

Rapport de Projet



EPITA 2028: Projet S3
Groupe CANA

RAMSTEIN Antoine
MULLER Célian
CORNU Noé
MEYER Aurélien

TABLE DES MATIÈRES

1	Introduction	2
2	Présentation du groupe	3
3	Répartition des charges	5
4	Etat d'avancement du projet	7
4.1	Chargement d'une image et suppression des couleurs	7
4.2	Détection de la grille et extraction des mots	11
4.3	Algorithme de résolution d'une grille de mots cachés	13
4.4	Concept du réseau de neurones	15
5	Aspect technique	17
5.1	Prétraitement	17
5.2	Détection de la grille et extraction des mots	17
5.3	Réseaux de neurones	18
5.4	Solver	20
6	Conclusion	22

1 INTRODUCTION

L'objectif de ce projet est de réaliser un logiciel de type OCR (Optical Character Recognition) qui résout une grille de mots cachés. L'application prendra en entrée une image représentant une grille de mots cachés et affichera en sortie la grille résolue. Dans sa version définitive, l'application devra proposer une interface graphique permettant de charger une image dans un format standard, de la visualiser, de corriger certains de ses défauts, et enfin d'afficher la grille complètement remplie et résolue. La grille résolue devra également pouvoir être sauvegardée. La première soutenance est une soutenance de suivi. L'objectif est de montrer l'état d'avancement de notre projet.

2 PRÉSENTATION DU GROUPE

Notre groupe se compose de 4 membres : Noé CORNU, Célian MULLER, Aurélien MEYER et Antoine RAMSTEIN. Ce projet a pour but de renforcer notre capacité à travailler efficacement en groupe tout en perfectionnant notre aptitude à gérer un projet complexe. Cette expérience nous enseignera des compétences essentielles en gestion de projet, telles que la planification, la répartition des tâches, la communication et la collaboration au sein de l'équipe. Pour le bon déroulement du projet, nous avons décidé de mettre en place des réunions, le plus souvent toutes les semaines, pour être le plus efficace dans l'avancement du projet. En somme, ce projet constitue une opportunité d'apprentissage holistique dans le domaine de la technologie et de l'informatique, nous préparant ainsi à des défis professionnels futurs.

Célian

L'informatique me motive profondément pour son potentiel d'innovation et d'impact dans divers secteurs. J'apprécie particulièrement les défis techniques, comme ce projet de WorldSearchSolver. J'aime l'idée que, grâce aux technologies, nous pouvons améliorer des systèmes, optimiser des processus, et transformer la manière dont nous vivons et travaillons. Dans le cadre de ce projet, je suis particulièrement intéressé par l'apprentissage d'un réseau de neurones et la création d'interface graphique. Cet aspect est très motivant pour moi car il me permet de mettre en pratique mes compétences vue en cours magistraux et d'apprendre à collaborer avec d'autres étudiant, ce qui est essentiel pour mon développement professionnel. Ainsi, ce projet est une source de motivation et d'épanouissement puisqu'il me permet d'apprendre d'avantage des notions qui me serviront dans le monde professionnel.

Noé

Ce projet me permet d'acquérir une expérience précieuse, bien différente de celle de projets classiques pour lesquels de nombreuses ressources existent déjà en ligne. Ici, le caractère unique du World Search Solver rendent les recherches de solutions plus difficiles, car peu de projets similaires sont disponibles pour s'en inspirer. Cette absence de repères nécessite de développer une réflexion autonome, de sortir des méthodes établies et de penser de manière créative pour construire un algorithme efficace. Ce processus, bien qu'exigeant, va me per-

mettre d'affiner ma capacité à résoudre des problèmes de manière innovante, renforçant ainsi ma confiance en mes propres capacités. En somme, ce projet m'offre une occasion de grandir techniquement et personnellement, tout en stimulant mon autonomie et ma persévérance face aux défis complexes.

Aurélien

je suis passionné par la cybersécurité et la programmation, avec une ambition pour le milieu militaire. Ce qui me motive particulièrement, c'est l'idée de relever des défis complexes et d'utiliser mes compétences pour sécuriser des systèmes.

Dans ce projet d'OCR, je suis en charge du réseau de neurones, une chose que je n'ai jamais faite, mais qui m'intéresse beaucoup car cela permet d'allier analyse de données et apprentissage automatique. Ces compétences sont cruciales pour comprendre et anticiper le comportement des systèmes informatiques, ce qui est essentiel en cybersécurité.

Travailler sur ce réseau de neurones me permet de renforcer mes compétences en programmation et en analyse, des compétences qui me seront utiles pour une carrière en sécurité informatique. De plus, ce projet est d'autant plus motivant du fait de travailler en groupe, ce rend l'expérience plus enrichissante. Nous avons déjà travaillé ensemble par le passé ce qui facilite la communication et l'entraide. Nous pouvons donc aborder des problèmes sous différents angles et apprendre les uns des autres, ce qui nous prépare à travailler efficacement en équipe pour le monde professionnel.

