Rapport de la soutenance finale

Samy YACEF – Arthur LE FLEM
Mathis VILLEMIN – Augustin GLORIAN
Promo EPITA 2026
Theia

11 décembre 2022

Projet S3 : OCR Sudoku Solver

Table des matières

1 Présentation du groupe

1.1 Présentation général du groupe et des répartitions

Ce projet consiste à créer un sudoku solver c'est à dire que lorsqu'on donne une image à notre programme, celui-ci détecte le sudoku grâce à un réseau de neurone puis le résout et affiche le résultat final qui est le sudoku résolut. Pour cette réalisation, nous nous sommes mis à quatre pour développer cette application.

Nous avons créé ce groupe facilement car dès notre première année à EPITA, nous étions déjà dans la même classe malgré que nous n'ayons pas fait de projet en commun avant. De ce fait, dès la mise en ligne du cahier des charges et au fait que nous étions dans la même classe, nous sommes venu l'un vers l'autre très rapidement afin de constituer ce groupe que l'on a nommé Theia en rapport à la déesse de la vue. En effet, la déesse de vue est une bonne métaphore concernant une application qui traite des images et donc du visuel.

Moins d'une semaine après la création du groupe, nous avons organisé une réunion afin de découper le projet et de se répartir les tâches. Nous avons donc pris la décision après plusieurs lectures du cahier des charges de répartir les tâches en deux grosses sections : le traitement de l'image et le réseau de neurone. Suite à ce découpage, Samy et Arthur ont choisit la partie consacré au réseau de neurone et Mathis et Augustin ont donc pris la partie consacré au traitement des images qui faisait première vue moins peur que le réseau de neurone.

Dans la section de traitement des images, nous avons découpé le travail en deux sous-parties : pré-traitement et post-traitement qui sont elles-même découpé en plusieurs fonctions.

Membre	Partie
Pré-traitement des images	Mathis et Augustin et Samy
Réseau de neurone	Samy et Arthur
Post-traitement des images	Mathis et Augustin
Interface Utilisateur	Mathis et Augustin

Nous avons tous pendant une semaine entreprit de faire des recherches sur notre section. Afin de savoir comment débuter et par quoi. Nous avons donc refait une réunion durant laquelle nous avons mis en place un repository sur GitHub pour faciliter le développement du projet. Nous avons également mis en place grâce au site Overleaf un dossier à partir duquel nous rédigeons tous nos documents en Latex (comme ce rapport). Et enfin, nous communiquons majoritairement via discord et grâce aux cours ou durant notre réunion hebdomadaire.

1.2 Présentation de chaque membre et de ses tâches

1.2.1 Samy

Étant étudiant étranger, j'ai rejoint l'EPITA, après un cursus scolaire dans un établissement français à l'étranger. J'ai pu rencontrer de nombreuses personnes d'origine diverses et variées, et cela m'a permis d'évoluer et d'en apprendre plus sur moi-même.

Après cette introspection, j'ai commencé une recherche concernant ma carrière future et après le stage de 3éme dans une entreprise de matériel informatique concurrente à celle de ma mère, j'ai pu découvrir le métier de technicien informatique dans leur service après-vente. J'ai ressenti une certaine attirance pour le domaine de l'IT.

De mère ingénieure électronique et cheffe d'entreprise en Informatique, j'ai été, et ce depuis le plus jeune âge, baigné dans ce domaine mais sans avoir une réelle attirance pour un domaine en particulier. Nous avons eu l'occasion l'année dernière notamment, de réaliser un jeu vidéo, nous plongeant directement dans ce domaine pour avoir un petit aperçu de ce dernier. J'espère ainsi à travers ce projet, évoluer encore une fois, et déterminer une possible attirance pour le domaine de l'imagerie.

J'attendais avec impatience le début du projet de S3 car je savais que le défi serait présent et un résultat concret serait visible. En effet la plupart des tâches à réaliser, notamment les TPs, que nous avons fait ont été assez théoriques et centrés sur la maîtrise d'une ou plusieurs composantes de la programmation informatique tel que les pointeurs.

De plus, ce projet m'a permis de mettre en pratique toutes les connaissances que j'ai acquises au cours de ces deux premières années universitaires, mettant en lumières les progrès que j'ai fait et ceux malgré certaines difficultés au début.

C'est pour cette raison que je me suis porté volontaire pour la mise en place du réseau de neurones, qui, pour être tout à fait honnête, nous a fait assez peur car sans réel base en IA, cette partie du projet semblait assez flou et très complexe. J'ai eu certes de nombreuses difficultés au départ mais des recherches approfondies quant au mode de fonctionnement des réseaux de neurones m'ont permis d'y voir plus clair quant à une potentielle implémentation dans le logiciel de reconnaissance et de résolution de sodoku.

De plus, l'ambiance conviviale et chaleureuse au sein de notre groupe m'a poussé à donner le meilleur de mes capacités dans ce projet. Un esprit d'entraide était aussi très présent, cela nous a permis d'avancer ensemble et d'échanger sur les problèmes que chacun rencontre et réfléchir ensemble à une solution si nécessaire. Ce travail d'équipe est d'autant plus nécessaire dans un projet comme celui-ci ou chacune des composantes du projet doit être en accord avec les autres.

1.2.2 Arthur

Au cours de mon parcours scolaire j'ai pu développer des premières compétences en informatiques dans certains langages tel que python, c#, html, css, interfaces utilisateur ou encore technologie web. Et je n'ai qu'une grande motivation, travailler sur ces sujets. C'est pourquoi le choix d'EPITA était logique.

Ce n'est pas la première fois que je réalise un tel projet de groupe, puisqu'au lycée mon groupe et moi avons mis au point un puissance 4 dans le cadre de la spécialité numérique et science informatique en classe de première. Nous étions 3 et nous avons développé le jeu en python.

Cela m'a permis de perfectionner mes compétences en python, interface utilisateur et expérience utilisateur puisque le jeu était lancé sur une fenêtre à part, ainsi qu'en intelligence artificielle car nous avons doté le jeu d'un robot que l'on peut défier. Le jeu se joue à deux joueurs ou alors à un joueur contre le robot.

D'autre part l'année dernière j'ai pu développer un jeu vidéo sur ordinateur. Nous étions un groupe de quatre exactement comme pour ce nouveau projet. Et nous sommes arrivée à produire un jeu comme convenu avec lequel nous pouvons jouer en multijoueur sur plusieurs ordinateurs différents.

Ainsi je vais pouvoir mettre à profit l'expérience que j'ai acquise au cours de ses fabuleux projets pour la réalisation d'un résolveur de sudoku à l'aide d'une reconnaissance optique de caractères. Ce nouveau projet est aussi excitant que redoutable puisqu'il repose sur un processus informatique complexe et extrêmement puissant.

Mais nous sommes également plus autonomes qu'avant sur ce projet dans le sens ou le cahier des charges est spécifique sur le fonctionnement mais ne nous guide pas pour accomplir le résultat attendu. Je suis donc convaincu que ce projet me permettra de renforcer mes compétences et d'en acquérir de nouvelles. De plus, je peux mettre à profit mes capacité afin d'aider mes camarades à avancer sur le résolveur de sudoku.

La partie que j'ai trouvé la plus intrigante et la plus intéressante est clairement l'implémentation du réseau de neurones qui va nous servir pour la reconnaissance optique de caractères. C'est donc sur cette partie que je me suis proposé de travailler, proposition que notre groupe a acceptée.

Travailler ensemble pour ce projet a été un choix évident pour tous les membres depuis le départ. Nous avons eu la chance d'être tous les quatre dans la même classe l'année dernière. C'est d'ailleurs là que nous nous sommes rencontrés. Nous nous faisons confiance, nous sommes exigeants et nous attendons toujours plus du résultat de l'équipe. C'est pour cela que travailler sur ce projet avec notre groupe est motivant et stimulant.

1.2.3 Mathis

Bonjour ! je suis Mathis Villemin. j'ai découvert l'informatique en première avec les cours de NSI. Je me suis tout de suite très intéressé a cette matière qui est devenue ma matière de prédilection très rapidement. j'y ai découvert le python et le web et j'ai pu m'épanouir avec des projets personnels comme un algorithme permettant de caché un message dans une image (stéganographie).

