Rapport première soutenance OCR

Thomas BALLER - Arthur HAMON

Romane MASSET - Xavier GHOSTINE

Studio : TRAX

EPITA Rennes - Année 2023

Prépa SPE - Promotion 2027

Table des matières

1 Introduction

1.1 Qu'est ce qu'un OCR ?

"OCR" qui vient de Optical Caractère Recognition, ou Reconnaissance optique de caractère en français, désigne les procédés informatiques pour la traduction d'images de textes imprimés ou dactylographiés en fichiers de texte. Il s'agit d'un logiciel permettant de reconnaître du texte sur une photographie, que ce soit des lettres ou des chiffres. Dans notre cas, nous devons seulement reconnaître les chiffres de 1 à 9.

1.2 Quel est le but de notre projet ?

Pour notre projet, nous devons donc développer un logiciel composé de 4 parties différentes, tout en gardant une compatibilité entre nos différentes parties afin de permettre un échange d'informations fluide entre ces dernières.

Dans un premier temps, il conviendra d'appliquer différents filtres (rotations, niveau de gris, flous, binarisation...) sur l'image choisit par l'utilisateur grâce à l'interface graphique. Ces filtres auront pour objectif de normalisé les entrées utilisateurs pour les transmettre au réseau de neurones.

Le réseau de neurones, composé de 3 couches (une couche d'entrée, cachée, de sortie) a pour objectif d'être capable, suite à un entraînement, de donner une probabilité sur le chiffre qui lui a été donné en entré.

Une fois que le réseau de neurones à donner une estimation des chiffres présents sur la grille de sudoku, une grille de sudoku au format texte est créé, c'est cette grille qui sera envoyé à la partie du solveur de sudoku.

Le solveur de sudoku aura pour objectif, via un algorithme de "backtracking", de résoudre rapidement le sudoku.

Une fois le sudoku résolu, notre programme reconstitue une grille de sudoku avec la solution trouvé par nos algorithmes, en vert.

2 Pré-traitement de l'image

2.1 Mise en niveau de gris

La mise en niveau de gris d'une image est l'une des étapes initiales essentielles à appliquer lors d'un traitement d'image. Elle consiste à "supprimer" les couleurs d'une image en faisant la moyenne des valeurs RGB pour chaque composante du pixel.

Ainsi, pour chaque pixel P, si l'on pose R, G, B (resp. R', G', B') les composantes RGB de P (resp. P'), on a :

$$R' = G' = B' = 31R + 13G + 13B \tag{1}$$

Il existeégalement une autre formule alternative qui prend en compte une vision plus réaliste de l'œil humain, où les coefficients ne sont pas uniformes. Cependant, le choix entre ces deux techniques n'a pas d'importance pour notre application, car une étape ultérieure de binarisation sera effectuée dans le processus.

L'étape de mise en niveau de gris est importante, car elle permet pour les algorithmes suivant de ne prendre en compte qu'une seule composante de couleur du pixel plutôt que les 3, simplifiant le développement des algorithmes.

Image 1 : Application d'un filtre de mise à niveau de gris.

2.2 Réduction de bruit

Le bruit d'une image signifie une fluctuation parasite et aléatoire de lumière ou de couleurs dans une image. Ce bruit est souvent présent lors de prise de photos et peut causer de nombreux problèmes pour plus tard. Cependant, il est très difficile de réduire le bruit entièrement, mais il est possible de le réduire un peu, au moins de façon à ce qu'il soit moins impactant. Il existe plusieurs façons de réduire le bruit.

Image 2 : Exemple de réduction de bruit. A gauche, l'image avant la réduction de bruit et à droite, après.

2.3 Flou gaussien

Le flou gaussien répartit les poids en fonction de leurs distance par rapport au pixel. Par conséquent, les poids attribués aux pixels situés au centre de la matrice seront plus importants, tandis que ceux situés plus près des bords et des coins auront des poids nettement plus faibles. L'avantage principal est que ce filtre préserve les bords des images.

Image 3 : Exemple d'application du flou gaussien sur une image.

2.4 Binarisation

L'étape d'après la réduction de bruit est la binarisation. Cela signifie passer d'une image en nuance de gris à une image en noir et blanc. Cette étape est très utile car elle permet de réduire encore plus l'information, et simplifie grandement les algorithmes d'après.