Antoine

Passionné par la programmation, je me spécialise actuellement dans le langage C. Dans le cadre de mes études, je travaille sur un projet de reconnaissance optique de caractères (OCR) pour les mots fléchés. Ce projet me permet de développer des compétences en traitement d'image et en analyse de texte, tout en approfondissant ma maîtrise du C. Mon objectif est de me spécialiser dans des domaines comme l'intelligence artificielle et le développement logiciel, avec toujours la volonté de relever de nouveaux défis techniques.

3 RÉPARTITION DES CHARGES



	Noé	Célian	Aurélien	Antoine
Chargement de l'image		X		
Supression des couleurs		X		
Détection	X			
Découpage	X			
Rotation de l'image	X			
Solver				X
Réseau de neurones			X	

Pour cette première soutenance, nous avons décider de répartir les tâches ainsi. En effet, puisque la détection le découpage et la rotation sont intrinsèquement liés, Noé a pris la décision de faire ces 3 tâches. Par ailleurs, Aurélien et Antoine ont des tâches bien différentes et ont donc eu une seule tâche. Célian devait charger une image et supprimer les couleurs. De plus, il a commencé le prétraitement complet de l'image pour aider Noé dans la détection de la grille ce qui nous a fait gagner du temps, pour la suite.

4 ETAT D'AVANCEMENT DU PROJET

4.1 Chargement d'une image et suppression des couleurs

Tâche réalisée par Célian MULLER

Pour charger une image nous utilisons les bibliothèques SDL et SDL-image. SDL fonctionne en utilisant un système de surfaces et de textures. En effet, une image peut être stockée en mémoire (surface) ou être utilisée par la carte graphique via le moteur de rendu (texture). SDL permet, dans un premier temps, de charger une image depuis un fichier et, par la suite, de la modifier. On peut redimensionner l'image pour l'ajuster à une partie de la fenêtre, gérer la transparence d'une image et la faire pivoter ou la transformer. Ainsi, SDL et SDL-image prennent en charge une grande variété de formats d'image (bmp, png, jpeg, gif, ...), et permettent un rendu graphique optimisé via la carte graphique, et facilitent la gestion multi-plateforme (Windows, Linux, macOS).

L'algorithme de suppression de couleur permet de filtrer les différentes couleurs de l'image pour ne rendre que du blanc et du noir. Cette tâche permet donc de faciliter la détection de la grille et des mots qui serviront pour le réseau de neurones. Le filtre permet aussi de rendre l'image plus nette, d'enlever les imperfections de celle-ci, et dans certains cas de supprimer des parties de l'image pour ne laisser passer que les mots et la grille de mot. Dans un premier temps, nous laissons apparaître un filtre noir et blanc sur les images qui en ont besoin.

➡ Find the words below and circle them (across, down, or diagonal):

- | | | |
|--------------------------------|-------------------------------|-----------------------------|
| <input type="radio"/> FLAMINGO | <input type="radio"/> TOUCAN | <input type="radio"/> SWAN |
| <input type="radio"/> KIWI | <input type="radio"/> PELICAN | <input type="radio"/> EAGLE |
| <input type="radio"/> TURKEY | <input type="radio"/> PEACOCK | |
| <input type="radio"/> CARDINAL | <input type="radio"/> PIGEON | |
| <input type="radio"/> PARROT | <input type="radio"/> SEAGULL | |



