OCR
Soutenance Finale

Julien BESTARD
Léa BONET
Hippolyte PIK
Benjamin DREYFUS

Contents

1 Introduction

1.1 Présentation du projet

Voici le rapport de soutenance pour notre projet de S3. Ce document est notre récit de bord pour l'OCR. Il présente les différentes étapes que nous avons réalisé pour vous présenter un OCR de qualité.

Notre groupe *Les Princesses* est composé de 4 étudiants en E1.

1.2 Le groupe du projet

- **Julien Bestard (Chef de Projet)**

  Julien est un génie de l'informatique originaire de la région 91, comme le célèbre groupe PNL, et mène une vie tranquille dans sa ville de campagne. Depuis le lycée, il est attiré par les nouvelles technologies et s'intéresse de plus en plus à ce domaine.
  Étudiant en 2ème année à EPITA, il est prêt à se lancer corps et âme dans ce nouveau projet. Grâce à ses compétences informatiques et son sens de l'humour inégalé, il est le garant du bon déroulement du projet et du maintien d'une bonne ambiance entre nous. Julien est une valeur sûre pour ce groupe !

- **Hippolyte Pik**

  Hippolyte Pik est un élève de la E1. C'est la première fois qu'il développe un réseau de neurones c'est donc une totale découverte à faire. Passionné tout particulièrement par l'intelligence artificielle ce projet est pour lui une super opportunité pour la suite de son projet de vie.

- **Léa Bonet**

  Léa Bonet est une étudiante en deuxième année à l'EPITA. Elle a beaucoup aimé faire le projet de S2 car il lui a appris beaucoup de compétences en C# mais également à travailler en groupe. Léa avait hâte de démarrer ce projet pour renouveler cette expérience. Elle adore résoudre des sudokus et s'est donc occupée de la partie solver.

- **Benjamin Dreyfus**

  Passionné de logique et de programmation, Benjamin (lui aussi élève de la très bonne classe E1) est touche-à-tout et s'intéresse à toute sorte de programme informatique. Le projet de S2 étant une expérience plutôt satisfaisante pour lui, il est prêt à mener un autre projet de ce genre à terme. Dans celui-ci, il s'occupera principalement de l'interface graphique.

2 Répartition des tâches

Tasks	Julien	Léa	Hippolyte	Benjamin
Traitement d'image				
Détection de la grille				
Sudoku Solver				
Interface				
Réseau de Neurones				

3 Réseau de neurones

3.1 Structure globale

3.1.1 Front Propagation

La propagation vers l'avant (ou en anglais *front propagation*) est une méthode dite de base dans la résolution du problème du réseau de neurones.

C'est une méthode où l'on ne se déplace que dans un seul sens, des neurones d'entrée en passant par les neurones cachés (le cas échéant) et puis, vers la sortie. Cette méthode consiste à envoyer dans chaque neurone de la première couche chacune des entrées du problèmes.

La propagation vers l'avant se calcule à l'aide d'une fonction d'activation notée $g^{(n)}$ (l'indice n indique la couche sur laquelle le neurone se situe, ici la n-ème), et d'une fonction d'agrégation notée ici $h_j^{(n)}$ (souvent un produit scalaire entre les poids et les entrées du neurone) (l'indice j indique le numéro du neurone sur la couche, ici le j-ème) et des poids synaptiques $\vec{w}_{jk}$ entre le neurone $x_k^{(n-1)}$ et le neurone $x_j^{(n)}$.

La notation est alors inversée : $\vec{w}_{jk}$ indique bien un poids de k vers j.

$x_j^{(n)} = g^{(n)}(h_j^{(n)}) = g^{(n)}\Big(\sum_k w_{jk}^{(n)} x_k^{(n-1)}\Big)$

Ainsi avec cette méthode on obtient finalement sur le(s) dernier(s) neurone(s) le(s) résultat(s) attendu(s).

3.1.2 Back Propagation

La propagation vers l'arrière (en anglais *back propagation*) est une méthode où l'on cherche à faire apprendre notre réseau de neurones. Cette méthode comporte ce que ne comporte pas la première méthode présentée, des boucles et des retours en arrières. Elle prend comme base la méthode présentée au dessus. Nous continuons donc le processus après avoir obtenu un résultat noté $\vec{y}$.

On calcule alors l'erreur entre la sortie donnée par le réseau $\vec{y}$ et le vecteur $\vec{t}$ désiré à la sortie pour cet échantillon. Pour chaque neurone i dans la couche de sortie, on calcule (g' étant la dérivée de g): $e_i^{\text{sortie}} = g'(h_i^{\text{sortie}})(y_i - t_i)$

Ensuite on propage l'erreur vers l'arrière $e_i^{(n)} \mapsto e_j^{(n-1)} e_i^{(n)} \mapsto e_j^{(n-1)}$ grâce à la formule suivante : $e_j^{(n-1)} = g'^{(n-1)}(h_j^{(n-1)}) \sum_i w_{ij}^{(n)} e_i^{(n)}$

On note qu'entre la dernière couche et l'avant dernière $e_i = e_i^{\text{sortie}}$

Et finalement on met à jour les poids dans toutes les couches : $w_{ij}^{(l)} = w_{ij}^{(l)} - \lambda e_i^{(l)} x_j^{(l-1)}$ où λ représente le taux d'apprentissage (de faible magnitude et compris entre 0,0 et 1,0).

3.2 Le xor

Le xor que l'on a implémenté utilise la méthode de back propagation. Pour ce faire nous avons créé un type *Neuron* auquel on a assigné un tableau d'entrée, un tableau de poids, un coefficient d'erreur, et un biais.

La première étape est d'assigner à chacun de nos neurones un poids aléatoire. Cela étant fait, on met à chacune de nos entrées de chacun de nos neurones les entrées de notre programme.

On calcule ensuite la sortie de notre neurone actuel, à l'aide d'une fonction sigmoïde ce qui donne: $\frac{1}{1+e^{\sum w_i x_i}}$ que l'on insère directement dans les entrées des neurones de sortie. Nous calculons finalement le résultat à l'aide de la même fonction sigmoïde.

Une fois le résultat obtenu, nous calculons le coefficient d'erreur de la sortie grâce à la formule: $\frac{y}{1-y}(t-y)$ avec y le résultat obtenu et t le résultat attendu.

On ajuste ensuite les poids et le biais avec la formule: $w_i = w_i + e_i x_i$ et $biais = biais + e_i$. Une fois la sortie ajustée ne reste plus qu'à ajuster les erreurs, les biais et les poids de chacun des neurones restants.

Les coefficients d'erreur sont modifiés avec la formule: $e_i = \frac{y_i}{1-y_i} e^{\text{sortie}} w_i$, cependant les poids et les biais ont la même formule que le neurone de sortie. Voilà, l'explication en détail de notre xor est terminée.

3.3 Reconnaissance d'images

La reconnaissance d'images est l'un des pôles essentiels à la réalisation de ce projet. En effet il permet de lier la partie de Julien "Traitement d'images" à celle de Léa "Résolveur de sudoku". Cette partie se résume à la création d'un réseau de neurone qui est capable à partir d'une image de reconnaître un chiffre. Pour ce faire j'ai implémenté plusieurs méthodes :

3.3.1 La fonction d'initialisation : Xavier initialisation

Cette méthode vient de Xavier Glorot, chercheur en intelligence artificielle chez Google. Cette méthode permet de ne pas mettre n'importe quel nombre aléatoire, elle permet de choisir un intervalle pour lequel les fonctions TanH ou Sigmoïde sont très efficace. Mais au delà de son efficacité, cette méthode permet à notre fonction d'apprentissage d'être plus performante et ainsi d'avoir des résultat suffisant. L'intervalle choisi dépend du nombre d'entrées de notre réseau de neurones et correspond à :

$[-\frac{1}{\sqrt{i}}, \frac{1}{\sqrt{i}}]$ où i correspond au nombre d'entrées

3.3.2 La fonction d'activation : TanH et SoftMax

La fonction d'activation TanH ou tangente hyperbolique est une méthode qui ressemble à la fonction sigmoïde à la différence près que la tangente hyperbolique a une courbe plus raide à l'approche du 0. L'utilisation de TanH est d'autant plus intéressante que le réseau de neurone que l'on crée est ce que l'on appelle un réseau de neurones récurrents.

3.3.3 La fonction d'apprentissage: Gradient Descent Stochastique

Le gradient Descent stochastique est une méthode dérivée du Gradient Descent qui se différencie grâce à sa pertinence dans l'utilisation du réseau de neurone avec de grandes quantités d'échantillon.

3.3.4 Reconnaissance d'images

Maintenant que les explications des fonctions sont fournies, place à l'explication du code:

Tout d'abord on initialise l'espace mémoire pour l'image que l'on stocke dans un pointeur de pointeur. Ensuite on initialise le réseau de neurone. Dans notre cas les neurones seront stockés dans un tableau par leur poids ainsi que leurs erreurs et leurs biais. On crée pour chacun des 3 tableaux un espace mémoire, et ensuite la fonction Xavier pour l'initialisation.

Ensuite vient la partie de la front propagation. Mais avant cela on ajoute dans l'espace mémoire créé avant, la première image issue du travail de Julien. On insère à tous les 784 neurones de la première couche les valeurs r,g,b de chacun des pixels divisés par 3. Ensuite on calcule les niveaux d'après à l'aide des niveaux précédent et de la fonction TanH.

Ensuite vient la phase de calculer le dernier niveau. Pour ce faire on utilise la fonction softmax.

On affiche le résultat en fonction de la valeur de softmax la plus grande.

Si on veut le faire apprendre, on utilise ensuite la fonction de back propagation qui met à jour les poids. On cacule d'abord l'erreur du dernier niveau puis on calcule l'erreur des niveaux précédents. Enfin on met à jour les poids de chacun des niveaux.

