Rapport de première soutenance

Projet OCR

Kaisetsu Corp

Tom Brossard, Kylian Bozec, Quentin Hay–Kergrohenn et Pierre Fonseca R1

10 Novembre 2022

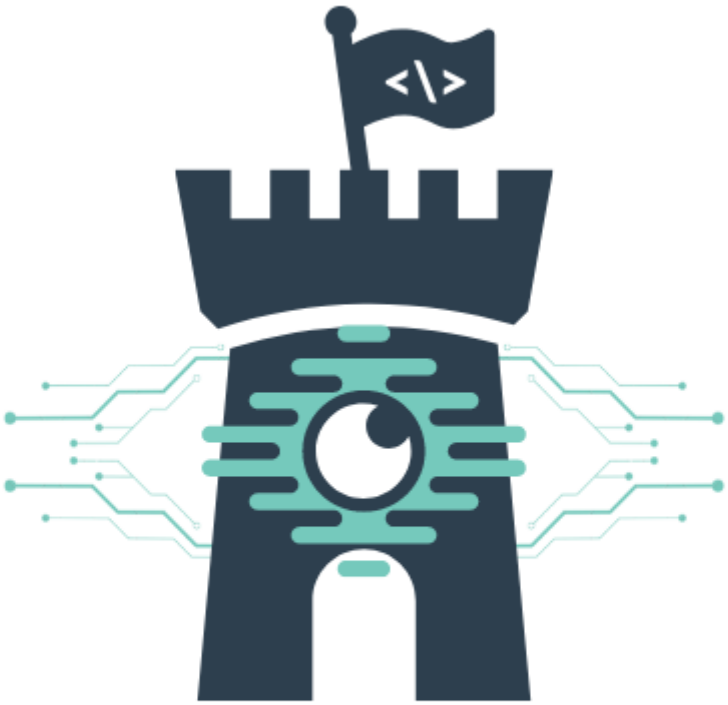


Table des matières

1 Introduction

1.1 Présentation du projet

Nous avons pour objectif de réaliser un logiciel de type O.C.R, c'est-à-dire une reconnaissance optique de caractères qui va résoudre une grille de sudoku à partir de l'image de cette dernière donnée en paramètre au logiciel. Pour ce faire, nous avons pour contrainte de le réaliser entièrement en langage C sur une durée d'un semestre et par groupe de quatre.

1.2 Présentation du groupe

Nous sommes la Kaiketsu Corp, composée de Tom Brossard, Kylian Bozec, Quentin Hay–Kergrohenn et Pierre Fonséca. Nous sommes tous les quatre étudiants en classe préparatoire dans l'école d'ingénieur en informatique EPITA Rennes.

2 Répartition des tâches

Ci-dessous, le tableau de répartition des tâches individuelles réalisées par chacun pour la première soutenance :

	TOM	KYLIAN	QUENTIN	PIERRE
Solver :			X	
Affichage image :				X
Prétraitement :				X
Rotation de la grille :			X	
Détection de la grille :		X		
Réseau de neurone :	X			
Rapport :				X

3 Avancement

3.1 Résolution du sudoku

Pour la résolution du sudoku, on a utilisé le principe de backtracking. Pour chaque case vide, on cherche les chiffres possibles, puis on teste le premier chiffre et on recommence avec la prochaine case vide jusqu'à résoudre le sudoku. Si pour une case vide il n'y a pas de chiffres possibles, on revient à la case testée précédente et on teste le chiffre suivant. En cas de résolution, un fichier avec l'extension ".result" est créé. S'il n'est pas résoluble, on ne change rien.

3.2 Traitement de l'image

Tout d'abord, il faut savoir que l'ensemble de notre traitement sur l'image a été réalisé grâce à la librairie SDL (Simple Direct-Media Layer) qui permet le développement de logiciels multiplateforme.

Avant de pouvoir faire tout type d'action avec notre image, il fallait tout d'avoir l'afficher. Pour cela, il suffit de lancer notre fonction main.c et de taper la commande ./test " nom du fichier de l'image" (commande temporaire). Ainsi, en lançant ce programme l'image s'affiche.

Image affichée de base.

Suite à cela la première partie du traitement de l'image pouvait commencer, par la suppression des couleurs, pour cela trois étapes étaient nécessaires. Tout d'abord, transformer chaque pixel de l'image en une nuance de gris selon sa propre couleur, c'est l'étape grayscale.

Image avec l'application du grayscale.

Puis la deuxième étape était le filtrage, c'est-à-dire de flouter l'image pixel par pixel afin d'uniformiser les zones de couleur dans l'image. Pour cela nous avons opté pour l'algorithme du flou gaussien. Le principe est le suivant : on travaille sur chacun des pixels de l'image, on dessine une matrice d'un rayon choisi arbitrairement de taille rayon * rayon. Suite à cela on récupère la médiane de l'ensemble des couleurs des pixels environnants (selon le rayon), puis on applique la médiane au pixel sélectionné et on recommence le processus sur l'ensemble des pixels.

Image avec le filtrage.