Pour ce faire, nous avons utilisé une méthode de seuillage. Un seuillage, comme son nom l'indique, consiste à définir une certaine valeur s en tant que seuil. Tout pixel dépassant ce seuil deviendra un pixel blanc, tandis que tout les autres pixels deviendront noir. La méthode la plus basique consiste à définir un s constant pour toute l'image, il s'agit du seuillage global mais cette technique basique est peu efficace pour des images ayant des variances de luminosités. C'est pour cette raison que nous ferons pour la prochaine soutenance la technique du seuillage adaptatif.

Le seuillage adaptatif fait varier le seuil s tel que s est égal à la moyenne des valeurs des pixels aux alentours, moins une constante C.

Ainsi, les pixels plus sombres que leurs voisins seront mis en arrière-plan tandis que les pixels plus clairs ou de même intensité seront au premier plan.

Image 4 : A gauche, une image en niveau de gris. A droite, la même image mais avec un filtre de binarisation.

2.5 Rotation d'image

Il existe plusieurs méthodes de rotation d'image. Nous avons opté pour la plus simple et efficace. Nous avons tout d'abord implémenté une méthode simple qui consiste à faire la rotation avec une seule matrice :

$$x'cos(\theta)sin(\theta)xy' = -sin(\theta)cos(\theta)y \tag{2}$$

Le problème est que nous avons des pertes de qualités de l'image, car certains pixels n'existent plus. Ils sont hors de l'image, ce qui est dû à une simplification et donne donc une imprécision de la formule.

Nous comptons utilisé le shearing qui consiste à prendre chaque pixel de l'image puis à appliquer 3 multiplications successives par des matrices dite de shearing. Le shearing est une transformation qui va consister à décaler chaque point de l'image en fonction d'un certain angle.

Après avoir calculé nos nouvelles coordonnée, ils nous faut leurs ajouter un décalage pour pouvoir replacer bien l'image en son centre.

La perte de qualité après rotation est minime cependant, il y a un peu de crénelage (c'est-à-dire qu'il y a des sortes d'escalier) sur l'image. Mais cela n'empêche en rien la détection de ligne. Ce qui est bien mieux que la première fonction de rotation ou nous avions de la perte qui pouvais occasionner une gêne dans la détection des lignes.

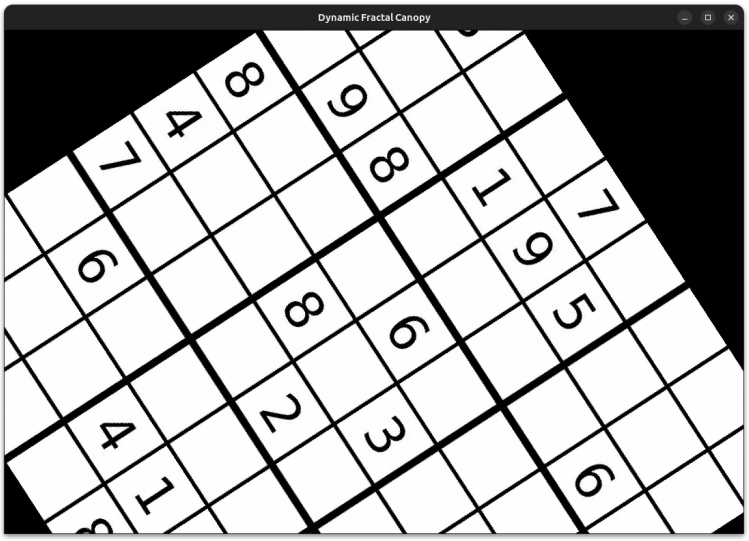

Image 5 : Rotation de 45° d'une grille de sudoku

2.6 Détéction des lignes

La détection des lignes est une étape cruciale pour la suite du traitement d'image. Elle permet de détecter le plus gros polygone à quatre côtés, qui sera notre grille de sudoku (à noter qu'il ne s'agira pas forcément d'un carré bien orienté, mais peut-être d'une grille penchée, elle ne sera pas strictement carrée).