4 Traitement de l'image

Notre traitement d'image n'est pas parfait, mais il nous permet d'obtenir de bons résultats. Il y a actuellement 4 étapes :

1. Réduction de la taille
2. Gamma
3. Contraste
4. Le niveau de gris
5. Le flou Gaussien
6. La méthode d'Otsu

Prenons l'image ci-dessous comme exemple :

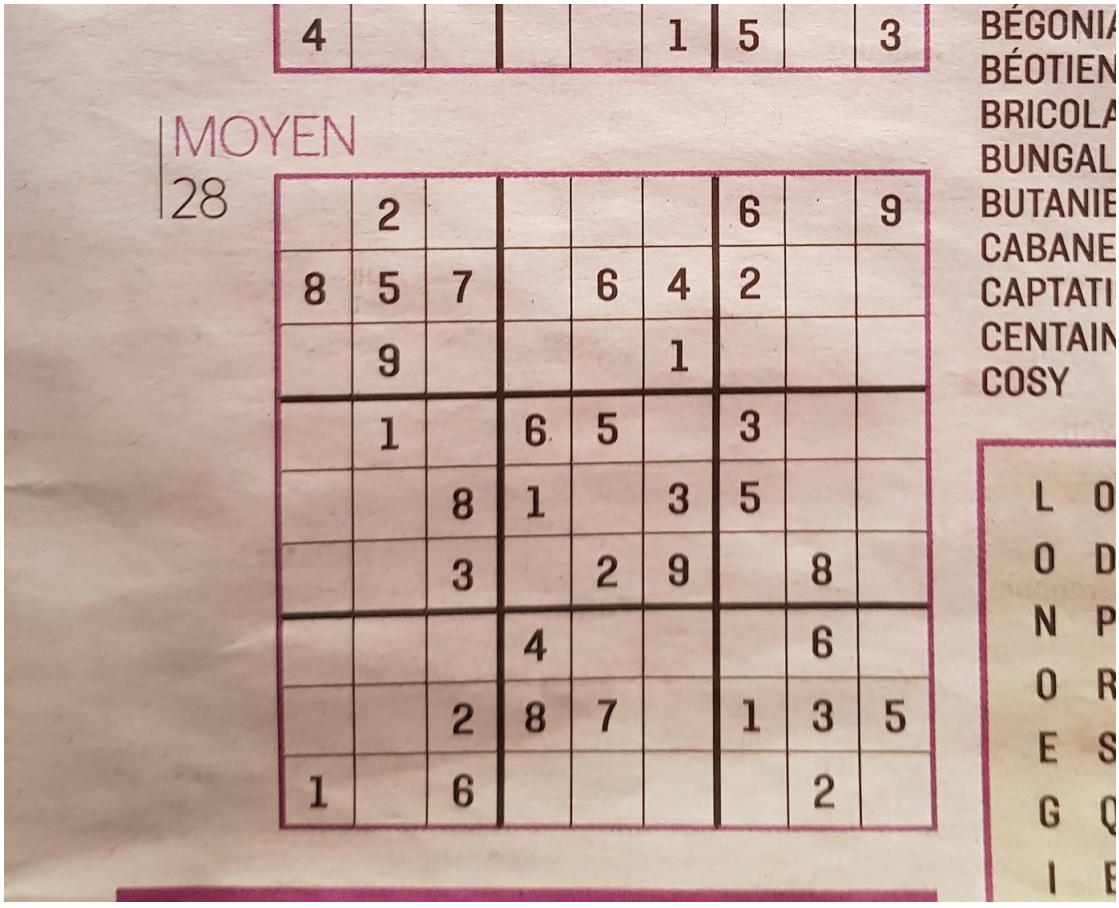


Figure 1: Image 04

4.1 Réduction de la taille

Si une image est trop grande, nous réduisons la taille de celle-ci pour le temps de traitement de celle-ci.

4.2 Ajustement des couleurs

4.2.1 Gamma

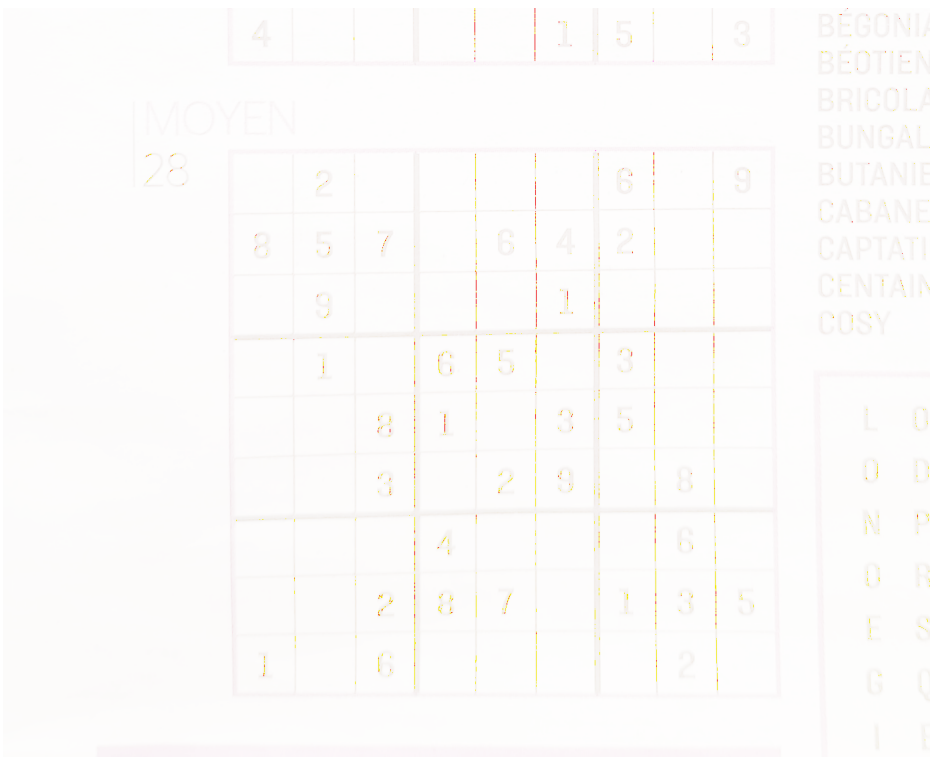


Figure 2: Après l'ajustement du gamma

Après avoir éliminé le bruit, nous appliquons un filtre gamma. Ce filtre, combiné aux précédents, nous permet d'avoir une réduction de bruit efficace, améliorant donc la détection des bords. Le gamma définit la relation entre la valeur des couleurs d'un pixel et sa luminosité actuelle.
Nous calculons sa valeur grâce à cette formule : $nC = 255 * \frac{C}{255}^{\gamma}$, avec ici $\gamma = \frac{2}{5}$ et c la valeur du rouge ou vert ou bleu.

4.2.2 Contrast

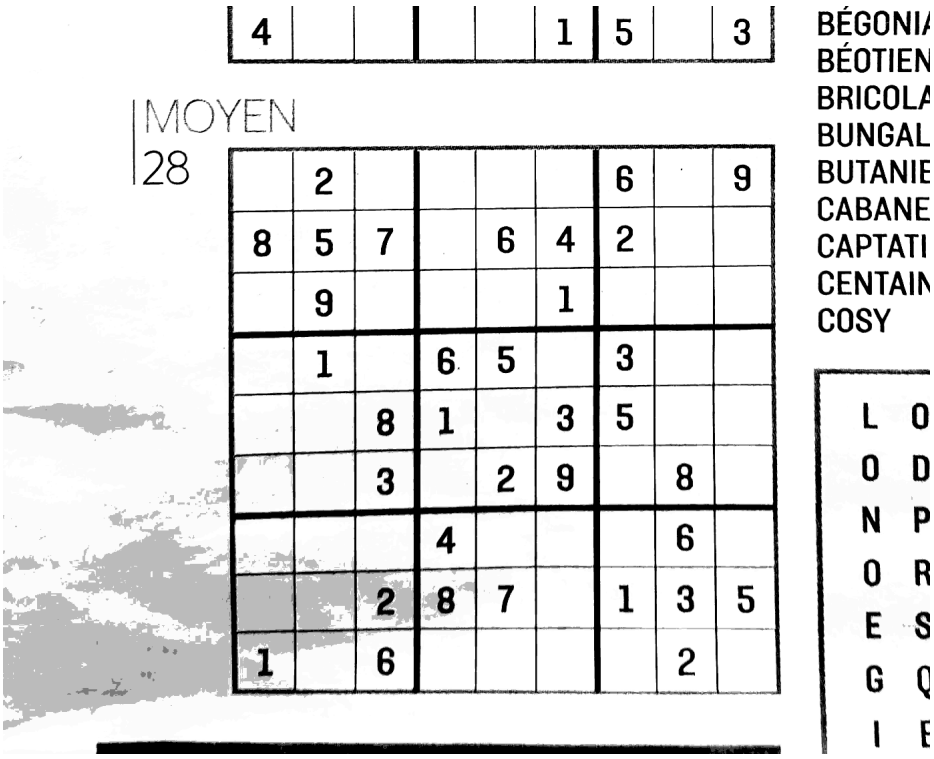


Figure 3: Après l'ajustement du contrast

Nous avons utilisé le contraste ici pour augmenter la lisibilité de la date sur l'image et faire ressortir les contours. Nous utilisons une formule mathématique pour calculer la correction du contraste : $F = \frac{259 \cdot (C+255)}{255 \cdot (C-259)}$

Le C est le facteur du contraste souhaité . Il sera utilisé pour modifier la valeur du rouge/bleu/vert avec cette formule : $Color = F \cdot (Color - C) + C$

La combinaison de ces deux procédés nous permet d'obtenir une image plus détaillée, et avec un peu moins de bruit.

4.2.3 Niveau de gris

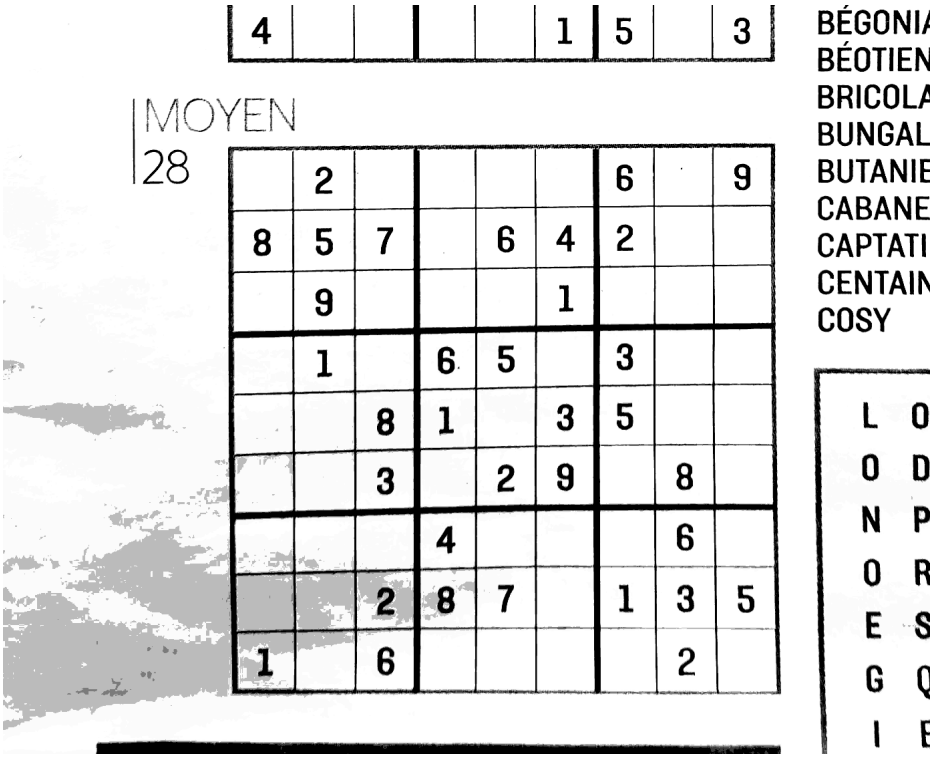


Figure 4: Après le niveau de gris

Le niveau de gris est la première étape de notre traitement d'image. De ce fait, nous n'aurons plus qu'à traiter un seul taux de couleur, et plus du RGB.
Pour le faire on applique cette fonction : $gris = 0.3 * r + 0.59 * g + 0.11 * b$.
La valeur "gris" remplacera chaque valeur de notre RGB pour le pixel actuel, qui deviendra donc gris.
Nous utilisons la méthode pondérée car nous sommes sensibles aux lumières dans cet ordre : vert, rouge, bleu.