Ce projet de S3 est pour moi une occasion de m'améliorer sur différents sujets.

Pour commencer la prise d'information.

Durant la conception du projet de S3 je me suis pose une quantité importante de question que ce soit sur la manière la plus simple de répondre à un problème ou le fonctionnement de certain fonction. j'ai donc développé mes capacités de recherche d'information, compréhension des codes et de réflexion personnelle. Par exemple j'ai lu une bonne partie de la documentation de SDL pour connaitre les fonctions utilisables et les possibilités que cette bibliothèque nous offre. Certaine fonction (notamment SDL_CreateRGBSurface) m'ont posé plus de problème. l'apparition d'un masque de bits ma perturbé et obligé à lire en détail les spécification pour comprendre "In fine "que le masque pouvait être ignoré. j'ai dû cherché et analyser le code de plusieurs personne afin de l'utiliser de la bonne manière.

Le partage d'information.

Puisque nous avons répartie les différentes étapes de l'OCR nous avons dû communiquer sur la manière dont l'information sera passé entre chaque partie ainsi que sur l'architecture final du projet. Nous avons aussi beaucoup communiqué sur l'avancement du projet les défaillances et les améliorations. Par exemple les reviews de push request devenait parfois l'occasion de passer plusieurs heures de plus sur le projet pour trouver la provenance d'un bug que l'on vient de trouver. Partager l'information sans inonder nos discutions est aussi important. Par exemple nous tenons à jour un tableau représentant l'avancement du projet pour partager cette information sans encombrer nos réseaux de communication.

Ma compréhension du langage C.

J'ai relu un grand nombre de fois le cours de programmation de Monsieur David afin de comprendre pourquoi un pointeur était NULL ou pourquoi mon programme retourné un segment fault. Tous ces erreurs m'ont permis d'augmenter mon niveau de programmation en C et en général.

J'attends beaucoup des opportunités d'apprentissage de ce projet que ce soit une amélioration technique ou humaine. Je sais que ce projet changera ma manière de procéder et j'espéré pour faire une partie des bonus qui semblent intéressant ou fun comme la possibilité de faire un hexadocu voir de rendre l'OCR compatible avec tous les tailles de sudoku.

1.2.4 Augustin

Quelques mois après mon arrivée au lycée, au moment de choisir mes spécialités pour mon année de première, je me suis remis en question et j'ai réfléchit à ce que j'aimais faire durant mon temps libre. Après de longues heures de recherche intérieur et d'échange avec mes proches, je suis arrivé à la conclusion que j'aimais passer du temps sur mon ordinateur principalement à jouer à des jeux vidéos. Je me suis donc naturellement tourné vers la spécialité Numérique et Sciences Informatiques qui me permettrait d'en savoir plus sur l'outil que j'utilise au quotidien.

Durant mes deux dernière année d'étude au lycée et grâce à ma spécialité NSI, je me suis davantage rapprocher de l'univers de l'informatique en dehors des jeux vidéos. Je me suis notamment intéressé à programmer mon propre bot discord étant donné que discord est quasiment omniprésent dans ma vie de part son originalité, sa simplicité, son esthétique, sa convivialité et sa privacité.

Grâce à ce rapprochement à l'informatique, je me suis dirigé vers EPITA à ma sortie de lycée étant donné que c'est l'une des seul école d'informatique qui délivre également un diplôme d'ingénieur. Grâce à mon entrée dans cette école, j'ai pu approfondir considérablement mes connaissances dans le fonctionnement d'appareil électrique et des programmes en leur sein.

De plus, j'ai réalisé déjà deux incroyables projets durant dès ma première année dans cette école avec notamment la création d'un jeu vidéo. De ce fait, j'avais hâte de commencer ce nouveau projet pour mon troisième semestre au sein de l'école.

De ce fait, grâce à mes premières années à EPITA, je comprends mieux les ordinateurs, leurs fonctionnements et je peux donc plus facilement développer des petits projets sympathiques qui viennent en complément de ma scolarité à EPITA. Je prendre l'exemple de plusieurs bots discord que je continue de développé avec des fonctionnalités qui s'adapte en fonction du serveur et de la demande des membres. J'ai également mis en place sur un serveur distant, un nuage qui me permet de facilement transférer des fichiers de manière sécurisé sans passer par des services tiers comme WeTransfer.

Je suis donc à très motivé pour réaliser dans son intégralité l'application demandé avec l'aide mon groupe afin de toujours chercher à apprendre de nouvelles choses et à développer mes compétences dans ce domaine que j'aime tant.

2 Éléments lors la première soutenance

Nous avons entreprit d'avancer par duo afin de pouvoir s'entre-aider pour facilement tout en avançant sur des parties différentes. Cette méthode s'est avéré payante et nous avons pu obtenir une bonne avancée suite à la mise en route du projet.

Vous pouvez observer nos avancées pour cette première soutenance et une projection de notre avancée pour la dernière soutenance dans le tableau ci dessous.

	1ère soutenance	Soutenance Finale
Pré-traitement des images	20 %	100 %
Réseau de neurone	25 %	100 %
Post-traitement des images	30 %	100 %
Interface Utilisateur	x	100 %

2.1 Pré-traitement de l'image

Dans cette partie, nous nous sommes concentré sur la mise en noir et blanc d'une image, sa rotation d'image et son découpage. En ce qui concerne la détection de la grille, nous l'aurons finalisé pour la prochaine soutenance.

2.1.1 Image en noir et blanc

Pour plus facilement traiter l'image et ainsi pouvoir aisément détecter la grille, nous passons par une mise en noir et blanc de tous les pixels d'une image.

En effet, nous récupérons le code RGB de chaque pixel afin de le transformer via des calculs, en pixel de teinte entre le noir et le blanc.

2.1.2 Rotation d'image

Dans cette partie, nous avons entreprit de créer une fonction qui permet de tourner une image d'un certain nombre de degré. En effet, une personne après avoir compilé les fichiers, peut exécuter l'exécutable avec en paramètre le nom du fichier contenant une image et le degré de l'angle.

La fonction utilise le principe des rotations matricielles c'est à dire que l'on récupère tous les pixels de l'image d'origine avec leur position, on applique la fonction matricielles afin de connaître la position de tous les pixel sur l'image final et on les place. Nous obtenons donc une image qui à pivoté d'un certain degré.

Cependant, avec cette méthode, nous perdons une partie des données lors du passage dans la fonction matricielle. En effet, étant donné que l'image va tourné et que nous allons effectuer plusieurs conversion de types, nous allons perdre en précision et donc certains pixels arrivent sur un pixel déjà occupé sur l'image finale ce qui l'écrase et procure des pixels "vide". Pour palier à ce problème, nous avons mis un fond de couleur blanc à l'image.

2.1.3 Découpage des cases d'un sudoku

Pour le découpage nous utilisons principalement la bibliothèque de SDL qui est très utile pour manipuler des images.

Le fonctionnement est très simple. on donne à la fonction la position de l'angle en à droite et les dimension du sudoku. puis le programme parcourra l'image en partant des coordonnés du sudoku en se déplacent en fonction des dimensions pour découper de manière propre chaque case du sudoku.

Les case seront ensuit enregistré en avec leur coordonnée dans le nom ce qui rendra plus simple la création du fichier text représentant le sudoku.

Ce programme est accompagné d'un make qui permet de supprimé les images de case une fois utilisé.

2.1.4 Détection de la grille

Nous avons malheureusement rencontré un problème pour détecter la grille. Nous nous y sommes pris trop tard ce qui a rendu sa résolution impossible. Bien que nous ayons cherche beaucoup d'information sur les fonctions de détection de grille nous avons échoué à rendre cette partie dans les temps. c'est pourquoi nous avons décidé de posé une deadlines une semaine avant la soutenance final pour s'assurer que ce genre de problème n'apparait plus.

