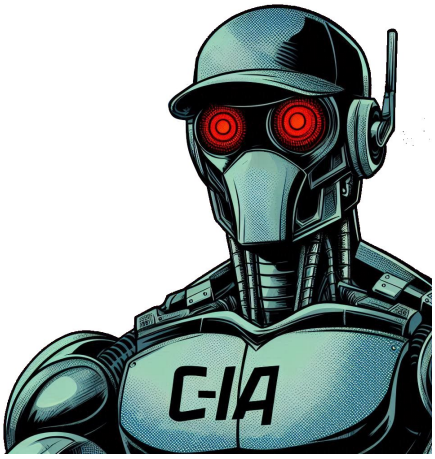
OCR Word Search Solver

Alexandre Divol
Théo Francois-Pichard
Andrew Youansamouth
Leopold Schneckenberger

Novembre 2024



Contents

1 Introduction

Bienvenue à la lecture de ce rapport sur le projet "OCR Word Search Solver". Ce document résume les efforts intensifs de notre équipe durant les deux derniers mois. À travers des heures d'acharnement et de recherche sur divers wikis et articles, nous avons cherché à comprendre des éléments techniques qui, aujourd'hui, nous paraissent plus simples mais qui ont été de véritables défis au départ. Ce rapport est destiné à détailler pas à pas notre développement et les solutions que nous avons apportées à chaque étape.

Notre projet a commencé par la création d'un solveur, un programme annexe crucial pour la résolution des grilles de mots. Ce solveur constitue la base de notre application, et sa conception a nécessité une compréhension approfondie des algorithmes de recherche et d'optimisation. Ensuite, nous nous sommes concentrés sur le réseau de neurones, un élément central de notre projet. Pendant un mois, nous avons exploré différentes architectures de réseaux, ajusté des paramètres et testé diverses approches pour obtenir les meilleurs résultats possibles en termes de reconnaissance de caractères.

En parallèle, deux membres de notre équipe ont travaillé sur un autre aspect fondamental : l'algorithme permettant de reconnaître une grille dans une image et de la découper correctement. Ce processus de découpage a été critique pour la précision du solveur, car il fallait s'assurer que chaque lettre soit correctement isolée et identifiée.

Enfin, nous avons développé le GUI (Graphical User Interface), une interface utilisateur ergonomique et intuitive qui simplifie l'utilisation du programme. Grâce à cette interface, même les utilisateurs non techniques peuvent facilement interagir avec notre application et résoudre des grilles de mots cachés en un clic.

Ce projet nous a également confrontés à de nombreux défis imprévus. De la gestion des erreurs dans le traitement d'image aux ajustements de l'algorithme de reconnaissance, chaque difficulté rencontrée nous a permis d'apprendre et de nous améliorer. Ce rapport vous offre un aperçu détaillé de ces défis et de la manière dont nous les avons surmontés.

Nous espérons que ce rapport reflète fidèlement le travail et la passion investis dans ce projet et qu'il servira de guide et d'inspiration pour de futurs développements dans le domaine de l'OCR et de la reconnaissance de grilles de mots.

2 Solveur de grille

Un programme en apparence simple, dont l'objectif est de retrouver des mots dans une grille de lettres. Cependant, un petit bémol : la grille est fournie sous forme de tableau. Il nous incombe donc d'identifier les mots à partir de ce tableau. Pour ce faire, nous nous sommes basés sur le pseudocode de l'article Wikipedia consacré au solving de mots. Cette méthode requiert de tester toutes les possibilités, c'est-à-dire d'examiner chaque cas de figure : lorsqu'une première lettre est identifiée, il faut alors rechercher dans les huit directions possibles (haut, bas, gauche, droite et les quatre diagonales) pour déterminer si le mot continue ou non.

```
[alexdieu@hirohito-laptop Solver]$ ./solver
solver: Usage: ./solver <Text File> <Word to find> <Debug/Optional>
```