NAME: _____

DATE: _____

Strange Words

WORD FIND

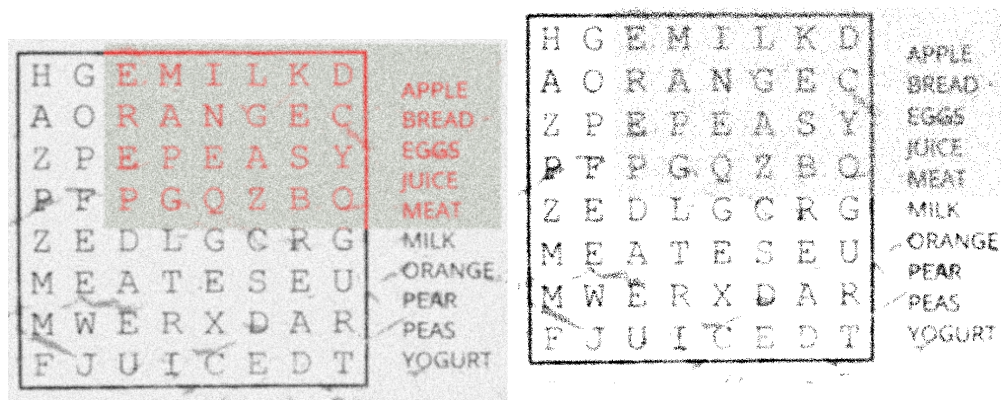


Y I M Z W J C E T A V I T S E R J K M X O H Y
 P A L I M P S E S T U X D T T E G C N D M K Y
 R B G N O I T A L U B A N N I T N I T E P D P
 O O Q I G N I K N U L E P S M E D F V T H E U
 P P A N G L O S S I A N Z D C M I T R A A F S
 R Y K J P E T R I C H O R N F O U N E L L E I
 I H F R I P P E T Q J A N C T N V A T U O N L
 O G U F S U S U R R U S X J A A J G X B S E L
 C T A T T E R D E M A L I O N M S A O O K S A
 E D E M O R D N I L A P U N O O T M Y B E T N
 P J R C N X D W E H G N A P S M M R Y M P R I
 T S X M L P E D I A N S W H U G E E G O S A M
 I G N I T A V R E N E J C L M Y S T Y C I T O
 O G E R Y T H R I S M A L I H H I X Z S S E U
 N R C F M O H F G N Y N A L T P S C Y I G P S
 R G G A I S E N M O T P Y R C S C E S D O H C

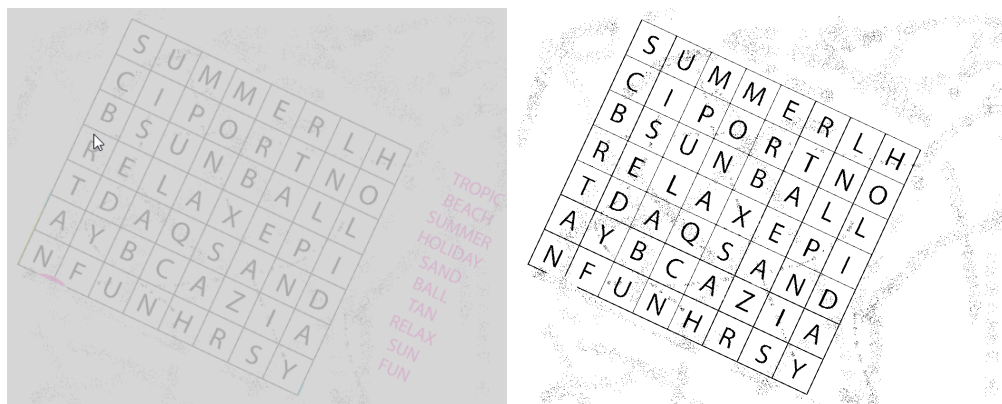
TINTINNABULATION	DEFENESTRATE	TERMAGANT
DISCOMBOBULATED	PANGLOSSIAN	SUSURRUS
OMPHALOSKEPSIS	ERYTHRISMAL	ESTIVATE
PROPRIOCEPTION	PALINDROME	SPANGHEW
TATTERDEMATION	ENERVATING	FRIPPET
PUSILLANIMOUS	PALIMPSEST	SYZYGY
CRYPTOMNESIA	SPELUNKING	TMESIS

M S W A T E R M E L O N	APPLE
Y T B N E P E W R M A E	LEMON
R R L W P A P A Y A N A	BANANA
R A N L E M O N A N E P	LIME
E W L E A P R I A B P R	ORANGE
B B I L B B W B R L A Y	WATERMELON
K E M P M A W L R A R B	GRAPE
C R E P R N R E R R G R	KIWI
A R Y A Y A O A N L A M	STRAWBERRY
L Y Y A R N E R K I W I	PAPAYA
B E B A A A N A A P R T	BLUEBERRY
Y R R E B P S A R N N W	BLACKBERRY
Y R R E B E U L B L G I	RASPBERRY
T Y P A T E A E P A C E	

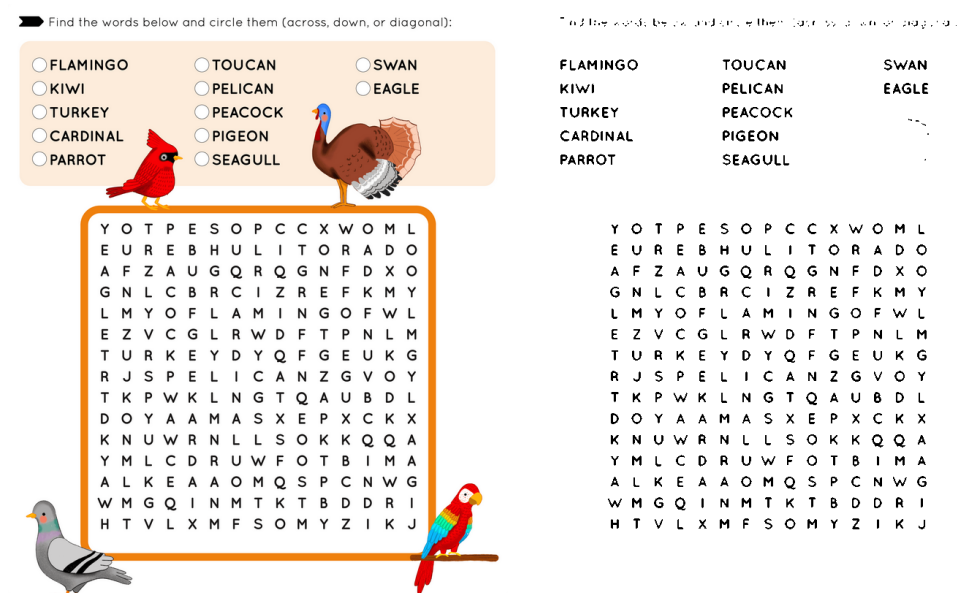
Il suffit de parcourir chaque pixel de l'image en utilisant la librairie SDL, et regarder si par rapport à une valeur seuil (choisi après différents essais) le pixel est inférieur (deviendra blanc) ou supérieur (deviendra noir).



