Rapport 1 OCR

Zachary COATES - Manon BIÈRE - Norah Beillevert

novembre 2025

Sommaire

Introduction

Ce document constitue le premier rapport intermédiaire pour le projet "OCR Word Search Solver" , réalisé dans le cadre du cursus S3 à l'EPITA durant l'année 2024/2025.

L'objectif de ce projet est de concevoir et de développer un logiciel de reconnaissance optique de caractères (OCR) capable de résoudre automatiquement des grilles de mots cachés à partir d'une image. Le défi technique repose sur la chaîne de traitement complète, allant du chargement et du prétraitement de l'image (incluant la correction des rotations et l'amélioration du contraste), à la détection et la reconstruction de la grille et de la liste de mots, en passant par la reconnaissance des caractères via un réseau de neurones que nous devons entraîner nous-mêmes. Enfin, un algorithme de résolution, développé en langage C, trouve les mots dans la grille reconstituée et le résultat est présenté via une interface graphique.

Ce premier rapport a pour but de présenter l'état d'avancement du projet à mi-parcours, conformément aux attentes de la première soutenance. Nous y détaillerons la répartition des missions au sein de l'équipe, les avancées significatives sur les trois axes principaux que sont les rotations, l'intelligence artificielle et le solver, ainsi que les difficultés rencontrées et les pistes d'amélioration envisagées pour la suite du développement.

Présentation des membres et de leurs missions

Zachary COATES :

Je m'appelle Zachary, et dans notre projet, je suis responsable de toute la partie réseau de neurones — autrement dit, c'est moi qui dois apprendre à la machine à reconnaître les lettres. Pour ça, j'ai d'abord dû me plonger dans le fonctionnement des réseaux de neurones, histoire de comprendre comment leur faire apprendre quelque chose sans qu'ils explosent au premier essai.

J'ai commencé avec un petit réseau simple capable de résoudre un XNOR, juste pour prouver que je pouvais lui faire "comprendre" quelque chose. C'était un peu la version entraînement avant le vrai combat.

L'objectif maintenant, c'est de transformer ce prototype en un réseau capable d'identifier les lettres de l'alphabet à partir d'images. En bref, je passe d'un mini-cerveau qui sait faire du binaire à un modèle qui apprend à lire

Manon BIERE :

Je m'appelle Manon, et au sein de notre groupe, je suis principalement en charge du solver ainsi que de la reconstruction de la grille et de la liste de mots à partir des résultats fournis par la reconnaissance de caractères. Mon travail s'inscrit donc dans la partie finale du projet, celle qui assure la liaison entre la phase d'analyse des images et la résolution effective de la grille.

Dans un premier temps, j'ai développé le programme solver en langage C, capable d'analyser une grille de mots cachés contenue dans un fichier texte et de déterminer la position d'un mot donné. Ce module constitue le cœur logique de la résolution et sert également de base de test pour valider les résultats issus du traitement d'image.

La seconde partie de ma mission consistera à reconstruire la grille de mots et la liste des mots à rechercher à partir des données issues du réseau de neurones.

Norah BEILLEVERT :

Je m'appelle Norah, et dans le cadre de notre projet, je suis responsable de toute la partie détection et prétraitement d'image. Mon travail consiste à transformer une image brute, souvent issue d'une photo ou d'un scan, en un ensemble d'éléments propres et exploitables : la grille de mots, la liste des mots à chercher et, enfin, les lettres individuelles prêtes à être reconnues. En résumé, c'est la partie qui permet à la machine de "comprendre" ce qu'elle voit, avant même de tenter de lire les caractères.

C'est une étape essentielle, car la qualité du prétraitement conditionne directement les performances du réseau de neurones et du solver. Si l'image d'entrée est floue, mal éclairée ou inclinée, tout le reste du pipeline peut échouer. Mon rôle est donc de garantir que les données envoyées aux autres modules soient les plus nettes et les plus structurées possibles.

Les Avancement

Traitement de l'image :

La phase de détection des contours et d'isolement de la grille a constitué un moment charnière dans le développement du pipeline de traitement d'images, représentant une part significative du travail technique accompli jusqu'à présent. Nous avons conçu et mis en œuvre un algorithme sophistiqué de remplissage par zones connexes (flood fill) qui parcourt systématiquement l'image pixel par pixel selon une approche voisinage-4 et voisinage-8, permettant de regrouper efficacement les régions noires adjacentes tout en respectant la connectivité des composantes. Chaque groupe de pixels ainsi identifié fait ensuite l'objet d'une analyse géométrique poussée aboutissant à la création d'une boîte englobante (BoundingBox) optimale, qui délimite avec précision l'ensemble des formes textuelles détectées dans le document source.

