RAPPORT DE SOUTENANCE FINALE

OCR

4puters can read

Hamza GAIZI
Martin LACAILLE
Bastien ROZEN
Loic SCHUSSLER

Promo 2024

Table des matières

1 Rappels de la premiere soutenance

1.1 Introduction

Le projet présenté dans ce document consiste en la création d'un logiciel de Reconnaissance Optique de Caractères (OCR), qui a pour fonction de prendre une image et d'en extraire le texte qu'elle contient.

Ce projet a été fait dans le cadre du projet de S3 de programmation de $2me$ année de l'EPITA.

Ce document a pour but de faire un état des lieux pour la première soutenance qui a lieu pendant la semaine du 26 octobre 2020.

L'équipe 4puter can read vous souhaite une bonne lecture du rapport et espère que vous avez apprécié notre (courte) présentation.

1.1.1 Historique

La première machine utilisant un logiciel de reconnaissance optique des caractères fut créée en 1929 par un ingénieur allemand du nom de Gustav Tauschek. Elle contenait un détecteur photosensible qui pointait une lumière sur un mot lorsqu'il correspondait à un gabarit contenu dans sa memoire.
En 1950, un cryptanalyste américain commença à travailler sur les procédures d'automatisation des données. Le problème principal se posait dans la conversion de messages imprimés en langage machine, de telle sorte que la machine puisse faire un traitement informatique du message.

La reconnaissance optique de caractères a évolué et s'est devéloppée en même temps que l'informatique générale.

Les premiers systèmes d'OCR apparurent dans le monde au sein d'entreprises privées, servant à automatiser les tâches de bureau ou encore gérer des données commerciales et les convertir en cartes perforées.

Au cours des années 1950 a été construite une machine utilisant la meilleure correspondance. Cela détecte un caractère par comparaison avec tous les caractères d'un alphabet.

Durant les décennies qui ont suivies, le système ROC s'est démocratisé et a commencé à être utilisé dans la plupart des grosses entreprises.

Au debut des années 1990, les scanners portables sont devenus très populaires. Les logiciels de reconnaissance optique des caractères sont alors devenus norme. Ils sont aujourd'hui beaucoup plus puissants qu'à leur création. Nous les utilisons quotidiennement sur nos smartphones et sur le web, pour traduire une carte de menu dans un restaurant étranger par exemple.

1.1.2 Principe d'un OCR

Un système OCR part de l'image numérique réalisée par un scanner optique d'une page (document imprimé, feuillet dactylographié, etc.), ou un appareil photo numérique, et produit en sortie un fichier texte en divers formats.

Certains logiciels tentent de conserver l'enrichissement du texte (corps, graisse et police) ainsi que la mise en page, voire de rebâtir les tableaux et d'en extraire les images.

Certains logiciels comportent, en outre, une interface pour l'acquisition numérique de l'image.

Le processus effectué par le logiciel peut se découper en plusieurs étapes :

— Pré-traitement de l'image : amélioration de la qualité de l'image en la redressant, la débruitant, en ajustant son contraste et en la passant en noir et blanc.

— Segmentation en lignes et caractères : isolation des lignes et des caractères dans chaque ligne, détection de texte souligné en gras...

— Reconnaissance des caractères : normalisation des caractères, comparaison d'une bibliothèque de formes connues et des réponses de l'algorithme.

— Post-traitement : réduction du nombre d'erreurs de reconnaisance en utilisant des méthodes linguistiques et contextuelles.

— Reconstruction du texte : génération du format de sortie ave une mise en page éventuelle

La partie traitant de la reconnaissance des caractères nécessitera l'utilisation d'un réseau de neurones qui identifiera les caractères et apprendra de ses erreurs pour devenir plus perfomant.

1.1.3 Répartition des tâches

Tâches	Hamza	Martin	Bastien	Loic
Chargement des images et suppression des couleurs		X		
Pré-traitement et renforcement des contrastes	X			
Détection et découpage en blocs, lignes et caractères		X		
Définition abstraite des classes utilisées dans le réseau	X	X	X	X
POC d'un neurone XOR	X			X
Jeu d'image pour l'apprentissage				X
Chargement et sauvegarde des poids du réseau de neurones			X	
Manipulation de fichiers pour la sauvegarde des résultats			X	X
Débruitage des images par convolution moyenne			X	X

Remarque : Certaines tâches qui sont considérées comme les pivots de ce projet seront effectuées par tout le groupe de façon à ce que chaque membre soit en mesure de rattacher son travail au cœur du projet.