Enfin, la dernière étape était la binarisation (mettre l'image en noir et blanc). Pour cela, il a fallu calculer un seuil de binarisation afin de savoir si tel ou tel pixel devait devenir noir ou blanc. Pour obtenir le seuil on effectue le calcul suivant :

Seuil = moyenne niveau de gris d'une image / nombre total de pixel

Ainsi, on parcourt l'ensemble des pixels de l'image et si le niveau de gris du pixel est inférieur au seuil trouvé, alors on le transforme en pixel noir, sinon en pixel blanc.

Image avec la binarisation.

3.2.1 Prétraitement

3.2.2 Rotation de l'image

La rotation d'une image est l'étape qui a monopolisé l'entièreté du temps que j'ai passé sur l'O.C.R depuis le début de ce projet. Nous pourrions nous dire que faire tourner une simple et assez simple et ne requiert pas tant d'effort qu'un réseau de neurones (par exemple). Cependant, la pratique est, comme souvent, bien différente de la théorie. Tout d'abord, la rotation d'une image manipule, de toute évidence, des images. Il m'a donc fallu dans un premier temps bien m'imprégner du fonctionnement de la SDL, librairie majoritairement utilisée pour notre projet. Une fois plongé dans l'univers de la SDL, le premier problème est venu assez rapidement : les ressources. En effet, la rotation d'une image en programmation n'est pas le sujet où nous pouvons trouver le plus de documentations et de ressources. Le peu que nous trouvons n'est pas dans le bon langage mais ce n'est qu'un petit détail résolvable très facilement. Le plus gros problème lié à ce manque de documentation est, de toute logique, de ne pas savoir comment procéder pour faire la rotation d'une image. Cependant, après un certain temps à creuser dans les tréfonds du web, certaines pages m'ont paru très intéressantes pour commencer cette rotation.

Pour effectuer cette rotation, j'ai procédé en plusieurs étapes. La première était d'afficher simplement une image, c'est-à-dire de prendre une image en paramètre et de l'afficher dans une fenêtre. La deuxième étape était déjà un peu plus ardue : La rotation à 90 et 180 degrés. En effet, ces rotations sont les plus "simples" car elles ne nécessitent qu'un changement simple de coordonnées des pixels de l'image de base. En l'occurrence, nous utilisons la largeur et la hauteur de l'image de base afin de redéfinir le positionnement des pixels dans la nouvelle surface. En effet, lorsque nous effectuons une rotation à 90° sur une image avec une hauteur "h" et une largeur "w", alors dans la nouvelle image rotate, "h" sera égale à "w" et vice-versa.

Ce principe nous permet de déplacer les pixels de la surface de base à la nouvelle surface. Le deuxième gros problème est apparu peu après : Le Segmentation Fault. Après de nombreuses recherches et plusieurs tasses de café, j'ai trouvé d'où venait le problème : la création de la surface finale (la surface contenant l'image tournée). En effet, la surface finale était initialisée à "NULL" et donc, lorsque le programme essayait d'accéder à sa largeur et sa hauteur, il y avait une segmentation fault. Ainsi, après avoir régler ce problème (et quelques petits autres...) nous avions une rotation fonctionnelle sur un angle fixe de 90 et 180 degré (cf. image 1).

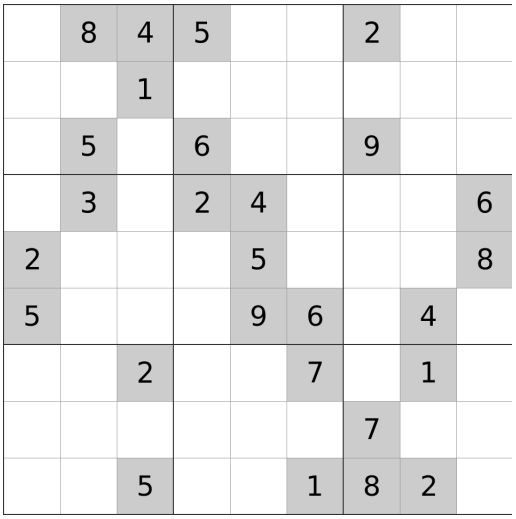


Image de base

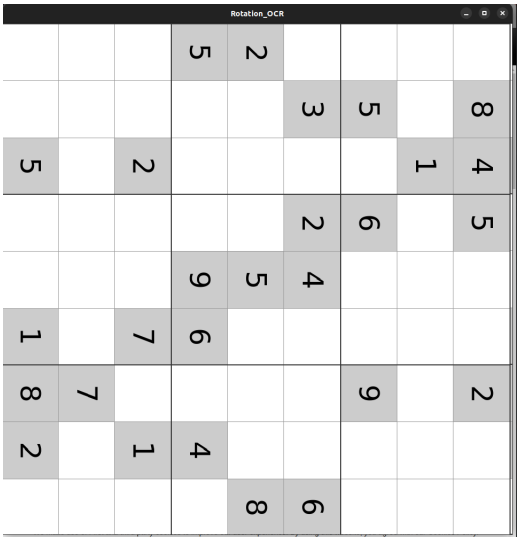


Image 1 avec rotation

Or, ce que nous voulons, c'est une rotation de n'importe quel angle entre 0 et 360 degré. Il nous fallait donc utiliser des formules mathématiques plus complexes que pour la première partie... Après quelques recherches, j'ai trouvé les formules correspondantes au calcul de la nouvelle position de chaque pixel selon un angle x. Pour mieux comprendre, voici un schéma illustrant le principe sur un point (cf. image 2).

