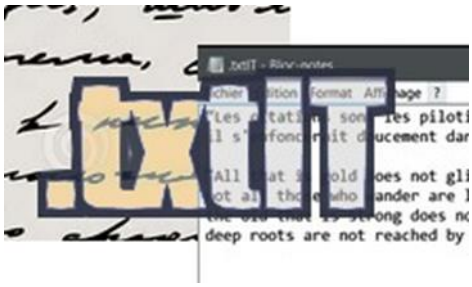
Juliette Malassé
Sacha Ntolo-Mvelle
Quentin Pagneux
Élodie Pouliquen
Louise Geny



RAPPORT PROJET TRANVERSE

.txtIT

Équipe « Chat » :
Juliette Malassé
Sacha Ntolo-Mvelle
Quentin Pagneux
Élodie Pouliquen
Louise Geny

EFREI 2021

Juliette Malassé
Sacha Ntolo-Mvelle
Quentin Pagneux
Élodie Pouliquen
Louise Geny

Introduction, concept et mise en œuvre

Le concept de .txtIT est né après avoir essayé notre texte de test sur des OCR intégrés à des sites web trouvés en ligne. Sur les trois que nous avons testé, aucun n'est arrivé à ressortir un texte cohérent avec l'image entrée, et ils en étaient tous très loin. De là est née notre idée de faire non seulement un OCR, mais d'essayer de l'intégrer à un site web et de le rendre adaptable. Pour ce faire, nous avons eu l'idée que les utilisateurs créeraient un compte et prendraient en photo un alphabet écrit par leurs soins, afin que notre réseau de neurones puisse être entraîné de manière personnalisée, propre à chaque client. En effet, la difficulté avec les OCR, surtout pour de la reconnaissance manuscrite, est que chaque personne a une écriture différente et il est très compliqué d'entraîner un réseau de façon efficace pour qu'il reconnaisse le texte à chaque coup.

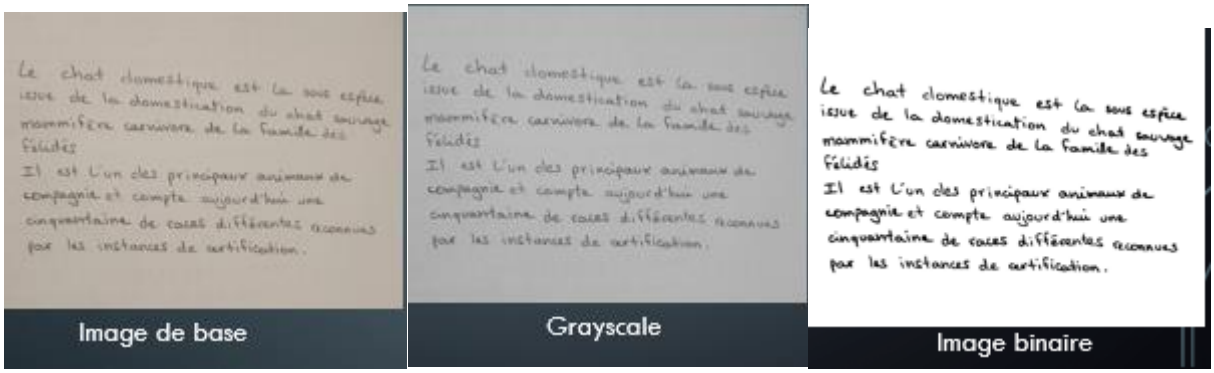
Nous avons donc décider d'utiliser le Python pour réaliser cet OCR, car c'est un langage polyvalent, qui nous permettrait de réaliser aussi bien le traitement d'image que le réseau de façon efficace, et qui dispose d'un documentation très étoffée dû à sa popularité. Il est également très accessible et cela était un plus pour certains membres de notre équipe qui ne sont pas très à l'aise, ni avec le fait d'apprendre un nouveau langage, ni avec la programmation en général. Le Python nous a donc semblé un bon compromis. Nous avons travaillé sur Spyder, IDLE Python et sur OpenCV vers la fin du projet. Pour mettre en commun notre travail et stocker notre code, nous avons utilisé GitHub.

Pour concevoir cet OCR, nous avons découpé le travail en trois parties : le pré traitement de l'image, c'est-à-dire tout ce qui concerne les niveaux de gris, la binarisation, la rotation de l'image, la suppression des lignes, des ratures... Ensuite, nous avons isolé la segmentation de l'image, donc tout ce qui est détection des blocs de texte, des lignes de texte, des caractères, mais également différentiation des images du texte etc. Enfin, nous avons mis le réseau de neurones à part, qui reçoit les caractères sous forme de matrice après segmentation et traitement de l'image et avec des formules mathématiques et des poids, permet de retrouver le caractère qu'on lui a donné et de le ressortir. Les caractères reconnus sont alors ajoutés à un fichier .txt.

