# Rapport de suivi du projet OCR

Groupe NVAM*

Décembre 2024

*Mael Minard (chef de projet), Victor Flament, Adrien Fonck, Jaisy Brahimi

# Table des matières

**1 Introduction** **4**

**2 Présentations** **4**
2.1 Le groupe NVAM . . . . . 4
2.2 Les membres . . . . . 5
2.2.1 Jaisy Brahimi . . . . . 5
2.2.2 Victor Flament . . . . . 6
2.2.3 Adrien Fonck . . . . . 6
2.2.4 Mael Minard . . . . . 7

**3 Différents éléments du projet** **8**
3.1 Interface graphique . . . . . 8
3.2 Résolution de la grille . . . . . 9
3.2.1 Première implémentation - naïve . . . . . 9
3.2.2 Implémentation retenue - Présentation . . . . . 10
3.2.3 Implémentaton retenue - Structures utilisées . . 10
3.2.4 Implémentaton retenue - Algorithme . . . . . 12
3.2.5 Implémentaton retenue - Programme solver . . 13
3.3 Traitement d'image . . . . . 13
3.4 Détection des éléments . . . . . 17
3.5 Identification des lettres . . . . . 18
3.6 Liens entre les éléments . . . . . 19

**4 Répartition des taches** **19**

**5 Planification des taches** **19**

**6 Expériences personnelles de chacun** **20**
6.1 Mael . . . . . 20
6.2 Adrien . . . . . 21

6.2.1 Ressenti . . . . . 21
6.2.2 Explications techniques . . . . . 22
6.3 Jaïsy . . . . . 24
6.4 Victor . . . . . 24
6.4.1 Ressenti . . . . . 25

**7 Avancement final des tâches** **25**
7.1 Traitement d'image . . . . . 26
7.2 Détection des éléments . . . . . 26
7.3 Réseau de neurones . . . . . 27
7.4 Résolution de la grille . . . . . 27
7.5 Interface graphique . . . . . 27

**8 Conclusion** **29**

# 1 Introduction

Ce second rapport a pour rôle de montrer l'état final du projet OCR de l'équipe NVAM pour la dernière soutenance. Rappelons que nous nous intéressons au logiciel de type OCR (Optical Character Recognition) pour nous permettre la résolution d'une grille de mot caché, de plus nous devions approfondir la détection des éléments, l'intelligence artificielle et l'interface graphique. La détection des éléments est en effet primordiale car elle désigne le processus par lequel un système identifie et localise les différentes composantes importantes d'une image afin de les analyser puis de les convertir en texte. Elle se découpe en plusieurs étapes que nous expliciterons plus tard. L'intelligence artificielle utilise des algorithmes, basés sur l'apprentissage automatique pour améliorer la capacité des systèmes OCR à détecter, interpréter et extraire du texte, notamment grâce aux réseaux de neurones qui s'appuient sur des données d'entraînements pour apprendre et améliorer leur précision au fil du temps. Enfin l'interface graphique désigne la manière dont est présenté un logiciel à l'écran pour l'utilisateur, en outre c'est le positionnement des éléments comme le menu, les boutons, les fonctionnalités dans la fenêtre. Pour cette ébauche finale nous étudierons principalement ces trois aspects énoncés plus haut, préciserons leurs fonctionnements et pourquoi nous avons utilisés ces choix techniques. Nous complémenterons ce rapport par les différents obstacles rencontrés pour chaque implémentations et choix techniques.

# 2 Présentations

## 2.1 Le groupe NVAM

NVAM est un groupe de projet de deuxième année formé par quatre étudiants en septembre 2024. Le groupe est composé de Jaisy Brahimi,

Victor Flament, Adrien Fonck et Mael Minard.

La mission de notre équipe est de réaliser le projet OCR, c'est a dire créer un programme capable de résoudre une grille de mots mêlés a partir d'une image.

## 2.2 Les membres

### 2.2.1 Jaisy Brahimi

Jaisy Brahimi est un étudiant en deuxième année à l'école d'ingénieur informatique EPITA, à Lyon Part-Dieu. Ayant une appétence dans le domaine de la finance il vient d'une licence mathématique-informatique et se consacre maintenant à plein temps à une école spécialisé en informatique. Au sein du groupe NVAM, Jaïsy a donc pour rôle l'assistance à la réalisation du réseau de neurones qui sollicite l'étude de concepts abstraits, notamment dans la descente de gradient qui a pour but la minimisation. En effet ce lien avec la finance n'est pas anodin car toute entreprise à pour but de minimiser ses coûts. Il devra donc assister à l'architecture du réseau de neurones, à son entraînement et à son exploitation. De plus il assistera aussi dans l'interface graphique.

### 2.2.2 Victor Flament

Victor Flament est un brillant élève au sein du campus de Lyon de EPITA. Tombant dans la passion des systèmes GNU/Linux durant sa seconde il se décida a faire des études de maths et d'informatique. Étant passionne de systèmes d'exploitations et de développement systèmes ce projet n'est pas des plus passionnants pour lui. Cependant l'algorithmique fait parti des domaines qu'il l'intéresse. C'est pour cela qu'il s'occupe des taches de solver et détection. Mais aussi du traitement d'images.

### 2.2.3 Adrien Fonck

Adrien a vécu à Gap, dans les Hautes-Alpes, mais il a passé son baccalauréat à l'école internationale de la région PACA, à Manosque.C'est aussi dans cet établissement qu'il a développé ses compétences informatiques, notamment au travers des spécialités Mathématique et Numérique Science Informatiques. Dans sa scolarité, Adrien a pu participer à différents concours d'informatique, particulièrement Prologin au niveau national (ou il finira sa course en demi-finale). Dans ce projet, il sera surtout présent pour la conception du réseau de neurones et aidera pour les détections des caractères.

### 2.2.4 Mael Minard

Mael Minard est un étudiant en deuxième année à l'école d'ingénieur informatique EPITA, à Lyon.