Figure 1: Fonctionnement du programme

L'élaboration de ce programme a commencé par une analyse approfondie du pseudocode. Nous avons dû adapter cet algorithme théorique à notre propre contexte, en tenant compte des spécificités de notre grille et des contraintes imposées par notre application. La phase de debugging a été cruciale pour garantir le bon fonctionnement du solveur. En mode debugging, notre programme permet de suivre pas à pas le processus de recherche des mots, en mettant en évidence les lettres testées et les directions explorées. Voici une démonstration du programme en mode débogage :

```
[alexdieu@hirohito-laptop Solver]$ ./solver grid horizontal 1
Welcome to solver.c 1.5.2 made by Theo and Alex [DEBUG MODE]!
File to look : grid
Word to find : HORIZONTAL
Launching solver ...
Opening the file grid
Grid is a 9 x 10
Transforming the file in an array
Array succesfully created
Here is the array:
[[ H O R I Z O N T A L ]
[ D X R A H C L B G A ]
[ D I K C I L E O K C ]
[ I G A J H Y L Y H I ]
[ H G F G O D T I O T ]
[ G D L R O W K B F R ]
[ P L N R D N E R G E ]
[ J H A I D U A J G V ]
[ U K G F F O L L E H ] ]
Possible horizontal match on (0,0)
Possible horizontal match on (1,0)
Possible horizontal match on (2,0)
Possible horizontal match on (3,0)
Possible horizontal match on (4,0)
Possible horizontal match on (5,0)
Possible horizontal match on (6,0)
Possible horizontal match on (7,0)
Possible horizontal match on (8,0)
Possible horizontal match on (9,0)
It's a horizontal match ! on (0,0)(9,0)
Closing the file grid
Closed the file grid
(0,0)(9,0)
[alexdieu@hirohito-laptop Solver]$
```

Figure 2: Démonstration du programme en mode debugging

On peut voir ici que le débugging nous indique toutes les étapes effectués pour trouver le mot. Sur un exemple plus complexe :

```
[alexdieu@hirohito-laptop Solver]$ ./solver grid world 1
Welcome to solver.c 1.5.2 made by Theo and Alex [DEBUG MODE]!
File to look : grid
Word to find : WORLD
Launching solver ...
Opening the file grid
Grid is a 9 x 10
Transforming the file in an array
Array succesfully created
Here is the array:
[[  H  O  R  I  Z  O  N  T  A  L ]
[  D  X  R  A  H  C  L  B  G  A ]
[  D  I  K  C  I  L  E  O  K  C ]
[  I  G  A  J  H  Y  L  Y  H  I ]
[  H  G  F  G  O  D  T  I  O  T ]
[  G  D  L  R  O  W  K  B  F  R ]
[  P  L  N  R  D  N  E  R  G  E ]
[  J  H  A  I  D  U  A  J  G  V ]
[  U  K  G  F  F  O  L  L  E  H ] ]
Possible horizontal match on (5,5)
Possible bottom-left to top-right diagonal match on (5,5)
Possible bottom-right to top-left diagonal match on (5,5)
Possible bottom-right to top-left diagonal match on (4,4)
Possible backwards horizontal match on (5,5)
Possible backwards horizontal match on (4,5)
Possible backwards horizontal match on (3,5)
Possible backwards horizontal match on (2,5)
Possible backwards horizontal match on (1,5)
It's a backwards horizontal match! on (5,5)(1,5)
Closing the file grid
Closed the file grid
(5,5)(1,5)
[alexdieu@hirohito-laptop Solver]$
```

Figure 3: Démonstration du programme en mode debugging sur un exemple plus complexe

Une fois le debugging achevé et le programme ajusté, nous avons obtenu une version fonctionnelle et optimisée du solveur. La capture d'écran ci-dessous illustre le programme dans son état final, capable de retrouver avec précision les mots cachés dans la grille :

Pour transformer notre fichier texte en tableau, rien de plus simple : nous avons créé un tableau en itérant sur chaque caractère du fichier texte jusqu'au retour à la ligne, où nous passons à la ligne suivante du tableau. Il est évidemment crucial de vérifier que l'entrée est correcte, en s'assurant que chaque ligne contient le même nombre de caractères.