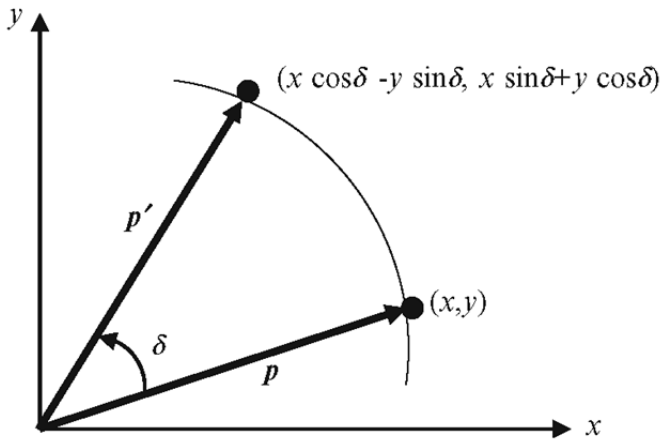


Image 2

Il nous fallait donc appliquer ce changement de coordonnées sur chacun des pixels de la surface de base afin d'avoir notre surface finale rotate. Un problème important s'est pourtant posé après cela, l'image obtenue lors de la rotation n'était pas vraiment... celle attendue (cf. image3).

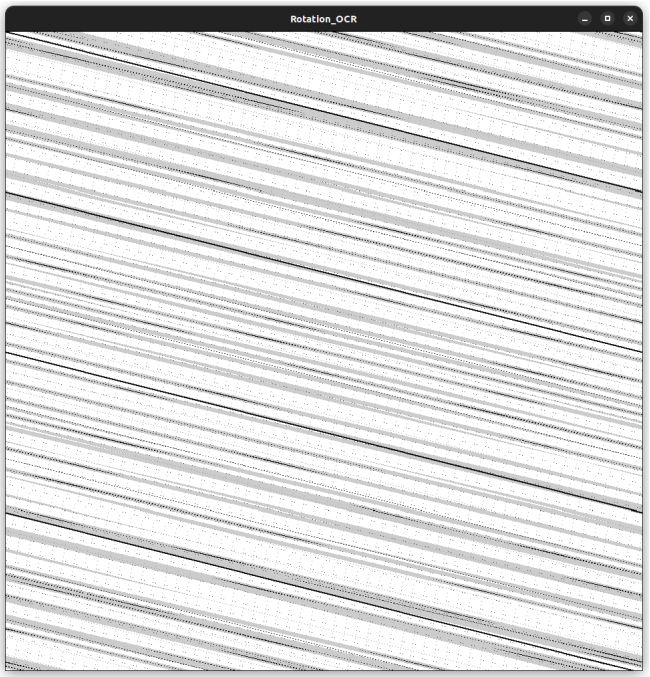


Image 3

Il s'est avéré plus tard que ce problème venait en fait d'une conversion forcée d'un nombre de type "double" en type "int". Cela affectait la coordonnée de changement du pixel qui donnait un résultat arrondie. Finalement, après beaucoup d'efforts, nous avions un résultat de rotation qui fonctionnait. Or, un dernier problème se posait : l'encadrement de l'image. En effet, lors d'une rotation autre que 90 ou 180, les bords de l'image sortent du cadre du fait de la rotation. Pour régler ce problème, l'idée était de définir une surface avec une largeur et une hauteur plus grande que celle de base afin d'avoir notre nouvelle image parfaitement encadrée et rotate. Le principe est de récupérer les coordonnées de chaque coin de notre image de base puis d'appliquer une "pseudo-rotation" sur ces coordonnées, en prenant la taille de notre surface rotate, afin d'obtenir un encadrement pile à la bonne taille.

Ainsi, après quelques changements de coordonnées et de formules, nous sommes arrivés à une rotation tout à fait correcte, qui était bien encadrée et qui respectait la rotation donnée. Enfin, après tout cela, un problème subsiste toujours : la superposition d'une partie de l'image sur les bords (cf. image 4). Malgré de nombreux efforts, ce problème n'est malheureusement pas encore réglé.