Pour ce genre d'image, le filtre noir et blanc à lui seul ne suffisait pas. Il a fallu mettre en place un algorithme qui détecte des nuages de pixels noirs espacés et les transforme en pixel blanc pour rendre l'image la plus nette possible pour ainsi faciliter la détection de la grille et des lettres pour le réseau de neurones.



Pour faciliter la détection de grille, il est possible pour certaines images de laisser apparaître que la grille et les mots.



En utilisant un algorithme qui détecte des nuages de pixels noirs, il est possible de les filtrer et de ne laisser que la grille et les mots ce qui rend plus facile la détection de grille par la suite. L'algorithme parcourt les groupes de pixels noirs pour déterminer leur taille. Si un groupe dépasse la taille définie en paramètre, il est remplacé par des pixels blancs.

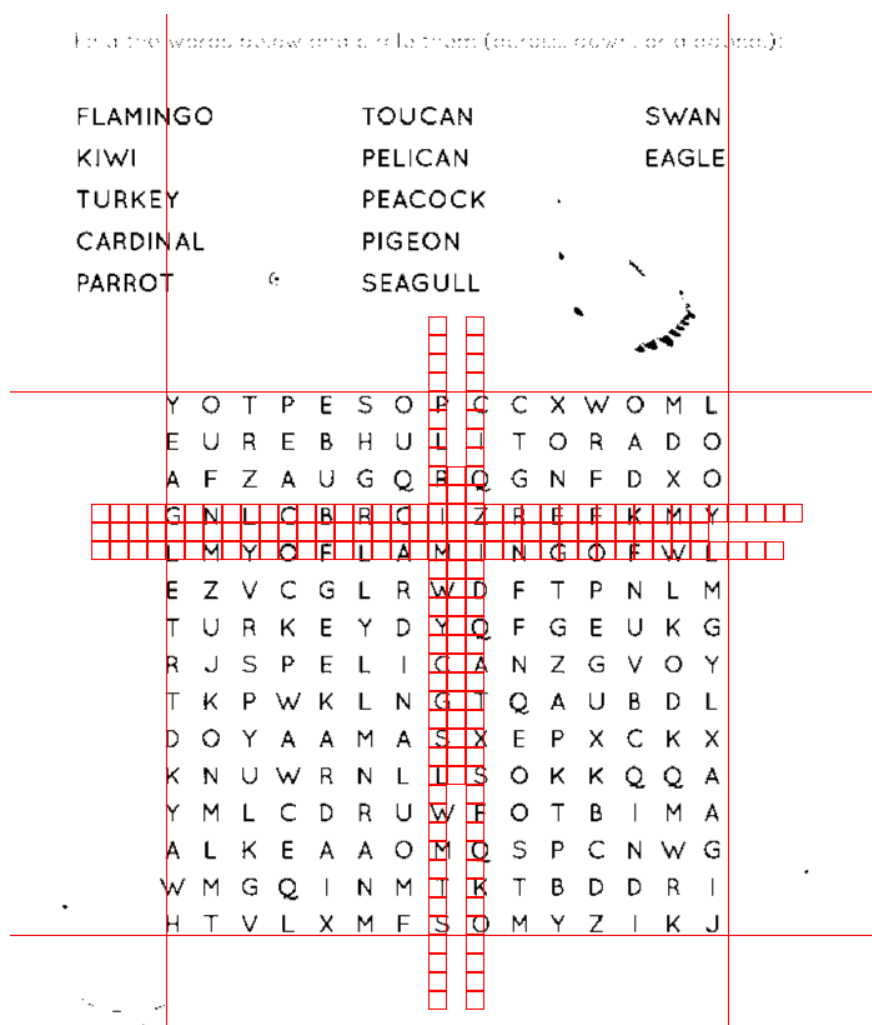
Comme la tâche de supprimer des couleurs et de charger une image était relativement simple, je me suis permis d'avancer et de commencer la suppression de bruit pour rendre l'image la plus nette possible, ce qui a fortement aidé pour la détection de la grille et des mots. Les images du niveau 2 étaient les plus dures à rendre nette puisqu'elle comportait beaucoup d'impuretés ce qui rendait les images complexe à voir, dans un premier temps, et à analyser dans un second temps. Comme le prétraitement complet n'est pas attendu pour la première soutenance, j'aurai donc plus de temps à m'accorder sur ces cas après la première soutenance. De plus, concernant le chargement des images, j'imaginai cette tâche simple comparé à la suppression des couleurs, mais je n'avais pas assez bien estimé ce temps. En effet, il a fallu se familiariser avec le langage C dans un premier temps (syntaxe et Makefile), puis installer correctement les bibliothèques SDL et SDL-image sur nos ordinateur personnels ce qui fut une tâche délicate.