1.1.4 Planning

Soutenance 1	Soutenance finale
Chargement des images et suppressions des couleurs	Pré-traitement
Détection et découpage en blocs, lignes et caractères	Réseau de neurones complet
POC du réseau de neurones	Reconstruction du texte et sauvegarde
Jeu d'images pour l'apprentissage (commencé)	Interface utilisateur
Manipulation de fichiers pour la sauvergarde des résultats (commencé)	Intégration d'un correcteur orthographique (éventuel)

Développer cet OCR permettra de nous initier au domaine de la reconnaissance d'images et de nous faire découvrir tous les principes mathématiques qui y sont liés. Nous allons devoir implémenter de nombreux algorithmes et en étudier les avantages et les inconvénients.
Tout au long de ce projet, nous acquérirons de nouvelles connaissances en traitement d'images et en machine learning.

2 Le groupe

2.0.1 Hamza GAIZI

Jeune nîmois passé par EPITA Rennes, c'est le couteau Suisse de l'équipe. Incollable sur tous les sujets, son accent du sud ne sèchera jamais devant un problême complexe.

2.0.2 Martin LACAILLE

Les roses sont rouges, les tulipes sont bleues, Martin est un gros bourrin. Capable de maîtriser son sujet si il concerne la programmation, son talent est un peu gâché par son inertie qui lui fait coder des centaines de lignes sans prendre assez de recul pour juger de la pertinence de son travail.

2.0.3 Bastien ROZEN

Officiellement, il regarde des vidéos pour se documenter sur le sujet. Dans les faits, Bastien tire son épingle du jeu en regardant vers le futur les différents problèmes qui nécessitent de solides fondations. Sa maxime : "Tant mieux si la route est longue, on pourra faire un peu plus de détours. L'avenir appartient à ceux qui s'lèvent à l'heure où j'me couche".

2.0.4 Loic SCHUSSLER

Véritable savoyard au sang bouillant, Loic est le responsable de l'IA du programme. C'est notre Rais à nous. Son seul défaut est d'utiliser le thème Discord en clair.

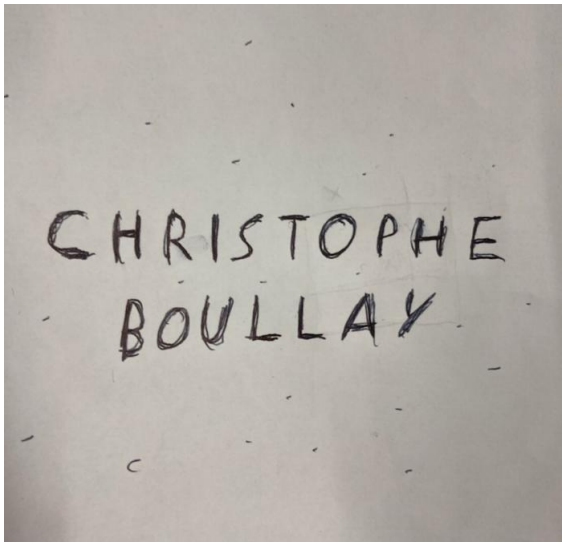
2.1 Pré-traitement de l'image

2.1.1 Transformation en niveaux de gris

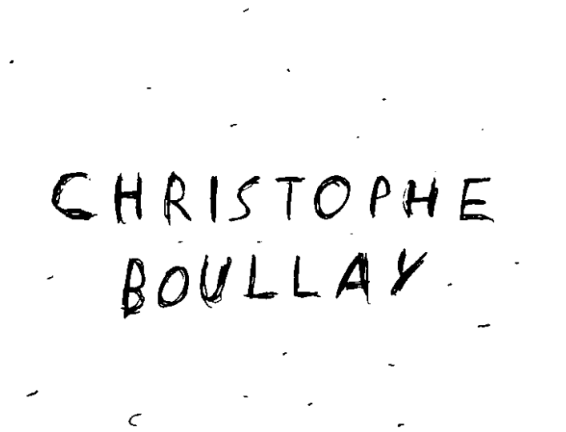
Dans un premier temps, on applique un niveau de gris à l'image. Ceci nous aidera pour passer ultérieurement l'image en noir et blanc.
Pour déterminer le niveau de gris, on calcule la luminescence de chaque pixel selon ses valeurs RGB. On donne ensuite la même valeur à calculer en RGB. En appliquant ce système pour chaque pixel, on obtient l'image en niveau de gris.