Cette méthodologie nous a permis d'identifier avec un taux de succès remarquable la zone correspondant à la grille principale, mais également de détecter de manière fiable les autres blocs de texte significatifs, en particulier la liste lexicale des mots à rechercher. L'algorithme intègre des mécanismes de filtrage avancés basés sur des ratios hauteur/largeur et des surfaces minimales, éliminant ainsi la majorité des artefacts et éléments parasites tout en conservant les caractères d'intérêt.

Une fois ces boîtes calculées et validées, le processus de reconstruction de la structure logique de la grille a été engagé. Par une analyse statistique fine de la répartition spatiale des centroïdes de chaque boîte détectée, le programme détermine automatiquement la topologie de la grille, inférant le nombre précis de lignes et de colonnes grâce à des algorithmes de clustering adaptés. Cette analyse géométrique sophistiquée permet un découpage régulier et systématique de la grille, où chaque cellule contenant un caractère est extraite sous la forme d'une image individuelle normalisée selon un format standardisé.

L'ensemble de ces images de caractères est soigneusement organisé et sauvegardé dans une arborescence de dossiers spécialisés – out/cells pour les lettres de la grille et out/words pour celles de la liste – constituant ainsi un jeu de données structuré, parfaitement adapté aux exigences de la phase d'apprentissage supervisé du réseau de neurones. Cette organisation méthodique facilite grandement le processus d'entraînement et de validation des modèles de reconnaissance.

Pour garantir la robustesse et la transparence de ce processus complexe, un module complet de visualisation graphique temps réel a été implémenté à l'aide de la bibliothèque SDL2. Ce module affiche de manière superposée sur l'image originale les rectangles colorés entourant chaque région détectée, avec un code couleur distinct pour la grille, la liste de mots et les éléments rejetés. Cette fonctionnalité de débogage visuel avancé s'est révélée indispensable pour valider instantanément le bon fonctionnement des algorithmes et pour ajuster de manière interactive les nombreux paramètres et seuils de détection.

L'intelligence artificielle :

Dans le cadre du projet, ma mission consiste à donner un peu d'intelligence au programme : créer le réseau de neurones chargé d'apprendre à reconnaître les lettres à partir d'images. Autrement dit, c'est la partie où la machine passe de "je vois des pixels" à "je sais ce que c'est". Pour l'instant, mon travail s'est concentré sur la compréhension du fonctionnement interne d'un réseau de neurones et sur la création d'un premier prototype capable d'apprendre quelque chose par lui-même.

Avant de m'attaquer à la reconnaissance de caractères, j'ai commencé par me plonger dans la théorie : comment un neurone artificiel prend une décision, comment il apprend de ses erreurs et comment toutes ces petites unités peuvent coopérer pour produire un résultat cohérent. Pour mettre tout ça en pratique, j'ai développé un petit réseau de neurones en C capable de résoudre le problème logique du XNOR. C'est un test simple, mais parfait pour s'assurer que tout fonctionne comme prévu — un peu la "preuve de vie" du cerveau avant de lui apprendre à lire.

La prochaine étape sera de transformer ce prototype en un véritable modèle de reconnaissance de lettres. Pour ça, il faudra adapter l'architecture du réseau, gérer un volume beaucoup plus important de données et utiliser un jeu d'images comme EMNIST pour l'entraîner. L'objectif est clair : passer d'un petit réseau logique à un modèle capable de reconnaître les lettres de l'alphabet à partir d'images réelles.

Le développement est entièrement réalisé en langage C, sans bibliothèque externe. Ce choix me permet de garder la main sur tout ce qui se passe dans le réseau — du calcul des poids à la propagation des erreurs — et d'assurer une parfaite compatibilité avec le reste du projet. Le travail effectué sur le XNOR constitue donc une base solide, sur laquelle je pourrai construire la suite du système.

En résumé, cette première phase m'a permis de comprendre les mécaniques de l'apprentissage automatique et de poser les fondations du futur réseau de reconnaissance de lettres. Le projet entre maintenant dans une phase plus concrète, où le défi sera d'apprendre à la machine non seulement à réfléchir, mais aussi à lire.