4.2.4 Gaussian Blur

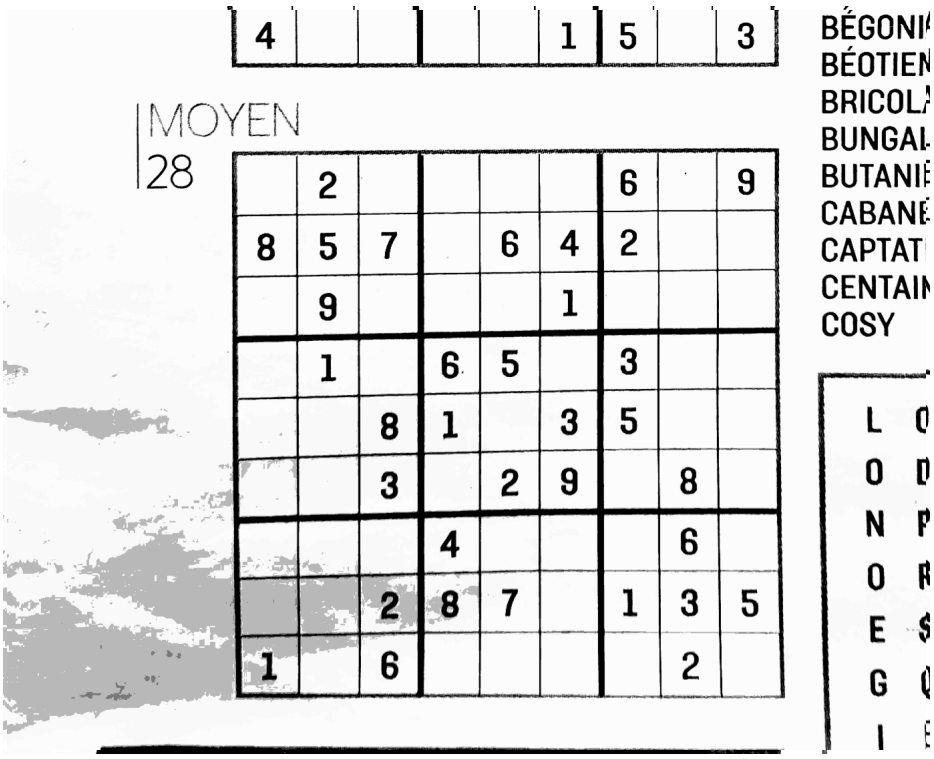


Figure 5: Après le flou

Après avoir fait le niveau de gris, nous appliquons le flou Gaussien, aussi connu sous le nom de lissage gaussien. Le flou Gaussien, qui tire son nom du mathématicien Carl Friedrich Gauss, est l'application d'une fonction mathématique à une image pour la flouter. Ce filtre de type passe-bas, lisse les valeurs inégales des pixels d'une image en supprimant les valeurs aberrantes extrêmes.
Ce filtre permet, pour une photo prise avec une luminosité faible et qui présente beaucoup de bruit, d'atténuer ceux-ci. Si vous voulez ajouter du texte sur une image, un flou gaussien permettra d'adoucir celle-ci pour le faire ressortir avec plus de netteté.
Ici il nous est particulièrement utile pour éliminer le bruit de l'image et l'adoucir un peu.

Pour calculer la transformation à appliquer à chaque pixel de l'image nous utilisons la formule pour une image à 2 dimensions : $G(x,y) = \frac{1}{2\cdot\pi\cdot\sigma^2} e^{-\frac{x^2+y^2}{2\cdot\sigma^2}}$

Où x est la distance par rapport à l'origine sur l'axe horizontal, y est la distance par rapport à l'origine sur l'axe vertical, et σ est l'écart type de la distribution gaussienne. Les indices de cette distribution sont utilisés pour construire une matrice de convolution qui est appliquée à l'image originale.

Ce processus de convolution est illustré visuellement dans la figure de droite. La nouvelle valeur de chaque pixel est définie comme une moyenne pondérée du voisinage de ce pixel. La valeur du pixel d'origine reçoit la pondération la plus forte (ayant la valeur gaussienne la plus élevée) et les pixels voisins reçoivent des pondérations plus faibles à mesure que leur distance au pixel d'origine augmente. Le résultat est un flou qui préserve mieux les limites et les bords que d'autres filtres de flou plus uniformes.

4.2.5 Méthode d'Otsu

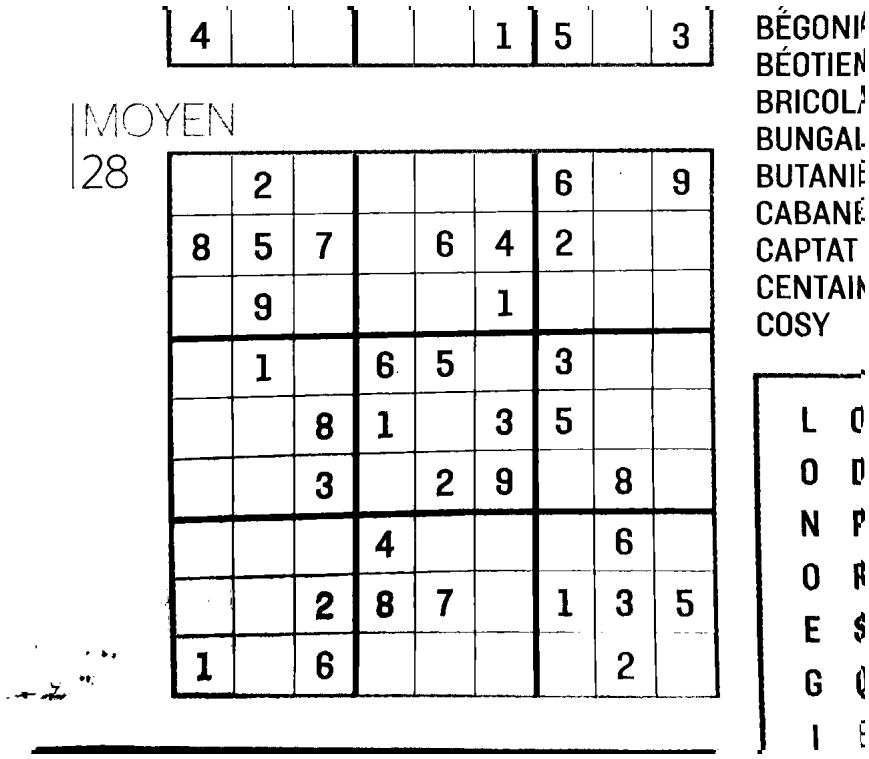


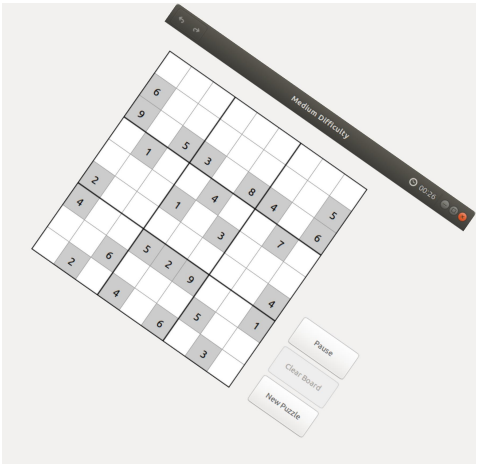
Figure 6: Après OTSU

Afin de transformer notre image en noir et blanc, nous utilisons la méthode d'Otsu. La méthode d'Otsu est utilisée pour effectuer un seuillage automatique à partir de la forme de l'histogramme de l'image, ou la réduction d'une image à niveaux de gris en une image binaire. L'algorithme suppose alors que l'image à binariser ne contient que deux classes de pixels, (c'est-à-dire le premier plan et l'arrière-plan). L'algorithme s'exécute de cette façon : il recherche itérativement le seuil qui minimise la variance intra-classe, définie comme la somme pondérée des variances de deux classes (arrière-plan et premier plan). Les couleurs en niveaux de gris sont généralement comprises entre 0 et 255. Ainsi, si nous choisissons un seuil de 150, tous les pixels dont la valeur est inférieure à 150 deviennent l'arrière-plan et tous les pixels dont la valeur est supérieure ou égale à 150 deviennent le premier plan.

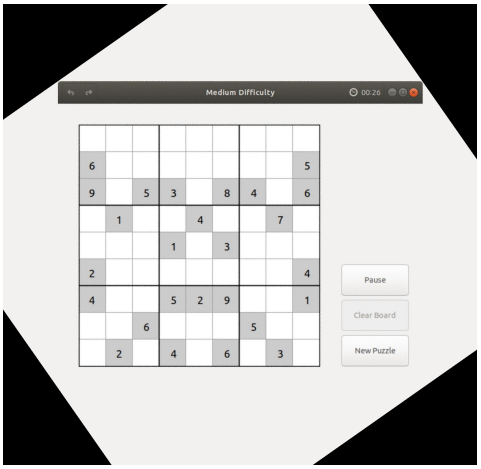
4.3 Rotation de l'image

Pour faire pivoter une image, nous avons décidé d'utiliser une matrice de rotation. Cette méthode est rapide et précise, elle nous donne des résultats sans bords irréguliers et nous permet d'effectuer des rotations avec des angles très précis. La matrice de rotation s'exprime par les formules suivantes pour trouver les coordonnées des pixels tournés (x, y) par l'angle θ sur l'image originale (x', y')

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \cdot \cos\theta - y \cdot \sin\theta \\ x \cdot \sin\theta + y \cdot \cos\theta \end{pmatrix}$$



(a) Avant rotation



(b) Après rotation de 35

5 Segmentation de la grille

Une fois l'image traitée, il faut détecter la grille du sudoku et découper la grille obtenu pour récupérer chaque cellule. On pourra ensuite, grâce au réseau de neurones, détecter chaque chiffre de la grille.