Ce projet, bien que simple dans son concept, a révélé de nombreux défis et a exigé une compréhension approfondie des algorithmes de recherche et d'optimisation.

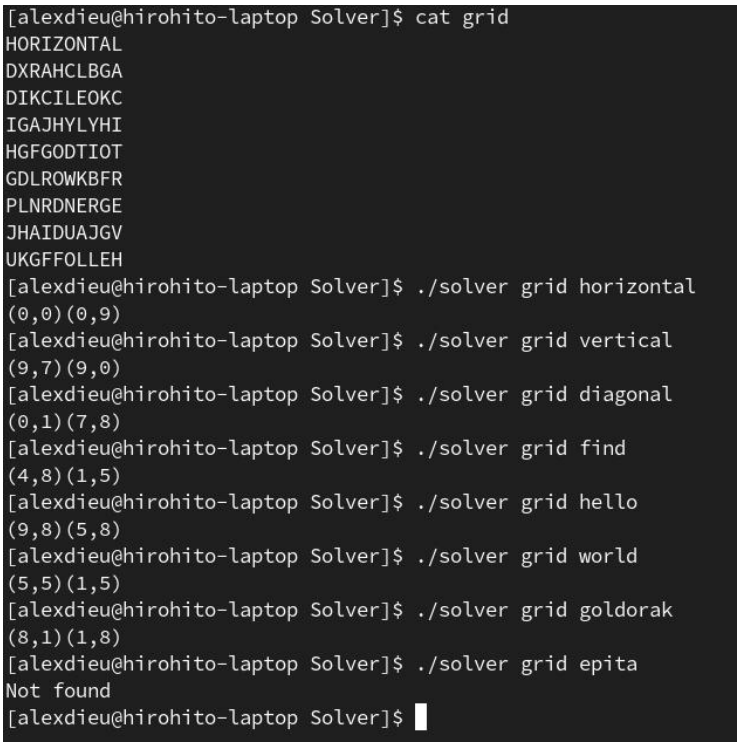


Figure 4: Programme fonctionnant correctement

Il a également mis en lumière l'importance d'une phase de debugging rigoureuse pour assurer la fiabilité et l'efficacité du programme. La satisfaction de voir notre solveur fonctionner correctement après plusieurs heures de travail acharné est une récompense en soi, démontrant la valeur de l'effort et de la persévérance dans le développement de solutions algorithmiques.

3 Réseau de Neurones

Nous entrons dans la partie la plus délicate du projet. Non pas parce qu'elle est particulièrement difficile en termes de codage, mais parce qu'elle est la plus complexe à comprendre. Simplifions tout cela. Saviez-vous que votre estomac abrite un réseau de neurones semblable à celui du cerveau d'un chien? Eh bien, oui! Lorsque votre estomac gargouille, ce n'est pas votre tête qui envoie l'ordre, mais bien un réseau de neurones dans votre système digestif. Certes, il ne contient "que" plusieurs centaines de millions de neurones, loin des milliards de notre cerveau, mais cela suffit amplement à réguler son fonctionnement.

Dans ce projet, nous allons imiter, de manière simplifiée, cette machine incroyablement complexe créée par la nature. Nous allons créer un réseau de neurones artificiel, lui fournir des entrées et lui permettre de générer plusieurs réponses possibles. Pour notre preuve de concept (POC), il devra être capable de comprendre des opérations logiques de base. Ensuite, nous développerons un réseau capable d'identifier les lettres.

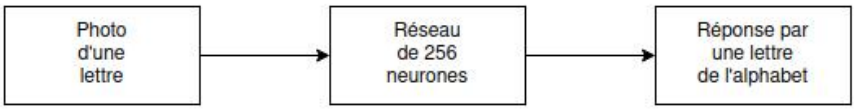


Figure 5: Diagramme représentant le fonctionnement de l'IA

3.1 Réalisation du PoC

Nous vous proposons un tour d'horizon complet de la réalisation d'un Proof of Concept (PoC) en intelligence artificielle. Que vous soyez novice ou expérimenté, vous trouverez ici toutes les étapes nécessaires pour maîtriser ce sujet passionnant.