2.2 Réseau de neurone

L'exigence la plus complexe pour le projet final est la reconnaissance des différents chiffres présents dans la grille et ce indépendamment de la qualité d'écriture de ces derniers par l'utilisateur ou de la qualité de la photo qui aurait prise, cette dernière sera certes a priori traitée avant d'être envoyés à l'OCR mais ce traitement a ses limites. Nous avons dû donc élaborer un algorithme qui puissent reconnaître des caractères et ceux malgré des circonstances peu favorables. Un algorithme mimant un système neuronal a donc été privilégié pour résoudre ce problème,

Notre binôme Arthur et Samy s'est occupé de sa mise en place. Pour la première présentation du projet, une implémention d'un réseau de neurones reconnaissant un OU exclusif a été demandé et cela pour nous permettre de nous familiariser avec le principe de fonctionnement d'un réseau de neurones à travers une application de ce dernier pour un problème relativement simple et basique. De la documentation nous a été offerte, qui nous a aussi permis, de mettre en place ce réseau de neurones mais aussi de comprendre et maîtriser son fonctionnement.

Quant aux spécifications du réseau de neurones XOR, il est constitué de deux neurones d'entrée (Inputs) qui peuvent recevoir un 1 ou un 0. Ces derniers transmettent ces informations vers la couche intermédiaire (couche cachée), constitué de deux neurones, après un traitement de cette donnée a travers la multiplication par un poids, spécifique pour chacune des liaisons entre un neurone d'entrée et un neurone de la couche cachée, ce dernier représentant l'importance que cette liaison a pour le résultat final, après cette multiplication, l'ajout d'un biais, cette fois ci, spécifique à la neurone destination, dans ce cas la neurone de la couche intermédiaire, permet d'affiner le traitement de la donnée. Enfin, le résultat obtenu sera entré comme paramètre d'une fonction sigmoïde qui nous assurera que le résultat sera compris strictement entre 0 et 1. Ce processus est répété à chaque passage de l'information d'un neurone à un autre jusqu'à être renvoyé à l'utilisateur à travers le neurone de sortie (Output).

Afin d'entraîner le réseau de neurones, on donne a ce dernier une série dite d'entraînement, cette dernière est constitué de deux entrées et d'une sortie attendue, on peut avoir par exemple, (1, 1, 0), l'algorithme, comporte une fonction d'erreur, va calculer le degré d'erreur qu'il a obtenue à la fin de chaque entraînement et modifier les poids et les biais de chacune des neurones pour réduire au prochain entraînement le degré d'erreur, améliorant ainsi les performances global du réseau de neurone.

2.3 Post-traitement de l'image

2.3.1 Résolution du sudoku (Solver)

Le post-traiment ce fait en trois parties. la récupération du sudoku, sa résolution et son affichage. Pour ce qui est de la récupération du sudoku nous avons écrit un algorithme de parsing capable de lire un fichier est extraire le sudoku sous forme d'une matrice d'entier. Bien sur le sudoku doit écrit de la manière d'écrite dans le sujet.

Une fois le sudoku extrait du fichier nous utilisons la méthode de résolution par backtracking étudier l'an dernier. Pour ce faire nous avons rédigé le fichier solver qui prend en paramètre le nom du fichier contenant le sudoku extré de l'image. Pour une facilité la lecture de l'algorithme nous avons ajouté une condition au nom du sudoku non résolu. Cette condition est que le fichier contenant le sudoku ne doit pas contenir d'extension. Elle permet de réduire la taille du code et donc de faciliter sa relecture. A noté que si le nom de fichier contient tout de même une extension l'exension ".result" sera juste rajouté à la fin à la fin ce qui peut porté à confusion. (ex : test.txt => test.txt.result). Une fois le sudoku résolu (si résoluble) nous enregistrons le résultat avec l'extension ".result" en respectant les normes imposés par le sujet. A noté que si le sudoku n'est pas réalisable le résultat enregistré est simplement le sudoku passé en paramètre.

Il est important de noté que toute la partie post-traitement permet de traiter à la fois les sudoku est les hexadoku. Nous pensons qu'il est important de prévoir cette fonctionnalité très tôt dans le développement du projet pour éviter une réécriture d'une grande partie du code pour ajouter cette fonctionnalité.

2.3.2 Reconstruire la grille

Le squelette de la fonction et les recherches ont été effectués et donc un code est en cours d'écriture. Malheureusement, suite à quelques erreurs et avertissements qui n'ont pas encore trouvé d'origine, nous n'avons pas pu la finalisé avant le clone du répertoire. Cependant, une version finit sera disponible lors du passage de la soutenance même si nous ne pourrons pas la présenter.

2.4 UI

Concernant l'avancement de l'interface pour notre application de résolution automatique de sudoku, nous avons très peu avancé afin de nous concentrer d'avantage sur les autres fonctionnalités jugées plus essentielles.

Nous avons cependant commencer à faire des recherches sur les librairies que nous comptons utiliser et qui son disponible sur les ordinateurs de l'école. Actuellement, on comptons partir sur la librairie Glade pour toute la gestion de l'interface graphique.

D'ailleurs, nous avons également réalisé la fonction qui permet d'afficher une image grâce à la librairie SDL qui permet de traiter des images.

3 Avancement du projet

Après la première soutenance, nous nous sommes réorganisé pour faire un point sur les dernières fonctionnalités à implémenter. De ce fait, Arthur et Samy sont restés sur l'implémentation du réseau de neurone. Mathis s'est principalement consacré à la fin de la détection de la grille ainsi que la partie post-traitement du traitement de l'image. Quant à Augustin, il s'est consacré la l'interface graphique ainsi qu'à la généralisation et la mise en commun des fonctions.

De ce fait, voici la répartition qui a été mise en place durant ce projet.

Membre	Partie
Pré-traitement des images	Mathis, Augustin et Samy
Réseau de neurone	Samy et Arthur
Post-traitement des images	Mathis
Interface Utilisateur	Augustin

D'ailleurs, Samy étant donné que le réseau de neurone à pris de l'avance, il s'est occupé de quelques fonctions de pré-traitement pour faciliter la reconnaissance des caractères.

3.1 Pré-traitement de l'image

3.1.1 Image en noir et blanc

Depuis la première soutenance, nous n'avons pas modifier cette fonction qui reste encore opérationnel après l'ajout des dernières fonctionnalités développé.

3.1.2 Rotation d'image

De même, pour la rotation de l'image, depuis la dernière soutenance,

3.1.3 Filtre Médian

Le filtre médian est un filtre numérique non linéaire, souvent utilisé pour la réduction de bruit. La réduction de bruit est une étape de pré-traitement classique visant à améliorer les résultats de traitements futurs (détection de bords par exemple).

En effet, il permet sous certaines conditions de réduire le bruit tout en conservant les contours de l'image. Cette technique consiste en le calcul de la valeur médiane de l'ensemble des valeurs RGB des 9 cases autour d'une case donnée, afin de remplacer la valeur de cette case par la valeur trouvée. Pour cela, on regroupe dans une liste, l'ensemble de ces valeurs, un tri dans l'ordre croissant est ensuite exécuté, ce qui permet de trouver assez facilement la valeur médiane.

De manière générale une recherche de la valeur médiane sur 9 cases suffit à avoir une valeur satisfaisante d'approximation sur un pixel. Mais suite a plusieurs testes réalisées par nos soins, nous avons décidé d'effectuer cette recherche sur 25 cases afin d'avoir plus de précision. Cette dernière approximation a donné des résultats plus que satisfaisants. Ci-dessous vous trouverez un exemple d'application de notre algorithme de recherche de valeur médiane.

3.1.4 Découpage de l'image

Le découpage de l'image est plutôt simple à réalisé une fois que le sudoku est droit.

Split-grid prend en paramètre :

- le nom de l'image
- les coordonnées x et y de l'image du coin haut gauche
- les dimention du réctengle (hauteur et largeur)

Avec ces information nous pouvons facilement délimiter l'emplacement du sudoku dans l'image et nous déplacer sur chaque case du sudoku afin d'en créer une copie grâce à SDL-BlitSurface. Les dimensions de chaque case sont identique et délimité en fonction de la hauteur et de la largeur du sudoku. Chaque case de sudoku est enregistrée sous un nom générique 'Suokucase(x,y)' avec x et y la position de la case dans le sudoku.