Le solver :

La première phase de ma contribution au projet a été consacrée à la conception et à l'implémentation du solver, un programme en langage C permettant d'analyser une grille de mots cachés et de localiser la position exacte d'un mot donné. Cette partie a été entièrement finalisée dans les délais, validée par plusieurs jeux de tests, et répond pleinement aux exigences du cahier des charges.

L'objectif principal était de développer un outil capable de parcourir une grille de taille variable et d'identifier un mot selon toutes les directions possibles : horizontale, verticale et diagonale, dans les deux sens. Le solver devait également respecter un format d'entrée et de sortie strict, avec une gestion rigoureuse des coordonnées et des cas où le mot est absent de la grille.

Le développement a été effectué en C standard, en utilisant exclusivement les bibliothèques stdio.h, stdlib.h et err.h, afin de garantir la compatibilité avec l'environnement fourni par l'école et la légèreté du code. L'implémentation repose sur une lecture structurée du fichier contenant la grille, ligne par ligne, suivie d'une conversion en tableau dynamique de caractères permettant un accès direct aux coordonnées. Une attention particulière a été portée à la gestion de la mémoire dynamique (malloc et free), afin d'assurer la stabilité du programme et d'éviter toute fuite mémoire.

L'algorithme de recherche repose sur une approche systématique : chaque case de la grille est explorée comme point de départ potentiel, et le programme vérifie successivement la correspondance des caractères du mot dans les huit directions possibles, en tenant compte des limites de la grille pour éviter tout dépassement d'indice. Ce choix garantit une recherche exhaustive, tout en conservant une bonne lisibilité et un comportement déterministe.

De nombreux tests ont été réalisés à l'aide de grilles issues du sujet, notamment l'exemple fourni dans le cahier des charges, afin de vérifier la précision et la robustesse du programme. Les résultats obtenus ont systématiquement correspondu aux coordonnées attendues, confirmant la fiabilité du solver. L'ensemble du code compile sans erreur avec les options -Wall -Wextra et respecte les conventions de lisibilité et de clarté imposées par le projet.

Cette étape, désormais totalement opérationnelle, constitue la base logique de la partie finale du projet. Elle permettra d'intégrer la reconstruction de la grille et la résolution automatique dans le processus complet, assurant la cohérence entre les données issues de l'OCR et le résultat final affiché à l'utilisateur.

Les difficultés et problèmes rencontres

Traitement de l'image :

L'implémentation de ce module exigeant n'a naturellement pas été exempte d'obstacles techniques, chacun ayant représenté une opportunité d'approfondissement conceptuel. Une difficulté majeure a concerné la détection fiable de caractères au tracé particulièrement fin, tels que le "I" majuscule ou le "l" minuscule. En raison de leur empreinte pixelique réduite, ces caractères génèrent des composantes connexes de taille critique souvent inférieure aux seuils de détection, conduisant à des omissions fâcheuses. Nous avons partiellement contourné ce problème par l'implémentation d'un pré-traitement de dilatation morphologique contrôlée, mais une solution plus élégante par analyse de la densité locale est à l'étude.

À l'opposé, nous avons constaté que certains éléments graphiques parasites, notamment des bordures décoratives, des taches d'encre ou des annotations marginales, pouvaient être interprétés à tort comme des lettres, générant ainsi des faux positifs problématiques. La distinction entre ces éléments et les caractères légitimes représente un défi de classification qui pourrait bénéficier à l'avenir de l'intégration de critères de forme plus avancés, voire d'un micro-réseau de neurones dédié.

La question épineuse de la rotation automatique des images inclinées constitue un autre front de recherche actif. Bien qu'une fonctionnalité de rotation manuelle par saisie angulaire existe d'ores et déjà, son automatisation fiable nécessite une analyse géométrique autrement plus complexe. L'implémentation d'une transformée de Hough probabiliste pour détecter l'orientation prédominante des alignements de caractères est une piste sérieusement envisagée pour estimer et corriger avec précision l'angle d'inclinaison de la grille. Des tests préliminaires sur des images synthétiques sont encourageants.