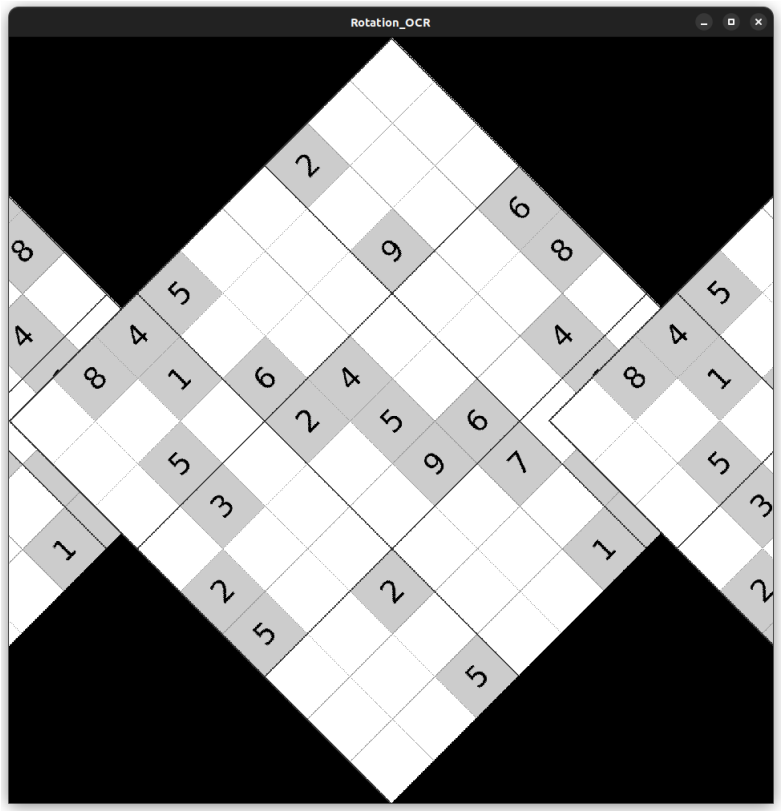


Image 4

3.3 Détection de la grille de sudoku

La détection de ligne nécessite un certain nombre d'étapes. Tout d'abord, on utilise un filtre de Canny pour la détection de contour. Il a pour avantages d'avoir une bonne détection (faible taux d'erreur dans la signalisation des contours), une bonne localisation (minimisation des distances entre les contours détectés et les contours réels), et une clarté de la réponse (une seule réponse par contour et pas de faux positif). Pour la bonne application du filtre, il applique un filtre gaussien sur l'image, précédemment fait, puis le calcul de gradient pour chaque pixel. Le gradient permet de détecter les variations d'intensité entre pixels horizontalement et verticalement. Ensuite on calcule l'orientation du contour avec la fonction suivante :

$$\text{theta} = \arctan(\text{Gx} \ / \ \text{Gy}).$$

On supprime après les points non-maxima, on supprime les pixels dont la dérivée des lignes de champs est nulle. Pour finir on applique un seuil en dessous duquel Gx et Gy sont trop faibles pour être considérés comme un contour.

3.4 Réseau de neurone

Afin de déchiffrer nos images pour en extraire les chiffres présents sur la grille, nous avons décidé d'implémenter, comme demandé, un réseau de neurones, élément très utilisé dans le domaine du "Machine Learning". Il existe de nombreux modèles différents de réseaux de neurones ayant chacun leurs avantages et défauts. Pour notre projet, nous avons décidé de partir sur un modèle expressif (peu profond) le plus facile à implémenter, mais aussi le moins optimisé. (voir figure 1) Si nous avons le temps, nous prévoyons d'implémenter un réseau de neurones à convolution qui est bien plus efficace pour le traitement d'images.

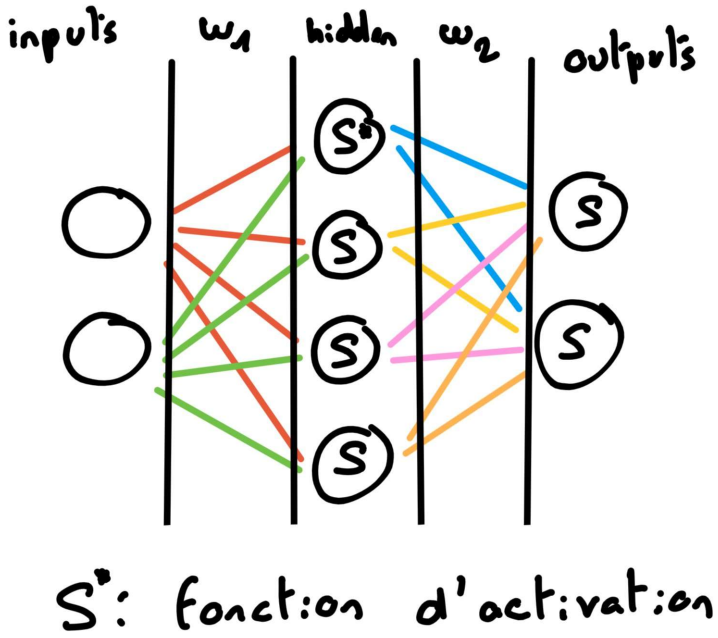


Figure 1 : Réseau de neurones expressifs.

Un réseau de neurones seul n'est pas vraiment utile, en effet qu'en faisons-nous ? Cette implémentation est une sorte de système de stockage qui fonctionne de la même manière que notre cerveau (en version simplifiée).