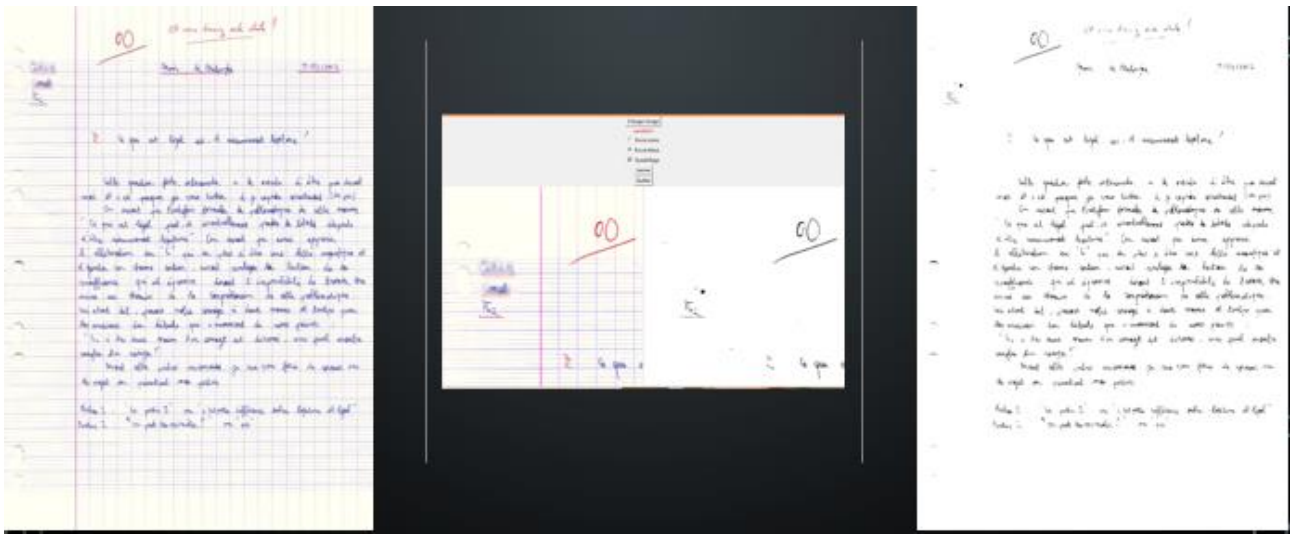
Juliette Malassé
Sacha Ntolo-Mvelle
Quentin Pagneux
Élodie Pouliquen
Louise Geny

Le pré-traitement de l'image et l'interface graphique

Avant de détecter les caractères pour pouvoir les passer au réseau de neurones, il a fallu prétraiter les images que nous avons utilisées. On a donc dû les passer en Grayscale, en niveaux de gris, en générant une valeur en utilisant des poids sur chaque composante (r,g,b) des pixels de l'image et ensuite en mettant tous les composants à cette nouvelle valeur. Nous avons dû également binariser l'image, ce qui consiste à fixer un seuil, et à ensuite comparer les pixels gris à cette valeur et à les assigner comme blancs ou noirs selon s'ils sont supérieurs ou inférieurs à ce seuil.



Pour le pré-traitement de l'image, nous avons aussi mis au point un algorithme qui supprime les lignes en arrière-plan d'une image. Il a été développé sous OpenCV et possède une interface. L'algorithme détecte les pixels nois alignés sur une image binarisée et les supprime. Le seul souci est que certains morceaux du texte se retrouvent également supprimés.



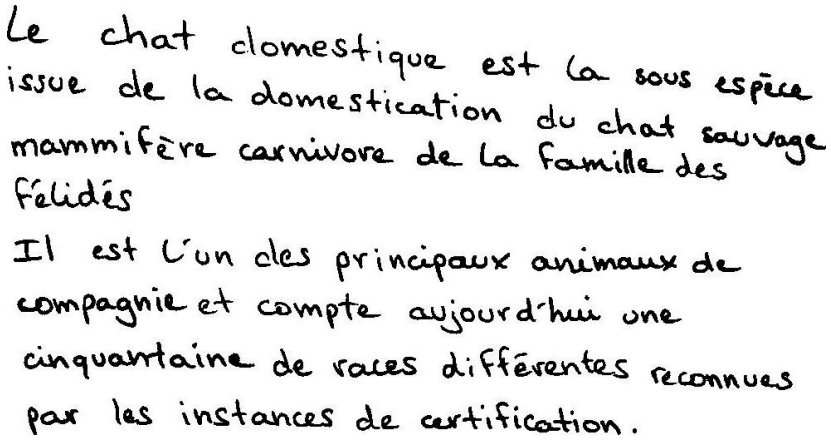
Juliette Malassé
Sacha Ntolo-Mvelle
Quentin Pagneux
Élodie Pouliquen
Louise Geny