Enfin, nous devons résoudre un problème subtil d'ordonnancement lors de la sérialisation des images de lettres. Bien que tous les fichiers soient correctement générés et nommés, leur séquence de sauvegarde ne reflète pas systématiquement l'ordre de lecture logique de la grille (de gauche à droite, de haut en bas). Cette limitation indique la nécessité d'implémenter un algorithme de tri spatial supplémentaire, probablement basé sur un double tri lexicographique selon les coordonnées (y, x) des centroïdes de chaque boîte, afin de garantir une correspondance parfaite entre l'ordre des fichiers et la structure sémantique de la grille.

Ces difficultés, bien que substantielles, sont intrinsèques au domaine exigeant du traitement d'images et de la vision par ordinateur. Leur résolution progressive, par itérations successives et validation empirique, s'avère extrêmement formatrice et contribue significativement à renforcer la robustesse et la précision globale de notre application. Les solutions envisagées et partiellement testées tracent une feuille de route technique claire et stimulante pour les semaines de développement à venir.

L'intelligence artificielle :

Cette partie du projet n'a pas été sans difficultés. Avant de commencer, je n'avais jamais codé de réseau de neurones, ni même travaillé en langage C. J'ai donc dû tout apprendre depuis la base — aussi bien le fonctionnement théorique d'un réseau que la logique parfois un peu... particulière du C. Autant dire que les premiers pas ont été remplis d'erreurs de segmentation, de pointeurs rebelles et de matrices capricieuses.

La principale difficulté a été de comprendre comment traduire les concepts mathématiques d'un réseau de neurones en code bas niveau. Là où d'autres langages offrent des bibliothèques toutes prêtes, tout devait ici être construit à la main : la structure des neurones, la propagation des signaux, la mise à jour des poids, et même le calcul des erreurs. Chaque ligne de code était donc un mélange entre mathématiques, logique, et un soupçon de débogage intensif.

Apprendre le C en parallèle n'a pas simplifié les choses, mais m'a permis de réellement comprendre ce que fait le programme à chaque étape. J'ai dû apprendre à gérer la mémoire manuellement, à manipuler des tableaux dynamiques, et à faire attention au moindre détail pour éviter qu'un simple oubli ne fasse planter tout le réseau. C'était parfois frustrant, mais aussi très formateur : à force de corriger des erreurs, on finit par mieux comprendre comment tout fonctionne.

Malgré ces obstacles, chaque petite victoire — qu'il s'agisse d'un neurone qui apprend enfin ou d'un code qui compile sans erreur — a rendu le processus d'autant plus satisfaisant. Cette phase d'apprentissage, bien qu'exigeante, m'a permis d'acquérir une vraie compréhension du fonctionnement interne d'un réseau de neurones et de renforcer mes compétences en programmation bas niveau.

Le solver :

La mise en œuvre du solver, bien que finalisée avec succès, n'a pas été exempte de défis techniques significatifs. Ces obstacles ont principalement concerné deux aspects fondamentaux du langage C : la manipulation avancée des pointeurs et la gestion précise des algorithmes de parcours.

La première difficulté majeure a émergé lors de la création et de la navigation dans la grille en deux dimensions. Utiliser un tableau de pointeurs de caractères (char **grid) s'est révélé plus complexe que prévu, notamment pour déterminer correctement les limites de la grille lors des accès mémoire. Des erreurs de segmentation survenaient fréquemment, causées par des dépassements d'indices ou des pointeurs non initialisés. J'ai dû consacrer un temps considérable à déboguer pas à pas, en ajoutant des affichages intermédiaires, pour visualiser l'état de la grille et identifier les cases accédées illicitement.

La seconde difficulté, plus subtile, concernait l'algorithme de recherche lui-même. Une première version de la fonction search parvenait à localiser le mot cible, mais ne retournait systématiquement que les coordonnées de départ (x0, y0), sans calculer correctement les coordonnées d'arrivée (x1, y1). Le problème provenait d'une mauvaise gestion du vecteur de direction dans la boucle de vérification. Plus précisément, l'incrémentation des coordonnées x et y se faisait de manière cumulative, mais la restitution des positions finales n'intégrait pas correctement le décalage induit par la longueur du mot et la direction empruntée.

La résolution de ces problèmes a été le fruit d'une démarche empirique et persévérante. Face au bug des coordonnées finales, j'ai choisi de réécrire entièrement la logique de parcours des directions. J'ai introduit un système de vecteurs dx et dy plus explicite, et j'ai méticuleusement recalculé la formule de sortie pour x1 et y1 en fonction du nombre de caractères parcourus (k-1). Cette approche, bien que fastidieuse, m'a permis de comprendre en profondeur la mécanique des déplacements dans la grille et de corriger définitivement l'algorithme.