À l'instar d'avoir des neurones, et des synapses biologiques, nous utilisons des neurones artificiels qui vont performer à l'aide de simple somme de valeurs en entrée et applique une fonction d'activation au résultat total pour ressortir une nouvelle valeur qui sera utilisée par les neurones suivants.

Pour connecter chaque neurone, nous remplaçons les synapses par des "poids" qui vont avoir le même fonctionnement que ces dernières, c'est-à-dire récupérer la valeur du neurone précédent pour l'affecter plus ou moins au neurone actuel. Un biais est presque toujours utilisé afin d'obtenir des résultats plus précis (voir figure 2 et 3).

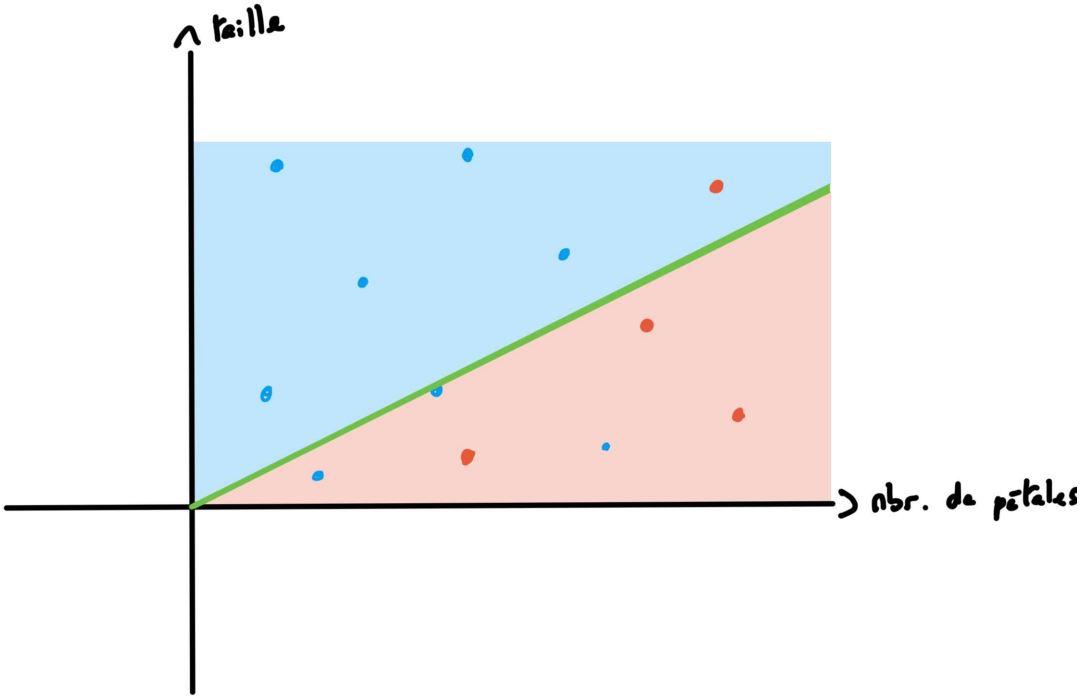


Figure 2 : Résultat sans biais sur un exemple de fleurs dont on compare leur nombre de pétales par rapport à leur taille.

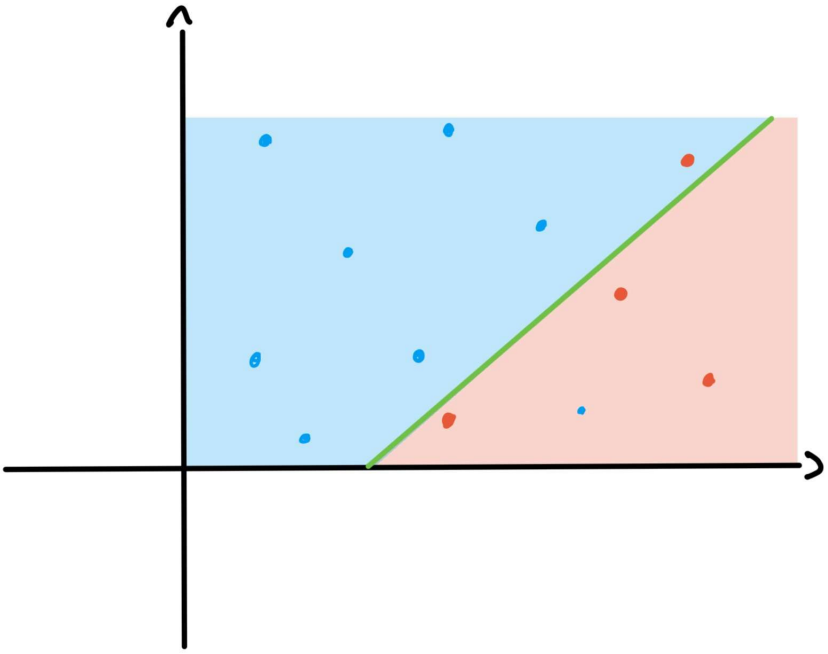


Figure 3 : Résultat avec biais

En effet, chaque neurone (les “inputs” exclus), sont comme des relais qui vont relayer de nouvelles valeurs en fonction de celles qui passent par ce relai. Pour éviter de traiter de trop grands nombres, nous utilisons des fonctions d’activations qui vont avoir pour effet de recentrer nos valeurs entre 0 et 1.