Comprendre les Réseaux de Neurones

Un réseau de neurones est constitué de différentes couches de neurones, chaque couche ayant un rôle spécifique dans le traitement des données. On distingue principalement trois types de neurones : les neurones d'entrée, les neurones cachés et les neurones de sortie.

Les Neurones d'Entrée

Les neurones d'entrée sont la porte d'entrée des données dans le réseau. Ils sont initialisés avec les valeurs d'entrée que nous fournissons. Par exemple, si notre réseau comporte deux neurones d'entrée et que nos valeurs d'entrée sont 0 et 1, le premier neurone d'entrée sera initialisé à 0 et le second à 1. Ces neurones véhiculent les données brutes vers les couches suivantes du réseau.

Les Neurones Cachés

Les neurones cachés se situent entre les neurones d'entrée et les neurones de sortie. Leur rôle est de traiter les données transmises par les neurones d'entrée.

Ils ajustent leurs poids (valeurs influençant leur comportement) au cours de l'apprentissage pour améliorer la précision des réponses du réseau. Ces neurones sont essentiels pour l'apprentissage des patterns complexes présents dans les données.

Les Neurones de Sortie

Les neurones de sortie constituent la dernière couche du réseau. Ils fournissent les résultats finaux. Lors des premiers entraînements, l'IA génère des réponses aléatoires pour apprendre des feedbacks qu'elle reçoit. Après plusieurs tentatives, le réseau ajuste ses poids, détecte des patterns et améliore ses réponses.

Processus d'Entraînement

L'entraînement du réseau de neurones est une étape cruciale. Nous mesurons la fiabilité du réseau grâce au pourcentage d'erreur à chaque itération du processus d'entraînement. Comme montré dans la figure ci-dessous, nous observons une réduction progressive de l'erreur au fur et à mesure que le réseau apprend.

4 Processus d'Entraînement

```
[alexdieu@hirohito-laptop Neural Network]$ ./ocr_solver
Epoch 1/30
End of Epoch 1 - Average Loss: 0.018351
Epoch 2/30
End of Epoch 2 - Average Loss: 0.018131
Epoch 3/30
End of Epoch 3 - Average Loss: 0.018816
Epoch 4/30
End of Epoch 4 - Average Loss: 0.018131
```

Figure 6: Pourcentage d'erreur à chaque itération du processus d'entraînement.

Facteurs Déterminants la Qualité du Réseau

La qualité finale de notre réseau de neurones dépend de trois facteurs principaux :

La qualité du dataset (ensemble de données) : Un dataset pertinent et bien annoté est essentiel pour un apprentissage efficace. Ce sont nos valeurs permettant d'entrainer notre réseau. Par exemple, pour l'OCR, on retrouve de multiples images de A, B, C, etc...

La taille du réseau : Un réseau plus grand avec plus de neurones et de couches peut offrir de meilleures performances, mais il est aussi plus coûteux en termes de calculs et de ressources.

Le temps d'entraînement: Plus le réseau s'entraîne longtemps, mieux il ajuste ses poids et plus il réduit les erreurs. Un bon compromis doit être trouvé pour éviter le surapprentissage où le modèle devient trop spécifique aux données d'entraînement et perd sa généralité.