3.1.5 Détection de la grille

Pour la détection de grille, nous partons du principe que la grille de sudoku est l'élément qui prends le plus de place dans l'image. La détection de grille utilise le flood fill pour trouver l'élément qui prend le plus de place dans l'image.

La fonction flood-pixel permet d'obtenir un tabelau de 8 éléments représentant les coordonées du carré dans lequel l'objet est strictement compris. pour ce faire flood-pixel s'appel récursivement sur les pixels adjacents. à chaque fois d'un pixel sors des coordonnées contenant l'objet on redéfinit les limites pour que ce pixel soit compris dans les coordonnées du tableau.

un fois le tableau obtenu on compare son aire avec l'aire de l'objet le plus grand jusqu'à maintenant est on garde le plus grand. on remarque à les coordonnées dans le tableau sont aussi les coordonné du sudoku dans l'image.

avec ces information nous pouvons faire la rotation et le découpage de l'image.

la fonction flood pixel est appel par la fonction flood-surface sur chaque pixel. c'est aussi flood-surface qui fait le calcul d'aire et qui conserve les coordonnées de l'objet de plus grand.

3.2 Réseau de neurone

La reconnaissance optique de caractères est un domaine d'application typique des méthodes de classification automatique. Outre son intérêt pratique tel que la reconnaissance de codes postaux, lecture automatique de chèques bancaires, etc. Pour notre projet nous cherchons a reconnaitre des chiffres écrits a la main dans un sudoku. La reconnaissance de caractères est une tâche connue pour être difficile et pour laquelle il n'existe pas encore de solution techniquement satisfaisante. Cette tâche est très compliquée parce que les chiffres manuscrits ne sont pas parfaits et peuvent être représentés de nombreuses manières différentes. La reconnaissance manuscrite des chiffres est la solution à ce problème qui utilise l'image d'un chiffre et reconnaît le chiffre présent dans l'image.

```c
float neural_network_gradient_update(mnist_image_t * image, neural_network_t * network, neural_network_gradient_t * gradient, uint8_t label)
{
    float activations[MNIST_LABELS];
    float b_grad, W_grad;
    int i, j;

    // First forward propagate through the network to calculate activations
    neural_network_hypothesis(image, network, activations);
    //feed_forward(image, network, activations);

    for (i = 0; i < MNIST_LABELS; i++) {
        // This is the gradient for a softmax bias input
        b_grad = (i == label) ? activations[i] - 1 : activations[i];

        for (j = 0; j < MNIST_IMAGE_SIZE; j++) {
            // The gradient for the neuron weight is the bias multiplied by the input weight
            W_grad = b_grad * PIXEL_SCALE(image->pixels[j]);

            // Update the weight gradient
            gradient->W_grad[i][j] += W_grad;
        }

        // Update the bias gradient
        gradient->b_grad[i] += b_grad;
    }

    // Cross entropy loss
    return 0.0f - log(activations[label]);
}
```

La fonction de perte que nous minimisons est la vraisemblance négative moyenne de la distribution conditionnelle de l'étiquette correcte compte tenu de l'entrée, log(p(y|x), parmi les exemples d'apprentissage (x,y) de l'ensemble de données S. Cela équivaut à maximiser l'objectif de régression logistique multinomiale. L'optimisation est basée sur le gradient et les gradients sont calculés à l'aide de l'algorithme de rétropropagation.

```c
void feed_forward(mnist_image_t * image, neural_network_t * network, float activations[MNIST_LABELS])
{
    int i, j;

    for (i = 0; i < MNIST_LABELS; i++) {
        activations[i] = network->b[i];

        for (j = 0; j < MNIST_IMAGE_SIZE; j++) {
            activations[i] += network->W[i][j] * PIXEL_SCALE(image->pixels[j]);
        }
    }
    neural_network_softmax(activations, MNIST_LABELS);
}
```

La rétropropagation est une technique de calcul rapide des dérivées, qui a été utilisée dans divers domaines techniques au-delà de l'apprentissage profond. La rétropropagation obtient les gradients de manière analytique. Formulée succinctement, la rétropropagation calcule les gradients d'expressions mathématiques par application récursive de la règle de la chaîne. La descente de gradient par rétropropagation est beaucoup plus rapide que l'utilisation des différences finies. La rétropropagation peut accélérer la formation des réseaux neuronaux modernes jusqu'à 10 millions de fois. La rétropropagation et la différenciation automatique s'appliquent le plus naturellement

du monde à l'optimisation d'une fonction en la décomposant en modules pour lesquels les gradients locaux peuvent être facilement dérivés, puis enchaînés. La différenciation automatique simplifie grandement la tâche d'obtention des gradients analytiques.

Le jeu de données MNIST contient 60 000 images d'entraînement de chiffres manuscrits de zéro à neuf et 10 000 images pour les tests. Ainsi, l'ensemble de données MNIST comporte 10 classes différentes. Les images de chiffres manuscrits sont représentées sous la forme d'une matrice 28×28 où chaque cellule contient une valeur de pixel en niveaux de gris.

3.3 Post-traitement de l'image

3.3.1 Résolution du sudoku (Solver)

Une grille de Sudoku est composée de 9 petits carrés de 3 cases sur 3, placés côte-à-côte afin de former un grand carré de 9 cases sur 9. Sur la grille, il faut placer des chiffres de 1 à 9 en respectant les règles suivantes :

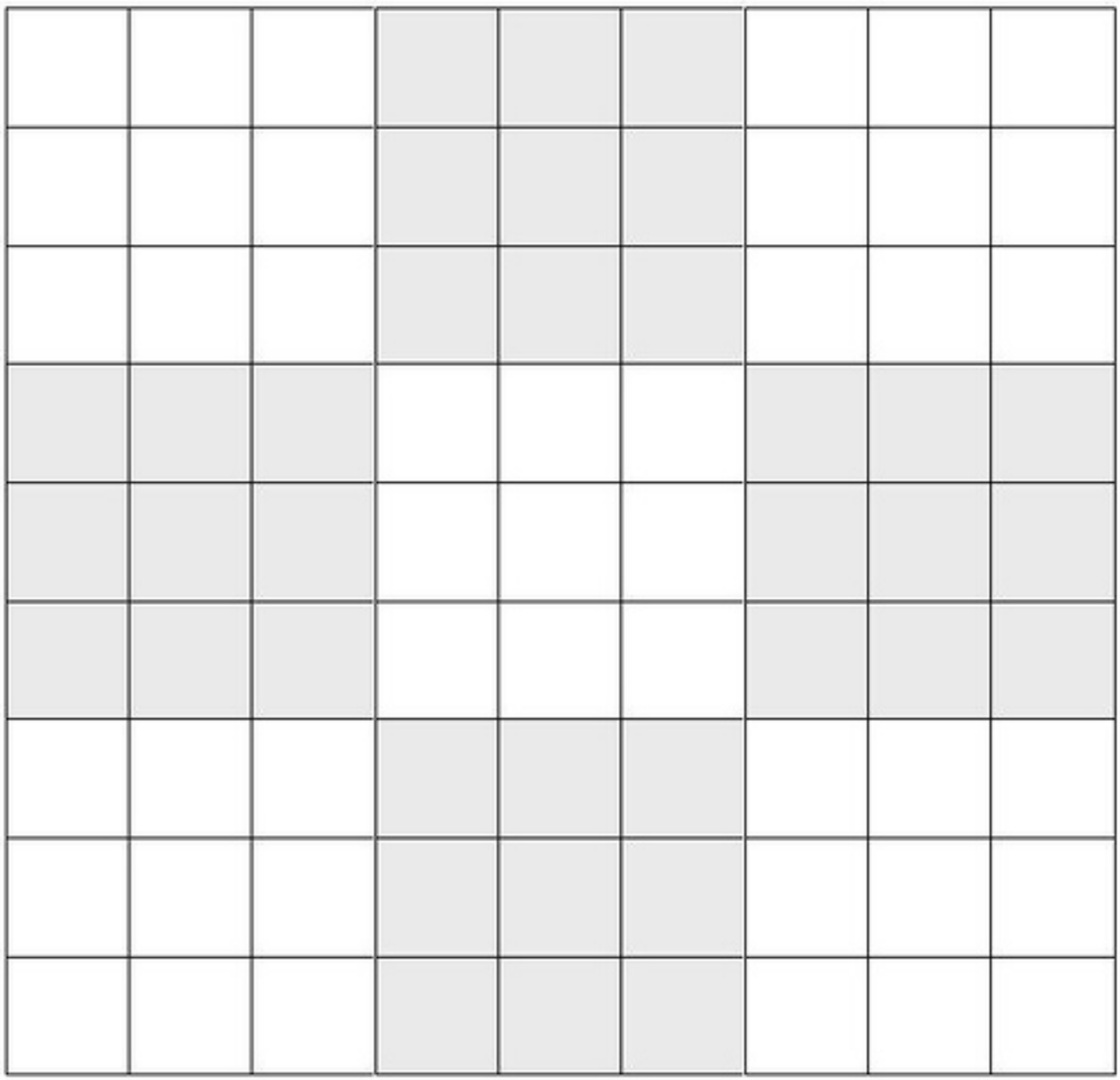
- les 9 chiffres doivent être présents dans chacun des 9 petits carrés
- chaque ligne du grand carré doit contenir les 9 chiffres (donc chacun une et une seule fois)
- chaque colonne du grand carré doit contenir les 9 chiffres (donc chacun une et une seule fois).

