Sodoko

GOPY

Rapport de soutenance n°1

Grégoire Lefaure Oscar Le Dauphin Yness Khouader Pierre Wagniart

29 octobre 2022

Introduction

Présentation du projet

Bienvenue dans le rapport de la première soutenance de notre projet d'OCR ! Pour présenter rapidement le projet [1], c'est un OCR, *Optical Character Recognition*, dont le but est de reconnaitre les chiffres que composent un sudoku classique, ainsi que leur position géographique. Ce programme devra ensuite le résoudre, afficher et enregistrer une nouvelle image, contenant le sudoku résolu.

Voilà pour ce qui est de la présentation générale de ce que notre programme doit faire.

Notre projet

Nous avons choisi, comme la première page le suggère, d'appeler notre équipe « GOPY », ce nom venant de la mise bout à bout des initiales de nos prénoms respectifs (**G**régoire, **O**scar, **P**ierre et **Y**ness).

Nous avons aussi choisi d'appeler notre projet « Sodoko ». Il n'y a pas vraiment de raisons à ce nom, si ce n'est que le nom de « Suduku » nous paraissait un peu vulgraire mais que nous aimions bien la répétition d'une même voyelle dans le mot « sudoku ».

Ce rapport

En ce qui concerne ce rapport, vous pourrez trouver une table des matière regroupant les différents points abordés dans ce rapport. Ce rapport est aussi l'occasion pour nous de faire un point écrit sur notre avancement, l'état de maturation du projet, notre répartition du travail jusqu'ici, etc... Il nous permet aussi d'exprimer notre ressenti sur ce projet depuis son début il y a quelques semaines.

Nous finissions cette introduction en vous souhaitant une excellente et heureuse lecture [2] !

1. Vous devez déjà bien le connaître ce projet :p
2. On s'en doute, ça n'est pas le type d'écrit forcément le plus enthousiasmant à lire :P

Table des matières

1 Qui sommes nous ?

Commençons ce rapport par nous présenter en tant qu'équipe, pour que vous puissiez plus facilement situer le caractère et les intérêts de chacuns.

Grégoire Lefaure

Hello hello ! Moi c'est Grégoire Lefaure, et je suis le chef de projet (même si ce statut est surtout utile pour l'école, et je n'ai pas de supériorité hiérarchique par rapport à mes co-équipiers). J'aime surtout faire du développement `backend` plus que du `frontend`. C'est pourquoi je m'occupe en partie, avec Oscar, de l'architecture du projet et des `Makefile`, dont nous allons parler plus tard dans ce rapport. J'ai d'ailleurs déjà un peu d'expérience par rapport à l'architecture d'un projet, puisque j'ai déjà géré celle de mon projet de S2. Voilà, je crois que le principal est dit me concernant.

Oscar Le Dauphin

Bonjour, moi c'est Oscar Le Dauphin, le O de l'équipe GOPY. M'interéssant beaucoup à l'intelligence artificielle en général, je tenais donc vraiment à m'occuper d'implementer la partie réseau de neuronnes sur ce projet. J'ai hâte de voir le résultat que l'on obtiendra et j'ai hâte de pouvoir jouer avec tous les paramètres du réseau de neuronnes pour mieux les comprendre.

Yness Khouader

Hey, je m'appelle Yness Khouader et je suis fière d'être membre de cette belle équipe ! J'aime beaucoup le design. L'année dernière lors du projet S2 je me suis occupée de la création de la map en 3D. En ce qui concerne ce projet, je m'occupe de la résolution du sudoku ainsi que de l'interface graphique, accompagnée par Pierre. Curieuse et portant un certain intérêt pour l'intelligence artificielle, j'aimerais également me pencher sur le développement du réseau de neurones dont s'occupe à merveille Oscar.

Pierre Wagniart

Bien le bonjour, je suis Pierre Wagniart et je suis l'un des 4 membres de cette équipe ! Tout comme ma collègue juste au dessus, j'aime beaucoup le design, ce qui m'a permis, au projet de S2, de me charger de tout l'aspect graphique du projet (shaders, materials, modèles, direction artistique, etc...). Je me chargerai/me suis chargé de la rotation, d'une partie du traitement de l'image et de l'UI avec Yness. Je pourrais donc me concentrer plus sur l'aspect code et j'ai hâte de faire ce projet que je n'aurais jamais fait seul !

2 Développement du programme

2.1 Nos outils pour développer en équipe et en toute sérénité

2.1.1 Discord

Commençons par l'outil qui nous est le plus indispensable : Discord. C'est en effet ce moyen de communication que nous avons choisi pour nous permettre de communiquer et surtout d'organiser notre communication en différents *channels*. Nous pouvons ainsi y poser toute sorte de questions et nous organiser, par exemple, sur l'écriture commune de ce rapport. Une question sur le comportement d'une fonction d'un co-équipier ? Il suffit de lui poser la question. Nous pourrions continuer ainsi pendant longtemps, mais vous avez probablement compris l'idée.

2.1.2 Github

Le second outil qui est lui aussi indispensable à la construction de ce projet est Github. En effet, c'est cette plateforme que nous utilisons pour héberger notre code, faire des *Pull Request*, des commentaires, etc... Nous n'utilisons en effet pas le répertoire `git` fourni par l'école, car nous le trouvons assez limité par rapport aux fonctionnalités de Github (Pull Request, Continious Integrations, etc...).