5.1 L'opérateur Sobel

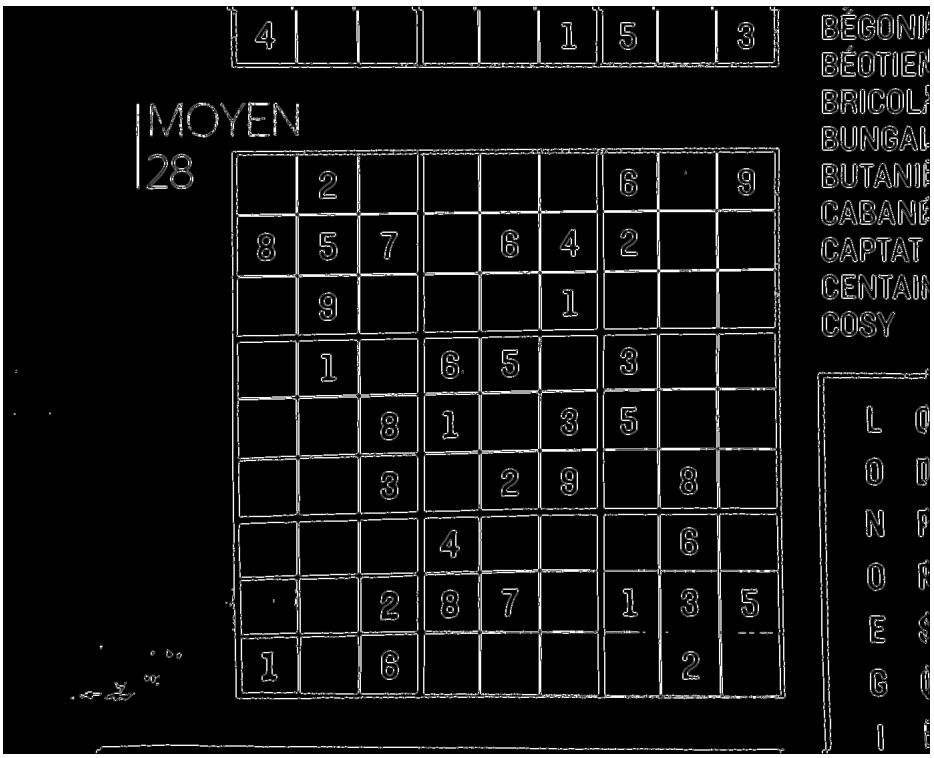


Figure 8: Après l'opérateur Sobel

Le première chose que nous ferons pour la détection de la grille sera d'utiliser l'opérateur Sobel. L'opérateur de Sobel utilise deux noyaux (un pour chaque direction) :

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ et } K_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

On calcule la convolution entre l'image (convertie en noir et blanc) et les deux noyaux séparément. Cela nous donne, pour chaque pixel, les valeurs mag_x et mag_y. La valeur du pixel courant est fixée à $\sqrt{(mag_x)^2 + (mag_y)^2}$. L'opérateur nous permet de faire ressortir les lignes de l'image, permettant ensuite à notre algorithme de détection de la grille de mieux les détecter.

5.2 Transformée de Hough

La transformée de Hough permet de détecter les formes d'une image. Pour le sudoku, nous cherchons à détecter la grille, nous appliquons donc cette technique pour détecter la grille. Le principe est de parcourir notre image et de stocker les valeurs associées grâce à des formules et des matrices.

La première chose à faire est de construire un accumulateur de $\rho \times \theta$. Les valeurs de cette accumulateur seront initialisées à 0.

θ correspond à l'angle d'étude pour le pixel. Étant donné que l'on cherche des lignes, θ prend ses valeurs entre $-90°$ et $90°$.

ρ correspond à la distance entre l'ordonnée à l'origine de l'image et la droite d'angle θ passant par le pixel étudié. Il prends ses valeurs dans l'intervalle $(-diag\,, diag)$. $diag$ est donné par $\sqrt{longueur_{image} + largeur_{image}}$.

Si le pixel étudié est blanc (de la couleur de notre grille après sobel), le calcul de ρ pour tout les valeurs de θ est donné par :

$\rho = x \cdot \cos\theta + y \cdot \sin\theta.$

Ainsi pour chaque valeur de ρ et θ, on incrémente de 1 la valeur aux coordonnées (ρ,θ) dans l'accumulateur.

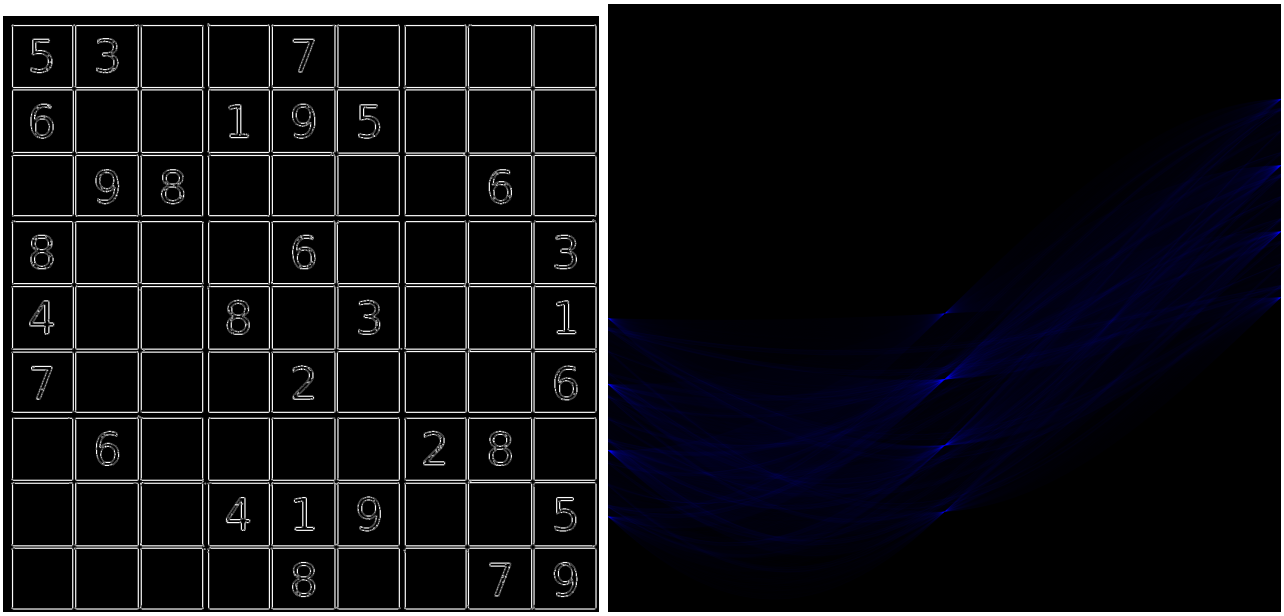


Figure 9: Image du sudoku et sa transformée de hough

Une fois chaque pixel étudié, on parcourt notre accumulateur. Si les valeurs dépassent un certain seuil, on considère qu'il s'agit d'une ligne possible. On récupère alors 2 couples de valeurs $(x_1\,, y_1)$ et $(x_2\,, y_2)$ tels que l'on retrouve les valeur du ρ étudié pour le θ associé. On prend les couples de valeurs les plus éloignés pour obtenir la ligne la plus exacte possible.

On cherche ensuite le plus grand carré. Pour cela, on cherche d'abord les intersections des lignes. On définit les angles minimal et maximal entre les lignes pour ne pas chercher une ligne presque identiques. On construit les vecteurs de chaque droite avec les couples $(x_1\,, y_1)$ et $(x_2\,, y_2)$ utilisés pour construire les droites. Pour l'étude de l'angle entre deux droites $d1$ et $d2$, on posera les vecteurs u et v tels que :

$$u_x = d1_{x2} - d1_{x1} \text{ et } u_y = d1_{y2} - d1_{y1}$$

$$v_x = d2_{x2} - d2_{x1} \text{ et } v_y = d2_{y2} - d2_{y1}$$

L'angle entre $d1$ et $d2$ est donné par :

$$\alpha = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \times \|\vec{v}\|}$$

On vérifie que l'angle soit dans notre intervalle. Si c'est le cas on garde les coordonnées. Une fois fini, on garde seulement les points qui ne se ressemble pas. On cherche ensuite les points qui forment les plus carré afin de trouver notre grille. Une fois ces points trouvés, on découpe l'image selon ces points.

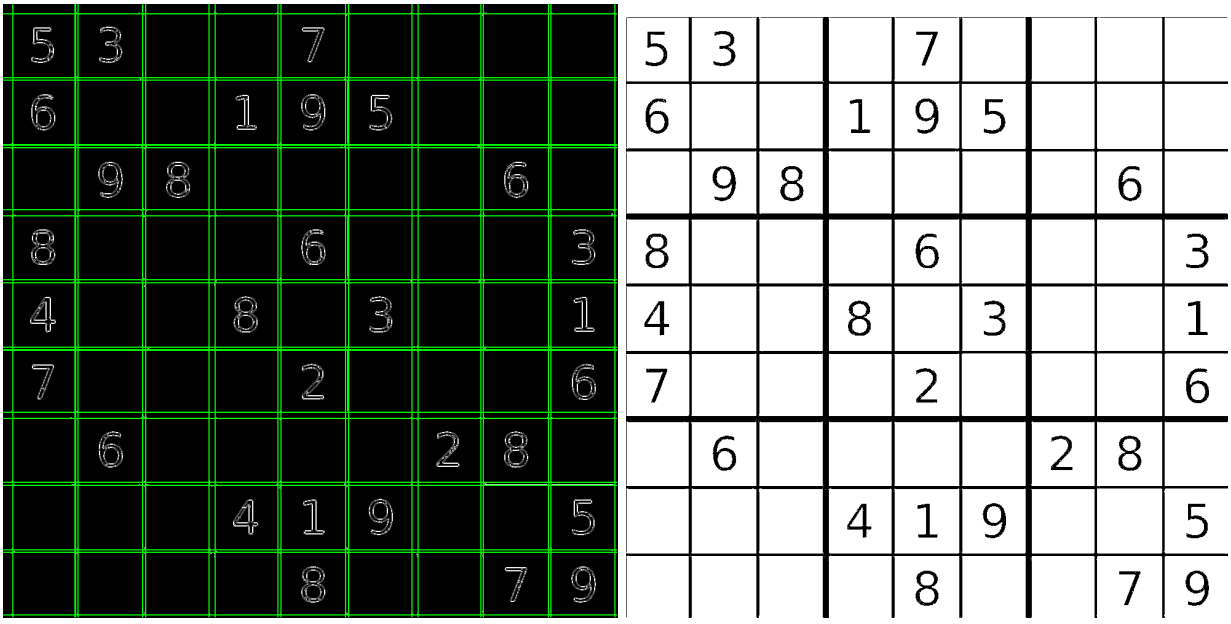


Figure 10: Lignes détectées et image découpée

5.3 Segmentation de la grille

Pour pouvoir détecter chaque chiffre de notre grille, il est indispensable de fournir chaque cellule du sudoku. Pour cela, nous réalisons une segmentation de la grille, on récupère alors 81 cases.

Pour fournir des données exploitables par le réseau, on procède d'abord à un nettoyage de l'image. Une fois le nettoyage fait, on redimensionne l'image en un carré de 28 pixels.

Figure 11: Case de la grille en 28x28 pixels

6 Résolution d'une grille

6.1 Le Solveur

6.1.1 Le principe

Pour résoudre une grille de sudoku on va utiliser la méthode du *backtracking* (retour sur trace). Cette méthode consiste à revenir en arrière sur les décisions prises afin de sortir d'un blocage. Dans le cas du sudoku il s'agit de tester toutes les valeurs pour chaque case vide en vérifiant si la grille reste valide. Dès qu'on arrive sur une valeur où la grille n'est pas valide on revient en arrière. Pour rappel, chaque lignes, colonnes et carrées de 3x3 doit contenir un chiffre entre 1 et 9 mais ne doit pas contenir deux fois la même valeur.