Ces obstacles, bien que frustrants sur le moment, se sont avérés extrêmement formateurs. Ils ont renforcé ma maîtrise de l'allocation dynamique et de la manipulation des pointeurs multiples en C, et m'ont appris l'importance cruciale du test incrémental et de la vérification des états intermédiaires dans le développement d'algorithmes complexes.

Pistes d'améliorations et prochaines étapes :

Traitement de l'image :

Au vu des avancements actuels et des exigences du cahier des charges, plusieurs axes d'amélioration et développements futurs se dessinent clairement pour mener le projet à son terme avec succès.

Perfectionnement du prétraitement d'image La détection automatique de la rotation par transformée de Hough constitue une priorité. Cette fonctionnalité, bien qu'amorcée, nécessite un calibrage fin pour corriger avec précision les déviations angulaires mineures qui perturbent actuellement l'extraction des caractères. Parallèlement, l'enrichissement des filtres de nettoyage d'image (réduction du bruit, amélioration du contraste) permettrait de traiter avec une fiabilité accrue les documents de niveau 2 et 3, notamment ceux présentant un fond granuleux ou un contraste hétérogène.

Optimisation du réseau de neurones La phase d'apprentissage du réseau de neurones représente le prochain chantier majeur. L'architecture actuelle devra être étendue et entraînée sur un jeu de données conséquent, constitué à partir des caractères extraits et enrichi par des données synthétiques. L'objectif est d'atteindre un taux de reconnaissance supérieur à 95

Renforcement de la robustesse du solver L'intégration d'un mécanisme de fuzzy matching au solver existant est une évolution stratégique. En tolérant un faible taux d'erreur dans la reconnaissance des caractères (par exemple via la distance de Levenshtein), le système maintiendra sa fonctionnalité même face à une grille reconstituée imparfaite. Cette amélioration, couplée à une gestion plus fine des mots à trait d'union ou des apostrophes, élargira significativement le champ d'application du logiciel.

Développement de l'interface graphique et finalisation La création d'une interface graphique intuitive et réactive constitue l'ultime étape d'intégration. Celle-ci devra orchestrer l'ensemble de la chaîne de traitement : chargement de l'image, paramétrage du prétraitement, lancement de la reconnaissance, résolution et affichage des résultats. La visualisation de la grille résolue, avec mise en évidence colorée des mots trouvés et possibilité de sauvegarde, sera au cœur des attendus. Une attention particulière sera portée à l'expérience utilisateur, en permettant des corrections manuelles ponctuelles (rotation, sélection de zone) si nécessaire.

Synthèse et feuille de route La coordination de ces différentes briques logicielles et la réalisation de tests intensifs sur une large gamme d'images constitueront la phase finale. L'objectif est de livrer une application homogène, performante et conforme au cahier des charges, prête pour la démonstration finale. La gestion rigoureuse des délais et la répartition claire des tâches au sein de l'équipe seront les facteurs clés pour concrétiser cette ambition dans le temps imparti.

L'intelligence artificielle :

La prochaine étape consiste à transformer le prototype actuel en un réseau capable d'identifier les lettres de l'alphabet à partir d'images. Cela impliquera d'adapter l'architecture du réseau pour gérer un plus grand nombre de neurones et de couches, d'entraîner le modèle sur un ensemble de données complet comme EMNIST, et de peaufiner les paramètres pour obtenir un taux de reconnaissance satisfaisant. L'objectif est de passer d'un réseau qui ne fait que résoudre un XNOR à un modèle capable de "lire" les lettres de façon fiable, même lorsque les images sont imparfaites ou bruitées.

Une fois cette étape franchie, il faudra harmoniser le réseau avec le reste du projet. Cela signifie que les lettres reconnues devront être transmises correctement au module de reconstruction de la grille et au solver, afin que chaque caractère puisse être exploité directement pour résoudre les mots cachés. Cette intégration demandera une réflexion sur le format des données, la communication entre les modules, et la robustesse globale du système. En d'autres termes, le réseau devra non seulement apprendre à reconnaître les lettres, mais aussi "parler le même langage" que le reste du projet.