La segmentation

Nous avons voulu faire la segmentation de notre image à partir de rien, car sur internet les méthodes décrites étaient très complexes et nous n'avons pas pensé à utiliser OpenCV par exemple, pour nous faciliter la tâche. Nous avons décidé de développer une segmentation « maison ». Elle est basée sur la reconnaissance de si un pixel est noir ou blanc et en fonction de sa position, on peut en déduire l'emplacement d'un bloc de texte par exemple.

Cette méthode a donc bien marché pour la détection des blocs, mais beaucoup moins pour la détection des caractères. En effet, nous nous sommes cette fois-ci basés sur la distance moyenne entre deux lettres et entre deux mots, mais le résultat n'a pas été concluant, car même si quelques caractères étaient bien détectés, l'algorithme en détectait souvent plusieurs en même temps ou détectait le chevauchement de deux caractères.

Le problème principal est que nous avons essayé de développer la segmentation pour un texte manuscrit et qui serait de plus écrit en écriture cursive et attachée à la fois, avec des majuscules et de la ponctuation, ce qui était beaucoup trop ambitieux.



Exemples de caractères bien détectés :

Juliette Malassé
Sacha Ntolo-Mvelle
Quentin Pagneux
Élodie Pouliquen
Louise Geny

Réseau de neurones

Il a deux sortes de composants : des neurones et des synapses, ces derniers sont répartis en plusieurs couches :
Une couche d'entrée
Une couche de sortie
Une ou plusieurs couches cachées

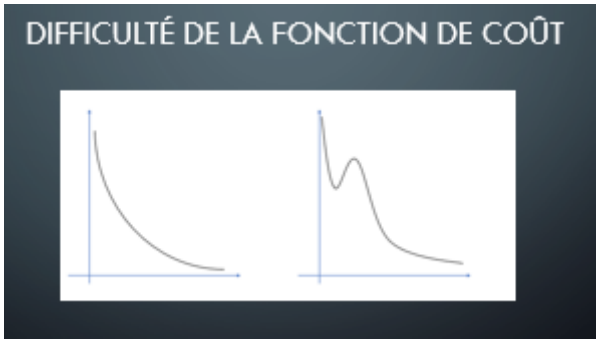
Les neurones appliquent une fonction sigmoïde aux valeurs qu'ils reçoivent pour pouvoir les normaliser
Les synapses quant à elles appliquent un poids à la valeur qu'elles reçoivent, le poids est une valeur très importante car c'est la valeur qui changera les résultats que renverra le réseau de neurones

Pour renvoyer un résultat, le réseau de neurones fait une propagation avant : chaque neurone d'une couche envoie une valeur à tous les neurones de la couche suivante qui sera modifiée par chaque synapse. Après avoir atteint la sortie, il renverra la valeur qu'il aura calculé.
Le souci est que passer uniquement par la propagation avant ne donne aucun résultat intéressant : le réseau n'ayant pas été entraîné, les poids ont été répartis de façon aléatoire et le résultat l'est donc aussi.

Pour régler cela, il faudrait pouvoir savoir le taux d'erreur et le corriger, ce qui est possible avec la rétropropagation, qui consiste à refaire le réseau de neurones en arrière pour pouvoir réajuster les poids et s'approcher du résultat final

Pour la propagation avant, les synapses reviennent simplement à faire des multiplications matricielles, avec les matrices de poids, et les neurones font passer une fonction sigmoïde ($f(x) = 1/1 + \exp(-x)$)

Pour calculer la rétropropagation il suffit de faire la dérivée du coût par rapport au poids pour pouvoir avoir un taux d'erreur lié au poids. Il suffit ensuite de soustraire ce nombre au poids pour réduire le coût et se rapprocher du résultat final



Une des difficultés rencontrées avec le réseau ont été liées au fait de devoir sauvegarder et charger les données, trouver un format adapté à la sauvegarde des floats n'a pas été facile et beaucoup de soucis, en particulier au niveau du chargement ont été rencontrés.