Finissant son cursus baccalauréat général au Lycée Rosa Parks en 2023, il se dirige vers des études d'informatique, sujet qui le passionne depuis son plus jeune age.

Au sein de ce groupe, Mael fait office de chef de projet, il s'occupe donc de coordonner l'équipe, ainsi que de la représenter durant leurs

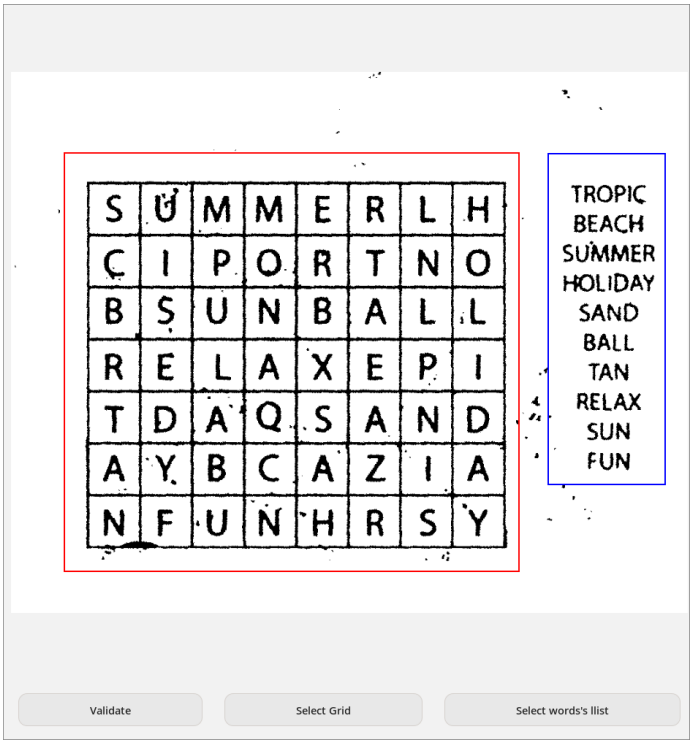
présentations.

Étant donné sa passion pour l'informatique, et sa soif d'apprentissage, Mael se dirigera au sein de ce projet vers la réalisation du réseau de neurone, une tache demandant la lecture d'un grand volume de documentation abstraite. Il sera donc chargé de réaliser l'architecture du réseau de neurone, ainsi que les différentes fonctions permettant de l'entraîner et l'exploiter. Il sera également chargé d'entraîner le réseau de neurone sur sa machine personnelle, de sorte a pouvoir le sauvegarder pour le réutiliser dans le programme.

# 3 Différents éléments du projet

## 3.1 Interface graphique

Le solver attendus doit pouvoir être utilise en format ligne de commandes, il prendrait deux paramètres: le premier un nom de fichier vers une grille de mots mêles uniquement en majuscules, et le deuxième serait un mot a trouver dans cette grille. Le programme renvoie sur la sortie standard deux couples d'entiers correspondants respectivement aux coordonnes de la première et de la dernière lettre du mot a trouver. Si le mot n'est pas trouve dans la grille le programme écrit "Not Found" toujours sur la sortie standard




## 3.2 Résolution de la grille

### 3.2.1 Première implémentation - naïve

La première version de notre programme était voulu naïve et brut-force. Le principe était simple: on commençait par charger le fichier texte de la grille et nous le parcourions deux fois, une pour compter les retours a la ligne et le nombre de lettres par lignes, ainsi on connaissait les dimensions de la grille et on pouvait vérifier qu'il n'y ait pas d'erreurs dedans, si un caractère n'est pas une lettre ou si toutes les lignes n'ont pas la même taille. Nous pouvions donc allouer la taille de la grille sur le heap et parcourir une deuxième fois pour remplir ce tableau.
Ensuite on parcourait la matrice représentant la grille et pour chaque lettre on vérifiait si elle correspondait a la première du mot cherche et

si c'était le cas on testait dans les huit directions si cela correspondait a notre mot.
A la fin si on avait des coordonnées on les affichait sinon on renvoyait "Not Found".
Cette méthode est simple a écrire, cependant elle est très coûteuse, encore plus si on se met dans le cas ou on cherche plusieurs mots dans la même grille, pour chaque mot on recharge la grille et on re-parcourt en vérifiant pour chaque lettre. C'est pour cela que nous avons changer d'idée pour l'implémentation retenue.

### 3.2.2 Implémentation retenue - Présentation

La version que nous utilisons actuellement du solver est pensée différemment: nous partons du principe que nous cherchons une liste de mots dans une grille. A partir de cette liste de mots nous allons construire un arbre ou les mots seront regroupes par rapport a leurs suffixes, ce qui facilitera la reconnaissance de lettres. Ensuite les différentes positions seront dans une liste chaînée.

### 3.2.3 Implémentaton retenue - Structures utilisées

Dans cette implémentation nous avons trois structures distinctes

- Grid
Cette structure nous permet juste de stocker le contenue du fichier de la grille ainsi que ces dimensions, elle est pas d'une grande utilité mais permet de simplifier le code et d'éviter d'avoir des fonctions a rallonges. Elle a différente fonctions associées, une recupérant les dimensions, une autre qui l'initialise et une qui la remplie. Ainsi qu'une dernière qui free ses différents attributs.