Ainsi, pour garder le répertoire fourni par l'école à jour, nous avons choisi un système de `« mirroring »` *fait maison* puisqu'il fonctionne sur une carte programmable type `raspberry pi` chez l'un d'entre nous (nous avons choisi de le faire localemement car nous souhaitions que tous les membres du groupe aient les même droits sur le répertoire Github, ce qui posait des problèmes de sécurité par rapport à la clé SSH de l'un d'entre nous).

Comme dit précédemment, nous utilisons donc beaucoup les `Pull Request` [3]. Nous trouvons en effet que les PR sont très pratiques puisqu'elles permettent aux autres membres de notre équipe de relire notre code et ainsi de limiter les erreurs bêtes, mais aussi d'avoir des retours sur notre coding style, la façons dont nous avons résolu un problème, et tout un tas d'autres remarques. Ces PR nous permettent aussi d'exécuter une CI [4] afin de compiler et tester notre code, pour que la CI vérifie automatiquement que tout se passe bien. Elle nous permet aussi d'analyser notre code et de mettre en évidence certains problèmes dans notre code (des problèmes de types pouvant engendrer des dépassements de capacité, ou bien des failles de sécurité ou des fuites mémoires). Héberger notre code sur Github nous permet aussi de rendre public notre projet, non pas pour que d'autres l'utilisent (vouloir résoudre un sudoku via OCR n'est pas forcément un problème quotidien), même si ce n'est pas impossible pour autant. Non, la portée publique de notre projet s'inscrit surtout dans volonté de maintenir un code le plus « open source »possible.

3. Que nous abrégerons en « PR »dans la suite de ce rapport
4. Continious Integration

2.1.3 Criterion

Pour tester notre code, nous avons choisi l'infrastructure logiciel (framework) de tests Criterion. Ces tests sont executés sur un serveur nous appartenant à chaque fois que l'un d'entre nous push son code. Cela nous permet de rapidement repérer quel est le commit dans une branche qui pose problème, puisqu'il y aura une croix à côté indiquant que nos tests de sont pas passés.

En plus de très bien s'intégrer avec Github, écrire des tests permet une expérience de développement bien plus rapide et agréable. En effet, la technique du `développement piloté par les tests`[5] est très répandue pour les projets de grande ampleur. Elle consiste à écrire les tests unitaires avant les fonctions en elle-mêmes. L'interêt de travailler de cette manière est que nous pouvons avancer plus vite, quand une fonction passe ses tests, on passe à la prochaine, quand on modifie une ancienne fonction, on vérifie directement que les tests passent toujours. Ainsi, en plus d'un développement plus agréable, on ne risque pas de régresser dans l'avancement du projet en cassant des fonctions préalablement écrite sans s'en rendre compte.

2.1.4 Notion

Notion est le dernier outil que nous utilisons souvent pour mener à bien notre projet. C'est un outil collaboratif permettant d'écrire des pages (un peu comme un site internet) de manière collaboratives, avec quelques fonctions intéractives très pratiques comme des rappels, une organisation claire des tâches, etc... La partie publique de ce Notion est d'ailleurs disponible ici : `https://gopy.notion.site`, pendant la durée du projet.

Dans la partie privée du Notion, nous pouvons retrouver quelques ressources qui nous sont utiles pour notre projet (le cahier des charges, pour ne pas avoir besoin de le rechercher dans moodle à chaque fois, etc...). Nous pouvons aussi retrouver une page spéciale dédiée à l'avancement du projet avec une todo list avec la ou les personnes s'occupant de telle ou telle tâche, des notes ou remarques sur la tâche en question, ainsi qu'une deadline pour une tâche donnée.

En ce qui concerne la partie publique de ce site web, elle concerne surtout la documentation de notre projet. Il se divise en trois parties disctinctes.

La première partie contient les pages classiques d'un projet : présentation de l'équipe, comment installer notre programme, un lien vers notre Github, ainsi que des pages donnant quelques tips que nous avons utilisé pour développer notre OCR (des alias git, des configurations vim, etc...).

La seconde partie est composée de *How to*, c'est-à-dire que ce sont des pages qui nous guident dans les différentes étapes de notre projet.

Enfin, la dernière partie, et la plus importante, est la documentation de notre code. En effet,

5. (ou `test driven development` en anglais)

nous souhaitons documenter notre code un maximum pour nos co-équipier, afin qu'ils puissent comprendre comment telle ou telle partie du programme fonctionne, ou même comment utiliser nos fonction dans une autre partie du programme.