Mais un cerveau a besoin d’être travaillé pour devenir performant. Pour notre première soutenance, nous avons choisis de travailler avec la méthode la plus simple d’apprentissage qui consiste à regarder à quel point chaque poids a une influence sur le résultat final de notre réseau. Pour faire cela, nous calculons un “coût” de notre modèle en regardant la différence entre le résultat souhaité et le résultat obtenu par notre réseau de neurones. Par la suite, ce résultat est monté au carré afin d’obtenir des valeurs plus simples à traiter (du fait que nous avons seulement des nombres positifs et que ces résultats deviennent d’autant plus importants).

Il existe deux méthodes connues :

— Une première qui va faire le calcul sur chaque poids et recalculer un coût (fastidieux).

— Une deuxième qui va utiliser les dérivées partielles afin de comprendre à quel point chaque poids influence le coût de notre modèle sans avoir à le recalculer à chaque boucle. (très efficace) (voir figure 4)

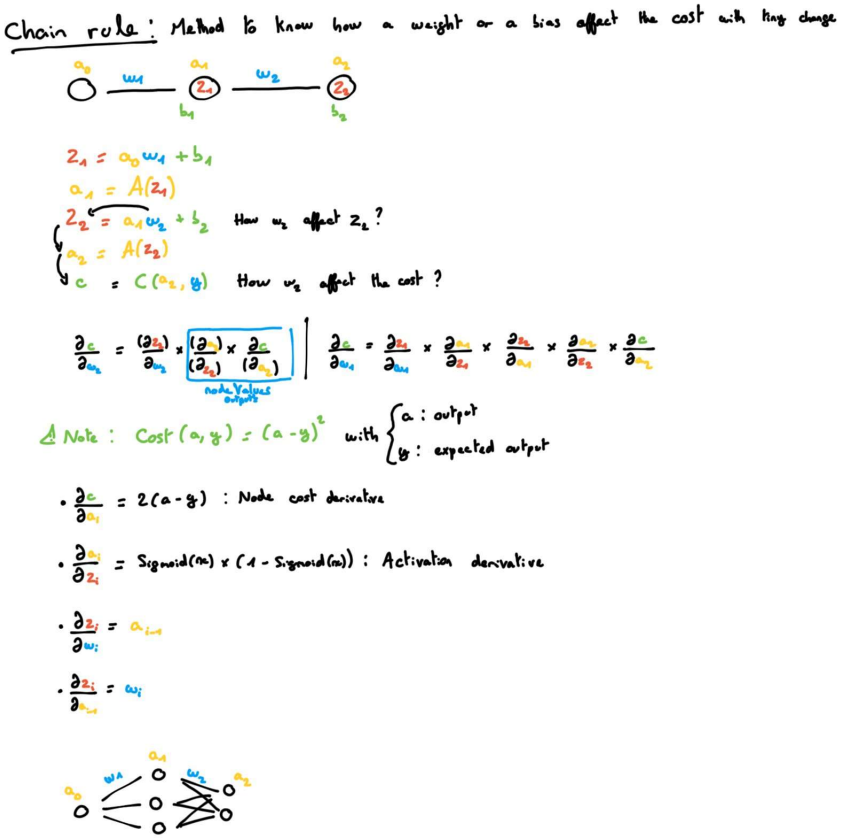
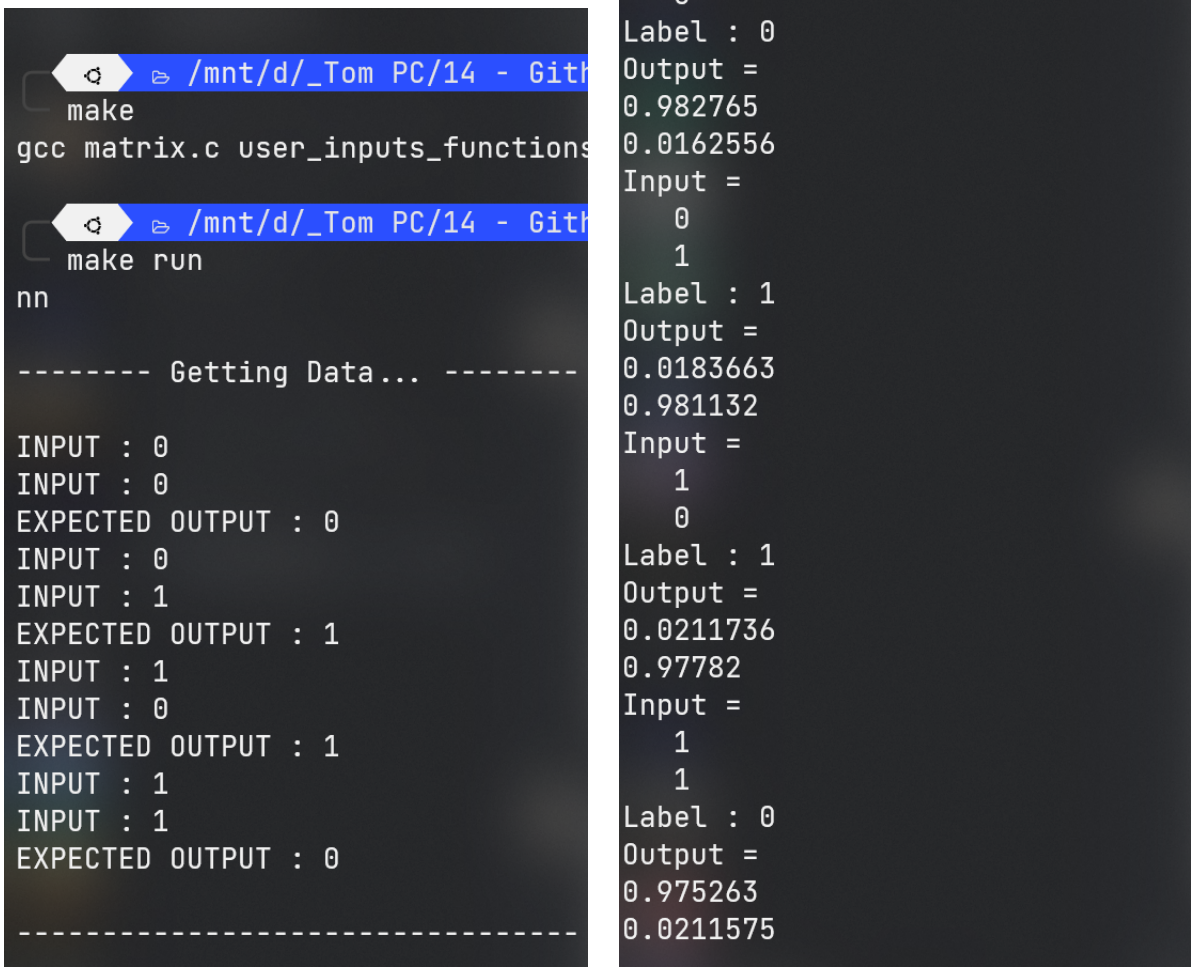