4.2 Détection de la grille et extraction des mots

Tâche réalisée par Noé CORNU

Dans le cadre du projet WorldSearchSolver, j'ai pris en charge la détection de la matrice de lettres ainsi que l'extraction de la liste de mots. La principale difficulté résidait dans l'absence de grille visible dans l'image, ce qui a limité l'utilisation de méthodes standard souvent appliquées aux grilles structurées (par exemple, les solveurs de Sudoku). Après avoir étudié diverses techniques disponibles, j'ai identifié qu'il serait nécessaire de développer une méthode adaptée spécifiquement à notre projet.

En analysant les images fournies, j'ai constaté que la matrice de lettres était systématiquement centrée. En partant du centre de l'image, j'ai développé un algorithme de propagation par zone : il parcourt l'image dans toutes les directions en divisant l'espace en carrés de pixels. Lorsque la densité de pixels noirs diminue en dessous d'un seuil défini, cela marque la fin de la matrice. Cette approche permet de délimiter avec précision la grille de lettres.



Lors de la détection de la grille et de la liste de mots, la complexité des images traitées a présenté des défis variés selon leur niveau. Les images des premiers niveaux sont relativement simples, permettant une détection efficace et rapide avec notre algorithme initial, sans ajustements particuliers.

Les images de niveau 2, en revanche, introduisent une difficulté supplémentaire en raison de leur bruit visuel. Actuellement, ces interférences empêchent une détection fiable, tant pour la grille que pour la liste de mots. Pour y remédier, nous travaillons sur l'intégration de filtres d'image visant à réduire le bruit et améliorer la lisibilité des caractères. Par ailleurs, certaines images présentent une inclinaison qui complique l'analyse, nécessitant des ajuste-

ments de notre approche afin de mieux aligner la grille lors de la détection.

Pour les images de niveau 3, qui comprennent des éléments visuels plus complexes, les filtres que nous avons appliqués jusqu'à présent s'avèrent majoritairement efficaces pour éliminer les artefacts gênants. Cependant, un cas particulier persiste : les listes de mots divisées en plusieurs colonnes. Pour ces images, il a été nécessaire d'adapter l'algorithme de recherche des mots en vérifiant autour de la première colonne détectée la présence d'autres mots à gauche ou à droite. Si une autre colonne de mots est identifiée, nous appliquons le même algorithme de segmentation sur chaque colonne.

Enfin, pour les deux dernières images, la détection de la grille s'est avérée réalisable sans difficulté majeure. La liste de mots, cependant, pose encore quelques problèmes en raison de l'alignement irrégulier des mots ou de la présence d'éléments perturbateurs. Ces derniers peuvent être confondus avec les listes de mots et nécessitent un affinement de notre approche pour distinguer clairement les mots des autres formes visuelles.

4.3 Algorithme de résolution d'une grille de mots cachés

Tâche réalisée par Antoine RAMSTEIN

L'objectif de cette partie du projet est de développer un programme en C appelé `solver`, capable de résoudre des grilles de mots cachés. Le programme doit lire une grille de caractères depuis un fichier, rechercher un mot donné dans cette grille, et retourner les coordonnées de la première et dernière lettre du mot s'il est trouvé. Cette partie permet de travailler sur la manipulation de fichiers, la gestion des chaînes de caractères et la mise en œuvre de techniques de recherche multi-directionnelle dans une grille.

1. Entrées du Programme :

- Le programme doit être exécuté en ligne de commande et accepter deux arguments :
 - Nom du fichier contenant la grille de caractères.

- Mot à rechercher dans la grille.
- La grille est un fichier texte formaté, où chaque ligne représente une rangée de la grille et chaque caractère est une lettre majuscule (A-Z).
- Le mot à rechercher peut être donné en majuscules ou en minuscules, sans distinction de casse.

2. Format de la Grille :

- La grille est composée de lettres majuscules et doit avoir un format rectangulaire avec un minimum de 5 lignes et 5 colonnes.
- Chaque ligne du fichier représente une ligne de la grille, avec des caractères séparés par des espaces ou collés.