Dans le programme, nous allons d'abord vérifier que la valeur que l'on souhaite ajouter n'est pas déjà présente sur la ligne, la colonne et le carré de 3x3. Puis dans la fonction principale, on va énumérer les valeurs possibles et les tester. On va appeler notre fonction récursive pour pouvoir retourner en arrière si la valeur est incorrecte.

6.1.2 Utilisation du programme

L'utilisation du solveur est simple, celui-ci prend en paramètre un fichier contenant une grille disposée sous cette forme (les points représentant les cases à combler):

Figure 12: Gille 00 non résolue

Une fois fini, le programme écrit dans un fichier <nom_de_la_grille>.result, la grille résolue.

7 Rendu visuel

7.1 Traitement et affichage des données

Une fois que nous avons reçu les données du réseau de neurones, nous devons les traiter. On a donc fait le choix d'écrire les nombres détectés en utilisant le même affichage que pour le solver afin de faciliter l'utilisation.

Nous avons à notre disposition une grille vide ainsi que des nombres ,dans une police de notre choix.

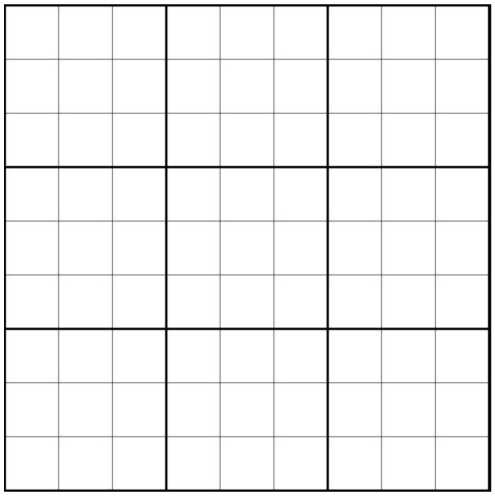


Figure 13: Modèle de la grille vide

Afin de reconstruire la grille résolue, on la compare avec la grille initiale.

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

Figure 14: Modèle de la grille initiale

Nous avons donc la grille résolue. On peut voir en rouge les nombres trouvés grâce au solver ultra puissant et optimisé et en noir la grille initiale (qui correspond à *image_03.jpg*)

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

Figure 15: Modèle de la grille résolue

8 Interface graphique

8.1 Logo

Pour commencer l'interface graphique, il nous a été nécessaire de concevoir un logo pour le groupe. Un symbole simple, qui nous représente. Quoi de plus approprié pour représenter le groupes des Princesses qu'un diadème (évidemment accompagné d'un symbole qui rappelle le sudoku) ?

Nous avons choisi de faire ce logo en pixel art. Étant donné nos capacités de dessin, c'était le plus simple à entreprendre. Le logo a donc été réalisé à l'aide du logiciel de dessin spécialisé en pixel art Aseprite.

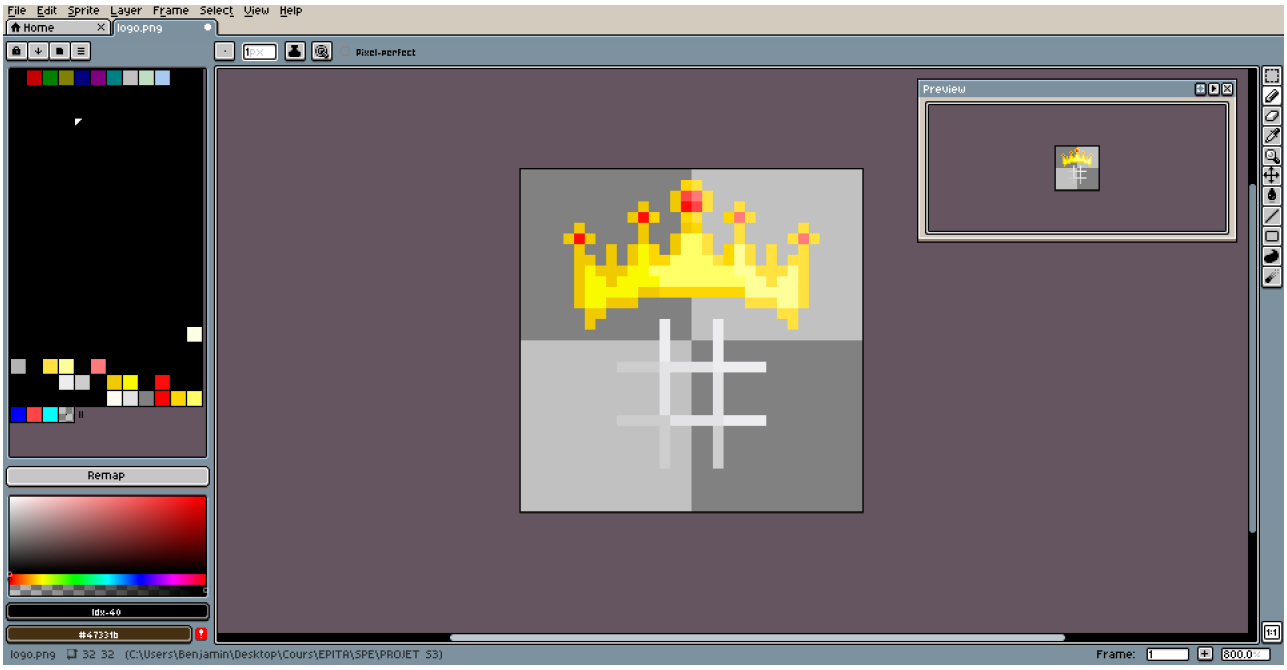


Figure 16: La création du logo dans le logiciel Aseprite

8.2 GTK

Le choix de la bibliothèque GTK s'est imposé par sa praticité et sa disponibilité sur les machines de l'école.

8.3 Glade

Pour simplifier la création de l'interface graphique, le logiciel Glade a été utilisé en complément de la bibliothèque GTK. Ce logiciel permet d'assembler une base d'interface graphique très facilement, à l'aide de sa propre interface graphique très pratique, permettant de faire glisser des "Widgets" (des composantes de l'interface graphique) à l'endroit souhaité.

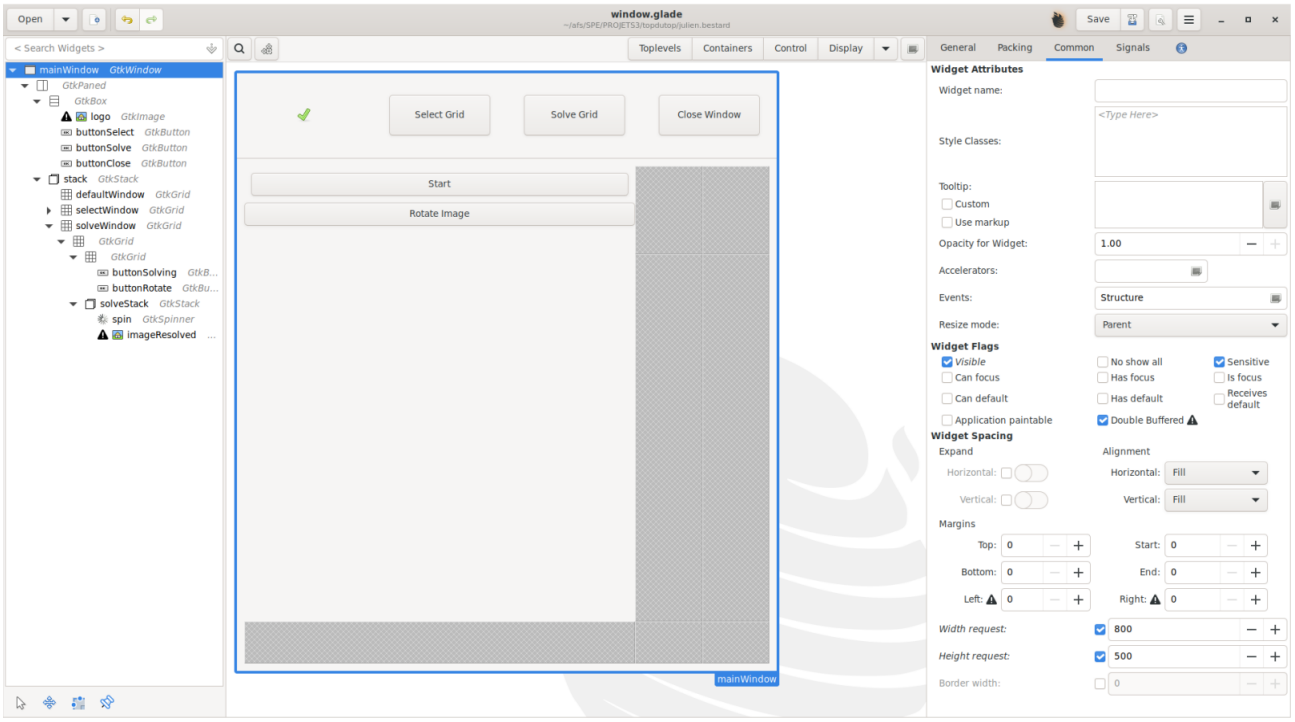


Figure 17: Le logiciel de conception d'interface graphique Glade

8.4 Conception de l'interface graphique

Maintenant, nous rentrerons un peu plus en détail sur la conception de l'interface graphique et ses fonctionnalités.

8.4.1 Structure de la fenêtre

L'application présente une "barre des tâches" sur son extrémité supérieure. Celle-ci comporte le logo de notre groupe et de l'application à sa gauche, puis le bouton d'ouverture de l'onglet de choix de la grille, puis le bouton d'ouverture de l'onglet de résolution de la grille, et enfin le bouton de fermeture de l'application. En haut, nous avons donc la barre des tâches, et en bas, l'emplacement pour les onglets de l'application.