2.2 Le développement commun

2.2.1 L'architecture du projet

Pour mener à bien un projet, nous devons utiliser de bons outils, comme présentés dans la section précédente, mais nous devons aussi suivre une architecture cohérente afin de s'y retrouver dans nos différents fichiers. Voici à quoi ressemble l'architecture actuelle du projet :

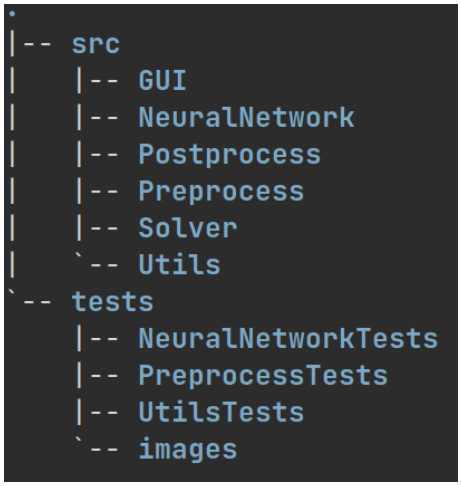


FIGURE 1 – Architecture de notre projet

Nous pensons que les différents noms des dossiers sont suffisament clairs pour ne pas être explicité ici.

2.2.2 Vive les `Makefile`

Cette étape est une partie importante de notre projet. Nous avons d'abord fait un Makefile très basique en se basant sur un template trouvé sur internet, mais plus nous avancions dans le projet, moins ces Makefile étaient adaptés à notre usage. C'est pourquoi nous avons restructuré nos Makefile presque entièrement afin de correspondre pleinement à nos besoins. Avant de commencer à expliquer comment nos Makefile fonctionnent, nous tenions à remercier nos ASM pour leur conférence sur le fonctionnement des Makefile qui nous a été extrêmement utile. Venons-en donc à la structure de nos Makefile.

Nous avons un Makefile principal à la racine de notre projet qui nous permet de compiler entièrement notre projet. Les fichiers de compilations sont placé dans un dossier spécial nommé `build/`, dont l'architecture interne suit celle de notre projet avec tous les fichiers `.o` et `.d`.

Nous avons ensuite un autre Makefile qui nous permet de compiler seulement certaines

parties de notre projet (uniquement le `Preprocess`, le `Solver` ou autre). Ces Makefile nous permettent donc de tester manuellement notre code sans pour autant compiler tout le projet.

Enfin, notre dernier Makefile est notre Makefile situé dans `tests/`, et, comme son nom l'indique, nous sert à exécuter des tests automatiques sur notre projet. Si ce n'est pas déjà fait, il appelle le Makefile principal pour compiler tout le projet, puis compile ensuite les tests avant de les exécuter.

Ce sont le Makefile principal et le Makefile de tests qui sont utilisés dans les CI pour tester notre code avant de le merge sur la branche principale de notre répertoire distant.

2.2.3 Notre boite à outils

Cette partie va présenter les fonctions communes à plusieurs parties de notre OCR. En effet, certaines fonctions sont communes à plusieurs parties de notre programme, que ce soit pour nous permettre de facilement tester notre code, ou simplement de nous éviter de réécrire plusieurs fois la même fonction. Dans cette partie nous allons trouver des fonctions classiques de chargement d'images, de sauvegardes d'images, des fonctions de manipulation de matrices, de chaines de caractères, de listes, ainsi que de nombreuses autres fonctions.

2.3 Le pré-traitement

2.3.1 255 nuances de gris

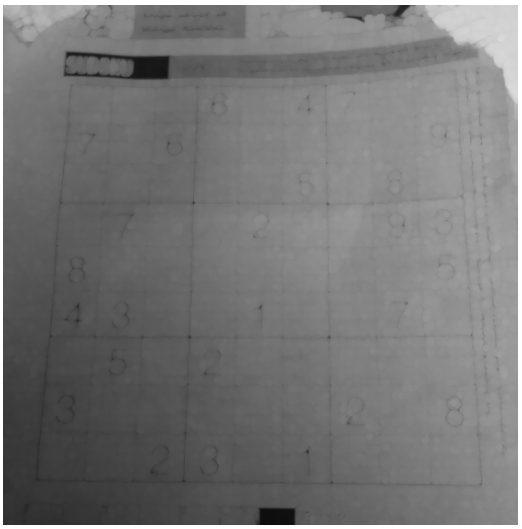
Première étape du prétraitement, elle est donc très importante. Ainsi, elle permet de simplifier tous les algorithmes suivants, non seulement dans le pré-traitement mais également pour le réseau neuronal. Et quelle meilleure manière peut-on avoir de simplifier un algorithme que de simplifier son entrée ? Le principe est donc ici de prendre en entrée une image classique et de parcourir chacun des pixels de celle-ci afin d'appliquer une formule mathématique sur les valeurs RGB des pixels $((rouge + bleu + vert)/3)$. Ainsi on fait une moyenne qui gomme toute couleurs "classiques" et on obtient, en sortie, une image en échelle de gris facilement utilisable.