3. Recherche du Mot :

- Le programme doit rechercher le mot en suivant huit directions possibles à partir de chaque lettre dans la grille :
 - Horizontalement : de gauche à droite et de droite à gauche.
 - Verticalement : de haut en bas et de bas en haut.
 - Diagonalement :
 - * Haut gauche vers bas droit.
 - * Bas gauche vers haut droit.
 - * Haut droit vers bas gauche.
 - * Bas droit vers haut gauche.
- Si le mot est trouvé, le programme retourne les coordonnées de la première et dernière lettre du mot dans la grille.
- Si le mot n'est pas trouvé, le programme affiche "Not found".

4. Format de la Sortie :

- **Mot trouvé** : Si le mot est trouvé, le programme affiche les coordonnées dans le format $(x0, y0) (x1, y1)$, où :
 - $(x0, y0)$ sont les coordonnées de la première lettre du mot dans la grille.

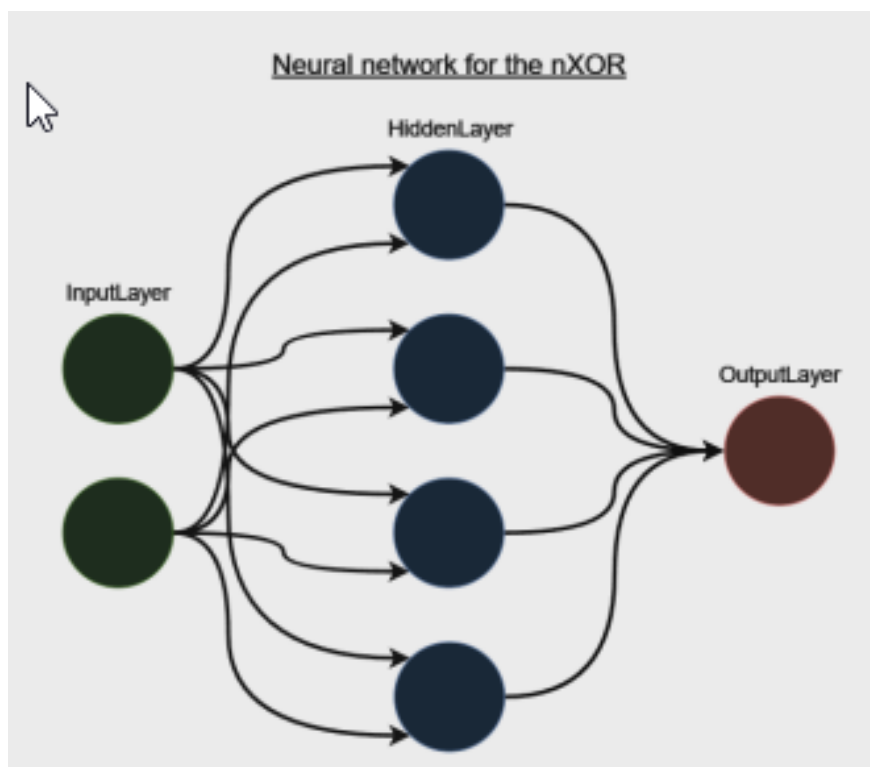
- $(x1, y1)$ sont les coordonnées de la dernière lettre du mot.
- **Mot non trouvé** : Si le mot n'est pas trouvé dans la grille, le programme affiche simplement "Not found".
- Les coordonnées $(0,0)$ correspondent au coin supérieur gauche de la grille.

4.4 Concept du réseau de neurones

Tâche réalisée par Aurélien MEYER

La conception du réseau de neurones a tout d'abord commencé par des recherches notamment sur le fonctionnement de celui-ci.

Pour cette première soutenance, mon avancement sur le développement du réseau de neurones devait se traduire par un exemple concret : faire apprendre à l'intelligence artificielle une fonction simple, celle du nXOR. Pour y parvenir, mon réseau de neurones devait ressembler à ceci:



-2 inputs d'entrée chacun pouvant prendre la valeur 0 ou 1

-4 couches cachées

-1 input de sortie pouvant prendre la valeur 0 ou 1

Je devais me munir d'une base de données pour entraîner mon programme. Dans ce cas, cela n'a pas été compliqué étant donné la simplicité de la fonction à reproduire.

A	B	$A \leftrightarrow B$ (XNOR)
0	0	1
0	1	0
1	0	0
1	1	1

Pour chaque paire d'input, je connaissais l'output attendue. J'ai donc pu répéter le processus d'entraînement un nombre assez conséquent de fois jusqu'à que les poids de chaque lien et les biais de chaque neurone soient parfaitement ajustés.

Mon premier réseau de neurones a donc appris la fonction nXOR. Ainsi, pour sauvegarder son résultat, il me suffit de stocker les poids de chaque lien et les biais de chaque neurone. Le début a été très difficile en raison de la difficulté de concevoir un réseaux de neurones, mais avec du temps et des exemples, j'ai pu comprendre les principes fondamentaux, affiner les paramètres et finalement obtenir des résultats prometteurs.