Exemple de transformation d'une image "colorée" d'un texte manuscrit en une image à niveaux de gris :

Avant modifications



Après modifications



2.1.2 Débruitage par convolution

On a donc obtenu des images de caractères en niveaux de gris à partir de la luminescence de chaque pixel. Il peut persiter sur l'image des imperfections. C'est ce qu'on appelle des bruits parasites, ou juste bruits. C'est la présence de certains pixels ayant un niveau de gris trop élevé ou trop faible. Ces petites perturbations de l'image peuvent entraîner des erreurs dans l'apprentissage de l'OCR car les caractères sont mal délimités. Cela peut provoquer une erreur plus importante du réseau, entrainant un décalage dans l'ajustement des poids de chaque connexion.

Pour tenter de supprimer ces bruits parasites, nous allons appliquer ce qu'on appelle une convolution moyenne sur chaque pixel de l'image.

Le concept de convolution consiste à appliquer une sorte de filtre représenté sous forme de matrice, que nous allons va faire passer sur une autre matrice plus grande.

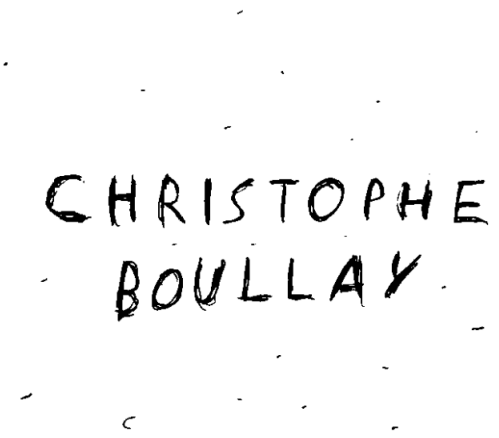
Dans le cas du débruitage, on applique à chaque pixel de notre image de caractères une convolution moyenne. La taille de la convolution peut varier au cas par cas, mais on considèrera pour l'exemple une convolution de taille 3x3, qui est la plus simple.

La convolution moyenne va consister à centrer la matrice de convolution de taille 3x3 sur le pixel, puis de faire la moyenne du niveau de gris de tous les pixels compris dans cette matrice. Une fois la convolution appliquée sur l'ensemble de l'image, on a en principe réduit le bruit à une quantité minime.

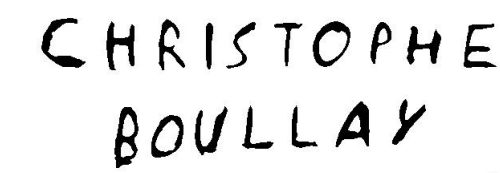
Principe de convolution moyenne :

Debruitage de l'image grisée et du texte manuscrit :

Avant modifications :



Après modifications :



2.1.3 Rotation

Il peut arriver qu'une image chargée soit inclinée ou déformée. Dans ce cas, on applique une rotation sur l'image.
Pour appliquer une rotation à une image en fonction d'un angle et d'un point d'origine de rotation, nous utilisons les formules suivantes :

$$fx(x, y) = \cos(\alpha)(x - px) - \sin(\alpha)(y - py) + px \tag{1}$$

$$fy(x, y) = \sin(\alpha)(x - px) - \cos(\alpha)(y - py) + px \tag{2}$$

α représente l'angle, x et y les coordonnées du point dans l'image source, px et py les coordonnées du centre de rotation.

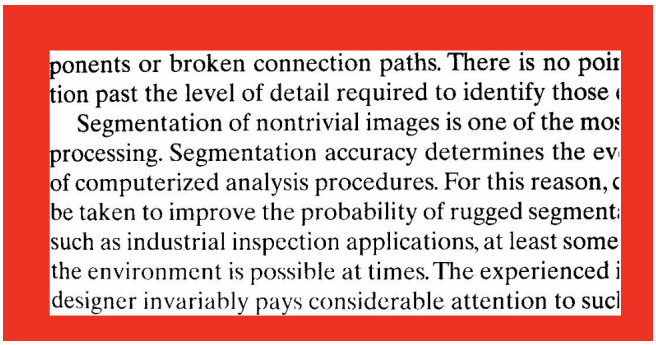
La détermination de l'angle se pose comme la partie la plus complexe de la rotation. Elle nécessite l'utilisation de la transformée de Hough, qui permet de reconnaitre une certaine forme dans une image. Elle découle d'un principe assez simple : en un point passent un nombre infini de lignes, la seule distinction entre elles étant l'angle. La transformée de Hough va détecter lesquelles de ces lignes passent au plus près du schéma attendu.