En résumé, mon travail à venir se concentre sur trois points principaux : développer et entraîner le réseau pour la reconnaissance de toutes les lettres de l'alphabet, tester et optimiser sa précision sur des images variées, et l'intégrer de manière fluide avec le module de reconstruction de la grille et le solver. Ces étapes permettront de transformer le prototype actuel en un composant pleinement fonctionnel, capable de convertir les images brutes en lettres exploitables, et de compléter ainsi la partie "intelligence" du projet.

Le solver :

Le développement du solver ayant atteint un stade fonctionnel satisfaisant, les prochaines étapes viseront principalement à renforcer sa robustesse et à parfaire son intégration au sein de l'écosystème logiciel complet. L'enjeu majeur réside désormais dans la gestion des imprécisions inhérentes à la reconnaissance optique de caractères. En effet, les sorties du réseau de neurones peuvent occasionnellement comporter des erreurs, produisant une grille reconstituée avec des lettres manquantes ou incorrectes. Pour garantir un taux de réussite élevé même dans ces conditions dégradées, une piste d'amélioration essentielle consiste à implémenter un mécanisme de recherche approximative, ou fuzzy matching.

L'idée sous-jacente est de permettre au solver d'identifier un mot même si sa représentation dans la grille comporte une légère divergence, par exemple un caractère erroné ou manquant. L'utilisation d'un algorithme de distance d'édition, comme la distance de Levenshtein, permettrait de quantifier la similarité entre le mot recherché et les séquences de la grille. En acceptant les correspondances dont le score de similarité dépasse un certain seuil, le solver gagnerait considérablement en tolérance aux défauts, une caractéristique indispensable pour traiter des images de qualité hétérogène correspondant aux niveaux 2 et 3 du cahier des charges.

Sur le plan de l'intégration, l'objectif est de faire du solver un module parfaitement interconnecté avec les autres composants. Cela impliquera de modifier son entrée pour qu'il lise la grille directement depuis la structure de données en mémoire générée par le module OCR, plutôt que depuis un fichier texte. Ses résultats, c'est-à-dire les coordonnées des mots trouvés, devront ensuite être transmis de manière structurée au module d'affichage graphique. Cette synergie est fondamentale pour permettre la superposition des solutions sur l'image originale de la grille, offrant ainsi à l'utilisateur une visualisation claire et intuitive des mots retrouvés, et achevant de la sorte la chaîne de traitement automatisé.

Conclusion

Ce premier rapport d'avancement témoigne des progrès significatifs accomplis depuis le lancement du projet OCR Word Search Solver. Conformément au calendrier établi, les fondations des trois piliers du système — à savoir le prétraitement d'image avec les rotations, le réseau de neurones pour la reconnaissance des caractères, et le solver pour la résolution de la grille — ont été posées avec rigueur. Chaque membre de l'équipe a pleinement investi sa mission, permettant une avancée coordonnée et méthodique.

Les fonctionnalités obligatoires pour la première soutenance, telles que la détection des éléments de la grille, l'extraction des lettres, et l'implémentation du solver, sont d'ores et déjà opérationnelles. Bien que certaines difficultés aient émergé, notamment dans la gestion des pointeurs en C et le calcul des coordonnées dans le solver, celles-ci ont été surmontées grâce à une approche de débogage systématique et ont offert à chacun une solide expérience d'apprentissage.

À ce stade, le projet respecte les délais et le cahier des charges. La prochaine phase, qui s'annonce aussi exigeante que stimulante, consistera à parachever les éléments encore en développement — comme l'entraînement final du réseau de neurones et l'interface graphique — et à parfaire l'intégration de l'ensemble des modules. L'objectif est clair : construire une application homogène, robuste et pleinement fonctionnelle, capable de relever le défi de la reconnaissance et de la résolution de mots cachés sur des images réelles, et ainsi livrer un projet abouti pour la soutenance finale.