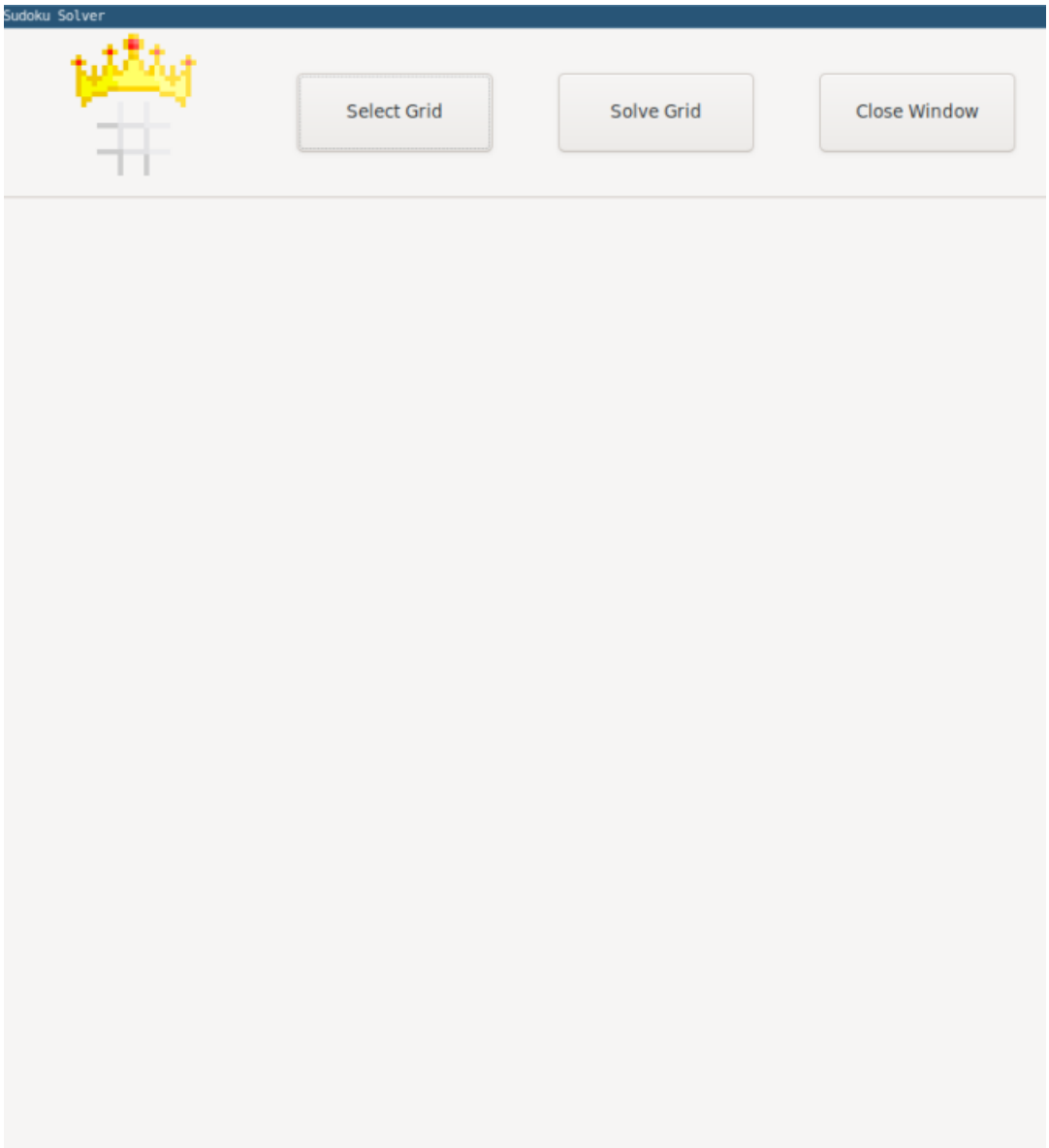


Figure 18: La structure de l'interface graphique

8.4.2 Onglet "Select Grid"

Lorsqu'on clique sur le bouton "Select Grid", un onglet apparaît en transition de par la droite de la fenêtre. Cet onglet dévoile un bouton accompagnée de l'instruction "Please select a grid".

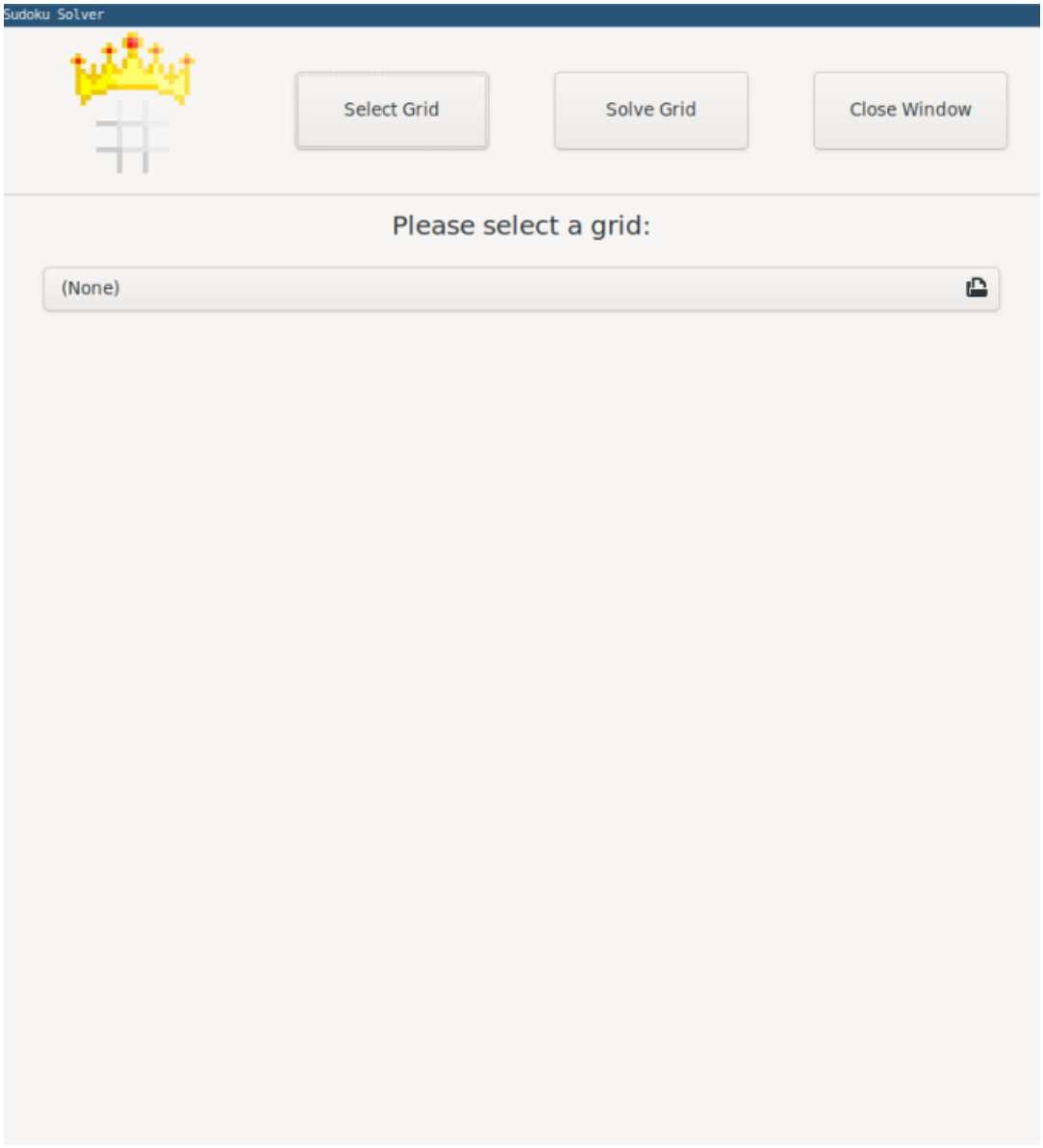


Figure 19: L'onglet "Select Grid"

Quand on clique sur ce bouton, une interface de choix de fichier apparaît. Elle nous permet de sélectionner n'importe quelle image stockée dans l'ordinateur. Pour la démonstration, nous choisirons la grille donnée s'appelant "image_01.jpeg". Ce menu d'ouverture de fichier est implémenté à l'aide du widget "FileChooser".

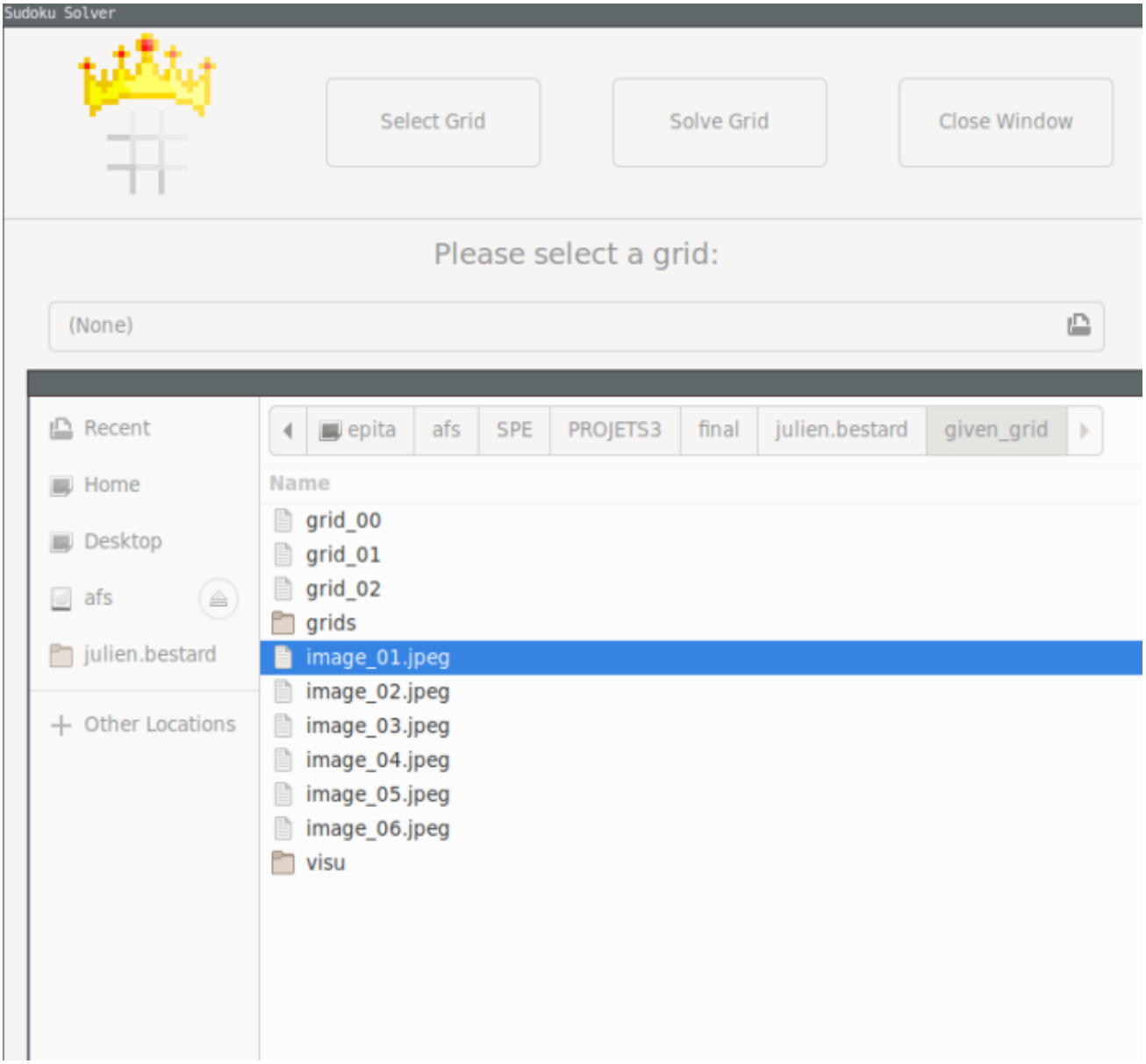


Figure 20: Choix de la grille dans l'interface

Lorsque l'image est sélectionnée, la grille est automatiquement affichée sur l'application, permettant de vérifier que la grille sélectionnée est bien celle que l'on souhaite effectivement sélectionnée. Ici, l'image "image_01.jpeg" est donc sélectionnée et affichée.