2.3.2 Aie, j'ai mal aux yeux !

Cette étape du pré-traitement est importante car en une seule étape, c'est-à-dire avec un seul algorithme, nous corrigeons la luminosité de l'image et son contrast, mais c'est aussi l'occasion de lisser l'image, c'est-à-dire de diminuer les bruits parasites qui pourraient poser problème au réseau de neurones. Pour faire cela, nous utilisons un filtre et plusieurs passages de ce filtres dans différents sens. Le premier passage est appelé `dilation`. Ce passage consiste à appliquer le maximum du filtre à un pixel, et ce pour chaque pixel de la surface. Comme une image vaut plus que mille mots, voici un exemple d'avant-après dilation d'une image :

(a)

Avant `dilation`



(b)

Après `dilation`

Sur le même principe, mais en applicant le minimum du filtre à chaque pixel, nous avons ensuite appliqué une `erosion`. Voici l'effet d'une erosion sur l'image de base :

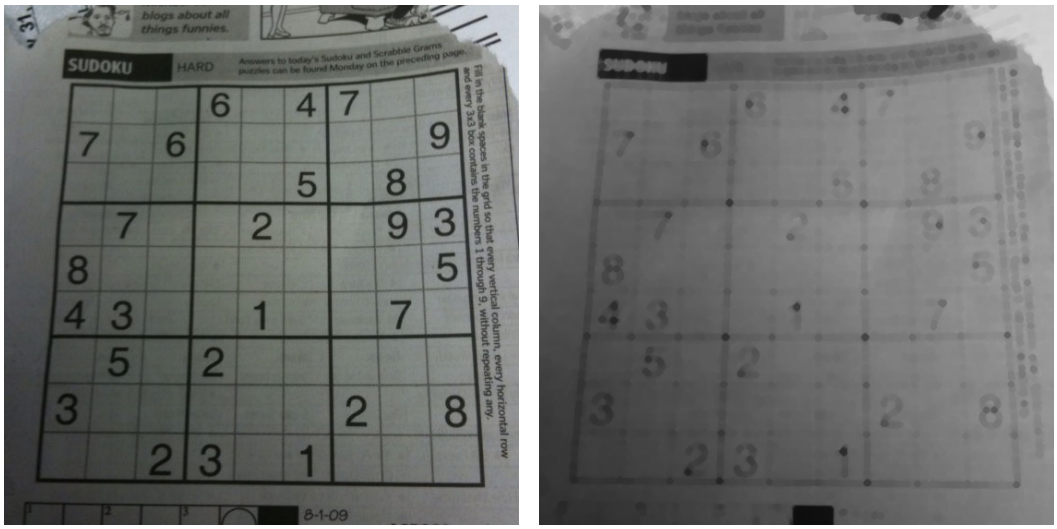
(a)

Avant `erosion`

(b)

Après `erosion`

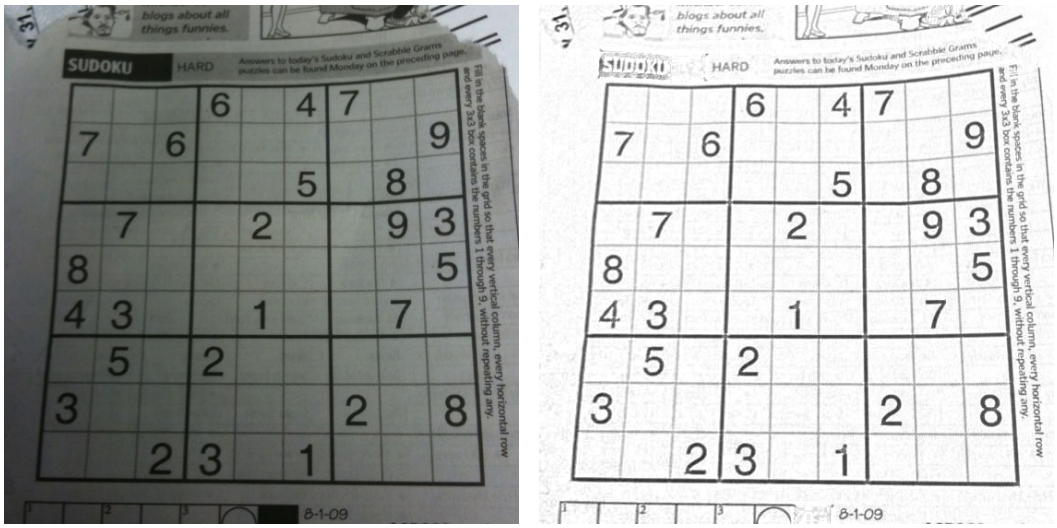
Nous avons donc combiné ces deux transformations dans ce que l'on appelle une étape de `closing`. Cette étape consiste à appliquer une `dilation`, puis appliquer une `erosion` au résultat. Voici le résultat avant et après l'étape de closing :



(a) (b)

Avant `closing` Après `closing`

Le résultat du closing peut paraitre assez peu utilisable tel quel, et c'est en effet le cas. Il va falloir lui appliquer une dernière transformation. Cette dernière transformation peut paraitre un peu magique (et c'est le cas), consisite en la division de l'image de base et de l'image obtenue après l'étape du closing. Vous allez me dire, comment on divise deux images ? [6] Et bien c'est simple, il suffit de diviser leurs pixels entre eux un à un, et comme par magie, nous obtenons ça :



(a) (b)

Avant correction Après correction

Voilà pour ce qui est de la partie sur la correction de la luminosité et du contraste et de

6. Vous ne direz probablement pas ça car nous ne devons pas être les premiers étudiants à avoir utilisé cette technique XD

la réduction de bruit. Nous espérons que ces comparaisons des images et ces explicitation des différentes étapes de cette partie vous aura permis de comprendre ce que nous avons voulu faire et comment cela fonctionne.