Pour détecter les lignes, nous utilisons la technique de la transformée de Hough. Pour rappel, il est possible d'écrire une droite sous forme paramétrique :

$$y = ax + b \tag{3}$$

, et chaque point de coordonnées blanc a des coordonnées (x, y). Le principe de la transformée est sous forme de vote, chaque point blanc a donc certains a et b qui valident l'équation $y = ax + b$, ce point-là va donc voter pour chacun des ses points. Et ainsi de suite pour chaque points blancs. Il est ensuite possible de récupérer les couples (a, b) ayant récupéré le plus de votes. Ce seront les lignes de notre image.

La seule différence avec l'application pratique de l'algorithme de Hough est qu'il ne s'agit pas de couple (a, b), tel que $y = ax + b$ mais de couple (ρ, θ) , tel que $\rho = xcos(\theta) + ysin(\theta)$. En effet, ce changement est nécessaire car il est impossible de représenter des droites horizontales avec le couple (a, b), cela aurait demandé $a = +\infty$ et c'est impossible.

Ci-dessous, un exemple de l'espace des couples (ρ, θ). Plus un point est blanc, plus il a reçu de vote.

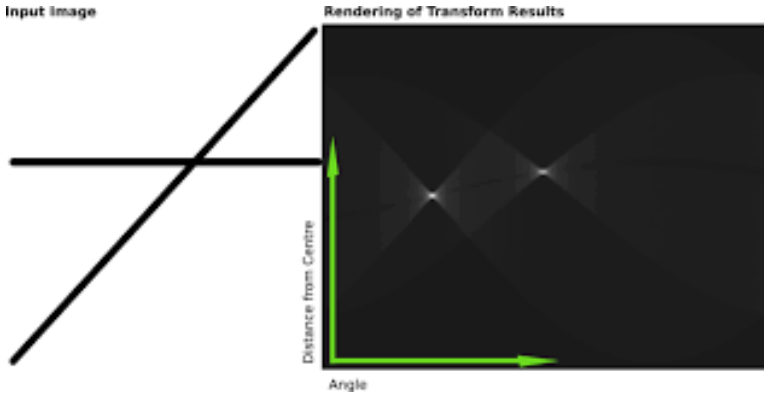

Image 6 : A gauche, une image possédant deux lignes. A droite, le rendu de la transformée de Hough avec l'espace de Hough.

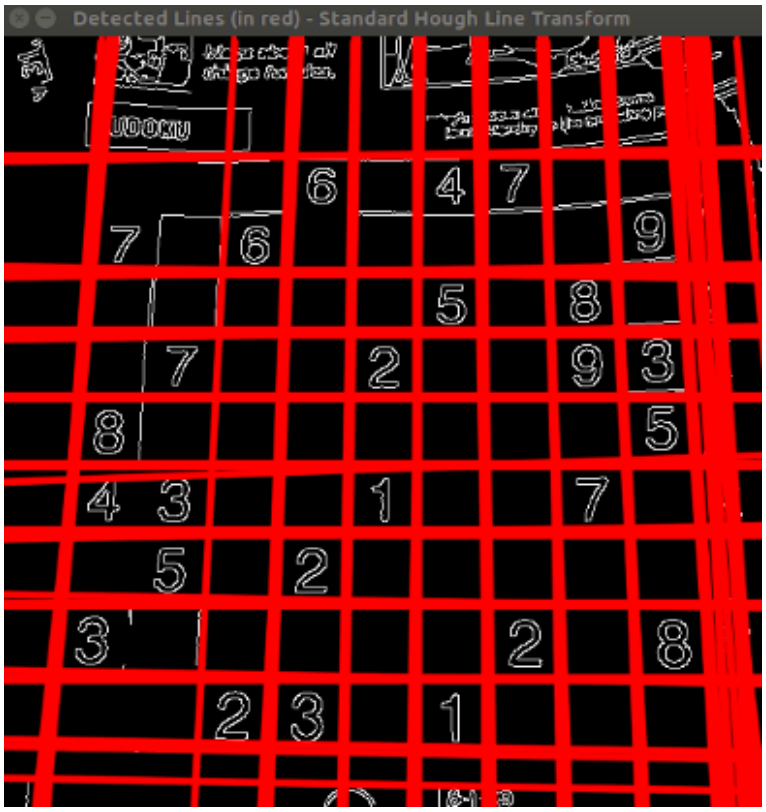

Image 7 : Résultat attendu d'une transformée de Hough sur une grille de Sudoku

2.7 Filtre de Canny

Le filtre de Canny est une méthode avancée de détection de grille. Ce filtre pourra remplacer le seuillage adaptatif afin de simplifier la détection des lignes de la transformée de Hough. Ce filtre est une amélioration d'un autre filtre (Filtre de Sobel), en appliquant d'autre étapes après pour minimiser le bruit et mettre en évidence les lignes.