5 ASPECT TECHNIQUE

5.1 Prétraitement

Le prétraitement utilise pour l'instant 3 fonction : `apply-black-and-white-filter`, `clean-large-black-groups` et `clean-image`. `apply-black-and-white-filter` est une fonction qui renvoie une image sous forme de pixels noirs et blancs.

$$\text{pixel} = 0.3 * \text{red} + 0.59 * \text{green} + 0.11 * \text{blue}$$

En effet, en utilisant cette expression nous pouvons convertir la couleur d'un pixel en un niveau de gris. Une fois cette valeur obtenue, il suffit de la comparer à une valeur de référence, ici 180, pour obtenir du noir ou du blanc.

`clean-large-black-groups` est une fonction permet de "nettoyer" les grands groupes de pixels noirs et de les rendre blanc. En effet, on parcourt chaque pixel pour vérifier s'il fait partie d'un groupe de pixels noirs. Pour chaque groupe de pixels noirs détectés, on utilise une fonction Flood-fill pour compter la taille du groupe et enregistrer ses coordonnées. Si la taille du groupe de pixels noirs dépasse une limite donnée en argument (ici 150), on remplace tous ses pixels par des pixels blancs, sinon on le garde tel quel.

`clean-image` fonctionne sur le même principe, mais pour les pixels trop espacé, permettant une fois supprimé de rendre l'image la plus lisible possible. La valeur arbitraire choisi est 7, s'il y a 7 pixels blancs ou plus autour d'un pixel noir, celui-ci devient blanc.

5.2 Détection de la grille et extraction des mots

Fonction principale : `detect_letter_grid`

1. **Initialisation des bornes** : Fixe les limites de la grille (haut, bas, gauche, droite) aux valeurs maximales/minimales de l'image.
2. **Point de départ et propagation** : Définit des points de départ proches du centre de l'image. Ces points servent de repères pour propager l'algorithme dans les quatre

directions (gauche, droite, haut, bas) pour identifier la grille.

3. Propagation :

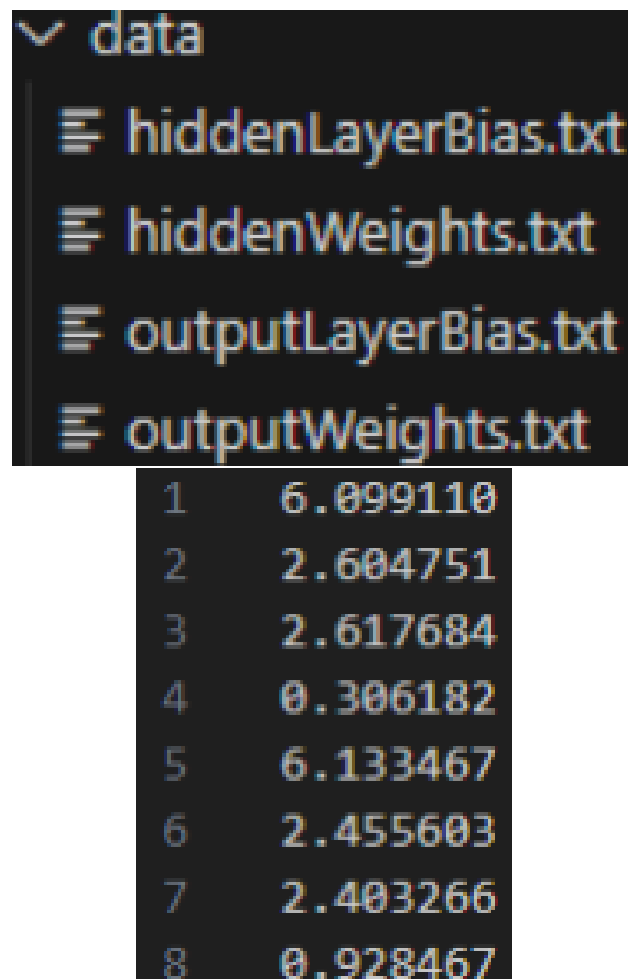
- Dans chaque direction (gauche, droite, haut, bas), l'algorithme compte le nombre de pixels noirs dans des blocs carrés.
- Si un bloc contient suffisamment de pixels noirs, il est considéré comme faisant partie de la grille et les bornes sont ajustées.
- Si un bloc contient trop de pixels blancs (en dessous d'un seuil de noirceur), l'algorithme arrête la propagation dans cette direction, marquant ainsi la limite de la grille.

4. Fonction de comptage de pixels noirs (`count_black_pixels_in_block`) : Compte les pixels noirs dans un bloc donné et retourne ce nombre. Les pixels noirs sont identifiés selon une tolérance.

Une fois la grille détectée, je procède au comptage du nombre de lignes et de colonnes pour découper chaque lettre. L'algorithme s'étend ensuite autour de la grille pour repérer la liste de mots, représentée par un bloc de pixels noirs distinct. Un traitement similaire est appliqué pour segmenter chaque mot en lettres individuelles, facilitant ainsi leur extraction et leur utilisation dans le solveur.