- Arbre de mots
Cette structure est centrale a notre algorithme, c'est elle qui va contenir tout nos mots, elle est sous forme d'un arbre général avec une implémentation premier fils/frère droit. La racine sera vide pour des raison d'implémentation mais les autres nœuds auront un char comme clé. Le premier niveau comportera la première lettre de chacun des mots, le deuxième niveau deuxieme lettre et ainsi de suite. Ainsi les mots 'papa' et 'pedantic' auront un nœud en commun, le premier p.
L'idée va donc être de construire cet arbre avec la liste des différents mots, il y a une fonction pour initialiser l'arbre avec la racine et une pour y ajouter un mot. Il y a aussi une fonction pour créer un arbre a partir d'une liste de mots et qui s'occupe de l'initialiser. Il y a aussi une fonction pour enlever un mot de l'arbre, ce sera pour quand on aura trouver un mot. Il y a la possibilité de détruire l'arbre et ainsi libérer la mémoire. Il y a d'autres fonctions pour connaître le nombre d'enfants d'un nœud, savoir si c'est une feuille ou connaître la taille de l'arbre.
Il y a aussi la fonction get-child qui prends en paramètres un arbre et un char et renvoie le fils de l'arbre donne ayant comme clé le char donne, ou null si il n'y en a pas. Cette fonction nous permettra de chercher les motifs dans la grille et ainsi voir si le construit dans la grille fait parti de la liste. Un exemple d'un arbre de mots:

- Liste de positions
Cette structure est une structure de liste chaînée un peu banal, chaque maillon de la chaîne contient un pointeur sur char qui est le mot auquel elle réfère, deux points de positions et un pointeur vers le prochain maillon. Un point de position est une structure de deux entiers, un x et un y, elle n'est pas vraiment utile mais sert juste a clarifier le code. Donc chaque mot de la liste est sensé être un mot trouve avec

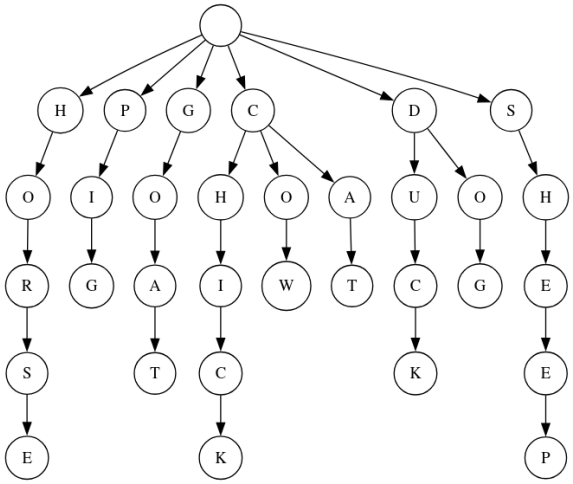



Figure 1: Exemple d'arbre de mots

sa position de début et sa position de fin.
Elle n'a que des fonctions banales de liste chaînée, la possibilité de l'initialise, y ajouter un élément, l'afficher ou encore la détruire pour libérer toute la mémoire.
Cette liste nous permet d'avoir le résultat pour chacun des mots trouves dans la grille.

### 3.2.4 Implémentaton retenue - Algorithme

Comme explique précédemment on va tout d'abord charger la grille de mots, construire notre arbre générale contenant touts les mots a chercher et initialiser une liste ou mettre les résultats.
Ensuite on va parcourir notre grille, pour chaque lettre de notre grille on va aller utiliser get-child de notre arbre et la lettre courante. Si le résultat est null alors aucun mot ne commence par cette lettre et on continue. Par contre si cela correspond on va essayer de voir si c'est la position de départ d'un des mots a trouver.

Pour ce faire on va chercher dans les 8 directions (gauche, droite, haut, bas, haut-gauche, haut-droite, bas-droite et bas-gauche). Et on va get-child de la lettre de ou on se trouve et tant qu'on est dans la grille et que ce résultat n'est null on continue. Si le nœud de l'arbre qu'on a récupérer est une feuille alors c'est qu'on a trouve un mot. Donc on ajoute a notre list un maillon avec ce mot, la position de ou on est partis, la position ou on est actuellement et un pointeur vers null car c'est la queue de la liste. On enlève le mot de l'arbre car on a plus besoin de le chercher.
Une fois tout la grille parcourue on détruit l'arbre et la grille car on en a plus besoin et retourne la liste de positions des mots.

### 3.2.5 Implémentaton retenue - Programme solver

A ce stade on pourrait se demander qu'en est il de l'implémentation demandée. En effet dans l'implémentation choisie on travaille avec une liste de mots or la on nous demande de faire un programme qui prends juste une grille et un mot en paramètres. Pour ce faire on a un autre programme qui réutilise l'implémentation présentée. On appelle notre programme avec la grille donnée et une référence vers notre mot, ce qui fait une liste de longueur 1. Si la liste renvoyée est null on écrit "Not Found" sur la sortie standard sinon on écrit les coordonnées de la lettre de début et de fin du mot.

## 3.3 Traitement d'image

Le principe de cette partie est de rendre l'image la plus utilisable possible pour la détection des lettres et de leur reconnaissance. Les principales fonctionnalités attendues sont l'augmentation des contrastes, la réduction du bruit donc les grains et la rotation et aussi tout ce

qui est niveau de gris puis noir et blanc. La manière dont nous avons procéder es de créer divers filtres pour des images et de les appliquer pour augmenter la qualité d'image. Pour ce faire nous nous sommes principalement bases sur la bibliothèque SDL2 qui nous permet de travailler sur les pixels d'une image. Pour optimiser le traitement d'image dans le logiciel, une refonte complète du système a été mise en œuvre afin de réduire considérablement les redondances dans le code et de rendre ce dernier plus modulaire et maintenable. L'approche retenue repose sur l'utilisation d'un modèle unifié pour tous les filtres, associé à des pointeurs de fonctions. Ce choix a permis de centraliser les opérations communes, de simplifier le processus d'ajout de nouveaux filtres et d'améliorer la lisibilité globale du projet.

L'idée principale a été de concevoir un prototype standard pour tous les filtres de traitement d'image. Chaque filtre est défini comme une fonction prenant en paramètre un pixel et son format, et retournant le pixel modifié. Ce format unifié a permis d'éliminer les implémentations spécifiques et redondantes pour chaque filtre. Par exemple, plutôt que d'avoir des boucles de traitement répétées dans chaque filtre, une fonction générique unique est désormais responsable du parcours de l'image. Cette fonction applique un traitement en appelant dynamiquement le filtre correspondant via un pointeur de fonction. Cela a permis de mutualiser toute la logique commune, comme le verrouillage de la surface ou la gestion des pixels, tout en rendant les filtres eux-mêmes modulaires et faciles à adapter.