Résultats Finaux

Pour notre PoC, nous avons réalisé 10 000 itérations d'entraînement avec un réseau de 5 neurones. Cette configuration est suffisante pour traiter des logiques simples impliquant des 0 et des 1. Voici l'output final de notre PoC :

```
[alexdieu@hirohito-laptop Neural Network (Proof of Concept)]$ ./reseau_neural
Entraînement ET...
Test ET...
Entrée : (0, 0) - Sortie prédite : 0.000125 - Sortie attendue : 0
Entrée : (0, 1) - Sortie prédite : 0.002643 - Sortie attendue : 0
Entrée : (1, 0) - Sortie prédite : 0.002640 - Sortie attendue : 0
Entrée : (1, 1) - Sortie prédite : 0.996209 - Sortie attendue : 1
Entraînement OU...
Test OU...
Entrée : (0, 0) - Sortie prédite : 0.003740 - Sortie attendue : 0
Entrée : (0, 1) - Sortie prédite : 0.997511 - Sortie attendue : 1
Entrée : (1, 0) - Sortie prédite : 0.997464 - Sortie attendue : 1
Entrée : (1, 1) - Sortie prédite : 0.999426 - Sortie attendue : 1
Entraînement NOT_A_AND_NOT_B_OU_A_ET_B...
Test NOT_A_AND_NOT_B_OU_A_ET_B...
Entrée : (0, 0) - Sortie prédite : 0.004624 - Sortie attendue : 0
Entrée : (0, 1) - Sortie prédite : 0.994778 - Sortie attendue : 1
Entrée : (1, 0) - Sortie prédite : 0.994779 - Sortie attendue : 1
Entrée : (1, 1) - Sortie prédite : 0.004849 - Sortie attendue : 0
```

Figure 7: Output du PoC après 10 000 itérations avec un réseau de 5 neurones.

Le réseau de neurones se révèle extrêmement précis dans ses prédictions, approximant avec une grande justesse les valeurs attendues. En effet, pour une entrée correspondant à 0, le réseau prédit une valeur de 0,0002, ce qui est très proche de la valeur correcte. De même, pour une entrée de 1, la prédiction du réseau atteint 0,99, démontrant ainsi sa capacité à s'ajuster et à fournir des réponses presque exactes. Ces résultats illustrent la performance remarquable du réseau dans l'apprentissage et la reconnaissance des patterns, validant son efficacité après l'entraînement.

Après plusieurs jours d'entraînement intensif, il apparaît clairement que l'IA OCR n'a pas encore atteint un niveau de performance optimal. Bien que des progrès aient été réalisés, certains défis restent à surmonter pour atteindre les résultats escomptés.

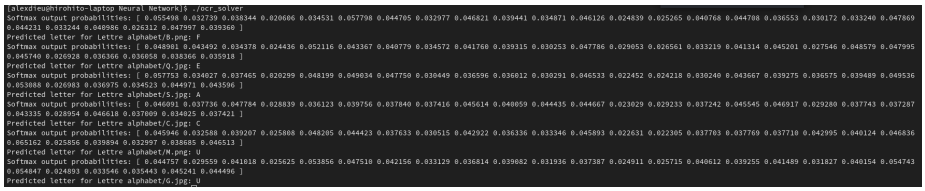


Figure 8: Test du réseau de neurones après 3 jours d'entrainement

On peut examiner les probabilités des 26 lettres dans les 26 neurones de sortie, et constater que l'IA n'a réussi qu'un seul test. Cela peut s'expliquer par la qualité insuffisante du dataset utilisé pour l'entraînement.

5 Avancement du GUI

L'avancement de l'interface graphique (GUI) n'a pas été notre priorité pour cette soutenance. Nous avons concentré nos efforts sur le bon fonctionnement de chaque composant pris individuellement. Nous avons cependant réalisé un prototype ALPHA non-fonctionnel de l'interface, utilisant GTK et SDL, pour illustrer à quoi elle pourrait ressembler.

Figure 9: Test du réseau de neurones après 3 jours d'entrainement

6 Découpage des images

Le découpage des images a représenté la partie la plus complexe et technique du projet. Pour aborder ce défi, nous avons développé un algorithme relativement simple mais ingénieux. Voici les étapes détaillées de notre processus :

Tout d'abord, nous commençons par convertir l'image entière en noir et blanc, en appliquant un seuil de luminosité. Ce seuil est ajustable, ce qui nous permet de moduler les résultats obtenus en fonction de la qualité de l'image initiale. Ensuite, nous procédons à une rotation de l'image pour la redresser et la mettre à l'horizontale, garantissant ainsi une base de travail correcte pour le découpage.