5.3 Réseaux de neurones

L'implémentation du réseau de neurone prend la forme de deux fonctions, l'une pour l'entraînement et l'autre pour son fonctionnement. Pour faciliter l'échange et la sauvegarde des données entre les deux fonctions les données comme les poids et les biais sont sauvegarder dans différent fichier.



```
data
├── hiddenLayerBias.txt
├── hiddenWeights.txt
├── outputLayerBias.txt
└── outputWeights.txt
```

1	6.099110
2	2.604751
3	2.617684
4	0.306182
5	6.133467
6	2.455603
7	2.403266
8	0.928467

Exemple de données sauvegarder dans le fichier hiddenLayerBias

Il est possible de se dire qu'avec un très grand réseau de neurones la sauvegarde de données est volumineuse, mais en réalité un tel fichier texte reste peu volumineux car l'on y enregistre que des caractères. En suivant les instructions et en manipulant correctement les structures de données, l'IA peut maintenant "apprendre" à partir d'exemples.

La partie principale de mes fonctions sont **le passage avant (Forward pass)** qui calcule le résultat du réseau de neurone en fonction des inputs et **la rétropropagation (Backpropagation)** qui applique les changements à effectuer pour corriger les erreurs.

5.4 Solver

La fonction principale de recherche parcourt chaque cellule de la grille. Pour chaque cellule correspondant à la première lettre du mot, le programme explore les huit directions pour vérifier si le mot complet est présent. La fonction `searchWord` utilise une boucle pour parcourir le mot dans la direction spécifiée. Si une lettre ne correspond pas ou si la position sort des limites de la grille, la recherche dans cette direction est abandonnée. Si le mot est trouvé dans une direction, les coordonnées de la dernière lettre sont stockées et retournées. Un tableau de directions est utilisé pour simplifier la gestion des déplacements dans la grille. Chaque direction est représentée par une paire de valeurs (dx, dy) indiquant le déplacement en abscisse et en ordonnée.

```
int directions[8][2] = {
    {0, 1},
    {0, -1},
    {1, 0},
    {-1, 0},
    {1, 1},
    {1, -1},
    {-1, 1},
    {-1, -1}
};
```

Fonctionnement des directions

Chaque direction est testée depuis une cellule jusqu'à ce que le mot soit trouvé ou que toutes les directions aient été explorées.

Enfin, la fonction `isValid` vérifie que les coordonnées (x, y) restent dans les limites de la grille, évitant ainsi les débordements de mémoire ou les accès invalides. Cette fonction est utilisée dans la boucle de recherche pour chaque direction.

Le programme lit la grille depuis le fichier spécifié en ligne de commande, et chaque ligne du fichier est lue dans un tableau de caractères à deux dimensions (une matrice). Chaque ligne est lue et stockée en respectant les dimensions maximales de la grille définies par les constantes `ROWS` et `COLS`.

Contraintes et limites

Le programme solver est conçu pour fonctionner avec des grilles de caractères de taille maximale définie par les constantes ROWS et COLS, pour le moment, fixées à 100x100. Les dimensions de la grille doivent respecter cette limite, et toute modification pour des grilles plus grandes nécessiterait des ajustements du code. La recherche de mots dans la grille n'est pas sensible à la casse, ce qui permet d'ignorer la distinction entre majuscules et minuscules lors de la correspondance des lettres. Le programme explore chaque cellule de la grille dans les huit directions (horizontal, vertical et diagonal), ce qui entraîne une complexité de recherche de l'ordre de $O(n * m * 8 * \text{len}(\text{word}))$, où n et m représentent les dimensions de la grille, et $\text{len}(\text{word})$ la longueur du mot à rechercher. Cette approche exhaustive assure une recherche complète mais peut ralentir le programme pour des grilles et des mots de grande taille, surtout si plusieurs occurrences du mot sont possibles.

6 CONCLUSION

Nous avons présenté la répartition des charges, l'état d'avancement du projet ainsi que les aspects techniques de celui-ci. Cette base nous permettra d'avancer efficacement dans le bon déroulement du projet. Pour rappel, nous avons élaborer des algorithmes capables de charger une image, supprimer les couleurs, faire une rotation manuelle de l'image, détecter la position de grille, mots et lettres, découper les images, résoudre une grille de mots cachés ainsi qu'un concept de réseau de neurones. Pour la soutenance finale, nous devrons présenter le prétraitement complet, un réseau de neurones complet et fonctionnel, la reconstruction de la grille, de la liste de mots et de la grille, l'affichage de la grille, la sauvegarde du résultat et une interface graphique. Les objectifs de ce projet vont au-delà de la réalisation du projet lui-même, que ce soit via le développement personnel et professionnel des membres du groupe mais également via l'apprentissage de compétences techniques et humaines qui nous aideront pour de futur projet.