Ce système offre plusieurs avantages majeurs. Tout d'abord, il a permis une réduction significative de la taille globale du code, en éliminant la duplication liée aux boucles et à la gestion des pixels dans chaque filtre. Ensuite, il facilite l'ajout de nouveaux filtres : il suffit de coder une fonction respectant le prototype standard, sans avoir à réécrire la logique de traitement d'image. Cette modularité simplifie aussi la maintenance du code et en améliore la lisibilité, car la structure est claire et cohérente. Par ailleurs, bien que l'impact

direct sur les performances soit marginal, cette approche réduit les risques d'erreurs liées à des implémentations spécifiques et garantit une meilleure uniformité dans le traitement des images.

La rotation automatique est une étape fondamentale dans le traitement des images pour l'OCR, car elle permet d'aligner une grille de mots mêlés même si elle est inclinée ou mal orientée dans l'image d'origine. Cette fonctionnalité repose sur un processus en plusieurs étapes intégrant la détection des bords, l'analyse des lignes à l'aide de la transformation de Hough, et la rotation de l'image en fonction de l'angle détecté.

Le premier stade consiste à analyser la luminosité des pixels de l'image pour les convertir en niveaux de gris. Cette conversion simplifie l'identification des caractéristiques importantes, telles que les contours de la grille, en réduisant les informations visuelles inutiles. Chaque pixel est transformé en une valeur de luminosité unique, ce qui prépare l'image pour les étapes suivantes.

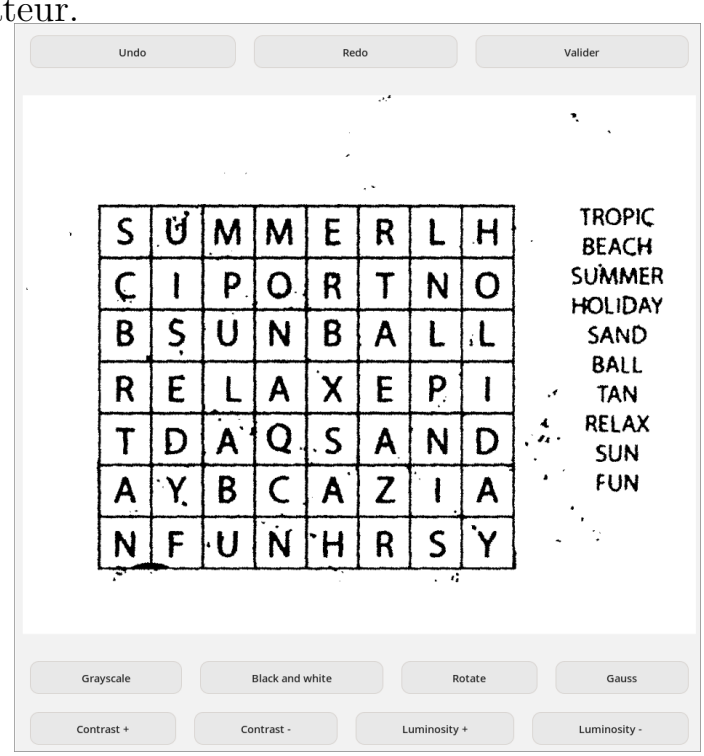
Ensuite, un filtre basé sur le gradient de Sobel est appliqué pour détecter les bords. Cette méthode examine les variations de luminosité entre les pixels voisins, ce qui permet d'identifier les transitions nettes, souvent caractéristiques des contours d'une grille. Les résultats de cette étape produisent une carte des bords qui met en évidence les zones de l'image où se trouvent des lignes significatives.

Une fois les bords détectés, la transformation de Hough est utilisée pour analyser ces données et repérer les lignes présentes dans l'image. Cette technique transforme les pixels de bords détectés dans un espace paramétrique défini par deux variables : p (distance à l'origine) et rho (angle par rapport à l'axe horizontal). Dans cet espace, chaque ligne droite de l'image correspond à un pic, permettant de déterminer les orientations dominantes. En analysant les accumulations dans cet espace, l'algorithme identifie l'angle principal correspondant à l'orientation de la grille.

Après avoir déterminé l'angle optimal, une étape de rotation est ap-

pliquée pour corriger l'inclinaison de la grille. À cette fin, une transformation graphique permet de faire pivoter l'image avec précision, tout en préservant la qualité visuelle. L'algorithme ajuste systématiquement l'angle pour aligner la grille à un multiple de 90 degrés, garantissant une disposition correcte pour les étapes suivantes du traitement.

Cette approche automatisée assure une correction robuste des orientations les plus courantes, facilitant ainsi la reconnaissance et la résolution des grilles. Néanmoins, elle présente quelques limites, notamment en cas de bruit excessif dans l'image ou d'erreurs dans la détection des bords, ce qui peut entraîner des rotations incorrectes. Dans de tels cas, une intervention manuelle pourrait être nécessaire pour finaliser l'alignement. Malgré ces éventuelles contraintes, la rotation automatique constitue un atout majeur, permettant d'améliorer la précision globale du logiciel OCR et de simplifier l'expérience utilisateur.



## 3.4 Détection des éléments

La détection des éléments est le système permettant au programme d'analyser l'image de la grille pour en sortir les coordonnées de chaque lettre dans l'image et leur rôle (fait parti de la grille ou d'un mot, position dans la grille/le mot). Tel que nous l'avons implémente, le système de détection requiert les coordonnées de la grille et de la liste de mots, fournies par l'utilisateur a travers l'interface graphique.