Le backtracking est un algorithme qui permet de trouver la solution à coup sûr. Le principe de cet algorithme est de prendre un candidat dans une cellule non encore résolue et de faire la résolution à partir de cette nouvelle grille. Si la résolution mène à une solution, c'est fini, sinon, si on arrive à une nouvelle impasse, on choisit un autre candidat possible dans une autre case et on continue et sinon si on arrive à une grille inconsistante on revient à notre point de départ. Cet algorithme est récursif. Dans la pratique, le backtracking de ce programme fonctionne comme

décrit dans la suite. Tout d'abord on sauvegarde la grille afin de pouvoir revenir en arrière si nécessaire. Puis on cherche la case non résolue qui a le plus petit nombre de candidats possibles. Puis on choisit de mettre dans cette case le premier candidat encore possible. Ensuite, on appelle la fonction qui solve la grille, donc on fait un appel récursif. Dans notre fonction, on teste si la grille est consistante à chaque fois qu'on réappelle les heuristiques, si la grille est inconsistante, on revient en arrière en sortant de cet appel récursif. Si la grille est consistante et qu'on a atteint un point fixe, alors on regarde si la grille est résolue ou non. Si elle est résolue, on affiche la grille et on quitte le programme. Si la grille n'est pas résolue, on resauvegarde la nouvelle grille, on recherche une nouvelle fois la case avec le plus petit nombre de candidats, et on refait un choix dans cette case. A la sortie de chaque appel récursif, la grille courante est remplacée par la grille précédemment sauvegardée. Puis on teste les autres candidats de la case, parce que le choix du candidat précédemment menait à une grille inconsistante.

3.3.2 Reconstruction de la grille

Afin de reconstruire la grille résolue, nous avons opté pour une méthode très classique. Nous avons d'abord créé un échantillon d'images de chiffres ainsi qu'un échantillon d'une grille de Sudoku vide. Nous avons choisi d'avoir une image de 800x800 pixels pour la grille vide et des images de 90x90 pixels pour chaque chiffre.



La méthode utilisée ici consiste à parcourir l'image de la grille vide et à remplacer chaque cellule par l'image du numéro correspondant dans la grille résolue. Pour ce faire, nous utilisons deux fonctions : La première remplace tous les pixels d'une image par ceux d'une autre image à des coordonnées précises sur l'image à modifier. La seconde permet de parcourir la grille résolue et non résolue et de changer la couleur du numéro sur l'image de la grille résolue s'il n'était pas dans la grille initiale.

3.3.3 Sauvegarde des résultats

Pour sauvegarder les résultats c'est-à-dire les images après traitement, on passe directement par l'interface graphique qui à chaque instant possède le chemin vers l'image issus du dernier traitement. De ce fait, l'utilisateur en cliquant sur le bouton *Save as*, il peut sauvegarder où il veut l'image en renommant le fichier comme il le souhaite.

3.4 UI

Pour la réalisation de l'interface utilisateur, nous avons utilisé glade afin de générer cette interface graphique. Avec glade, nous avons utilisé gtk afin de pouvoir intéragir avec cette interface.

Cette interface permet à un utilisateur de pouvoir facilement utiliser les différentes fonctions réaliser dans le cadre de ce projet notamment toutes les fonctions de pré-traitement, le réseau de neurone et les fonctions de post-traitement.

L'interface graphique affiche à tout moment l'image qui va ou qui est en cours de traitement. De plus, à tout moment, l'utilisateur peut sauvegarder l'image affiché ou recharger l'image initialement chargé. Tous les boutons utilisent le système d'interaction en dynamique c'est-à-dire que l'interaction sur les boutons sont détectée en partie grâce à glade qui appel une fonction dans le fichier c.

Derrière tous les boutons, nous pouvons retrouver des fonctions gtk afin de modifier l'interface en changeant certains boutons ou en changeant l'image affiché. De plus, pour appeler les fonctions dans les autres fichier c comme les fonctions pour mettre l'image en niveau de gris ou pour appeler le réseau de neurone ou encore pour reconstruire l'image, nous faisons grâce à un string et à la fonction *system()* ce qui permet d'exécuter des commandes terminales via un fichier c. Une autre méthode qui aurait pu fonctionner et d'utiliser grâce à des .h directement les fichiers dans les fichiers ce qui aurait pu limiter les erreurs de chemin. Malheureusement, étant donné que les fonctions ont été fait indépendemment pour la première soutenance, elle n'ont pas de fichier .h correcte, ce qui complique l'utilisation de cette méthode dans notre projet.

4 Présentation dans sa version final

4.1 Introduction de l'application

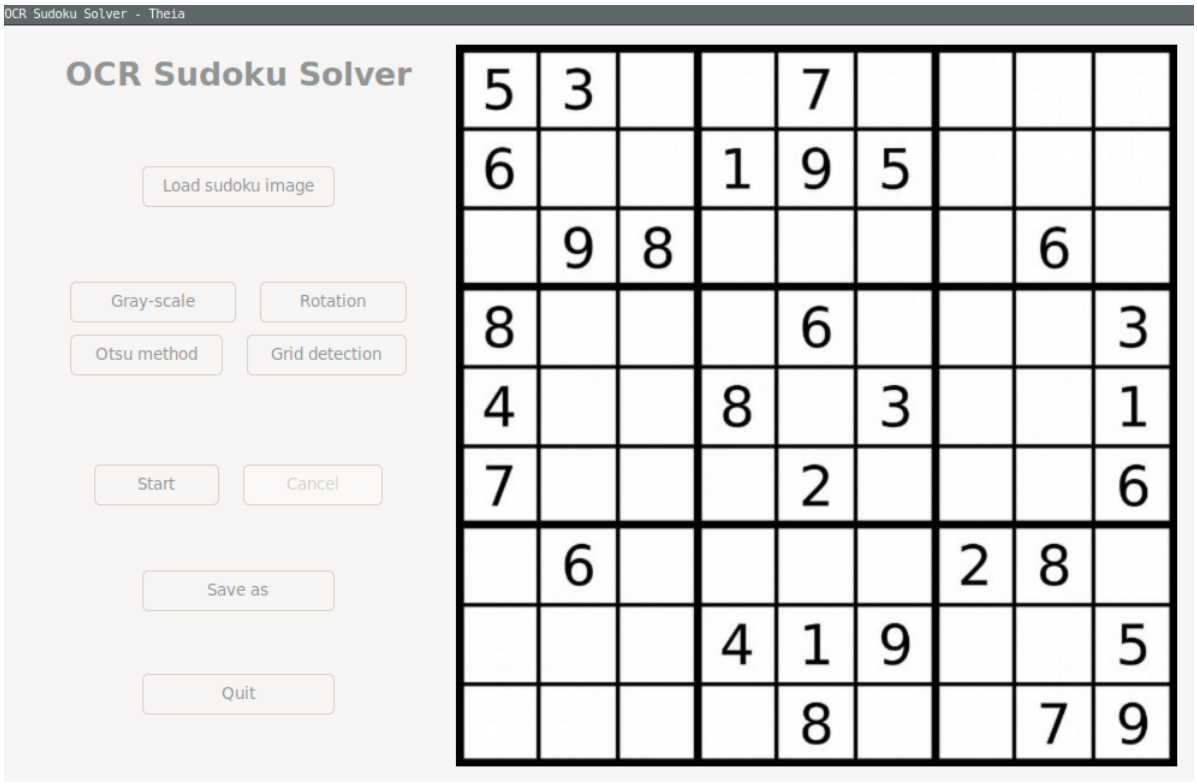
Nous allons maintenant, vous présenter l'application dans sa version finale. Nous verrons toutes les fonctionnalités disponibles pour l'utilisateur grâce aux images ci-dessous.