Pour la rotation automatique, il faudra tester la taille du paragraphe avec l'algorithme présenté après avec différents angles. L'image avec la meilleure rotation sera celle dont les dimmensions seront les plus petites

2.1.4 Segmentation

Après avoir pré-traité l'image pour en optimiser la qualité , cette dernière est prête pour la segmentation.

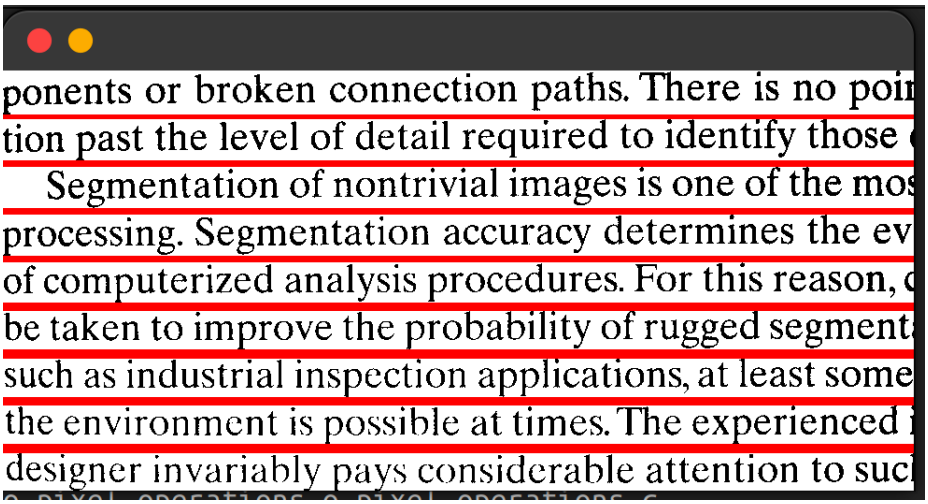
2.2 Détection du paragraphe



(l'image avait dès le départ les lettres de droites rognées, ce n'est pas dû à une erreur dans l'algorithme).

La première étape dans la segmentation est la détection du paragraphe de texte dans l'image.
Pour cela, pour chaque ligne de l'image, toutes les lignes et colonnes de pixels à partir des extrémités sont rognées si les lignes ou colonnes sont blanches, jusqu'à avoir des lignes ou colonnes qui contiennent des lignes de pixels noirs, c'est à dire une ligne de texte.

2.2.1 Détection de lignes



Une fois le paragraphe isolé, nous pouvons délimiter les lignes du texte. Pour cela, nous avons identifié les interlignes. Les lignes de pixels constituées uniquement de lignes blanches sont considérées comme des interlignes. Comme les interlignes sont identifiées, une ligne est facile à repérer car située entre 2 interlignes. Il suffit d'isoler

et de mettre ces lignes dans un tableau pour avoir un tableau constitué de chaque ligne du texte à analyser.