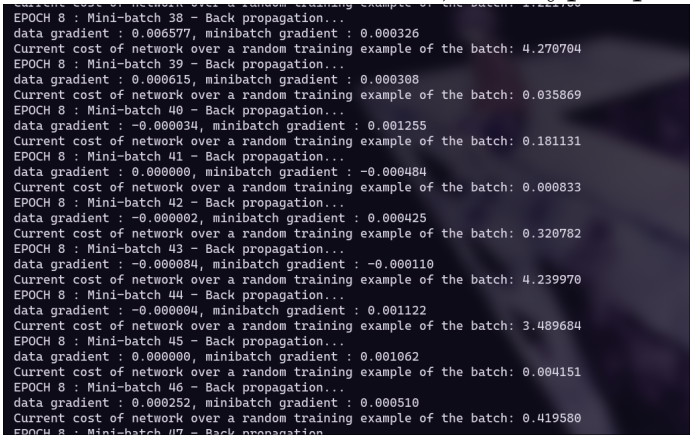
Le système de détection va ensuite traverser l'image au sein de la grille et liste de mots, cherchant des pixels noir, desquels ils va former des formes qui seront les lettres, pour enfin encadrer ces formes. Le programme va finalement filtrer ces encadrements pour enlever les formes parasites comme les dessins, puis enfin il va organiser ces encadrements dans une matrice représentant la grille, et une liste de listes représentant la liste de mots.

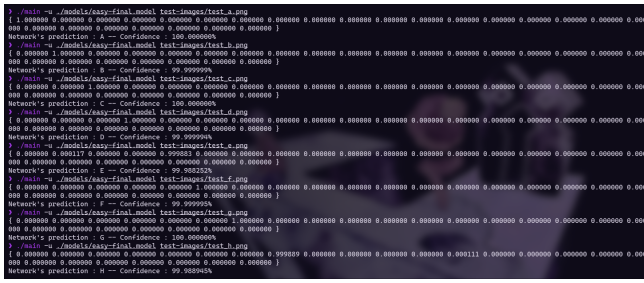
## 3.5 Identification des lettres

L'identification des lettres est la partie faisant rentrer en jeu le réseau de neurone, qui doit être capable pour n'importe quelle image de lettre, de reconnaître de quelle lettre il s'agit. Le système de réseau de neurone est principalement décomposé en deux sous systèmes, le réseau lui même, et son entraînement.

Le réseau peut être charge depuis un fichier, ou crée de toute pièces en quelques ligne, avec un nombre de neurone, nombre de couches, taille de couches, et autres paramètres spécifies. La tache étant une reconnaissance de caractères, le réseau utilise la fonction d'activation ReLu pour ses couches cachées, ainsi qu'un SoftMax pour sa couche de sortie, afin d'obtenir une probabilité normalisée sur les 26 labels correspondants aux 26 lettres. Le réseau de neurone peut enfin évidemment être sauvegardé dans un fichier.

L'entraînement du réseau de neurones se fait a l'aide de structures de données chargées a partir de fichiers avec des formats spécifiques, le chargement des structures d'entraînement est surtout programmé pour marcher avec les fichiers qui ont réellement été utilisés, ainsi qu'un format utilise dans une fonctionnalite de création de fichier de donnée d'entraînement personnalisé. L'entraînement est programmable au niveau du nombre d'itérations, des hyper paramètres, et est optimisé.



### 3.6 Liens entre les éléments

Cette partie rassemblent plusieurs fonctionnalités essentielles pour relier les autres éléments du programme. Il y a par exemple la fonctionnalite de redimensionnement de l'image, de convertissement d'image en tableau d'entiers prêt a être donnés au réseau de neurone, ou encore la conversion entre les résultats de la fonction de détection, couple aux résultats du réseau de neurone, en une grille de caractère et liste de chaînes de caractère prêtes a être donnés au programme de résolution de la grille. Ou encore la conversion des résultats du programme de résolution de la grille en affichage a l'écran pour l'utilisateur. Ces deux dernières fonctionnalités n'ont pas pu êtres implémentées à temps.

## 4 Répartition des taches

| | |
|---|---|
| Traitement d'image | Victor Adrien |
| Détection des éléments | Victor Mael |
| Intelligence artificielle | Adrien Mael Jaisy |
| Résolution de la grille | Victor |
| Interface graphique | Victor |

## 5 Planification des taches

| Tache | Septembre | Octobre | Novembre | Décembre |
|---|---|---|---|---|
| Traitement d'image | 50% | 90% | 100% | 100% |
| Détection des éléments | 0% | 40% | 80% | 100% |
| Intelligence artificielle | 30% | 60% | 90% | 100% |
| Résolution de la grille | 100% | 100% | 100% | 100% |
| Interface graphique | 0% | 10% | 40% | 100% |

# 6 Expériences personnelles de chacun

## 6.1 Mael

Durant ce projet, j'ai consacre le plus grand de mon temps a la conception du réseau de neurones. J'ai commence tôt dans le semestre a étudier les réseaux de neurones a l'aide du livre PDF dont le cahier des charges fait mention. Cette expérience fut plutôt passionnante. A la fin du premier chapitre du livre, j'avais appris les bases : Multi-Layer Perceptron, fonction sigmoid, fonction de coût mean squared, feedforward, descente de gradient. Je me suis donc lancé dans l'implémentation du réseau de neurone. J'ai crée les structures, les fonctions pour sauvegarder, charger, etc.. J'ai également rédige l'algorithme du feedforward. Puis j'ai fait l'erreur de me lancer dans le développement de l'algorithme de backpropagation sans bien me renseigner préalablement. En effet, je n'avais pas compris que le principe de la backpropagation est de calculer le gradient de la fonction de coût de manière optimisée. J'avais donc naïvement implémenté un algorithme calculant chaque composante du gradient a l'aide d'une dérivé partielle du coût. L'entraînement du réseau de neurone était donc extrêmement lent, de plus, il ne pouvait pas minimiser son coût convenablement a cause de l'utilisation du sigmoïde, ainsi que la méthode d'initialisation des poids et des biais. Je suis malheureusement resté bloqué longtemps a ce stade, ce qui fut une erreur majeure, avant d'enfin demander de l'aide sur un forum, ce qui m'a permis de voir et corriger mes erreurs. Après implémentation de la backpropagation, changement des fonctions d'activation, de coût, correction de certains bugs, le réseau de neurone pouvait convenablement s'entraîner. Je l'ai alors entraîne durant plusieurs nuits et j'ai sauvegarde le modelé, que nous pouvons désormais utiliser dans le programme.