L'algorithme de découpage suit les étapes suivantes : il commence par détecter toutes les masses de points noirs présentes sur l'image. Si ces masses sont trop grandes ou trop petites, elles sont ignorées, car elles ne correspondent pas aux critères d'intérêt. Une fois ces masses pertinentes identifiées, nous procédons à la détection des rectangles formant la grille.

La detection de ces lettres repose sur des principes clés. Nous les identifions grâce à leur espacement régulier et à leurs dimensions similaires en termes de longueur et de largeur. Cette uniformité nous permet de déterminer avec précision quels rectangles appartiennent à la grille. Les éléments qui ne respectent pas ces critères sont alors considérés comme des textes de réponse.

Ce processus, bien qu'il puisse paraître simple, se révèle extrêmement efficace. Il permet une analyse rigoureuse et précise, garantissant que chaque lettre et chaque caractère sont correctement détectés et traités.

Nous vous présentons ici deux exemples des résultats obtenus grâce à notre algorithme. Dans ces exemples, il est évident que toutes les lettres ont été

détectées avec une grande précision, démontrant l'efficacité et la fiabilité de notre méthode :

Figure 10: Lettres detectés par l'algorithme

Figure 11: Lettres detectés par l'algorithme

Ensuite, le processus de découpage s'effectue en fonction de la catégorisation des éléments, selon qu'ils appartiennent à une réponse ou qu'ils font partie du tableau :

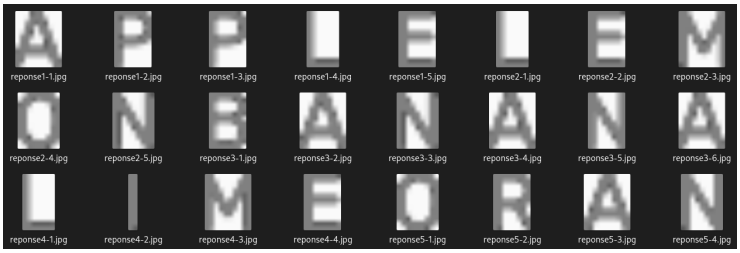


Figure 12: Résultat du découpage

Nous pouvons observer plus en détail les résultats obtenus grâce à la grille reconstruite à l'aide de SDL :

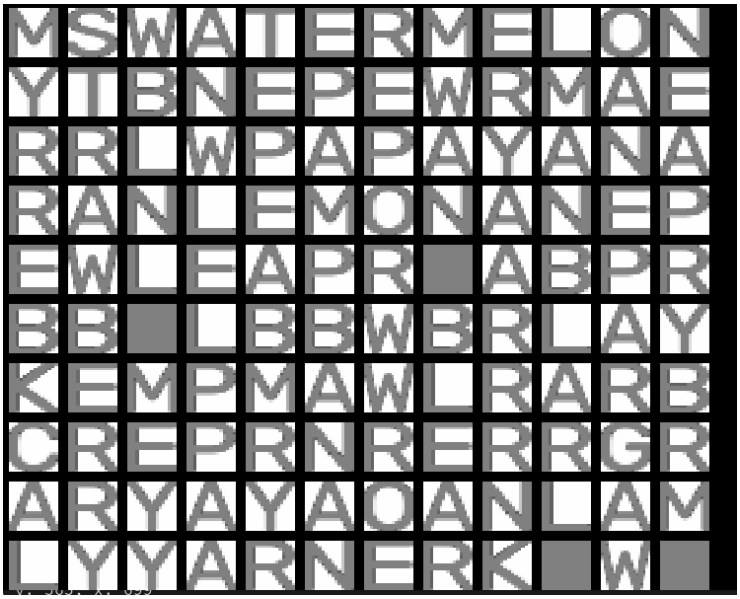


Figure 13: Grille reconstruite avec SDL

Ainsi que les réponses extraites :