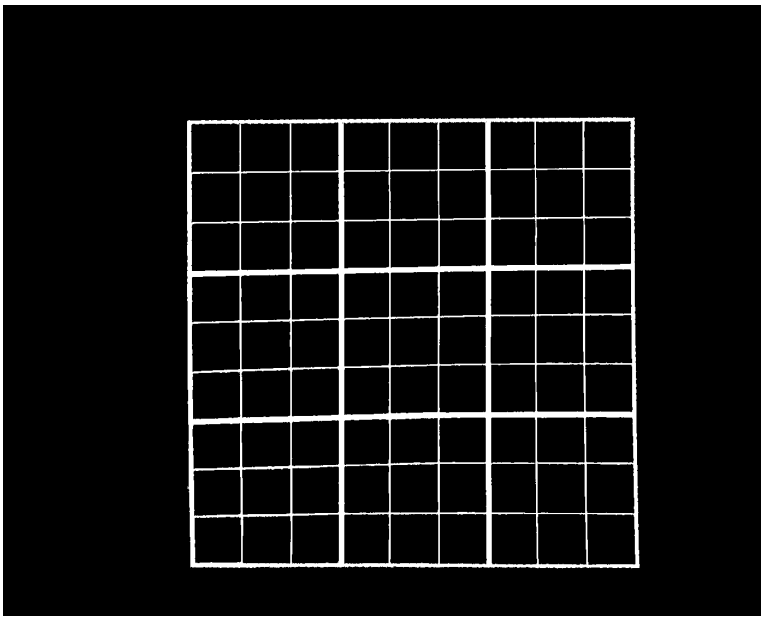
Passons maintenant à la binarisation des images, qui est la suite logique ce la correction de contraste et de la réduction de bruit.

2.3.3 Il fait tout noir !

Vous l'aurez probablement compris, cette partie concerne la binarisation des images, c'est-à-dire la mise en noir et blanc des surfaces. Cette partie du pré-traitement est une partie qui a été assez rapide et facile à implémenter puisqu'il nous a suffit de faire une comparaison de chaque pixel avec une valeur seuil. Nous avons arbitrairement pris `127` comme valeur de seuil pour définir la valeur d'une composante d'un pixel, et donc de définir s'il est soit tout blanc, soit tout noir.

2.3.4 Mais où est la grille ?

Pour trouver la grille c'est simple ![7] On commence par diviser les images en différents sous-composants [8] grâce à un algorithme de remplissage par diffusion (flood-fill en anglais). Une fois une liste de composant établie, on ne garde que le composant qui a l'air le plus grand (le plus de pixels connectés entre eux). Notre image ressemble alors maintenant à ça :



De cette image, il ne nous reste plus qu'à chercher les pixels blancs aux extrémitées de la grille, ce qui est plutôt trivial.

7. (non)
8. Une sous-composante d'une image est un ensemble de pixel tous reliés entre-eux

2.3.5 Tournez les images !

Il est nécessaire pour le réseau de neurones d'avoir une image bien droite, or, force est de remarquer que toutes les images ne sont pas prises droites. C'est face à ce défi incommensurable que notre équipe a pris la première marche menant à la solution qu'est la rotation automatique. En effet, nous sommes désormais capables de corriger l'angle d'une image manuellement afin de la remettre droit. La prochaine étape sera de détecter l'angle de correction automatiquement. Pour en revenir à l'implémentation, celle-ci ne fut pas sans peine. En effet, après s'être rendu compte qu'il n'existait pas de fonction permettant directement d'effectuer une rotation dans SDL2, nous avons cherché une manière de le faire entièrement par nous même. La solution trouvée était de faire une interpolation bilinéaire qui permet de calculer la nouvelle position d'un pixel en un point quelconque, à partir de ses deux plus proches voisins dans chaque direction. L'implémentation fut ardue, notemment car, contre toute attente, les cours de physique étaient nécessaires, avec l'utilisation des coordonnées polaires, simplifiant **beaucoup** les calculs. On se retrouvait ainsi avec un monstre de fonction faisant quelques 200 lignes et c'est ainsi que lors du débuggage, une fonction pré-faite dans une librairie annexe de SDL et faisant exactement la chose que l'on voulait, fut trouvée. Après quelques objets cassés et un très grand soupir, les 200 lignes n'en devinrent qu'une seule, laissant un goût amer dans nos bouches ainsi qu'une belle leçon sur la documentation.

2.3.6 Découpons-les...

Cette partie du pré-traitement est la dernière étape avant la reconnaissance des chiffres par le réseau de neurones. C'est donc une étape importante puisque si les découpes ne sont pas correctes, les chiffres pourraient être coupé et donc inreconnaissables.

La technique que nous avons choisi d'utiliser pour cette partie est simple : nous partons du principe que le sudoku est correctement cadré, et nous découpons simplement l'image en 81 cases en divisant la largeur et la hauteur par 9. Cette technique à l'avantage d'être simple à implémenter et rapide à exécuter. Le cadrage de l'image n'est quant à lui pas un soucis très important puisque ce cadrage est assuré par les différentes étapes précédentes du pré-traitement. Le recadrement automatique de l'image n'est pas encore disponible, mais les informations nécessaires sont mise à disposition par la partie détection de la grille. La seule étape manquante pour le recadrage automatique, et donc le fonctionnement parfaitement autonome de cette ultime étape de pré-traitement, est l'interpolation des grilles de sudoku qui ne sont pas parfaitement droites.