Ces étapes sont :

— Filtre de Sobel
— Suppression des non-maxima
— Hystérésis

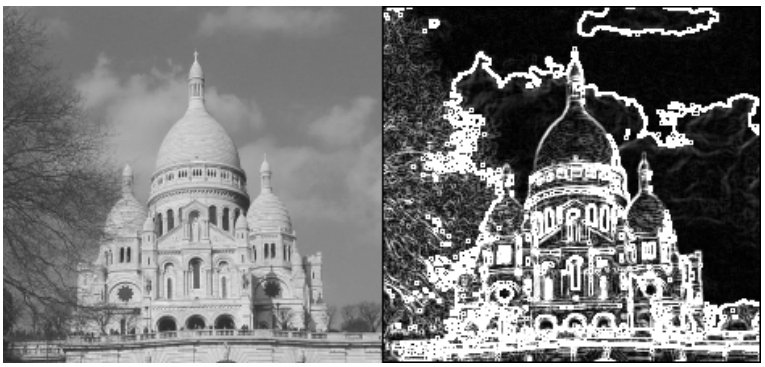

Image 8 : Application du filtre de Sobel

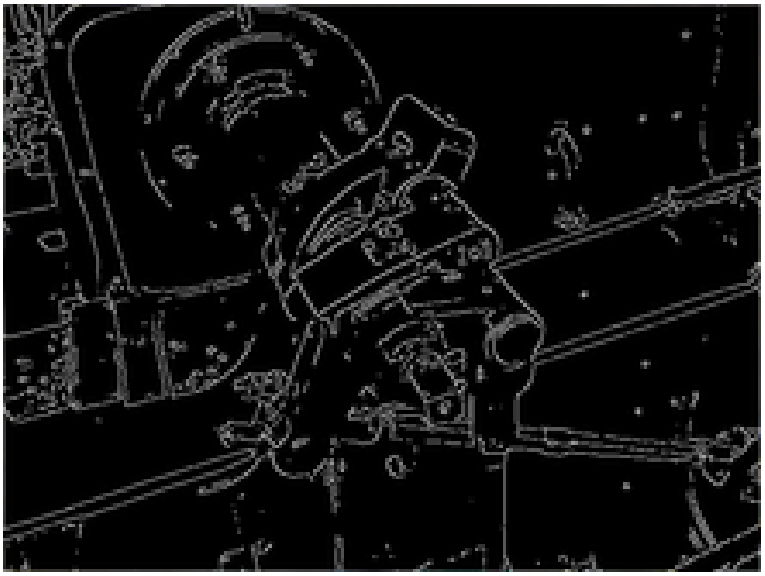

Image 9 : Image sur laquelle un filtre de Canny a été appliqué.

3 Le solver de Sudoku

3.1 Le principe de fonctionnement

Le Sudoku est un jeu de logique fascinant qui se joue sur une grille de 9x9 cases, divisée en neuf régions de 3x3 cases. Voici les règles de base :

Remplissage initial : Certaines cases de la grille sont déjà remplies avec des chiffres, formant un modèle initial.

Objectif : L'objectif du jeu est de remplir toutes les cases vides de la grille avec des chiffres de 1 à 9, de telle manière que chaque chiffre apparaisse une seule fois dans chaque ligne, chaque colonne et chaque région de 3x3 cases.

Indices : Les chiffres déjà donnés dans la grille servent d'indices pour résoudre les cases vides. Utilisez-les pour déduire les chiffres manquants en utilisant la logique.

Résolution : Le Sudoku est un jeu de logique, donc il n'y a pas besoin de deviner. En suivant des règles de déduction et d'élimination, vous devrez progressivement remplir la grille jusqu'à ce qu'elle soit complète.

Le Sudoku peut être un jeu complexe, mais avec de la pratique, vous améliorerez votre capacité à résoudre des grilles de différentes difficultés, allant de facile à diabolique. C'est un excellent exercice cérébral pour améliorer la concentration, la logique et la résolution de problèmes.
Afin de rester dans la médiocrité intellectuelle nous vous proposons dans notre projet un programme qui solvera n'importe quel sudoku du moment que celui- ci est soluble

Un sudoku est soluble si :

-Taille de la grille : Une grille de Sudoku standard doit être composée de 9 lignes et 9 colonnes, formant une grille de 9x9 cases.