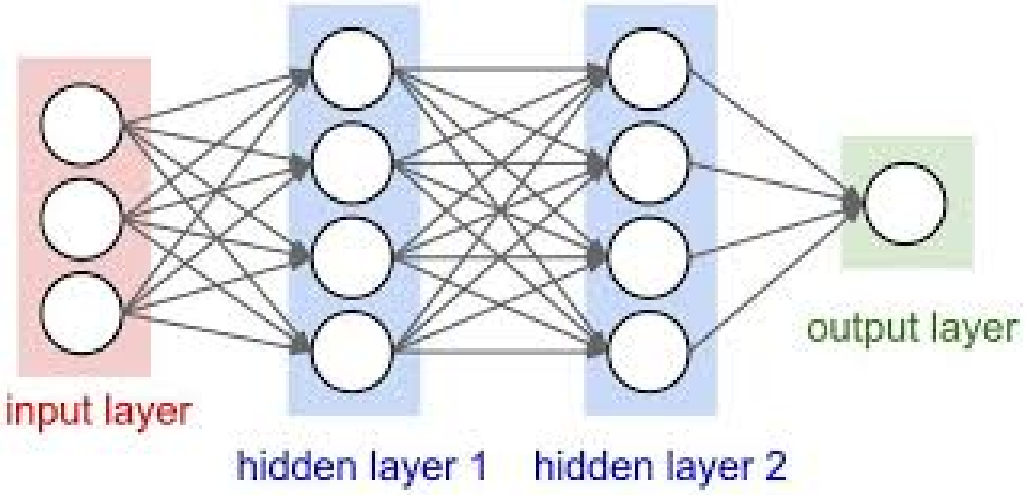
2.2.2 Détection de caractères

Pour chaque ligne, il faut repérer l'espace entre les caractères, c'est-à-dire les colonnes blanches. Les lettres sont donc logiquement situées entre les espaces (oui, je sais ce n'était pas facile à deviner...). Une fois les lettres repérées, il suffit de les placer dans une liste pour avoir la liste des images des caractères.

Et la détection des espaces dans tout ca ?

Justement, nous allions en parler.
Les espaces entre les caractères plus longs que le quart de la largeur du caractère sont considérés comme des espaces. Voila, c'est tout.

3 Réseau de neurones : Avancée Actuelle



3.0.1 Définition

Le réseau de neurones est un élément qui va permettre l'identification des caractères pour reconstruire le texte.
Un réseau de neurones artificiels est un système s'inspirant du fonctionnement des neurones biologiques. Le réseau va donc simuler le comportement d'un cerveau.
Dans le cadre du projet OCR nous utilisons un réseau de neurones dit supervisé. Celui-ci prendra des entrées qui seront modifiées au fur et à mesure de la transmission des données dans le réseau, à partir des poids entre les neurones et leur unique biais. A la sortie du réseau la valeur obtenue sera comparée avec une valeur cible définie pour ces entrées. Selon l'écart obtenu, des modifications vont être effectuées sur les biais et poids du réseau. On appelle cela la propagation inversée.
Dans un réseau supervisé, l'élaboration d'un ensemble de données d'entrainement est primordial, c'est à dire des n-uplets de données d'entrées avec n étant le nombre de neurones de la couche d'entrée ainsi que le ou les résultats attendus en sortie.

Un réseau de neurones est constitué de différentes couches :

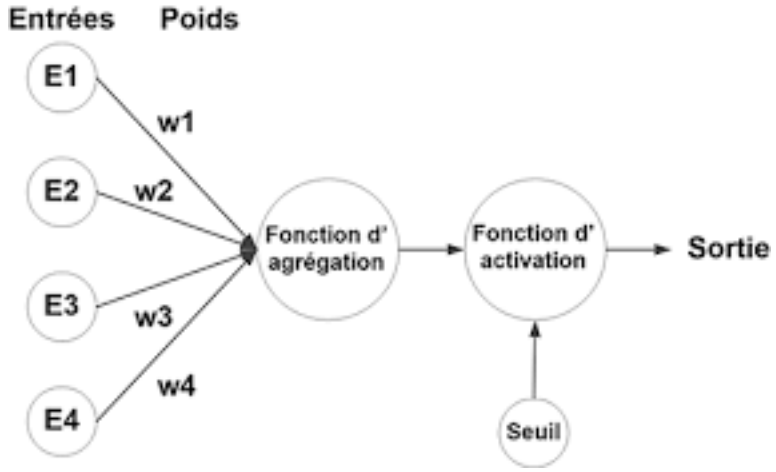
— une couche d'entrée : ses neurones contiendront les informations que l'utilisateur souhaitera tester.

— une couche de sortie : ses neurones nous donneront le résultat que le réseau aura calculé.

— une ou plusieurs couches cachées : ses neurones ne serviront qu'au calcul.

Un réseau de neurones est donc assimilable à une fonction mathématique.

Les neurones de la couche d'entrée sont chacun reliés à tous les neurones de la première couche cachée (si il y en a) ; ceux de la première couche cachée sont reliés à tous ceux de la couche suivante, et ainsi de suite jusqu'au neurone de sortie. Il peut arriver que les neurones d'entrée soient connectés à la couche de sortie malgré la présence de couches cachées, ce qui peut permettre d'améliorer l'apprentissage du réseau.
Chaque connexion entre deux neurones possède un poids qui sera utilisé dans différents calculs. Chaque neurone possède également une valeur de sortie (comprise entre 0 et 1), qui est la valeur à tester pour la couche d'entrée et le résultat d'un calcul pour les autres couches. Cette valeur correspond à l'information que chaque neurone transmet aux neurones suivants.