2.3.7 Du noir et blanc à la reconnaissance...

Nous allons donc maintenant conclure sur l'avancement du pré-traitement, qui est une partie importante de notre OCR. Nous pouvons dire que nous avons bien avancé et que le pré-

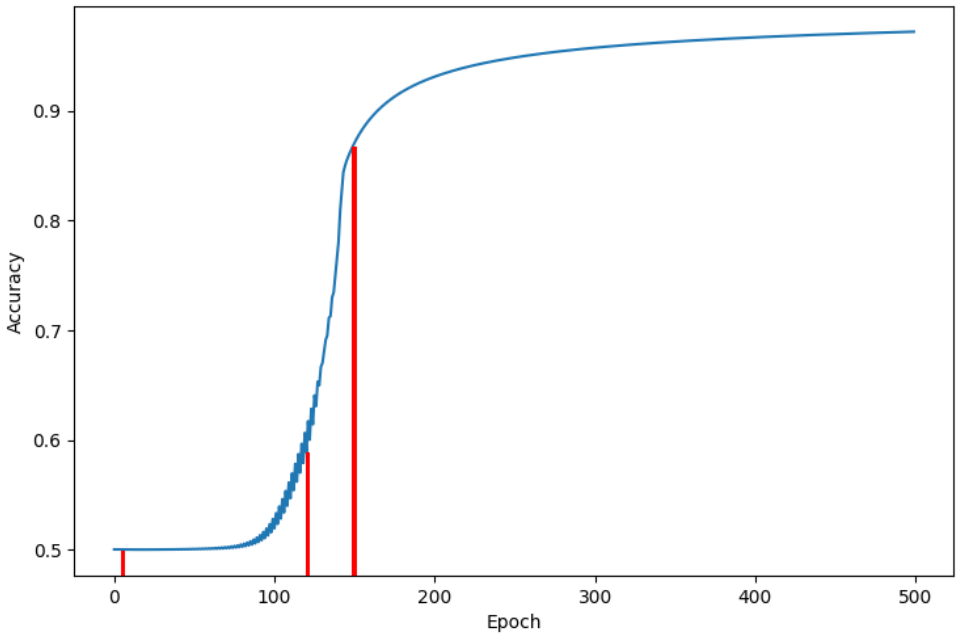
traitement est disponible dans sa forme presque définitive. Les principales fonctionnalités sont disponibles et fonctionnent correctement.

Notre objectif pour la dernière soutenance est donc de finir complètement le pré-traitement en ajoutant la rotation automatique ainsi que l'interpolation des images, mais ces deux étapes ne devraient pas poser trop de problèmes par la suite du projet. Nous devrons aussi peut-être régler avec plus de précision les filtres utilisés dans la partie correction du contrast, de la luminosité et suppression du bruit, pour les problèmes évoqués plus haut dans cette même partie.

Passons maintenant à l'étape suivante de notre OCR : le réseau de neurones.

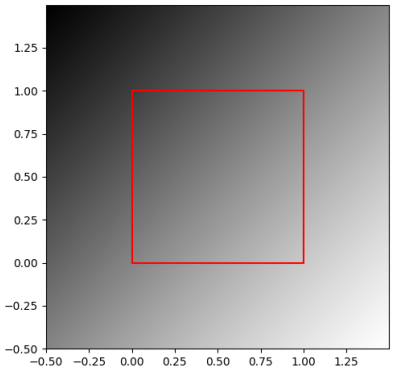
2.4 Le réseau de neurones

On ne va pas vous présenter les réseaux de neurones ici, on considère que les centaines de rapports d'anciens étudiants l'on déjà très bien fait.
On notera les réseaux de neurones (entrées,[couches[1],couches[2],...,couches[n]]). Cette notation représente le nombre de d'entrées suivi d'une liste de nombre de neuronnes pour chaque couche. Le nombre de neurones de la dernière couche est aussi la taille de la matrice de retour du réseau de neurones (nombre de sorties).
Par exemple, le réseau de neurones xor le plus simple est le suivant : (2,[2,1]). Il possède 2 entrées, une couche de 2 neurones, et une couche de 1 neurone. Il a donc 1 sortie.

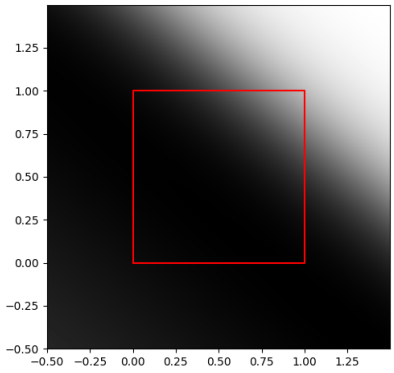


On peut visualiser la fonction que représente ce réseau de neurones au cours de son apprentissage :

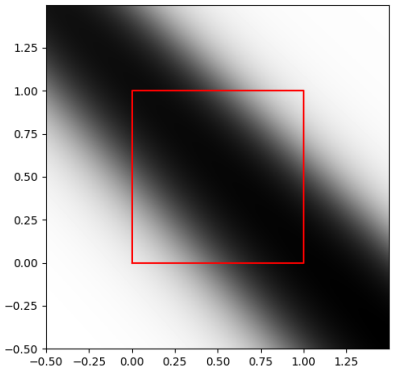
On appellera epochs, le nombre de fois que le réseau de neurones est passé sur l'entièreté du jeu de données.