-Régions de 3x3 cases : La grille est divisée en neuf régions de 3x3 cases. Chacune de ces régions doit contenir tous les chiffres de 1 à 9, sans répétition.

-Chiffres pré-remplis : La grille doit contenir un certain nombre de chiffres pré-remplis pour guider le joueur. Ces chiffres doivent être répartis de manière équilibrée dans la grille, de manière à ce que la solution soit unique.
Seuls les chiffres de 1 à 9 sont autorisés dans la grille. Aucun autre chiffre ou caractère n'est accepté.

-Unicité de la solution : La grille doit avoir une unique solution. Il ne doit pas y avoir de confusion ou d'ambiguïté sur la manière de la résoudre. Ainsi avant de résoudre toute grille notre programme vous avertira si la grille saisie en entrée est soluble ou ne l'est pas.

Afin de résoudre le sudoku, nous avons décider de travailler avec un algorithme récursif suivant ce principe :

Commencez par parcourir la grille pour trouver la première case dont la valeur est 0.

Pour la case vide trouvée, commencez à essayer les chiffres de 1 à 9 pour voir s'ils sont valides. Vérifiez si le chiffre respecte les règles du Sudoku (pas de répétition dans la même ligne, colonne ou région de 3x3 cases).

Si vous trouvez un chiffre valide, placez-le dans la case et passez à la case suivante vide. Si vous atteignez une impasse, c'est-à-dire qu'aucun chiffre n'est valide pour la case actuelle, revenez en arrière.

Lorsque vous revenez en arrière, révisez le chiffre précédemment placé dans la case et essayez un autre chiffre. Cela revient à explorer toutes les possibilités jusqu'à ce que la solution soit trouvée.

Répétez ces étapes de manière récursive jusqu'à ce que la grille soit remplie

Retour en arrière : Si vous atteignez un point où aucune solution n'est possible, revenez en arrière à la dernière étape de récursion valide et explorez une autre possibilité.

Une fois que la grille est entièrement rempli elle ne peux être que rempli correctement.

Pour résoudre notre programme il vous faut compiler le programme avec la commande make, puis résoudre une grille avec la commande ./solver arg (avec arg le nom du fichier de la grille à résoudre).

Le programme commence par lire le contenu du fichier texte qui contient une grille de Sudoku. Le format du fichier doit respecter les règles du Sudoku, avec des chiffres de 1 à 9 pour les chiffres pré-remplis et des espaces ou des zéros pour les case vides. Si le programme parvient à résoudre la grille avec succès, il crée un nouveau fichier, en utilisant le même nom que le fichier d'entrée, mais avec une extension différente, ".result". Ce nouveau fichier contient la grille résolue.

Le fichier résultant contient la grille de Sudoku complète, y compris les chiffres pré-remplis et les chiffres nouvellement remplis par le programme pour résoudre la grille. Il respecte le même format que le fichier d'entrée, garantissant que la grille résultante est également valide en termes de format.

4 Réseau de neurone

Nous avons développé un réseau de neurones artificiels. Nous lui avons fait apprendre la fonction ou exclusif aussi appelé XOR, afin de démontrer le bon fonctionnement de notre réseau.

Le modèle que nous avons utilisé est le perceptron multi-couches, c'est un système qui apprend par l'expérience. Son objectif est d'apprendre à classifier les données que l'on lui donne en entrée. Pour cela chaque neurone va créer une courbe pour séparer les données.

4.1 Le principe de fonctionnement

Un perceptron multi-couches est un ensemble de neurones organisé en plusieurs couches. On trouve une couche d'entrée, une couche cachée et une couche de sortie. Les neurones entre chaque couches sont reliés avec des liens pondérés, les valeurs des liens pondérés sont appelées les poids. On peut résumer le fonctionnement ainsi : on entre des valeurs puis on propage ces valeurs au travers de chacune des couches. Au final, dans la dernière couche on obtient la probabilité d'avoir un 1 pour le résultat de l'opération XOR par exemple. La couche d'entrée et la couche cachée possèdent des matrices de poids et de biais. Les biais sont des valeurs qui permettent de décaler vers la gauche ou la droite les courbes que dessine chaque neurone.