Annexes :

Le traitement d'image :

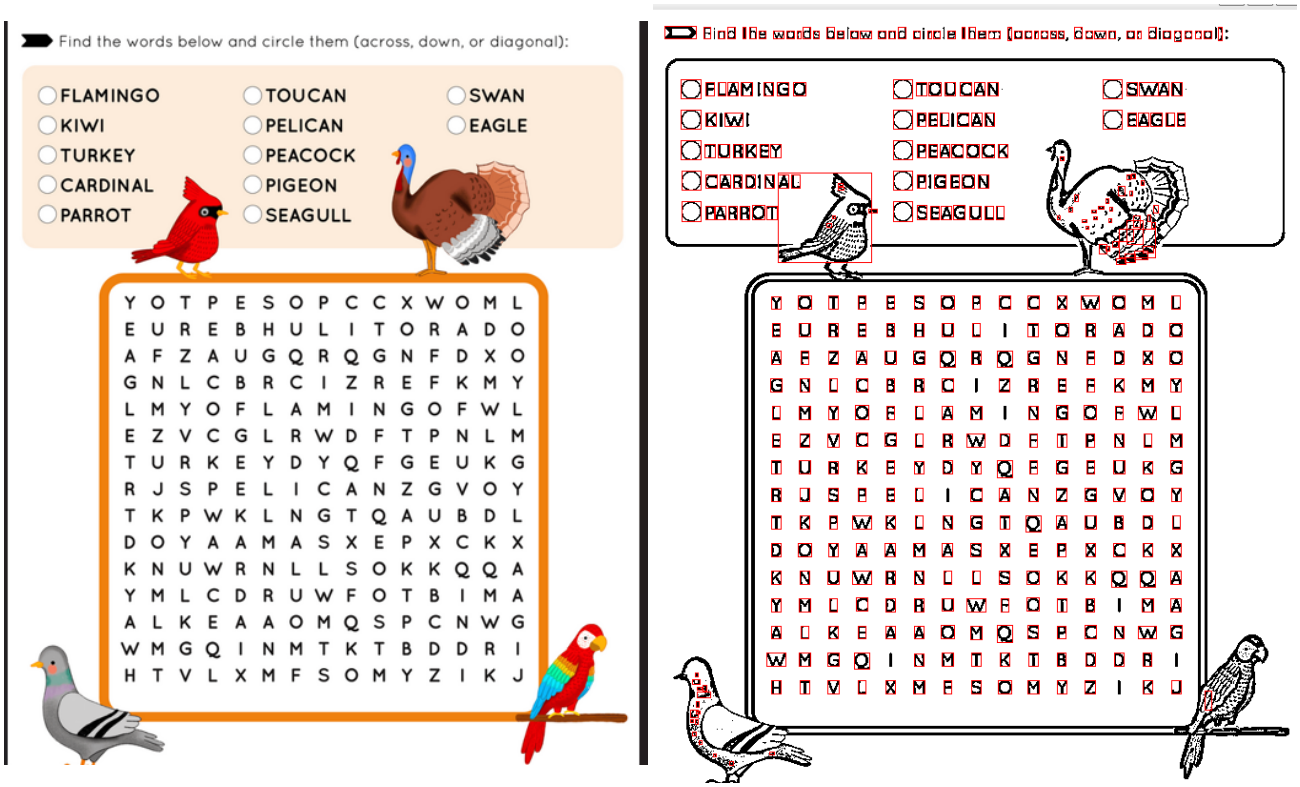
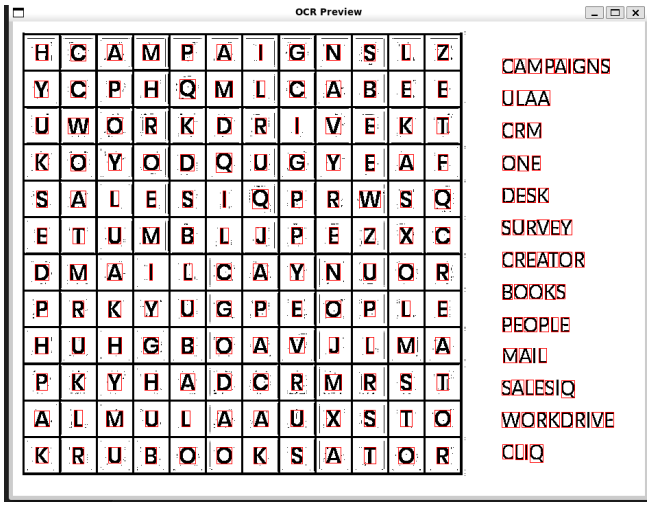


Image 1 : Resultat avec puis sans couleurs



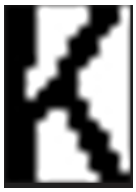


Image 2 : Resultat final avec lettre detoure