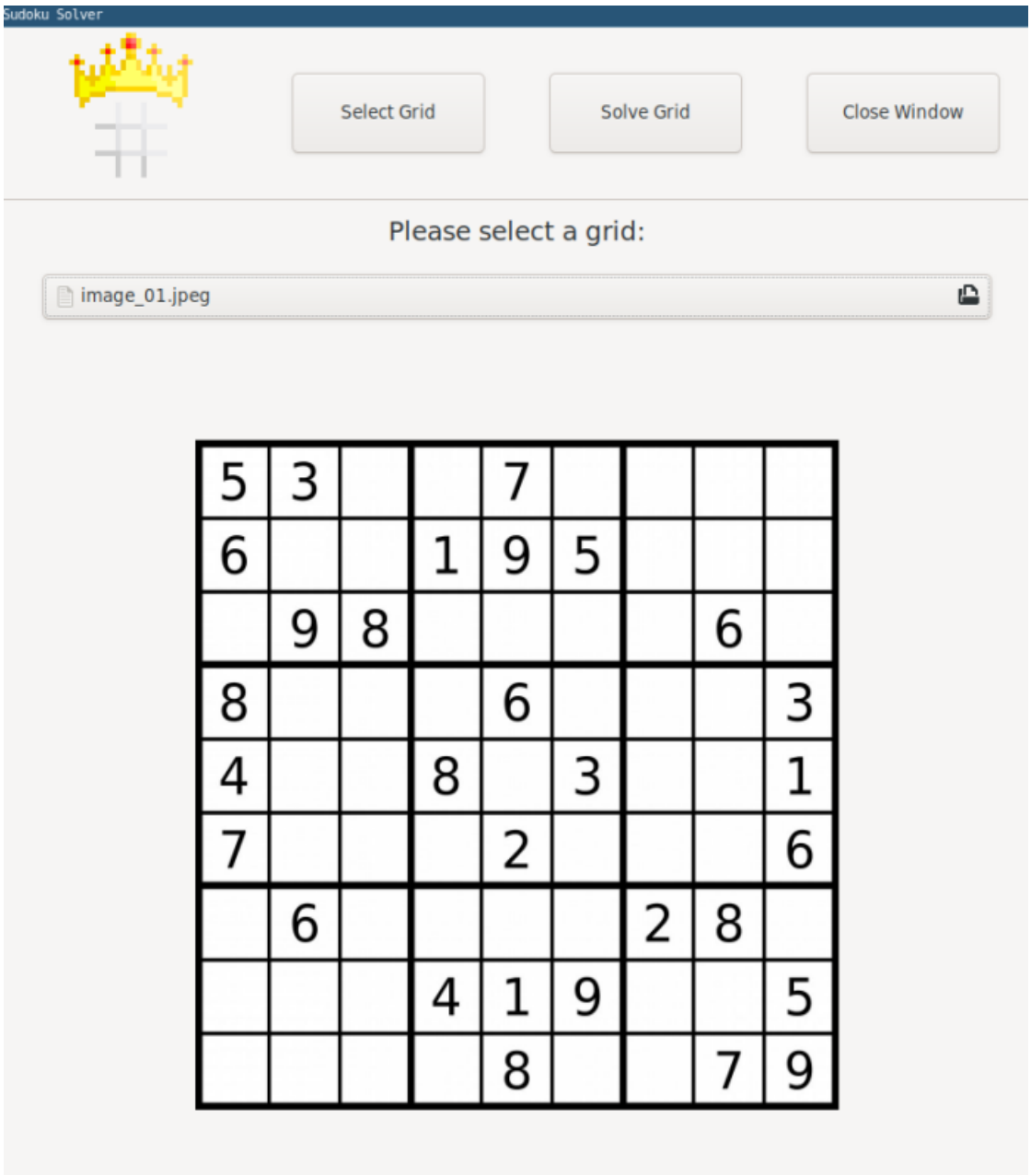


Figure 21: Affichage de la grille lorsqu'on l'a sélectionné

8.4.3 Onglet "Solve Grid"

Maintenant qu'une grille est sélectionnée, il est possible de sélectionner l'onglet "Solve Grid", laissant apparaître, à nouveau avec une transition venant de la droite de l'image, un nouvel onglet, comportant deux boutons, un bouton "Start", et un bouton "Rotate Image".

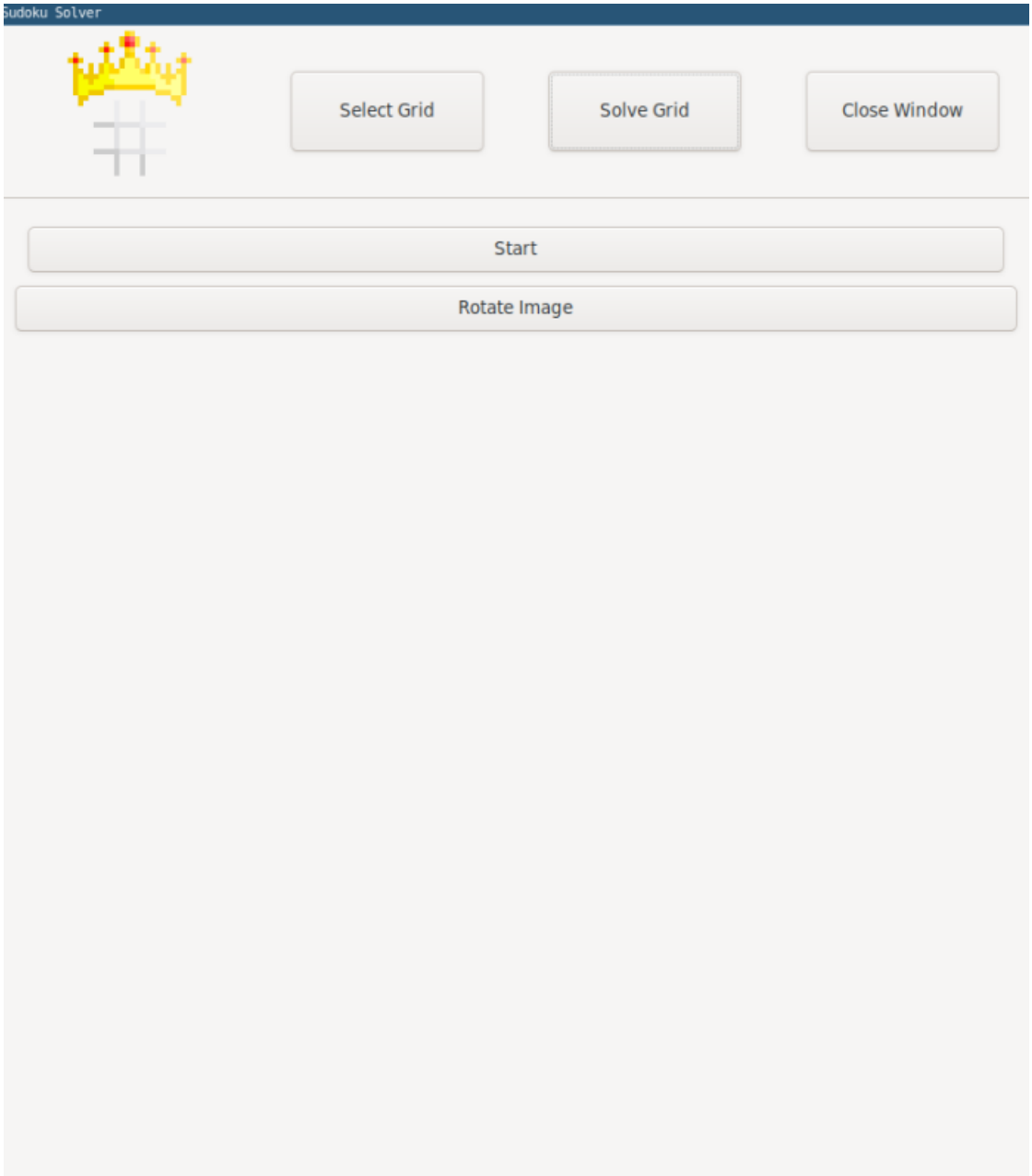


Figure 22: L'onglet "Solve Grid"

Lorsqu'on appuie sur le bouton "Start", il y a un petit temps d'attente, puis la grille résolue s'affiche sur l'écran. À noter que, si aucune grille n'est sélectionnée dans l'onglet "Select Grid", le bouton "Start" n'aura aucun effet.

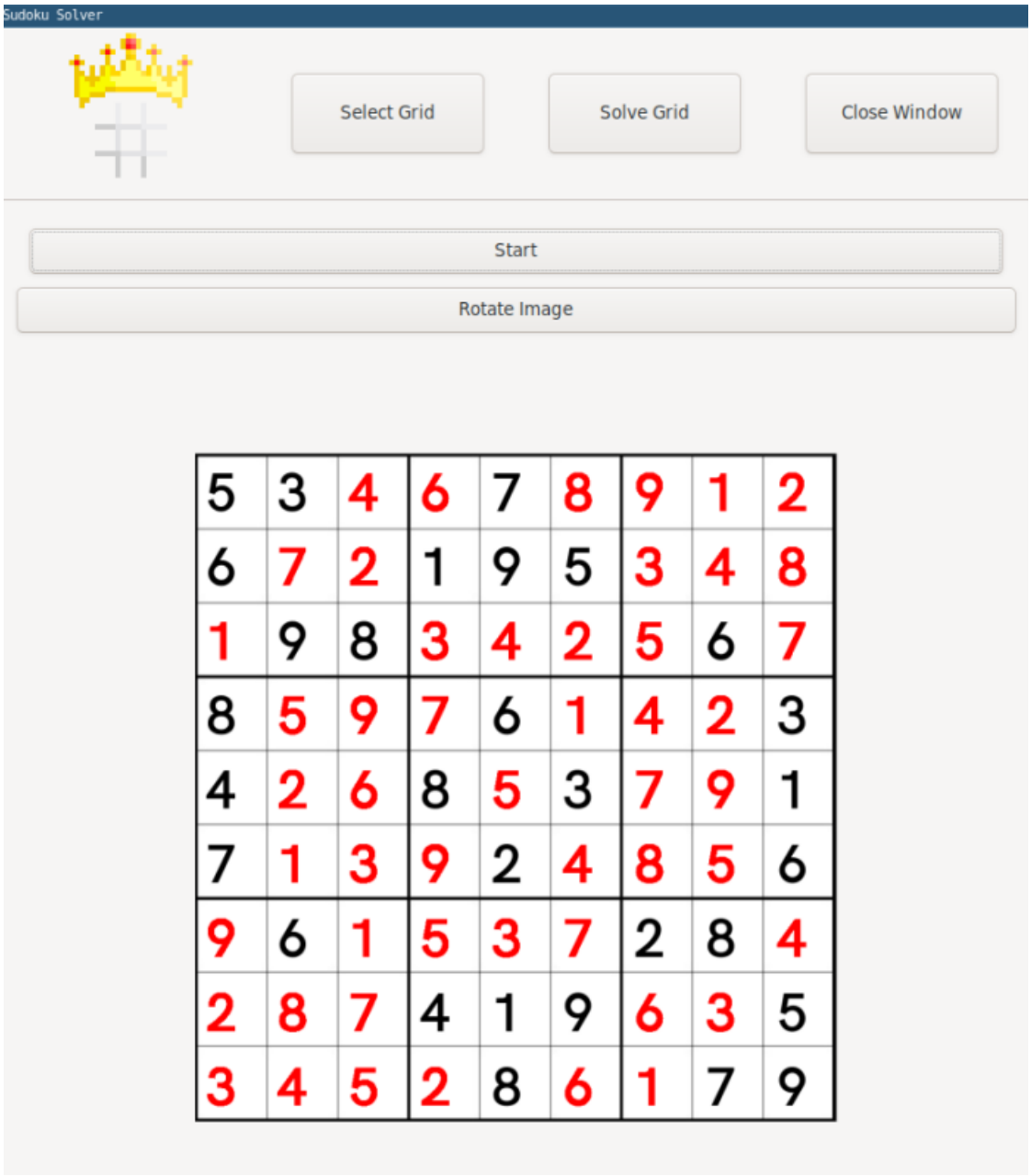


Figure 23: La grille résolue

Une fois la grille affichée, il y a possibilité de la pivoter de 90° par 90° (4 orientations possibles) à l'aide du bouton "Rotate Image". Ce bouton n'a aucun effet si la grille n'a pas été résolue.