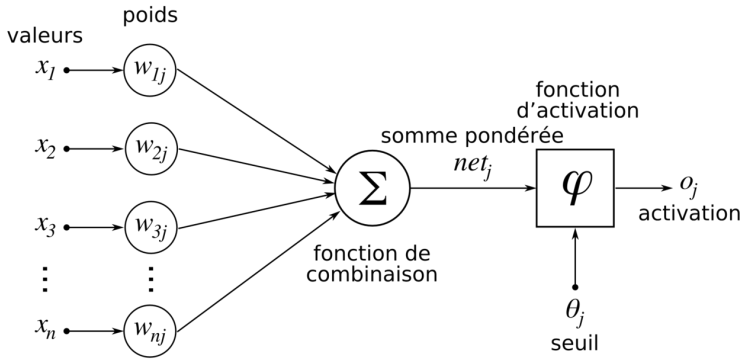

Image 10 : Illustration du fonctionnement d'un réseau de neurone.

4.2 Le principe d'entraînement du réseaux de neurone

Nous allons détailler le fonctionnement de notre réseaux. Dans un premier temps, il va falloir entraîner le réseau de neurones. La procédure d'entraînement se divise en 4 étapes. D'abord, on initialise le réseau de neurones avec la loi uniforme pour les poids et on initialise les biais à 0. La loi uniforme est la formule suivante, soit n le nombre de neurone de la couche à laquelle appartiennent les poids, tel que les poids appartiennent à l'intervalle suivant $[-y, y]$ avec $y = \frac{1}{\sqrt{n}}$

Ensuite, on propage les valeurs de la couche d'entrée. Pour calculer les valeurs d'une couche à une autre on utilise la formule suivante :

$$nYl = X\delta(xl - 1 * wl - \theta l)i = 1 \tag{4}$$

Ici l représente l'indice de la couche, n représente le nombre de neurone de la couche précédente et xi est la valeur du neurone i de la couche précédente. Le poids du lien est wi, le biais est ρ et enfin Ω est la fonction d'activation. La fonction d'activation que nous utilisons est la fonction sigmoïde, elle permet d'obtenir une valeur entre 0 et 1, peu importe la valeur d'entrée. Il suffit de multiplier la matrice des valeurs des neurones de la couche précédentes par la matrice des poids et d'ajouter la matrice des biais. Enfin on active le résultat avec la fonction sigmoïde. La troisième étape consiste à appliquer la backpropagation. Dans un premier temps, on calcule l'erreur de la couche de sortie, puis on la propage en arrière aux autres couches. La formule de l'erreur est la soustraction des valeurs attendues aux valeurs de la couche de sortie. Ensuite grâce à cette erreur, on peut calculer la correction des poids et des biais aussi appelées le gradient noté δ. Ainsi pour le calcul du gradient d'un poids on trouve la formule suivante :

$$\delta wil = xli * l$$

. La formule du gradient des biais est : $\delta\rho = \alpha\Delta$ Enfin, on applique un algorithme appelé gradient descent. Grâce au gradient on peut mettre à jour les poids et les biais. Mais pour éviter que le résultat des neurones oscillent, en corrigeant les poids avec des trop grandes valeurs, on ajoute une valeur. Un pas noté α. On obtient la formule suivante : $wl + 1 = \alpha * wl + \delta wliii$ Une fois le réseau de neurones entraînés il suffit d'entrer des valeurs, de les propager et de regarder la valeur des neurones de sorties.