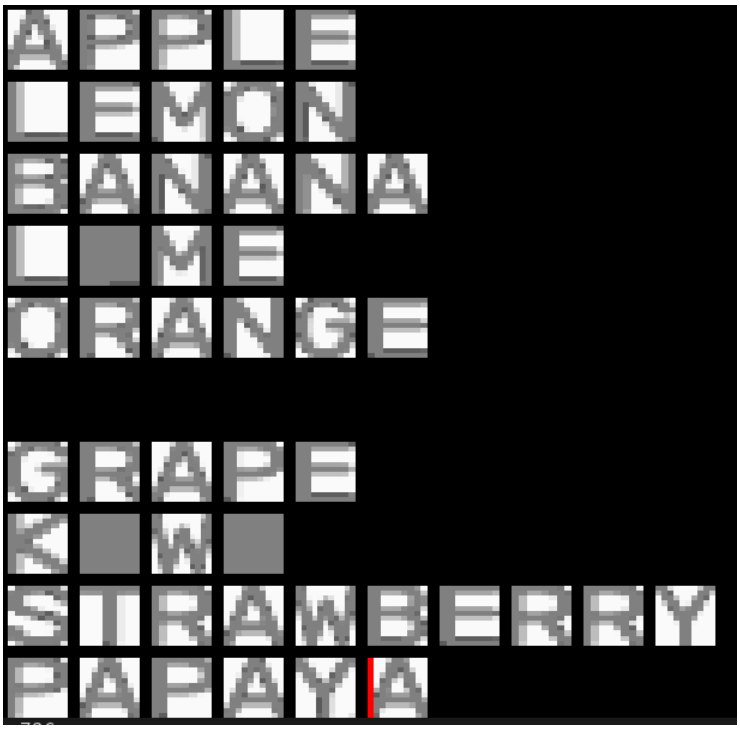


Figure 14: Réponses extraites par l'algorithme

Dans notre plan de dévolopper des outils de manipulation d'images, nous avons également développé un programme complémentaire intitulé rotate. Ce programme offre la possibilité de faire pivoter manuellement une image en spécifiant le degré de rotation souhaité.

Le programme rotate utilise les bibliothèques SDL et SDL-image pour charger différents formats d'images, y compris BMP, PNG et JPG. Voici un aperçu de son fonctionnement :

- Chargement de l'image : L'image à manipuler est chargée à partir d'un fichier spécifié par l'utilisateur.

- Calcul de la rotation : Grâce à un algorithme basé sur la trigonométrie, le programme calcule les nouvelles dimensions et les positions des pixels après rotation.

- Enregistrement de l'image : L'image pivotée est ensuite enregistrée sous un nouveau fichier, permettant à l'utilisateur de comparer facilement l'original et la version modifiée.

Le programme rotate est un outil pratique pour tous ceux qui ont besoin de manipuler les images de manière précise et flexible. Avec ce programme, la rotation des images devient simple et efficace.

Ici l'algorithme utilisé :

```
int nx = (x - cx) * cos(radians) - (y - cy) * sin(radians) + ncx;
int ny = (x - cx) * sin(radians) + (y - cy) * cos(radians) + ncy;
```

L'algorithme de rotation d'image repose sur des transformations trigonométriques pour recalculer la position des pixels. Voici une décomposition détaillée du processus:

Centre de l'Image:

- cx, cy : Coordonnées du centre de l'image originale.
- ncx, ncy : Coordonnées du centre de l'image après rotation.

Coordonnées Relatives :

- (xcx)(x - cx) et (ycy)(y - cy) sont les coordonnées du pixel par rapport au centre de l'image. Cela recentre le calcul autour du point pivot.

Imaginez que chaque pixel de l'image est attaché à une ligne qui part du centre de l'image. Lors de la rotation, chaque ligne pivote de l'angle spécifié, déplaçant ainsi le pixel à sa nouvelle position. C'est ce déplacement qui est calculé par les formules de rotation.

Ce processus est appliqué à chaque pixel de l'image pour obtenir l'image finale, pivotée selon l'angle désiré, tout en conservant la structure de l'image originale.

7 Problèmes rencontrés au cours du projet

Au cours de ce projet, nous avons rencontré plusieurs défis, notamment en ce qui concerne l'entraînement de l'IA OCR. Après quatre jours de formation intensive, il est apparu que l'IA n'avait pas atteint le niveau de performance escompté. Ce résultat décevant peut être attribué principalement à la qualité du dataset utilisé.