Après avoir implémenté le réseau de neurone, j'ai travaillé sur le système de détection, or je n'ai pu que peu travailler cette fonction-

nalité, par manque de temps.

J'ai donc eu une expérience positive au sein de ce groupe, dans le cadre du projet de premier semestre. J'ai pu apprendre beaucoup sur les réseaux de neurones, sujet passionnant, ainsi que sur le développement sur Linux. J'ai cependant mal géré mon temps et ma répartition de travail, ce qui m'a amené a une fin de projet stressante et décevante.

## 6.2 Adrien

### 6.2.1 Ressenti

Durant ce projet j'ai principalement contribué au réseau de neurones et au traitement d'images. J'ai adoré me renseigner sur les réseaux de neurones, notamment grâce à la chaîne 3Blue1Brown sur YouTube, qui m'a permis de mieux comprendre certains concepts clés. Travailler sur ce projet a été une expérience enrichissante, non seulement sur le plan technique, mais aussi sur le plan humain. L'ambiance dans le groupe était vraiment bonne, avec beaucoup d'entraide et de discussions constructives, ce qui a rendu le travail encore plus motivant.

L'un des moments marquants a été le choix d'utiliser l'interpolation bicubique plutôt que l'interpolation par voisin le plus proche pour le redimensionnement des images. Cette décision est née après avoir comparé leurs précisions respectives et après quelques recherches. Intégrer cette méthode n'a pas été simple, notamment parce que je ne maîtrisais pas bien SDL2 au début, mais grâce à l'aide de Viktor et Maël, nous avons réussi à surmonter ces défis.

Ce projet m'a aussi appris que, dans un projet collaboratif, chaque fonction peut être liée aux autres, ce qui peut être source de confusion, mais aussi une occasion d'apprendre, chaque fois que je ne comprenais pas la fonction de quelqu'un d'autre, ce dernier me l'expliquer ce qui m'a beaucoup aider pour l'ensemble du projet (compréhension

du code et meme idées pour implémenter mes fonctions). Les séances de brainstorming pour résoudre des erreurs de mémoire ont été particulièrement formidables, grâce à elles j'ai compris l'importance de partager ses idées et de travailler en groupe pour surmonter les obstacles.

Je ressens une grande satisfaction d'avoir réussi à avancer dans ce projet et d'avoir appris autant sur les réseaux de neurones, le traitement d'images et la collaboration en équipe. Cette expérience m'a renforcé dans mes compétences et m'a prouvé que je pouvais relever des défis techniques.

### 6.2.2 Explications techniques

Le réseau de neurones, grâce à son processus d'entraînement, est capable de trouver un minimum local sur la courbe de coût, ce qui lui permet de minimiser l'erreur et d'améliorer ses prédictions. Cependant, l'entraînement d'un réseau de neurones classique peut être un processus long et coûteux en termes de temps de calcul. Afin d'optimiser ce processus, nous avons choisi de lancer simultanément une dizaine de processus enfants. En procédant ainsi, l'entraînement devient beaucoup plus efficace, car au lieu de former un seul modèle à la fois, le programme entraîne plusieurs réseaux en parallèle. Ce mécanisme permet de tester différentes configurations et de comparer les résultats en temps réel, en sélectionnant automatiquement le modèle ayant donné les meilleurs résultats, c'est-à-dire celui qui a atteint le minimum le plus bas sur la courbe de coût. Cette approche est née d'un constat simple : lors de l'entraînement d'un réseau de neurones traditionnel, les ressources telles que le CPU et la RAM ne sont pas pleinement utilisées, et une grande partie des capacités de l'ordinateur reste inutilisée. En exploitant la fonction fork, qui permet de créer des processus enfants simultanément, nous pouvons tirer parti de la puissance de calcul disponible sur l'ensemble de l'ordinateur. Cela per-

met d'entraîner plusieurs réseaux de neurones en parallèle, accélérant ainsi considérablement le processus. Cette méthode ne se contente pas seulement de réduire le temps d'entraînement, elle permet également de trouver un réseau fonctionnel plus rapidement et, surtout, un modèle plus précis. En multipliant les essais avec différentes initialisations et en parallèle, nous avons plus de chances d'atteindre un modèle performant et bien optimisé.

Afin d'utiliser les images découpées dans le réseau de neurones, nous devons les convertir. En effet, le réseau de neurones final ne fonctionne qu'avec des images de 28×28 pixels. Il était donc impératif de créer une fonction de redimensionnement (resize) qui prenne n'importe quelle image et la transforme de manière à respecter cette contrainte. Notre première idée a été de créer une fonction basée sur l'interpolation par les plus proches voisins pour l'agrandissement (UpScale) et de simplement découper des lignes ou des colonnes pour le rétrécissement (DownScale). Le principal problème de ces procédés est la perte de qualité. C'est pourquoi, avant de commencer à coder, nous nous sommes renseignés sur d'autres techniques, et c'est ainsi que nous avons découvert l'interpolation bicubique.

Contrairement à des méthodes comme l'interpolation par les plus proches voisins, qui se contente de dupliquer le pixel le plus proche, l'interpolation bicubique utilise les 16 pixels voisins autour de chaque pixel à recalculer (quatre pixels dans chaque direction). Suite à cela, la couleur du pixel que l'on souhaite créer est calculé grâce à des formules mathématiques. Cette approche permet de lisser les transitions entre les pixels et de réduire les effets de pixellisation souvent présents dans les techniques plus simples. Un des gros points positifs de cette technique est qu'elle peut gérer le UpScale et le DownScale.