Ces neurones reçoivent un nombre X d'entrées, chaque entrée représentant la sortie d'un neurone précédent. Les valeurs sont ensuite multipliées par un coefficient propre à chaque connexion : c'est le poids de la connexion. Ces valeurs sont ensuite sommées, puis cette somme est traitée par une fonction inhérente à chaque neurone et qui représente généralement une fonction de seuil. La valeur de la somme va dépendre de la comparaison avec ce seuil, et le résultat en sortie en découlera.

Après avoir compris le fonctionnement d'un neurone, il a fallu étudier l'architecture du réseau pour déterminer la manière dont les neurones communiquent entre eux.
La structure d'un réseau de neurones peut être généralisée à un enchainement fini de couches de neurones.

3.0.2 Apprentissage

L'apprentissage est ce qui va permettre au réseau de neurones de s'améliorer en fonction de ses erreurs. Cela consiste en une modification des poids internes à chaque connexion, le tout en plusieurs étapes pour s'approcher de plus en plus de la valeur attendue.
A chaque étape, les poids des connexions se rapprochent de leurs valeurs optimales. C'est la marge d'erreur de l'état précédent qui sera utilisée pour corriger la valeur du poids.

Un apprentissage est dit supervisé lorsque l'utilisateur force le réseau à s'approcher d'un état final précis, connu. Dans ce cas on utilisera un Training Set contenant des exemples dont le résultat est connu. Ainsi on pourra comparer ce résultat à la valeur donnée par le réseau en sortie et ajuster les poids en fonction de cette différence.

Il est possible d'influencer la vitesse d'apprentissage du réseau en augmentant le nombre de neurones ou en connectant les neurones d'entrées directement aux neurones de sortie.

On peut également agir sur le pas d'apprentissage, une valeur permettant de régler la vitesse à laquelle le réseau apprend, mais aussi la précision. Plus la valeur du pas sera grande, plus l'apprentissage se fera rapidement, mais moins il sera précis.

3.0.3 Calcul en avant (Feed Forward)

L'algorithme de calcul en avant (Feed Forward en anglais) permet de déterminer les valeurs de sortie du réseau de neurones. Comme l'indique son nom, c'est un calcul qui se fait de l'entrée vers la sortie.

La fonction d'activation est une fonction qui va permettre la verification des valeurs en sortie du reseau de neurones. Elle retournera un reel proche de 1 si les bonnes informations d'entree sont donnees, et un reel proche de 0 dans le cas contraire.

3.0.4 Retro-propagation

La retro propagation est l'étape qui consiste à "corriger" le réseau de neurones à partir de l'erreur calculée en sortie du réseau par rapport aux données d'entrée. L'algorithme que nous allons utiliser pour effectuer les corrections consiste à appliquer à chaque poids et biais du réseau la modification suivante. A chaque étape, la valeur d'un biais ou un neurone devient :
$X_n = (X_{n-1} - (\alpha) * grad(f(S_{n-1})))$ avec :

— X_{n-1} : valeur du biais ou du poids à l'étape précédente

— S_{n-1} : données du retour de la fonction d'erreur avec la sortie du réseau à l'étape précédente