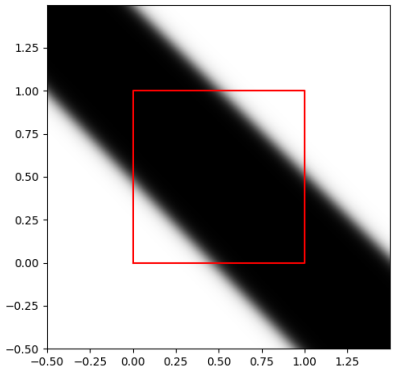
(a) Avant l'apprentissage



(b) Après 120 epochs

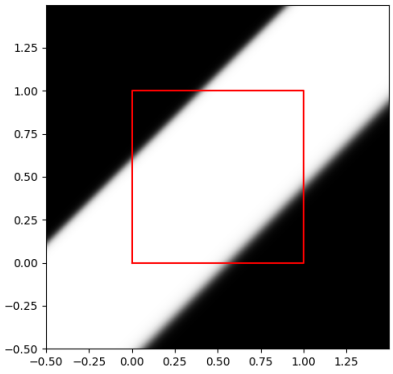


(c) Après 150 epochs

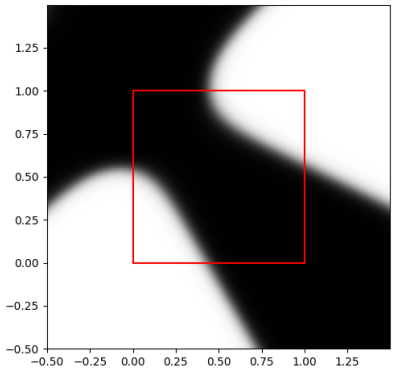


(d) Après 1000000 epochs

Cette fonction (d) est bien un xor, si on regarde les coins du carrés, ceux situés en (0,0) et (1,1) on une valeur de 1 (blanc) et ceux situés en (1,0) et (0,1) on une valeur de 0 (noir).
Voici des examples de sorties de réseaux de tailles alternatives :



(a) réseau 2,[2,1,1]



(b) réseau 2,[3,2,1]

2.5 La résolution

2.5.1 Et si nous chargions la grille ?

Le chargement de la grille n'est rien d'autre qu'une lecture d'un fichier on ne peut plus classique, puisque nous partons du principe de fichier passé en paramètre est correctement formaté.

2.5.2 Résolvons-la !

Bien ! Maintenant que nous avons pu transformer l'image en une fichier texte puis une matrice, nous allons pouvoir entreprendre la phase de résolution de ce sudoku. Résolution de sudoku ? Quelle coincidence ! L'année dernière nous avons justement eu la chance de travailler sur ce sujet en cours de programmation. Ainsi, nous avons pu aisément développer les différentes fonctions nécessaires à la résolution d'un sudoku. Parmi elles nous pouvons citer : - IsLineValid (respectivement IsColumnValid) qui vérifie qu'une ligne (respectivement colonne) de la grille est valide - IsSquareValid qui détermine si une grille de 3 par 3 est valide - IsBoardValid qui renvoie si la board est valid ou non en utilisant donc les fonctions précédemment citées - IsSolved qui determine si la grille est résolue - Et enfin Solve, qui résout la grille de sudoku

C'est bien beau mais à présent, il est temps de tester notre solver. L'élaboration de tests unitaires pour nous permettre de tester toutes ces fonctions fu quelque peu plus compliquée. Certains membres ont du se pencher sur les warning pour tenter de les résoudre. Un vrai travail d'équipe a été nécéssaire. Bien joué à nous ! Nous avons résolu le sudoku !

Quant au délai de livraison, l'objectif était de vous fournir un solver fonctionnel à la première soutenance. Chose faite. Nous avons pour objectif de l'optimiser pour la dernière échéance.

A présent il nous faut enregistrer cette grille résolue. Passons donc à l'étape suivante !

2.5.3 Enregistrement en cours...

Sur le même principe que le chargement du fichier, la sauvegarde se fait simplement en traduisant la matrice résolue en chaine de caractères, puis en écrivant le tout dans un fichier.

2.6 C'est déjà la fin...

Nous en avons fini pour ce qui est la partie développement de notre projet. Comme vous avez pu le voir, il manque encore quelques parties à notre projet pour qu'il soit complètement fonctionnel : une interface utilisateur, l'affichage du résultat de la résolution sous forme d'image, certaines étapes du pré-traitement qui ne sont pas faites ou pas finies, mais aussi 1bien plus important, la mise en relation de toutes nos parties. Ces différents points seront donc encore à travailler dans le futur et seront prêts pour la soutenance finale pour que nous puissions vous présenter un projet complet et fonctionnel. Nous arrivons donc à la presque fin de ce rapport.