4.2 Comment ouvrir l'application ?

Afin de pouvoir utiliser l'application, il vous faut récupérer au préalable le répertoire contenant le projet (avec la commande *git clone*). Une fois le répertoire récupéré, rendre dedans avec la commande *cd* puis exécuter la commande *make* afin de compiler tout le projet. Enfin, pour lancer l'application, il vous faudra exécuter la commande *./ui*.

4.3 Présentation à l'ouverture de l'application

Lorsque l'on ouvre l'application, nous arrivons directement sur cette page :



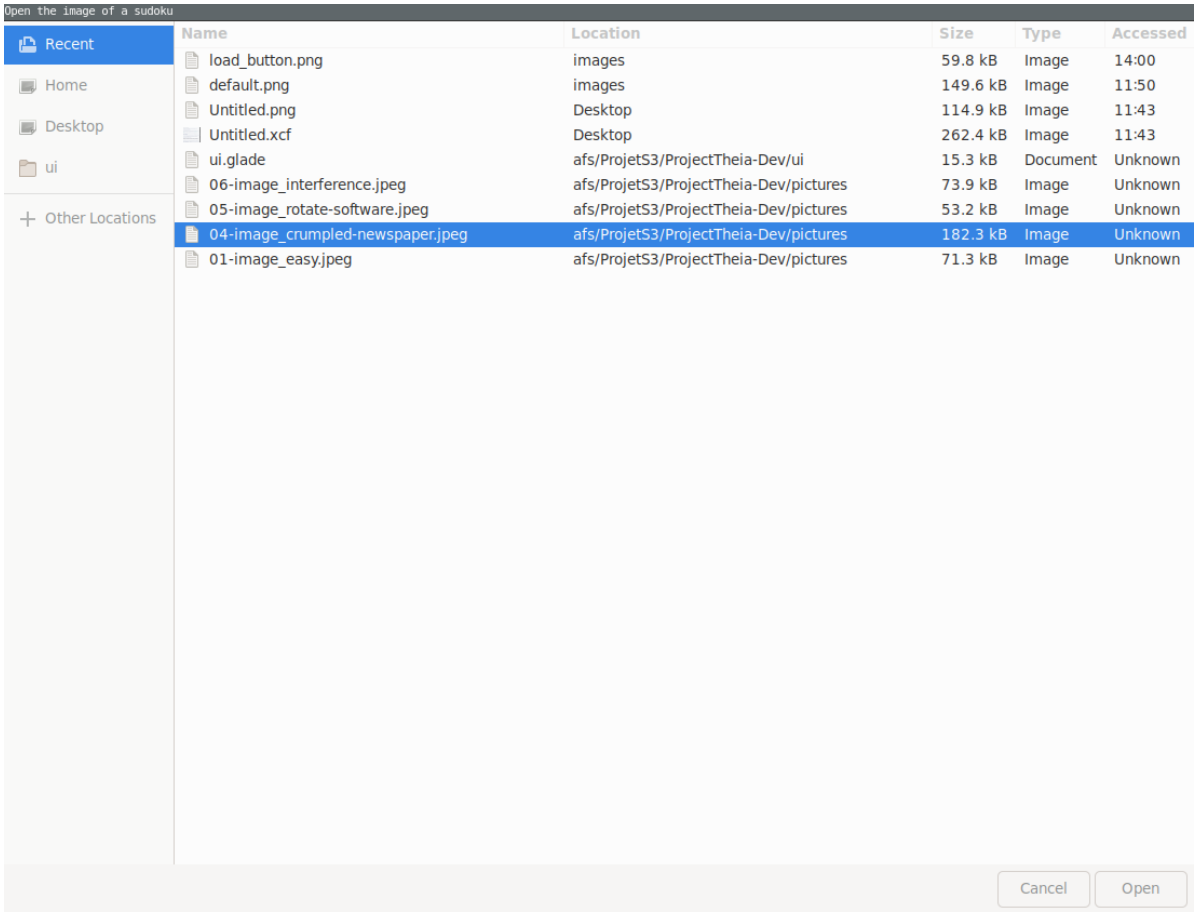
Nous pouvons constater que cela ouvre une fenêtre avec comme titre : *OCR Sudoku Solver - Theia*. A gauche, nous pouvons retrouver tous les boutons qui vont permettre à l'utilisateur d'interagir facilement avec l'application. Nous verrons en détail l'utilité des boutons ci-dessous. De plus, nous avons également mis par défaut une grille de sudoku afin de pouvoir facilement tester l'application et ce même si nous n'avons pas à notre disposition une banque d'image représentant des sudoku.

4.4 Présentation des boutons

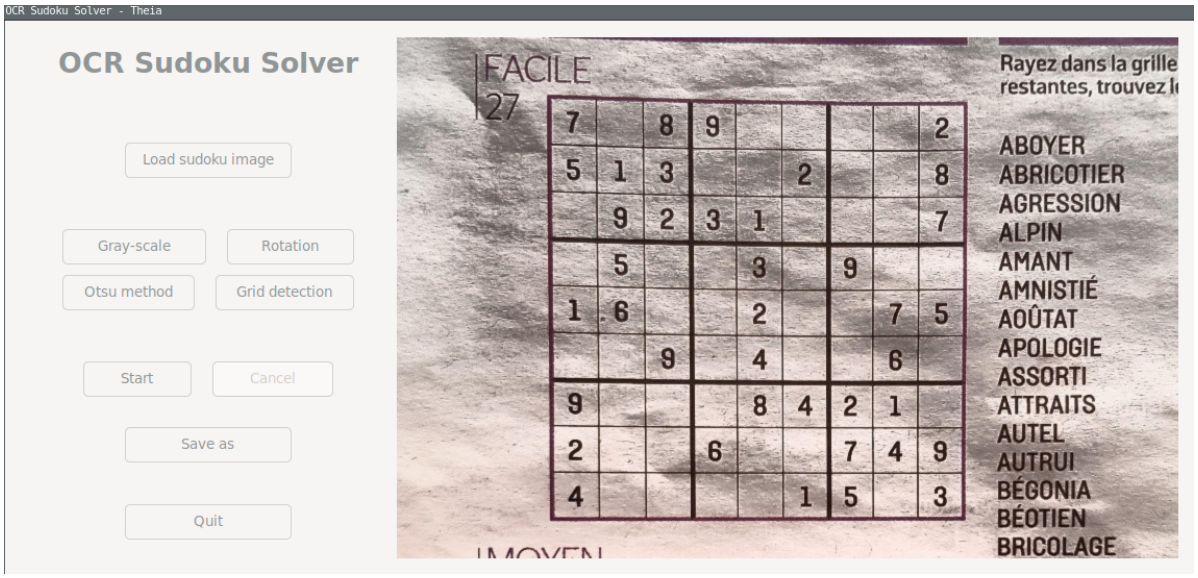
Voyons en détail l'action réalisé par les boutons quand l'utilisateur applique un clique gauche sur celui-ci.