Illustration de la deuxième méthode

Pour la première soutenance, nous avons implémenté la première méthode. (La seconde nous ayant posée des problèmes, car plus compliquée à mettre en place).

Pour cette première soutenance, nous avons réussi à implémenter une méthode assez efficace pour reconnaître la table XOR, mais encore trop lente pour reconnaître des images, même si en théorie, cela reste possible avec notre implémentation.

Nous avons aussi implémenté des méthodes nous permettant de sauvegarder dans un fichier “save” nos poids et biais ainsi qu’une autre pour les importer dans notre réseau de neurones.



Résultats obtenus

4 Ressenti des membres

4.1 Tom Brossard

4.1.1 Joies

J'ai dès le début été très accroché par le projet. En effet, je m'intéresse beaucoup au machine learning, par conséquent, il m'était déjà arrivé de faire du traitement d'image et de me renseigner dessus. Je n'ai malheureusement jamais pris le temps de faire un projet complet comme celui-ci et souvent dans des langages bien plus laxistes que le C, ce qui rend ce projet un vrai défis, surtout que les cours qui sont à côtés ne doivent pas être délaissés, il faut donc bien réussir à gérer notre temps.

4.1.2 Peines

J'ai commencé très rapidement à travailler sur ma partie, mais pour des raisons que j'ignore encore, toutes les méthodes que j'utilisais avant dans d'autres langages sont bien plus compliqués à mettre en place en C. J'ai donc passé énormément de temps sur les réseaux de neurones sans vraiment avancer finalement, ce qui m'a forcé à faire une pause dessus et les midterms sont très rapidement arrivés ce qui nous a fait courir à la fin. Je ne pense pas avoir très bien réussi à gérer mon temps, je vais donc essayer d'apprendre de mes erreurs pour la prochaine soutenance.

4.2 Kylian Bozec

4.2.1 Joies

Ce projet est très différent de tous les autres travaux que nous avons pu faire. Ce que j'apprécie beaucoup sur ce travail c'est qu'il nous permet de mieux appréhender et utiliser le langage C, notamment avec l'utilisation de la SDL. De plus, ce n'est que personnel, mais je trouve que la satisfaction d'avoir réussi une étape dans ce projet et bien plus grande dans le projet OCR. Enfin, je trouve qu'il nous apprend beaucoup sur notre manière de fonctionner. En effet, et cela est le cas pour plusieurs groupes, je fais partie d'un groupe nouveau qui est totalement différent par rapport à l'OCR. Les méthodes de travail et l'organisation n'étant donc pas les mêmes, il a fallu s'y adapter, ce que j'ai trouvé enrichissant.