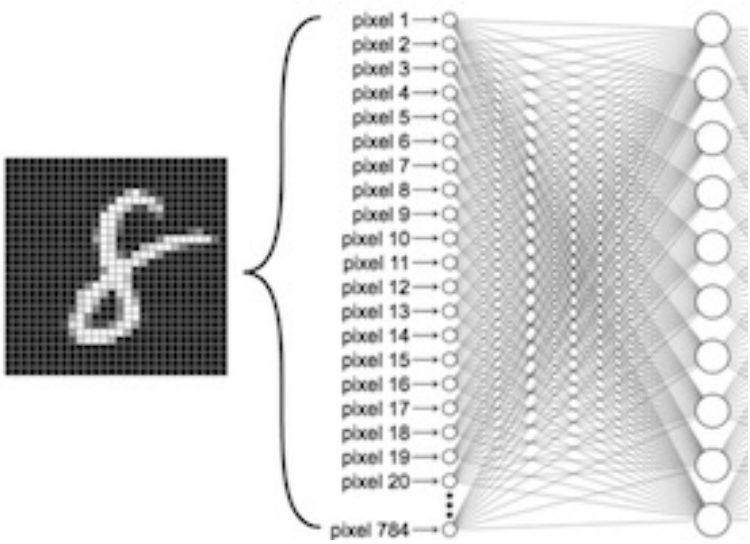

Image 11 : Principe d'entrainement d'un réseau de neurone

5 Conclusion

5.1 Bilan de la première soutenance

Tout d'abord, nous vérifions et, si nécessaire, corrigeons l'orientation de l'image en utilisant la rotation de l'image. Ensuite, nous convertissons l'image en niveaux de gris à l'aide du filtre de mise à niveau de gris, ce qui simplifie le traitement ultérieur. Pour améliorer la qualité de l'image et éliminer les imperfections, nous appliquons un filtre de réduction de bruit. Le flou gaussien est utilisé pour lisser l'image, réduire le bruit résiduel et améliorer la détection des contours. Nous poursuivons avec la binarisation de l'image, la transformant en une version binaire pour faciliter l'extraction des chiffres du Sudoku. Ensuite, nous utilisons le filtre de Canny pour détecter les contours, un élément essentiel pour extraire la grille du Sudoku et les chiffres qu'elle contient. Enfin, l'espace de Hough est employé pour détecter les lignes droites dans la grille du Sudoku, ce qui est crucial pour l'extraction précise des cases. Notre détection de lignes n'est pas encore finalisée nous arrivons à obtenir un espace de Hough mais nous travaillons pour en extraire des lignes.

Le solveur est capable de résoudre des Sudoku de manière efficace et précise. En parallèle, nous sommes également en train de développer un réseau de neurones, qui sera un atout majeur pour la reconnaissance des chiffres extraits de la grille du Sudoku. Ce réseau offre un potentiel d'amélioration significatif pour notre solution.

Dans l'ensemble, notre projet combine habilement des techniques de traitement d'image avancées, un solveur de Sudoku et le développement continu d'un réseau de neurones.

5.2 Objectifs seconde soutenance

Notre projet en informatique OCR est en constante évolution, avec un ensemble d'objectifs ambitieux à l'horizon. Au centre de nos priorités se trouvent plusieurs composantes clés, chacune destinée à renforcer notre capacité à résoudre des grilles de Sudoku à partir d'images. Tout d'abord, nous nous sommes engagés à finaliser la transformation de Hough. Cette technique de détection de lignes est essentielle pour notre projet, car elle permet d'identifier les lignes droites dans la grille du Sudoku. En perfectionnant cette étape, nous améliorerons la précision de l'extraction des cases, ce qui est fondamental pour une résolution précise du Sudoku. L'adaptation de notre algorithme de découpage de cases à la détection de lignes est une étape cruciale qui va de pair avec la transformation de Hough. Cette adaptation permettra une meilleure correspondance entre les limites des cases et les lignes détectées, assurant ainsi une correspondance optimale entre les cases de la grille et les chiffres qu'elles contiennent. Outre ces améliorations techniques, nous envisageons également d'ajouter un mode utilisateur. Ce mode permettra à l'utilisateur de prendre un

rôle actif dans le processus. Enfin, nous continuons à investir dans l'amélioration de notre réseau de neurones. Cette composante est cruciale pour la reconnaissance des chiffres extraits de la grille du Sudoku. Nous visons à optimiser les performances de notre réseau, à renforcer sa capacité à traiter une variété de polices et de styles d'écriture, et à améliorer la précision globale de la reconnaissance des chiffres. En somme, notre projet est en constante évolution, avec des objectifs clairs pour perfectionner chaque aspect du processus. La finalisation de la transformation de Hough, l'adaptation du découpage de cases, l'introduction d'un mode utilisateur et l'amélioration du réseau de neurones sont autant de jalons essentiels pour renforcer notre solution et la rendre plus performante, précise et conviviale. Nous sommes résolus à relever ces défis pour offrir une expérience OCR de Sudoku exceptionnelle.