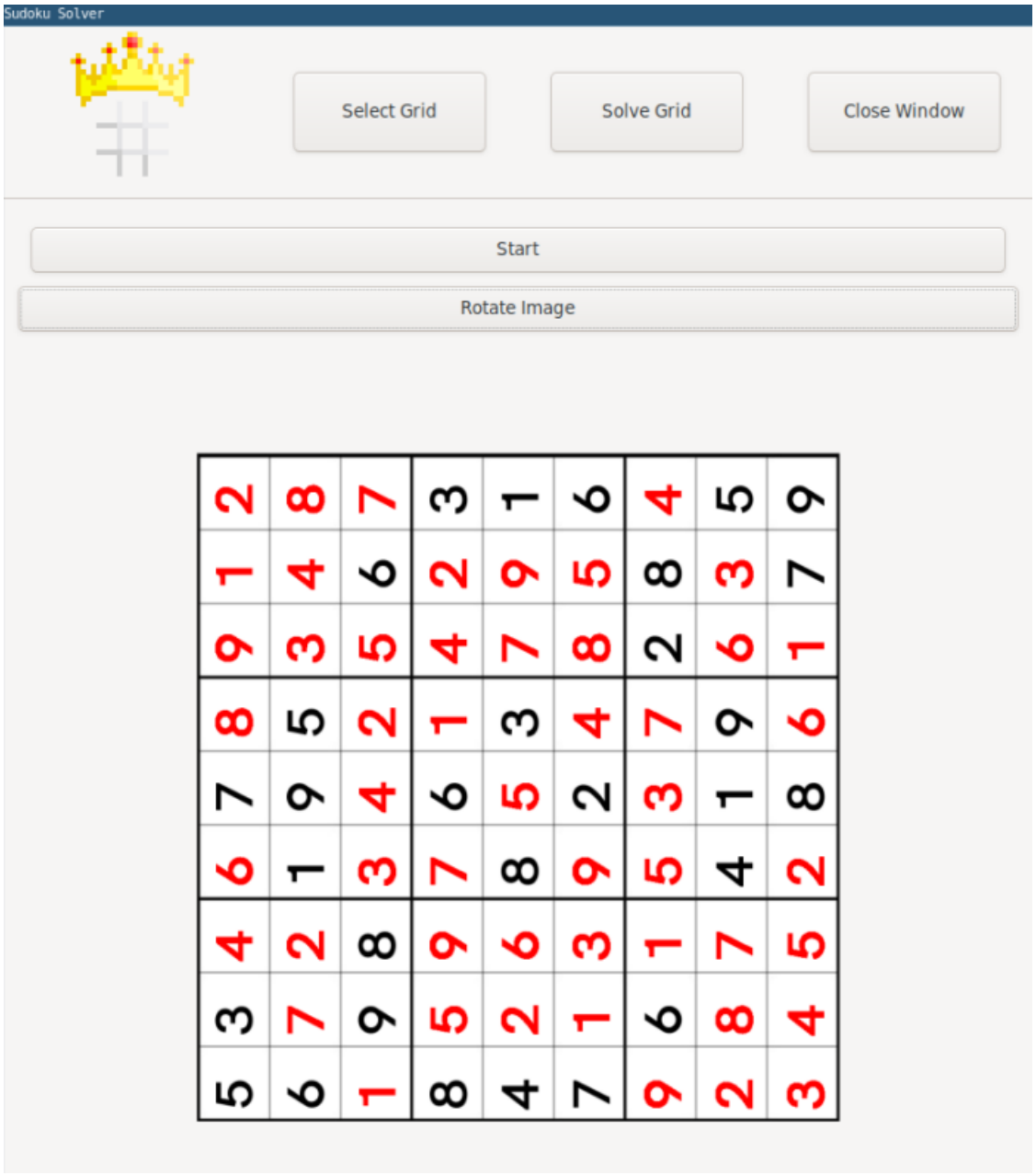


Figure 24: La grille résolue, tournée à 90°

8.4.4 Bouton "Close Window"

Enfin, le bouton "Close Window" permet de fermer la fenêtre très facilement.

8.4.5 Précisions sur l'interface

Les onglets sont accessibles à tout moment pour changer de grille. Ainsi, il est simple de résoudre plusieurs grilles de sudoku en une utilisation de l'application.

9 Compte rendu individuel

Voici un retour de chaque membre du groupe *Les Princesses* sur le projet du troisième semestre.

9.1 Julien Bestard

L'Optical Character Recognition était le projet que j'attendais le plus de ma prépa à EPITA. La réalisation d'un tel logiciel sur 4 mois de cours et d'examens était un challenge, il fallait réussir à s'organiser avec son groupe pour mener à bien le projet tout en suivant les cours.

Dans ce projet, j'ai principalement travaillé sur le traitement de l'image. Au début, j'étais vraiment confus quant à ce que je devais utiliser, quel type de filtre je devais appliquer à mon image et comment le faire. J'ai regardé et lu beaucoup d'articles sur le traitement d'image et cela a commencé lentement à être plus clair.

J'étais aussi en charge de la segmentation et de la détection de la grille. Je pense que cette partie était difficile car nous n'avions pas beaucoup d'indices pour commencer la détection du carré. Mais c'est vraiment gratifiant quand on arrive à faire une étape de la détection. Et j'étais si heureux lorsque les dernières fonctions ont fonctionné parce qu'au début, ce n'était pas gagné.

Comme la fin est presque là, je peux clairement affirmer que nous avons réussi et que nous avons fait un excellent travail. Nous sommes fières de ce ce qu'on a accompli, la seule partie qui aurait pu être meilleure est le fait que nous avons commencé à travailler trop tard. Si nous avions eu plus de temps, nous aurions fait mieux et avec moins de stress.

Pour conclure, je suis content de ce que nous avons fait globalement, l'équipe était géniale et j'ai trouvé le sujet intéressant. Nous avions beaucoup plus de contraintes que pour le projet S2 et c'est aussi ce qui fait la difficulté de ce projet. J'ai appris beaucoup de nouvelles choses, gérer un type de projet différent, à travailler en C et à créer des Makefile.

9.2 Léa Bonet

Ce projet m'a apporté beaucoup de choses durant ce semestre. J'avais hâte de le commencer car j'ai beaucoup apprécié réaliser le projet du deuxième semestre en SUP.

Tout d'abord des compétences techniques. J'ai pu utilisé les compétences acquises cette année en programmation en C pour coder le solver. J'ai utilisé mes compétences en LaTeX, acquises l'année dernière, pour rédiger les deux rapport de soutenance sur *Overleaf*. Même si je n'ai pas réalisé le réseau de neurones j'ai appris comment cela fonctionnait ce qui va m'être utile pour la suite de mon parcours scolaire à l'EPITA.

Mais ce projet m'a également apporté des compétences humaines. En effet, ce projet est avant tout un projet en équipe. J'ai dû apprendre à travailler avec mon groupe et à respecter les délais afin qu'on puisse tous avancer sans problème. Grâce à ce projet nous avons appris à se connaître, se faire confiance et s'entraider.

J'ai également appris à faire des rapports écrits, c'est-à-dire d'expliquer le travail que l'on a réalisé. Et de présenter à l'oral notre projet, ce qui m'a permis de gagner de l'aisance à l'oral.

Dans l'ensemble ce projet paraissait insurmontable au début, mais on est tous fiers du résultat final. On a réussi à travailler main dans la main jusqu'à la dernière minute pour ce projet. Ce fut une expérience riche qui va servir pour le cycle ingénieur.

9.3 Hippolyte Pik

Ce projet m'a beaucoup apporté étant donné mon souhait de parcours. En effet, voulant soit faire de l'intelligence artificielle, soit de l'image, un projet d'OCR était une réelle opportunité pour mon parcours futur.

Ayant réalisé le réseau de neurones je suis d'autant plus content que le résultat est très satisfaisant, me confortant dans mes choix d'orientation dans mon parcours scolaire au sein de l'EPITA.

Ce projet m'a aussi apporté une expérience positive avec un groupe qui travaille et qui est sérieux dans ce qu'ils doivent faire, basé sur la confiance de chacun.

9.4 Benjamin Dreyfus

Ce projet a été bénéfique pour moi sur de nombreux aspects. Il a été difficile de concilier l'avancement du projet avec les cours d'EPITA, mais j'ai bel et bien terminé ce que j'avais voulu faire sur ce projet. Ce projet m'a forcé à maintenir un rythme de travail convenable pour mener à bien ce projet, ce qui est plutôt utile pour m'entraîner à faire de même dans mon futur.

J'ai beaucoup aimé découvrir les dessous de la création d'interface graphique, tout en restant accessible à mes compétences, avec la bibliothèque GTK et le logiciel Glade. Cette expérience est encourageante pour la suite de mon parcours, m'ayant appris de nombreuses choses sur les interfaces graphiques. Je suis prêt à recommencer un projet du genre dans un futur proche ou non.

L'équipe que j'ai choisi de rejoindre pour compléter le projet a également été très encourageante, étant à l'écoute et apte à communiquer sur l'avancement du projet, et comptant une dévotion et une patience remarquable. Je remercie donc toute l'équipe du projet pour son professionnalisme, mais surtout pour son humanité.

En conclusion, ce projet a été une occasion rêvée pour, une nouvelle fois, établir des compétences de programmation et de travail d'équipe. Je m'en souviendrais fièrement, avec le projet de S2 que j'ai aussi beaucoup apprécié.

10 Conclusion

En conclusion, ce projet nous a tous appris beaucoup de compétences techniques mais aussi humaines. Nous avons affronté ensemble les différents obstacles qui se sont dressés devant nous.

Nous avons réussi à tout concevoir dans le temps qui nous était imposé : le traitement des images, le réseau de neurones, l'interface graphique et le solver.

Nous n'avons pas ajouté de bonus car nous voulions que notre OCR soit bien fonctionnel avant de rajouter des fonctionnalités.