonctionnalité sert à entraîner le réseau de neurones, contribuant à une amélioration continue des performances de résolution. Elle offre une approche pratique pour affiner la compréhension du modèle en permettant à l'utilisateur de fournir des données d'entraînement spécifiques.

12.1.8 Solve

Le bouton "Solve" orchestre l'ensemble du processus, appliquant les traitements sélectionnés et résolvant la grille de sudoku. Cette fonctionnalité centralisée

génère des résultats clairs et précis qui sont présentés de manière compréhensible dans la fenêtre principale. Elle représente l'aboutissement du parcours de l'utilisateur à travers les différentes étapes de prétraitement et de résolution, simplifiant ainsi le processus global.

13 Convivialité et Accessibilité

13.1 Convivialité

L'accent mis sur la convivialité dans l'interface graphique du Sudoku Solver se manifeste à travers plusieurs aspects essentiels :

13.1.1 Contrôles Intuitifs

Les contrôles, tels que les boutons et les menus, sont soigneusement conçus pour être intuitifs, facilitant ainsi la navigation de l'utilisateur sans nécessiter une courbe d'apprentissage complexe. L'utilisation de conventions visuelles familières et de libellés clairs contribue à une expérience utilisateur sans friction.

13.1.2 Messages d'Erreur Informatifs

En cas d'erreurs ou de situations exceptionnelles, l'interface génère des messages d'erreur informatifs. Ces messages sont formulés de manière à guider l'utilisateur vers la résolution du problème, favorisant ainsi une expérience utilisateur positive même lorsqu'il y a des obstacles.

13.1.3 Structure Visuelle Cohérente

L'interface maintient une structure visuelle cohérente tout au long de l'application. Les couleurs, les styles et la disposition des éléments sont harmonisés pour créer une expérience homogène. La cohérence visuelle contribue à une navigation fluide et à une compréhension intuitive de l'interface.

13.2 Accessibilité

L'accessibilité est au cœur de la conception de l'interface graphique, conformément aux normes d'accessibilité standard. Les principaux aspects incluent :

13.2.1 Contraste et Taille du Texte

Les choix de couleurs assurent un contraste adéquat, facilitant la lisibilité du texte. De plus, la taille du texte est ajustée pour garantir une visibilité optimale, répondant ainsi aux besoins des utilisateurs ayant des déficiences visuelles.

13.2.2 Navigation au Clavier

L'interface est pensée pour permettre une navigation fluide au clavier, offrant une alternative efficace à la souris. Les raccourcis clavier sont disponibles pour les principales actions, permettant ainsi aux utilisateurs d'accéder rapidement aux fonctionnalités sans dépendre exclusivement de la souris.

13.2.3 Compatibilité avec les Outils d'Assistance

L'interface est testée et optimisée pour être compatible avec les outils d'assistance, tels que les lecteurs d'écran. Les éléments interactifs sont correctement étiquetés, et les informations visuelles sont également disponibles sous forme textuelle pour garantir une expérience équitable pour tous les utilisateurs.

14 CSS dans le Design

L'intégration de CSS (Cascading Style Sheets) joue un rôle crucial dans le processus de conception de l'interface graphique GTK du Sudoku Solver. Bien que CSS soit plus traditionnellement associé au développement web, son utilisation dans le contexte de GTK offre plusieurs avantages significatifs en matière de personnalisation visuelle et de maintien d'une apparence cohérente.

14.1 Personnalisation Visuelle

14.1.1 Contrôle des Propriétés d'Apparence

CSS permet aux développeurs de définir et de contrôler les propriétés d'apparence des éléments de l'interface GTK. Ils peuvent spécifier des couleurs, des polices, des marges, des espacements et d'autres propriétés visuelles pour chaque widget. Cela offre une flexibilité considérable pour adapter l'apparence de l'interface aux besoins spécifiques du Sudoku Solver.

14.1.2 Thèmes Visuels

L'utilisation de CSS permet d'appliquer des thèmes visuels à l'ensemble de l'interface. Cela garantit une cohérence visuelle à travers différents éléments, créant une esthétique globale homogène. Les thèmes peuvent être personnalisés pour répondre à des préférences spécifiques ou pour intégrer l'interface graphique dans un environnement visuel plus large.

14.2 Maintien de la Cohérence

14.2.1 Réutilisabilité du Code

En séparant la logique de style de la logique fonctionnelle, l'utilisation de CSS favorise la réutilisabilité du code. Les styles définis dans une feuille de style CSS peuvent être appliqués à plusieurs widgets, évitant ainsi la redondance du code et facilitant la maintenance à long terme.