4.4.1 Charger une image de sudoku (Load sudoku image)

Le bouton le plus en haut, nommé *Load sudoku image* et un des boutons qui ouvre une autre fenêtre par dessus l'application afin de pouvoir sélectionner le fichier qui contient l'image que l'on veut traiter c'est à dire une image contenant un sudoku. Voici ci-dessous la fenêtre qui s'ouvre lorsque l'utilisateur clique sur le bouton.

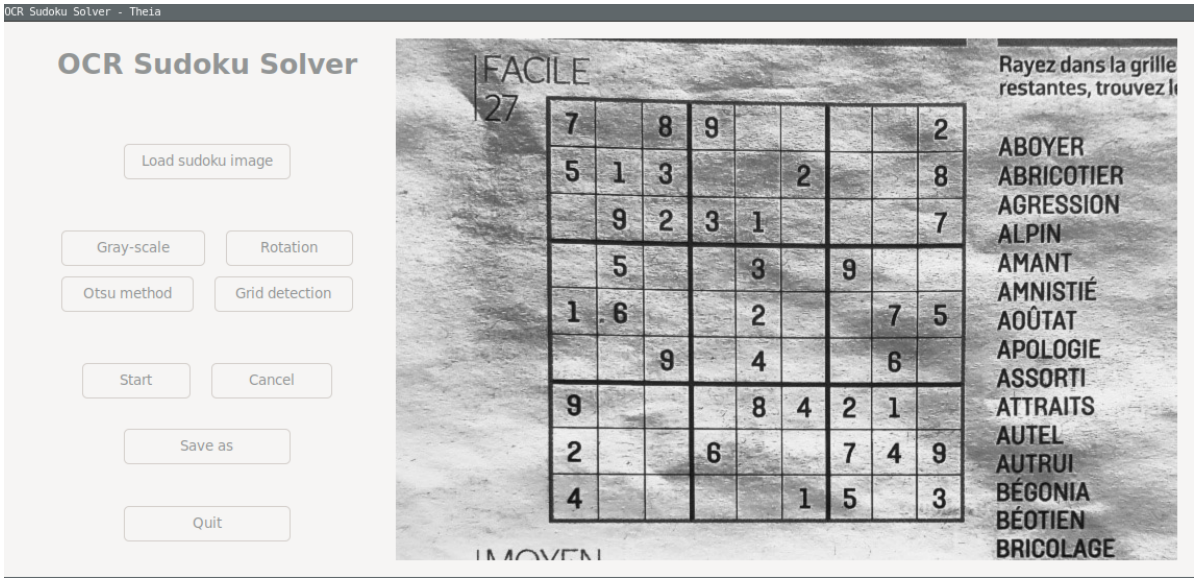


Quand l'utilisateur à sélectionné une image, le fenêtre de sélection se ferme et l'image apparaît à la place de l'ancienne, dans la partie droite de l'application comme nous pouvons l'observer dans la capture d'écran ci-dessous.



4.4.2 Niveau de gris (Gray-scale)

Le deuxième bouton nommé *Gray-scale* permet d'appliquer un traitement en gray-scale sur l'image de la partie droite. Si nous reprenons l'image précédente, nous obtenons l'image ci-dessous.



Comme précédemment, l'image à droite à changer afin d'afficher le résultat de l'image après le traitement de la fonction gray-scale.

4.4.3 Rotation (Rotation)

Nous utilisons les données produites par l'algorithme de la transformation de Hough. Nous parcourons l'accumulateur de Hough et comptons les votes pour chaque angle thêta trouvé. Si l'angle le plus voté est supérieur à |2 deg | et inférieur à |45 deg |, nous appliquons une rotation manuelle avec cet angle. Sinon, nous parcourons à nouveau l'accumulateur jusqu'à ce que nous trouvions un angle valide pour la rotation, jusqu'à un certain seuil.

4.4.4 Méthode d'otsu (Otsu method)

Le seuillage est utilisé pour binariser les images (rendre les couleurs 0 ou 255) afin de ne garder que les caractéristiques que nous voulons sur l'image. L'algorithme que nous avons décidé d'utiliser est la méthode d'Otsu, du nom de Nobuyuki Otsu. Elle nous permet d'effectuer un seuillage automatique efficace sur des images dont le fond est uniforme. Cette méthode utilise la variance d'un histogramme rempli du nombre de pixels à chaque valeur (de 0 à 255) pour trouver une valeur seuil située entre les couleurs du premier plan et du fond de l'image.

4.4.5 Commencer (Start)

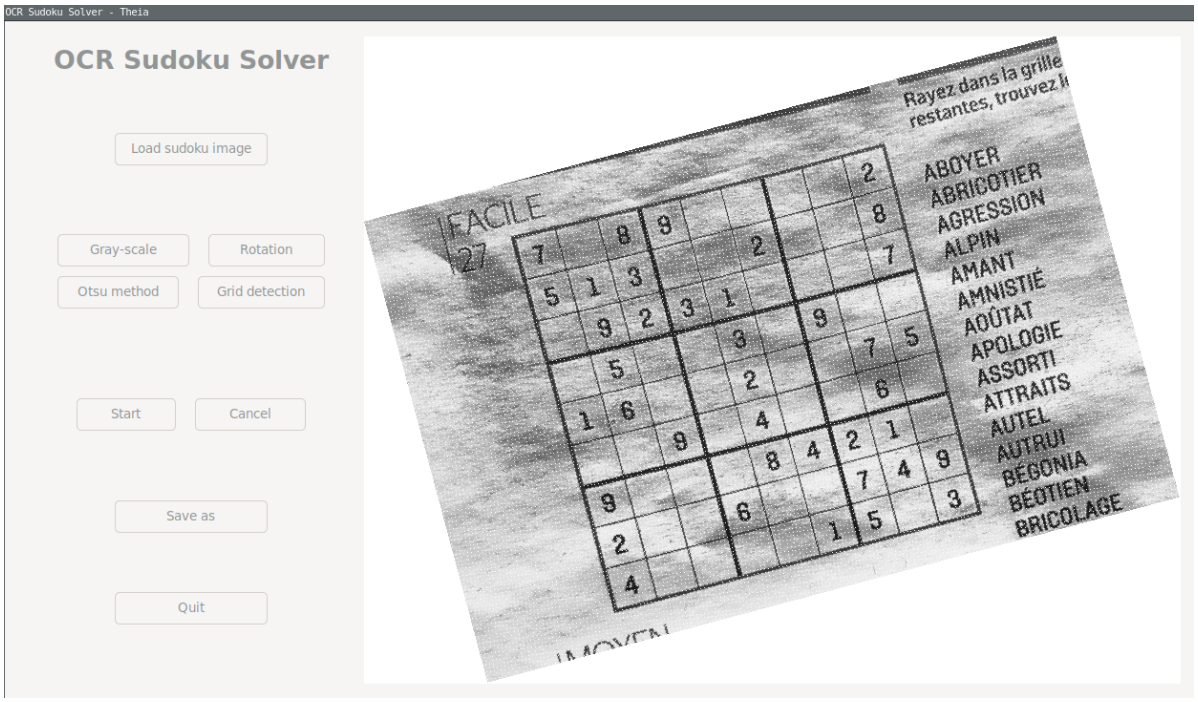
Grâce à ce bouton, l'utilisateur peut lancer tous les traitements pour résoudre une sudoku à partir d'une image.

En effet, ce bouton va appeler les fonctions pour mettre l'image en niveau de gris, appliqué le median filtre, la méthode otsu. Elle permet également de détecter la grille ainsi que mettre dans le bon sens la grille, puis d'appliquer le découper les cases de la grille avant de les passer dans le réseau de neurone pour ensuite appeler la fonciton qui résout le sudoku à partir d'un fichier txt. Et enfin, on reconstruit la grille et on l'affiche dans dans l'interface graphique.