Nous allons, dans la prochaine partie, exprimer nos ressentis par rapport à ce projet et à notre avancement.

3 Notre ressentis sur le projet

Grégoire Lefaure

Nous revoilà entre nous comme au début de ce rapport lors des présentations. Je vais d'abord donner mon ressenti en tant que chef de projet, avant d'exprimer un ressenti plus personnel sur mon travail et mon organisation. Pour commencer, nous ne nous connaissions pas tous avant de commencer ce projet, mais cela n'a pas semblé être un problème pour la bonne entente au sein du groupe. Malgré le fait que l'entente au sein du groupe n'ait pas été un problème, notre organisation globale n'a pas été parfaite pour autant. En effet, nous nous étions dit que la partie code devait être finie une semaine avant la date de rendue, mais nous n'avons pas réussi. Je crois que c'est d'ailleurs le principal problème que je vois au sein du groupe en tant que tel, avec le fait que les conséquences du dépassement de ces deadline ait pu impacter nos révisions de midterm, puisque pour ceux qui ont due faire du travail en plus, ce temps a été pris au dépend de leurs révisions.

En ce qui concerne le ressenti personnel sur ce projet, je dois dire que je ne pensais pas qu'il m'intéresserait autant. En effet, au début du projet, j'ai due me forcer à travailler car le sujet, à première vue, ne me passionnait pas plus que ça, mais lorsque j'ai commencé à m'y intéresser, j'ai vue que c'était assez satisfaisant que les transformations appliquées aux images fonctionnent. Gérer les Makefile et le git est aussi une partie qui m'a beaucoup intéressé dans cette première partie du projet, et j'espère que ça m'intéressera toujours autant dans cette deuxième partie du projet.

Oscar Le Dauphin

Ce projet d'OCR a vraiment été une activité plaisante pour moi au cours de ces dernières semaines. Je dois avouer que même si la détection de la grille m'a donné envie de tout abandonner (les résultats n'avaient aucun sens durant de longs jours), le rendu en a largement valu le coût. De même pour le réseau de neurones XOR, une fois que mes fonctions marchent je prends toujours plaisir à jouer avec ce que j'ai codé afin de le rendre meilleur ou simplement de mieux comprendre leurs comportements avec différentes entrées ou d'autres paramètres. Ce que j'apprécie tout particulièrement dans notre projet, c'est notre système de tests. La fonction d'apprentissage du réseau de neronnes est dépendante du bon fonctionnement de plusieurs dizaines de sous fonctions, et si les tests n'étaient pas là pour m'assurer qu'elles sont correctes, le débogage auraient sûrement été douloureux.

Yness Khouader

Pour faire un bilan de cette première échéance, j'estime ne pas avoir passé assez de temps à travailler sur ce projet, dû à un certain laxisme dans les débuts et une organisation à revoir bien que je sois très intéressée par ce projet que je trouve amusant. Ainsi, j'aurais pu essayer d'être plus autonome face à certains obstacles au lieu d'être constamment épaulée à partir de l'élaboration de tests pour le solver par Grégoire et Oscar. Plus motivée, je souhaite gagner en autonomie afin d'améliorer mes compétences en programmation en langage C et m'organiser de telle sorte que le délai de livraison soit bel et bien respecté.

Pierre Wagniart

L'organisation a été particulièrement compliquée lors de la première partie du projet. Il m'a toujours été trop facile de sacrifier l'OCR pour bosser sur d'autres matières/projets, parfois par manque de temps, mais également par manque d'envie, malheureusement. Heureusement, je me suis amélioré sur ce point-là vers la fin, notamment grâce à notre merveilleux chef de projet et je suis prêt à aborder la deuxième phase avec une ardeur renouvelée ! D'un autre côté, la bataille que j'ai menée avec SDL2 fut épique et m'a permis de m'améliorer sur la recherche de documentation, la prise en main du C et de git (qui était assez ésotérique jusqu'ici).Ainsi, pour garder le répertoire fourni par l'école à jour, nous avons choisi un système de « `mirroring` » *fait maison* puisqu'il fonctionne sur une carte programmable type `raspberry pi` chez l'un d'entre nous (nous avons choisi de le faire localemement car nous souhaitions que tous les membres du groupe aient les même droits sur le répertoire Github, ce qui posait des problèmes de sécurité par rapport à la clé SSH de l'un d'entre nous).

4 Conclusion

Nous arrivons déjà à la fin de ce rapport de soutenance. Nous y avons décrit notre projet, son avancement, nous en équipe et nous en tant qu'individu. Nous espérons que vous avez trouvé toutes les informations nécessaires à propos de notre projet dans ce rapport. Il reste encore beaucoup de choses à faire dans ce projet, mais cela ne nous décourage pas pour autant, et nous pourrons donc nous retrouver dans quelques semaines avec un nouveau rapport qui, cette fois, présentera notre projet entièrement terminé et foncitonnel.

Nous vous remercions de votre lecture et espérons surtout que le contenu proposé aura su vous intéresser.

La suite dans quelques semaines...