14.2.2 Facilitation des Modifications

CSS simplifie le processus de modification de l'apparence de l'interface. Les ajustements visuels peuvent être apportés de manière centralisée dans la feuille de style sans nécessiter des modifications directes dans le code source, ce qui facilite la gestion des évolutions visuelles.

14.3 Responsivité de l'Interface

14.3.1 Adaptabilité à Différentes Tailles d'Écran

CSS est essentiel pour rendre l'interface graphique du Sudoku Solver réactive aux différentes tailles d'écran. Les règles de style peuvent être adaptées pour garantir que l'interface conserve son esthétique et sa lisibilité, quel que soit le dispositif utilisé.

14.3.2 Media Queries

L'utilisation de media queries en CSS permet de définir des règles spécifiques en fonction des caractéristiques de l'écran, comme la largeur et la hauteur. Cela permet d'ajuster dynamiquement le style en fonction du contexte d'affichage.

En conclusion, l'intégration de CSS dans le design de l'interface graphique GTK du Sudoku Solver apporte une flexibilité et une facilité de maintenance essentielles. Cela permet aux développeurs de créer une interface visuellement attrayante, cohérente et adaptative, offrant ainsi une expérience utilisateur optimale. L'utilisation judicieuse de CSS contribue à l'esthétique globale et à la convivialité de l'application, enrichissant ainsi l'expérience des utilisateurs.

15 Conclusion

L'interface graphique développée avec GTK, sans l'utilisation de Glade, représente une solution complète et sophistiquée pour la résolution automatisée de sudokus à partir d'images. Les points saillants de cette conclusion comprennent :

15.1 Minutie dans la Conception

Chaque bouton et fonctionnalité a été élaboré avec minutie pour garantir une expérience utilisateur exceptionnelle. L'attention portée aux détails permet à l'interface de répondre de manière efficace aux besoins variés des utilisateurs, qu'ils soient novices ou expérimentés.

15.2 Puissance et Flexibilité

L'interface combine puissance et flexibilité en intégrant des fonctionnalités avancées telles que la transformation d'image, la détection de lignes, la rotation automatique, et bien d'autres. Cela offre aux utilisateurs un ensemble complet d'outils pour résoudre des sudokus de manière efficace et personnalisée.

15.3 Expérience Utilisateur Enrichissante

L'objectif ultime de l'interface est de fournir une expérience utilisateur enrichissante. Cette expérience est construite sur un équilibre entre la convivialité, l'accessibilité, et la sophistication fonctionnelle, créant ainsi une solution globale pour la résolution automatisée de sudokus.

En résumé, l'interface graphique OCR Sudoku Solver représente un exemple réussi d'intégration de technologies avancées, de concepts de conception conviviaux, et d'outils puissants pour répondre de manière exhaustive aux défis de la résolution automatisée de sudokus à partir d'images. Cette réalisation démontre

la capacité à créer une interface sophistiquée tout en offrant une expérience utilisateur optimale.

List of Tables

1	Répartition des taches	5
---	----------------------------------	---

List of Figures

1	Victor Agahi	4
2	Mehdi AZOUZ	4
3	Jules MAGNAN	5
4	Gaspard TORTERAT SLANDA	5
5	Avant filtre grayscale	6
6	Après filtre grayscale	6
7	Avant Flou de Gaus	7
8	Après flou de Gauss	7
9	Avant filtre noir et blanc	7
10	Après filtre noir et blanc	7
11	Avant Canny Edge	9
12	Après Canny Edge	9
13	Avant rotation	9
14	Après rotation	9
15	Accumulateur normalisé	10
16	Avant Hough	11
17	Après Hough	11
18	Avant Flood	12
19	Après Flood	12
20	Avant extraction	12
21	Après extraction	12
22	Avant Regularisation	13
23	Après Regularisation	13
24	Exemple de grille non resolution en format original	16
25	Exemple de grille non resolution en format ".result"	17
26	Avant Résolution	17
27	Affichage	17
28	Schéma réseau de neuronesXOR	19
29	Affichage du terminal XOR	20
30	Schéma réseau de neurones OCR	22
31	Exemple image d'entraînement réseau de neurones	26
32	Affichage réseau de neurone	28

References

- [1] John Canny (1986) *A Computational Approach To Edge Detection*, IEEE.