Juliette Malassé
Sacha Ntolo-Mvelle
Quentin Pagneux
Élodie Pouliquen
Louise Geny

Le site web

Nous avons décidé réaliser en plus de l'OCR un site web. Mais vous allez vous demander pourquoi rajouter une tâche difficile en plus de notre projet initial. C'est tout simplement parce que nous trouvons que c'est le moyen le plus viable économiquement parlant pour notre application. En effet nous pouvons facilement imaginer un abonnement ou un paiement au coup par coup pour l'utilisation de notre logiciel.

Mais un problème se pose dès le début : l'utilisation du Python dans un site web.
Nous disposons alors de 2 solutions évidentes :
-Recoder l'intégralité de notre programme en JavaScript
-Utiliser une implémentation du python dans un site web (Brython)

Recoder notre projet n'était pas quelque chose d'envisageable car nous ne connaissons pas le JavaScript et l'apprentissage d'un nouveau langage pour réaliser ce genre de tache aurait pris énormément de temps ce qui rend cette solution inenvisageable

Il nous reste donc que l'utilisation du Bryhton, une implémentation du python pour site web. Mais malgré plusieurs heures de travail nous n'avons pas réussi à utiliser cette implémentation, ce qui a stoppé l'avancée du site.

Les problèmes rencontrés

Nous avons rencontré de nombreux problèmes lors de ce projet, le problème majeur fut que nous avions une communication et une gestion du temps imparfaite, du fait que tous les membres n'aient pas à leur disposition des outils de communication excessivement utiles comme Discord ou Messenger.
De plus avec les révisions des autres DE et TAI nous avons eu du mal à nous organiser correctement pour bien avancer.

Ce projet fut trop ambitieux pour notre niveau. En effet, faire un OCR fonctionnel et un site web optimisé avec un design agréable en même temps fut une grande erreur, car a force de vouloir bien faire nous n'avons rien réussi à finir à 100% et au lieu d'utiliser des programmes connus nous avons essayé de recréer ces programmes à notre manière, ce qui nous a fait perdre aussi énormément de temps. Nous aurions dû utiliser OpenCV pour toute la partie traitement d’image, et mettre plus de personnes pour travailler sur le réseau de neurones.

Juliette Malassé
Sacha Ntolo-Mvelle
Quentin Pagneux
Élodie Pouliquen
Louise Geny

L’avenir du projet

Pour l'avenir de ce projet nous allons essayer de développer une version bien plus légère et optimisé dans un langage compatible a la programmation web comme le JavaScript par exemple. Nous allons également continuer le développement de l’application en Python pour avoir un OCR puissant et fiable, qui pourra également être commercialisé, et toujours adapté à l’écriture de l’utilisateur.

On pourra donc améliorer le projet avec des algorithmes plus élaborés et complexes afin de de rendre notre programme bien plus performant et polyvalent tel que les algorithmes de pré-traitement, de rotation ou bien de suppression d'images ainsi que l'utilisation d'un dictionnaire pour reconnaître un mot éventuellement mal reconnu pour un fonctionnement optimal de l’application. Toute la segmentation sera à refaire également, pour qu’elle soit plus efficace et moins lourde et complexe au niveau du code.

Notre volonté est aussi de rendre l'application actuelle beaucoup plus « User-Friendly » grâce à une interface, pour que notre application soit utilisable par tous sans contrainte, en mettant par exemple l’image à traiter à droite, les boutons pour les différentes étapes du traitement au milieu, et la sortie en fichier texte à droite.

Conclusion

Pour conclure, ce projet a été extrêmement enrichissant pour tous les membres du groupe, et même s'il n'est pas encore abouti, cela nous a justement motivés à le continuer que ce soit en L3 ou bien simplement comme un projet personnel. Nous avons pu en effet découvrir grâce à cet OCR un nouveau langage : le Python, ainsi que de nouveaux Framework, des nouvelles méthodes de travail et de nouvelles notions, car un réseau de neurones est basé sur des concepts mathématiques complexes et ouvre une porte sur l'IA, que personne dans notre groupe n'avait encore expérimenté.

Nous en avons également appris beaucoup sur le traitement d'image, les différents masques, comment modifier une image etc.

Pour ce qui est du travail en groupe, chacun a pu identifier ses forces et ses faiblesses, et ce même si cela nous a pris un peu de temps, nous avons pu nous organiser même si certains membres n'ont pas pu être très assidus et ainsi fournir un travail plus poussé.

Finalement, nous sommes tout de même satisfaits d'avoir travaillé sur ce projet car c'est un sujet passionnant et nous sommes motivés pour le continuer à l'avenir.