## 6.3 Jaïsy

Etant nouveau au sein de l'école, j'ai démarré avec un an d'expérience informatique en moins. Ce projet à premièrement permis de m'intégrer à un groupe et prendre mes marques plus rapidement dans le campus. De plus j'ai pu observer, apprendre et acquérir de nouvelles compétences que je n'aurai peut-être pas eu la chance d'avoir. Concernant le projet OCR, venant d'une licence mathématique, j'ai pu lors de la première partie du projet, travailler sur la descente de gradient. En outre ayant une moindre expérience en programmation, lors de la seconde moitié du projet j'ai eu plus un rôle d'assistant, notamment sur l'intelligence artificielle et les réseaux de neurones qui sont finalement, la continuité logique de la descente de gradient. Enfin ce projet à été très constructif pour moi car il m'a permis de créer de forger un esprit d'équipe, une meilleure écoute et des nouvelles compétences en programmation que je n'avais auparavant.

## 6.4 Victor

Pour ce projet, je me suis investi dans plusieurs tâches : le traitement d'images, la détection, le solver et l'interface graphique. Cela m'a permis de toucher un peu à tout et de varier les différentes problématiques : des aspects plus simples, comme GTK, des sujets plus algorithmiques, comme le solver, ou encore des réflexions sur la structuration du code pour le traitement d'images. Certaines parties, très complexes, comme la détection, ont été confiées à Maël.

J'ai largement préféré ce projet à celui du S2, car nous avions une idée plus claire de la direction que nous souhaitions suivre, et nous avons pu expérimenter la création d'un programme en C. Nous avons également appris beaucoup de choses, notamment sur les réseaux de neurones.

Cependant, le projet a parfois été frustrant en raison de sa tech-

nicité, surtout au niveau de la détection, où la difficulté semblait particulièrement élevée. Malgré cela, nous avons su tenir le cap et produire un rendu final dont nous sommes fiers, car il reflète tout ce que nous avons réussi à accomplir.

### 6.4.1 Ressenti

Pour ce projet, je me suis investi dans plusieurs tâches : le traitement d'images, la détection, le solver et l'interface graphique. Cela m'a permis de toucher un peu à tout et de varier les différentes problématiques : des aspects plus simples, comme GTK, des sujets plus algorithmiques, comme le solver, ou encore des réflexions sur la structuration du code pour le traitement d'images. Certaines parties, très complexes, comme la détection, ont été confiées à Maël.

J'ai largement préféré ce projet à celui du S2, car nous avions une idée plus claire de la direction que nous souhaitions suivre, et nous avons pu expérimenter la création d'un programme en C. Nous avons également appris beaucoup de choses, notamment sur les réseaux de neurones.

Cependant, le projet a parfois été frustrant en raison de sa technicité, surtout au niveau de la détection, où la difficulté semblait particulièrement élevée. Malgré cela, nous avons su tenir le cap et produire un rendu final dont nous sommes fiers, car il reflète tout ce que nous avons réussi à accomplir.

# 7 Avancement final des tâches

Voici le tableau représentant l'avancement actuel des différents éléments du projet.

| | |
|---|---|
| Traitement d'image | 90% |
| Détection des éléments | 40% |
| Réseau de neurone | 100% |
| Résolution de la grille | 100% |
| Interface graphique | 80% |

## 7.1 Traitement d'image

Le traitement d'image a pu être complété a 90%, en effet, le programme met a disposition de l'utilisateur une interface graphique robuste et complète pour préparer son image avant la détection. De nombreux filtres sont a sa disposition. La fonctionnalité manquant du traitement d'image est l'automatisation. En effet, l'utilisateur doit manuellement appliquer ces filtres, au lieu de simplement laisser le programme les appliquer de manière automatisée afin de traiter l'image le plus favorablement pour la détection.

## 7.2 Détection des éléments

La détection des différents éléments de l'image est la fonctionnalité majeure que nous n'avons pas réussi a implémenter. Le programme demande actuellement a l'utilisateur d'encadrer la grille de lettres ainsi que la liste de mots, de manière a alléger la tache du programme en enlevant la nécessité de trouver automatiquement la position de la grille et de la liste de mots. Le programme va ensuite chercher dans la grille et la liste de mots toutes les formes remplissant certains critères, et les encadrer, puis former a partir des coordonnées des formes une liste et une grille de formes. Le problème actuel de la détection est le manque de traitement des cas problématiques, par exemple, si deux lettres sont collées, elles ne vont former qu'une seule forme. Inversement, si une lettre possède des imperfections et est discontinue, elle pourrait former plusieurs formes.

## 7.3 Réseau de neurones

Le réseau de neurones est entièrement fonctionnel, il est facilement intégrable dans le code, entraînable, et donne des résultats adéquats. Après entraînement sur un ensemble de données, il montre une fiabilité d'environ 99% sur ce même ensemble, et après différents tests, il est capable de généraliser convenablement son entraînement sur des lettres qu'il n'a jamais vu, généralement avec une confiance au dessus de 90%.

## 7.4 Résolution de la grille

La résolution de la grille est parfaitement fonctionnelle, elle permet a l'utilisateur de chercher un mot spécifique dans la grille et au programme complet de chercher tous les mots dans la grille.

## 7.5 Interface graphique

L'interface graphique, développée avec GTK3, constitue un élément central de notre logiciel OCR dédié à la résolution de grilles de mots mêlés. Nous avons conçu cette interface en nous inspirant du thème graphique Orchis Pink, un design moderne et épuré qui privilégie la clarté et la simplicité. Grâce à des éléments visuels arrondis et des touches colorées subtiles, l'ensemble vise à créer une expérience utilisateur agréable et intuitive. Principes de conception

Notre objectif principal a été d'offrir une interface qui accompagne l'utilisateur tout au long du processus, de manière claire et progressive. Nous avons mis un point d'honneur à réduire la complexité des actions nécessaires à l'utilisation du logiciel, afin que même un utilisateur novice puisse aisément manipuler les fonctionnalités proposées. Fonctionnalités principales

L'interface permet à l'utilisateur :

De charger une image de grille : Cette étape initiale est simplifiée pour garantir une prise en main rapide. D'appliquer des filtres : Plusieurs traitements d'image (comme le noir et blanc, la rotation ou encore l'ajustement de contraste) sont disponibles. Ces options aident à optimiser l'image pour une meilleure reconnaissance des éléments. D'utiliser les systèmes de "Undo" et "Redo" : Ces fonctionnalités permettent de revenir en arrière ou de réappliquer une modification, garantissant une flexibilité et une précision accrues lors du traitement de l'image. De visualiser les détections automatiques : L'utilisateur peut prévisualiser le résultat des détections effectuées par le programme (grille et liste de mots). En cas d'erreur, une sélection manuelle est possible pour ajuster les zones concernées. De lancer la résolution : Une fois la grille et les mots correctement détectés, l'utilisateur peut demander au programme de résoudre la grille.