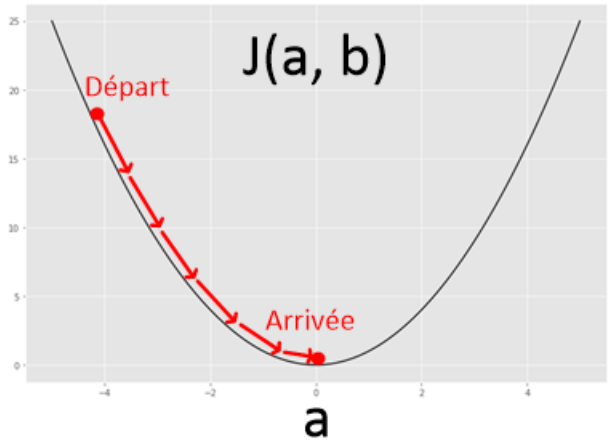
— f() = fonction d'erreur

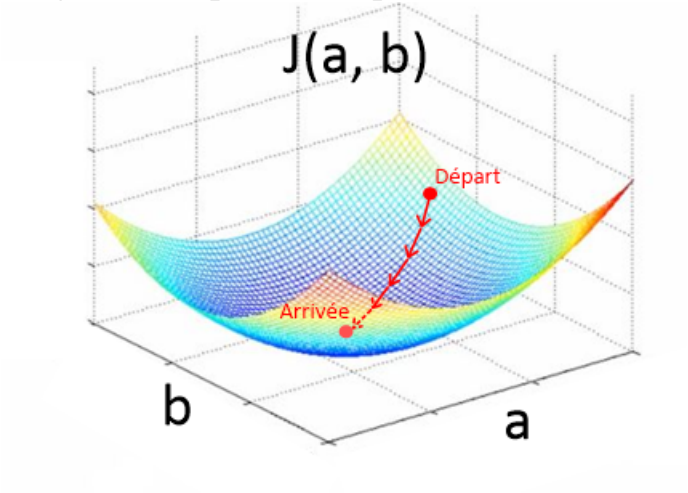
— $grad(f)$ il s'agit du gradient de f par rapport à l'ensemble des paramètres de f.

On répète ce processus jusqu'à observation d'une convergence de X_n.

Pour imager ce concept on prend souvent l'exemple d'un randonneur qui se serait perdu dans une montagne et qui chercherait à rejoindre le point le plus bas d'une vallée. L'algorithme de descente du gradient consiste à suivre la pente pendant une certaine distance (il s'agit du learning rate ou pas dans la formule) avant de recommencer à suivre la pente. A force d'appliquer ce processus, le randonneur devrait arriver au point le plus bas de la vallée. Bien sûr imagé, le concept de descente de gradient à plus de 2 paramètres devient compliqué, car nécessitant un nombre de dimensions trop grand pour être représenté.

Voici un exemple d'une image de descente de gradient à partir de 2 paramètres :





3.0.5 Réseau de neurones : Porte XOR

Dans notre projet, nous avons la possibilité d'implémenter n'importe quelle porte logique pour l'instant. Pour entrainer le programme sur le XOR, il faut lui apporter les 4 portes logiques ainsi que les 4 réponses. Voici comment marche, pour l'instant, notre réseau de neurones.

```
hamza@TheBread:~/martin.lacaille/NeuralNetwork$ cc -g -Wall
hamza@TheBread:~/martin.lacaille/NeuralNetwork$ cc -pthread
hamza@TheBread:~/martin.lacaille/NeuralNetwork$ ./main
```

```
There are 3 layers.
 Parameters of the network :
 input layer 2 neurons hidden layer
 3 neurons -output layer
 1 neurons

Created Layer: 1
Number of Neurons in Layer 1: 2
Neuron 1 in Layer 1 created
Neuron 2 in Layer 1 created

Created Layer: 2
Number of Neurons in Layer 2: 3
Neuron 1 in Layer 2 created
Neuron 2 in Layer 2 created
Neuron 3 in Layer 2 created

Created Layer: 3
Number of Neurons in Layer 3: 1
Neuron 1 in Layer 3 created


Initializing weights...
0:w[0][0]: 0.840188
1:w[0][0]: 0.394383
2:w[0][0]: 0.783099
0:w[0][1]: 0.798440
1:w[0][1]: 0.911647
2:w[0][1]: 0.197551
0:w[1][0]: 0.335223
0:w[1][1]: 0.277775
0:w[1][2]: 0.477397

Neural Network Created Successfully...

There is a learning rate of 0.15

There are 4 loop of training.

Enter the Inputs for training example[0]:
```

```
Created Layer: 1
Number of Neurons in Layer 1: 2
Neuron 1 in Layer 1 created
Neuron 2 in Layer 1 created

Created Layer: 2
Number of Neurons in Layer 2: 3
Neuron 1 in Layer 2 created
Neuron 2 in Layer 2 created
Neuron 3 in Layer 2 created

Created Layer: 3
Number of Neurons in Layer 3: 1
Neuron 1 in Layer 3 created


Initializing weights...
0:w[0][0]: 0.840188
1:w[0][0]: 0.394383
2:w[0][0]: 0.783099
0:w[0][1]: 0.798440
1:w[0][1]: 0.911647
2:w[0][1]: 0.197551
0:w[1][0]: 0.335223
0:w[1][1]: 0.277775
0:w[1][2]: 0.477397

Neural Network Created Successfully...

There is a learning rate of 0.15

There are 4 loop of training.

Enter the Inputs for training example[0]:
0 0

Enter the Inputs for training example[1]:
0 1

Enter the Inputs for training example[2]:
1 0

Enter the Inputs for training example[3]:
```

```
Number of Neurons in Layer 3: 1
Neuron 1 in Layer 3 created


Initializing weights...
0:w[0][0]: 0.840188
1:w[0][0]: 0.394383
2:w[0][0]: 0.783099
0:w[0][1]: 0.798440
1:w[0][1]: 0.911647
2:w[0][1]: 0.197551
0:w[1][0]: 0.335223
0:w[1][1]: 0.277775
0:w[1][2]: 0.477397

Neural Network Created Successfully...

There is a learning rate of 0.15

There are 4 loop of training.

Enter the Inputs for training example[0]:
0 0

Enter the Inputs for training example[1]:
0 1

Enter the Inputs for training example[2]:
1 0

Enter the Inputs for training example[3]:
1 1

Enter the Desired Outputs (Labels) for training example[0]:
1

Enter the Desired Outputs (Labels) for training example[1]:
0

Enter the Desired Outputs (Labels) for training example[2]:
0

Enter the Desired Outputs (Labels) for training example[3]:
0
```

```
Output: 0

Input: 1.000000
Input: 0.000000
Output: 0

Input: 1.000000
Input: 1.000000
Output: 0

Input: 0.000000
Input: 0.000000
Output: 1

Input: 0.000000
Input: 1.000000
Output: 0

Input: 1.000000
Input: 0.000000
Output: 0

Input: 1.000000
Input: 1.000000
Output: 0

Input: 0.000000
Input: 0.000000
Output: 1

Input: 0.000000
Input: 1.000000
Output: 0

Input: 1.000000
Input: 0.000000
Output: 0

Input: 1.000000
Input: 1.000000
Output: 0
```

```
Input: 1.000000
Input: 0.000000
Output: 0

Input: 1.000000
Input: 1.000000
Output: 0

Input: 0.000000
Input: 0.000000
Output: 1

Input: 0.000000
Input: 1.000000
Output: 0

Input: 1.000000
Input: 0.000000
Output: 0

Input: 1.000000
Input: 1.000000
Output: 0

Enter input to test:
1
1
Output: 0

Enter input to test:
0 0
Output: 1

Enter input to test:
1 0
Output: 0

Enter input to test:
0 1
Output: 0
```