Le dataset employé pour l'entraînement de l'IA n'était pas parfaitement adapté à nos besoins. La diversité et la représentativité des données étaient insuffisantes, ce qui a entravé la capacité de l'algorithme à généraliser efficacement. De plus, certaines images du dataset présentaient des défauts de qualité, tels que des résolutions inappropriées ou des artefacts visuels, compliquant davantage le processus d'apprentissage.

En outre, nous avons également rencontré des problèmes liés à l'Address Sanitizer et aux erreurs de segmentation (segfaults). L'utilisation d'Address Sanitizer nous a permis d'identifier plusieurs failles de mémoire dans notre code, mais ces détections ont parfois retardé notre progression en raison du temps nécessaire pour localiser et corriger les erreurs. Les segfaults, en particulier, ont été une source fréquente de frustration, nécessitant une attention minutieuse pour tracer et éliminer les causes sous-jacentes de ces erreurs.

Ces difficultés ont mis en évidence l'importance cruciale de sélectionner un dataset de haute qualité et de bien le préparer avant de lancer l'entraînement de l'IA. Une réflexion approfondie et des ajustements sur les paramètres du modèle et sur le dataset lui-même seront nécessaires pour améliorer les performances de l'IA OCR. De même, des efforts constants pour garantir la robustesse du code

à travers des outils tels qu'Address Sanitizer et des techniques de débogage efficaces restent essentiels pour éviter les erreurs de segmentation.

En conclusion, bien que le projet ait progressé de manière significative, il est évident que des améliorations sont nécessaires, notamment en ce qui concerne la qualité du dataset, les paramètres d'entraînement et la robustesse du code, afin de renforcer les capacités et la précision de notre IA OCR. Nous espérons atteindre nos objectifs pour la prochaine soutenance !

8 Ressenti du groupe

Andrew (chef)

En tant que chef de projet, cette expérience a été extrêmement enrichissante et stimulante. La coordination des différentes phases du projet et la gestion de l'équipe ont représenté un défi important, mais cela m'a permis d'améliorer mes compétences en leadership et en organisation. Observer l'évolution du projet et voir notre équipe surmonter les obstacles ensemble a été une source de grande satisfaction. Je suis particulièrement fier de notre capacité à rester soudés et à travailler de manière collaborative pour atteindre nos objectifs.

Theo (responsable IA)

Travailler sur l'intelligence artificielle a été à la fois passionnant et complexe. Les défis liés à l'entraînement de notre modèle OCR et les ajustements constants nécessaires pour améliorer les résultats ont été très formateurs. Malgré les difficultés rencontrées, je suis fier des progrès réalisés et de notre persévérance. Collaborer avec une équipe dédiée et trouver des solutions ensemble a renforcé ma passion pour l'IA et m'a donné de précieuses leçons pour les projets futurs.

Alex (responsable algorithmes)

En tant que responsable des algorithmes, j'ai eu l'opportunité de concevoir et d'implémenter des solutions techniques innovantes pour notre projet. Le développement de l'algorithme de découpage d'image a été particulièrement captivant et m'a permis d'approfondir mes connaissances en traitement d'image. Chaque étape de test et d'ajustement, bien que parfois frustrante, a été incroyablement gratifiante. Voir nos algorithmes fonctionner correctement après de nombreux essais a été une immense satisfaction.

Leopold (responsable interface)

La conception de l'interface utilisateur a été un défi stimulant qui m'a permis d'explorer différentes approches pour rendre notre application à la fois intuitive et esthétiquement agréable. Travailler avec SDL et GTK pour créer un prototype fonctionnel m'a offert de nombreuses occasions d'apprentissage et de développement de compétences. Je suis très satisfait du résultat final et de

la manière dont notre équipe a pu collaborer efficacement pour intégrer tous les composants. Ce projet a été une expérience précieuse qui a renforcé mes compétences en développement d'interfaces et en travail d'équipe.