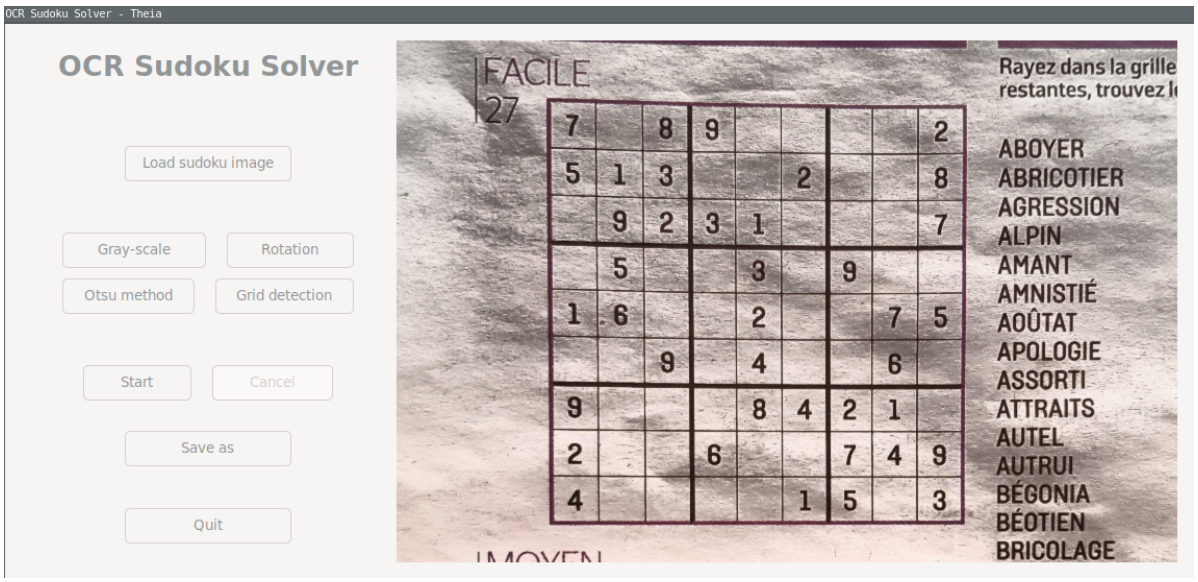
4.4.6 Annuler (Cancel)

Grâce au bouton *Cancel*, on peut à tout moment, retrouver l'image d'origine que l'on a initialement chargé avant tout les traitements appliqués à l'image. De ce fait, le bouton est désactivé tant qu'aucun traitement n'a été réalisé sur l'image. Donc, dès qu'un traitement est appliqué, on peut cliquer sur le bouton.

Prenons l'exemple après la mise en niveau de gris et après une rotation manuel. L'image avant de presser le bouton.



L'image après avoir actionné le bouton.

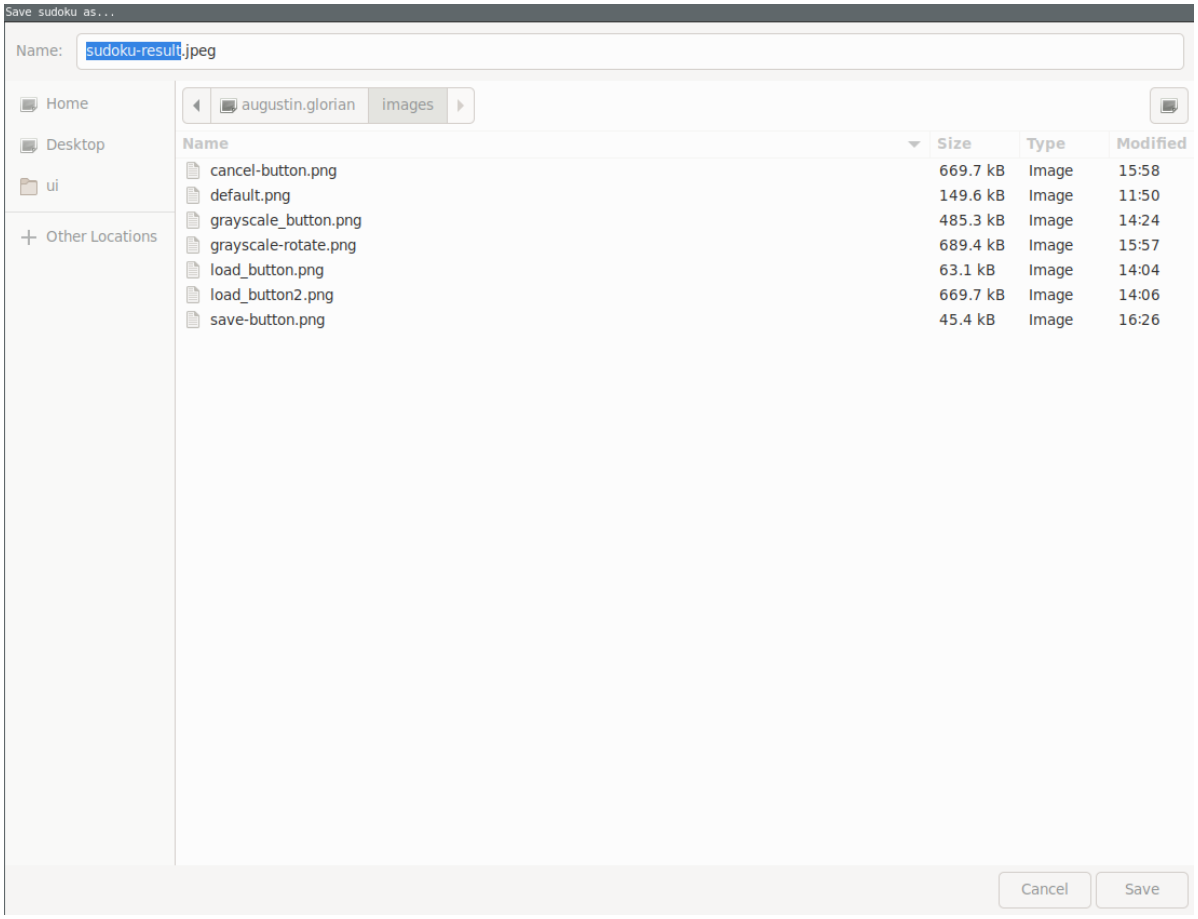


Nous pouvons donc facilement retrouver l'image initiale sans devoir la recharger à chaque traitement effectué sur celle-ci.

4.4.7 Sauvegarder sous (Save as)

Grâce au bouton *Save as*, l'utilisateur peut à tout moment sauvegarder l'image affiché à droite. En effet, grâce à ce bouton, une fenêtre va s'ouvrir et vous pourrez choisir le chemin ainsi que le nom du fichier où vous voulez enregistrer l'image. Par défaut, le fichier sera sauvegarder avec le nom *sudoku-result.jpeg*.

Voici la fenêtre qui s'ouvre lors de l'interaction avec le bouton.



D'ailleurs, elle comporte en titre : *Save sudoku as...* ce qui permet à l'utilisateur de se rappeler ce qu'il doit faire avec cette fenêtre même si le bouton en bas à droite laisse un indice.

4.4.8 Quitter (Quit)

Dès que l'utilisateur à finit ce qu'il voulait faire dans l'application, celui-ci peut la quitter l'application à tout moment en cliquant sur le bouton *Quit* ce qui va fermer la fenêtre de l'application.

5 Conclusion

A travers ce début de projet prometteur, nous avons déjà appris de nombreuses choses concernant le traitement d'image ou le fonctionnement et la mise en application d'un réseau de neurones.

En effet, le réseau de neurone n'a quasiment plus de secret pour Arthur et Samy, l'interface graphique avec glade et gtk est bien connu d'Augustin et Mathis maîtrisent correctement le traitement d'image notamment grâce à le bibliothèque SDL.

C'est grâce aux difficultés rencontrés que l'on à pu acquérir de nombreuses connaissances dans de nouveaux domaines. De plus, ce projet à renforcer notre amitié et notre cohésion d'équipe. Nous sommes donc prêt à affronter les prochains projets et épreuves qui nous attendent dans la suite de notre scolarité à EPITA voir même dans notre vie de développeur.

Après des mois de travail acharné, de résolution de problèmes et plus encore. Nous sommes tellement heureux que notre projet fonctionne. Nous avons tellement appris de ce projet, pas seulement en C mais aussi en gestion de projet et en cohésion de groupe. Ce projet était beaucoup plus difficile que le projet S2, il y avait beaucoup de spécifications, des choses difficiles à comprendre mais nous l'avons fait.