Accompagnement de l'utilisateur

Nous avons conçu l'interface pour guider l'utilisateur à chaque étape du processus, en suivant une logique séquentielle claire. L'idée est que, même si le programme ne parvient pas à résoudre la grille automatiquement, l'utilisateur soit suffisamment accompagné pour réussir en intervenant manuellement.

Voici quelques exemples d'éléments mis en place pour garantir cette simplicité d'utilisation :

Messages et indications visuelles : Des encadrés et des couleurs aident à repérer rapidement les zones clés (comme la grille et la liste des mots). Actions contextuelles : Les boutons et options disponibles s'adaptent aux étapes en cours, minimisant les distractions inutiles. Guides interactifs : Des suggestions ou des alertes peuvent apparaître pour indiquer des actions potentielles à l'utilisateur, comme l'amélioration de la rotation ou le recalibrage d'une détection.

Limites actuelles et développements futurs

Bien que l'interface soit presque complète, certaines fonctionnalités restent à implémenter ou à finaliser :

Visualisation des mots trouvés : Actuellement, le programme ne peut pas afficher la grille résolue avec les mots surlignés ou encadrés directement sur celle-ci. Enregistrement des résultats : Il manque une option permettant de sauvegarder l'image de la grille avec les mots trouvés en couleur. Intégration complète : Les différents éléments du logiciel (interface graphique, moteur OCR, et algorithmes de résolution) ne sont pas encore entièrement connectés entre eux.

Conclusion

Avec son design moderne et son ergonomie soignée, notre interface graphique vise à rendre la résolution de grilles de mots mêlés accessible à tous. Bien que certaines fonctionnalités nécessitent encore des développements pour aboutir à une version entièrement fonctionnelle, l'approche adoptée garantit une expérience utilisateur intuitive et adaptable. En intégrant des outils interactifs et des systèmes de correction manuelle, nous espérons offrir une solution robuste et agréable, capable de guider l'utilisateur du début à la fin du processus. ‘

# 8 Conclusion

Nous pouvons maintenant conclure sur l'état final et l'expérience du projet. Ce projet fut une expérience positive et enrichissante pour nous tous. Nous avons tous beaucoup appris sur diverses sujets lors de ce semestre dans le cadre du projet OCR.

Malheureusement, nous n'avons pas pu entièrement réaliser les fonctionnalités attendues du programme. Nous pouvons distinguer deux raisons à cet échec.

Premièrement, notre équipe a souffert d'un manque d'organisation. En effet, la répartition des taches n'était en fait pas convenable. Deux membres du groupe ont travailles séparément sur le réseau de neurone en début de projet, Adrien et Mael. Cependant, ces travaux étaient entièrement sépares, Adrien se concentrant sur un réseau XNOR pro-

gramme rapidement pour la soutenance, et Mael se concentrant sur un réseau plus complexe et adapte a la tache finale, mais qui ne serait pas prêt pour la soutenance. Également, tandis que le premier bimestre avait été assez rythmé, le début du second bimestre fut marqué d'un manque de travail général sur le projet. Ceci est du au fait qu'il restait encore assez de temps pour remettre le travail crucial a plus tard. Cependant, c'est a cause de cette remise a plus tard que notre groupe a manque de temps pour réaliser les objectifs. En effet, ce n'est qu'a la fin du projet que le groupe s'est rendu compte de la quantité de travail restant ce qui nous amène à la seconde raison de l'échec.

Secondement, la complexité de ce projet a été sous estimée par le groupe. En effet, bien que ce projet ne paraisse aucunement trivial à première vue, il cache néanmoins de nombreuses complexités qu'il est difficile d'anticiper avant que leur nécessité ne se révèle. Au début du projet, Victor avait déjà réalise le programme de résolution de grilles de mots croises, et avait pris l'initiative de travailler sur le traitement d'image, ce qui laissait principalement le réseau de neurones en tâche principale la plus complexe, c'est donc sur ceci que le reste de groupe a décidé de se pencher. La détection des éléments de la grille fut largement négligée, or il s'agit d'une composante principale du programme sans laquelle il ne peut pas fonctionner. La détection est une tache complexe qui demande du temps de conception, or elle fut réalisée presque entièrement lors des derniers jours avant le rendu. Également, la conception d'un système permettant de relier toutes les composantes du programme, avec les taches supplémentaires que ce système implique (redimensionnement des images, par exemple), fut négligée jusqu'à la dernière semaine du projet.

Si le projet était à refaire, le groupe adopterait donc une stratégie bien plus claire, se réunissant durant plusieurs heures pour réfléchir aux différentes pièces composant le programme final, à ce que ces pièces impliques, et repartissant correctement les taches.

Malgré le fait que le programme complet ne soit pas fonctionnel,

chaque membre de l'équipe NVAM reste néanmoins fier de ce qu'il a réalisé dans les différents programmes séparés. Chaque membre est satisfait de ce qu'il a pu apprendre sur les différents sous domaines de l'informatique, et nous voyons tous notre passion pour ce domaine renforcée. Enfin, le groupe a fondé lors de ce projet un véritable esprit d'entre-aide, et de collaboration.