4.2.2 Peines

Selon moi, ce projet est difficile. En effet, d'une part, même si nous "connaissons" le langage C, la majorité des outils que nous utilisons sont nouveaux et les apprendre "sur le tas" peut être parfois déroutant (notamment la SDL au début). De plus, les notions utilisées sont plus avancées que ce que nous connaissions lors du projet jeu vidéo qui était plus pour "débutant". On le voit, l'un des buts principal de ce projet et de nous améliorer à coder contrairement au projet jeu vidéo qui nous amenait à "découvrir" le travail de groupe. En ce sens, nous travaillons logiquement plus dur que sur d'autres projets tel que le projet de deuxième semestre.

4.3 Quentin Hay–Kergrohenn

4.3.1 Joies

Grâce à ce projet, je découvre le traitement d'image, domaine au premier abord obscur mais qui s'avère très intéressant et passionnant. C'est incroyable le nombre de choses que l'on peut faire avec une image et en tirer.

4.3.2 Peines

Selon moi, ce projet est difficile. En effet, d'une part, même si nous "connaissons" le langage C, la majorité des outils que nous utilisons sont nouveaux et les apprendre "sur le tas" peut être parfois déroutant (notamment la SDL au début). De plus, les notions utilisées sont plus avancées que ce que nous connaissions lors du projet jeu vidéo qui était plus pour "débutant". On le voit, l'un des buts principal de ce projet et de nous améliorer à coder contrairement au projet jeu vidéo qui nous amenait à "découvrir" le travail de groupe. En ce sens, nous travaillons logiquement plus dur que sur d'autres projets tel que le projet de S2.

4.4 Pierre Fonséca

4.4.1 Joies

En ce qui concerne le projet, contrairement au projet jeu vidéo, il est totalement différent, car nous nous sommes lancés dans un projet plutôt difficile avec un nouveau langage. Mais suite à ces premiers mois, j'ai pu approfondir mes connaissances sur le langage C, avec des méthodes et une manière de pensées nouvelles, en plus de celle développée avec les TP.

Enfin, en ce qui concerne le groupe, nous avons une bonne ambiance de travail, une très bonne écoute entre les membres du groupe, ce qui pour moi est essentiel dans un travail de groupe.

4.4.2 Peines

Je trouve que ce projet est beaucoup plus dur que le projet jeu vidéo, de part sa courte durée, les cours qui s'enchaînent très rapidement et la découverte d'un nouveau langage.

Même si je me suis amélioré en C, j'ai durant ces premiers mois de travail eu un certains nombres de difficultés avec ma partie en particulier vis-à-vis du langage et de comprendre certaines mécaniques. De plus, je suis frustré d'avoir fait perdre un peu de temps à mon groupe et de voir que ma partie porte encore quelques problèmes. Malgré cela j'ai pour objectif de corriger tout cela pour la fin du projet et de tenir mes engagements sur ce projet.

5 Outils utilisés

5.1 Github

Github est un service en ligne nous permettant de sauvegarder en ligne l'entièreté de nos fichiers et dossiers consituant notre projet. Ce service nous permet de mettre à jour régulièrement notre projet à l'aide de "push" et donc de travailler ensemble sur à même emplacement.

5.2 Overleaf

Ce logiciel en ligne nous a permis de rédiger en LaTeX (un langage pour la rédaction de documents scientifiques), l'ensemble des documents liés à notre projet tels que notre rapport de soutenance, notre plan de soutenance, et tous futurs documents liés à notre projet.

5.3 Vim

Vim est un simple éditeur de texte qui nous a permit d'écrire et également de compiler l'ensemble de notre code.

6 Conclusion

Au terme de cette première partie de notre projet, nous avons partiellement atteint les objectifs fixés pour la première soutenance malgré des problèmes qui nous sont encore inconnus mais que nous corrigerons pour la fin de notre projet. Pour ce qui est du projet, nous nous sommes tous amélioré que cela soit sur notre connaissance sur le C, ou pour notre travail de groupe, c'est-à-dire sur le point de vue d'organisation des tâches ou sur l'écoute entre membres du projet. Malgré ces problèmes nous sommes tout de même satisfait de notre début de projet et avons bien l'objectif de le concrétiser.

7 Bibliographie

— Prétraitement :

— https ://zestedesavoir.com/tutoriels/1014/utiliser-la-sdl-en-langage-c/les-textures-et-les-images/
— https ://zestedesavoir.com/tutoriels/1014/utiliser-la-sdl-en-langage-c/tp-effets-sur-des-images/

— Rotation :

— https ://www.codingame.com/playgrounds/2524/basic-image-manipulation/transformation
— http ://www.leunen.com/cbuilder/rotbmp.html

— Réseau de neurones :

— https ://www.lebigdata.fr/machine-learning-et-big-data
— https ://www.youtube.com/c/SebastianLague
— https ://www.ibm.com/fr-fr/cloud/learn/neural-networks

8 Nous contacter

Pour plus de question vis-à-vis de notre projet de deuxième année, vous trouverez ci-dessous les différents moyens de nous contacter :
Adresse mail : kaiketsu.corp@gmail.com