3.1 Conclusion (de la première soutenance)

Au cours de ces premières semaines de projet nous avons réussi à réaliser les étapes de conception de l'OCR demandées pour cette date de soutenance et même un petit peu plus. A ce jour nous avons donc :

- un programme qui permet de charger une image, de supprimer les couleurs, d'enlever le bruit, et d'appliquer des rotations.
- un programme qui permet la détection de paragraphes, de lignes puis de caractères.
- une architecture de réseau de neurones utilisée pour apprendre le XOR facilement modifiable pour la réalisation du réseau de neurones qui reconnaît les caractères.
- Bien sûr, nous avons aussi acquéri une multitude de connaissances sur le traitement des images, sur les réseaux de neurones...qui nous permettra d'avancer beaucoup plus vite et mieux pour la prochaine soutenance.

Notre travail pour la prochaine soutenance va être colossal mais réalisable. Nous espérons pouvoir vous présenter un réseau manipulable par une interface utilisateur capable d'identifier les caractères composant un texte. Si notre travail avance bien, pourquoi pas se pencher sur un correcteur ortographique utilisant ce programme.

L'ensemble de l'équipe 4puter Can Read vous remercie d'avoir pris le temps de lire notre compte rendu ainsi que d'avoir écouté notre présentation. A bientôt pour la suite de l'éducation de nos chers amis numériques.

4 Et maintenant ?

4.1 Ce que l'on a retenu de la première soutenance

Nous avions jugé la première soutenance positive : malgré une organisation et un travail de présentation qui pouvaient être améliorés, nous avions l'envie et la capacité de faire plus. Nous nous sommes donc concentrés à travailler en groupe, et à faire un code lisible par tous les membres du groupe. Le principal obstacle à notre bonne volonté a été le confinement et la distance qu'il a engendré entre les membres du groupe : la coordination et l'envie d'en faire plus ont été fortements réduits. Néanmoins, là ou certains y verraient une excuse pour moins travailler, nous y avons vu un défi supplémentaire.

5 Répartition des tâches

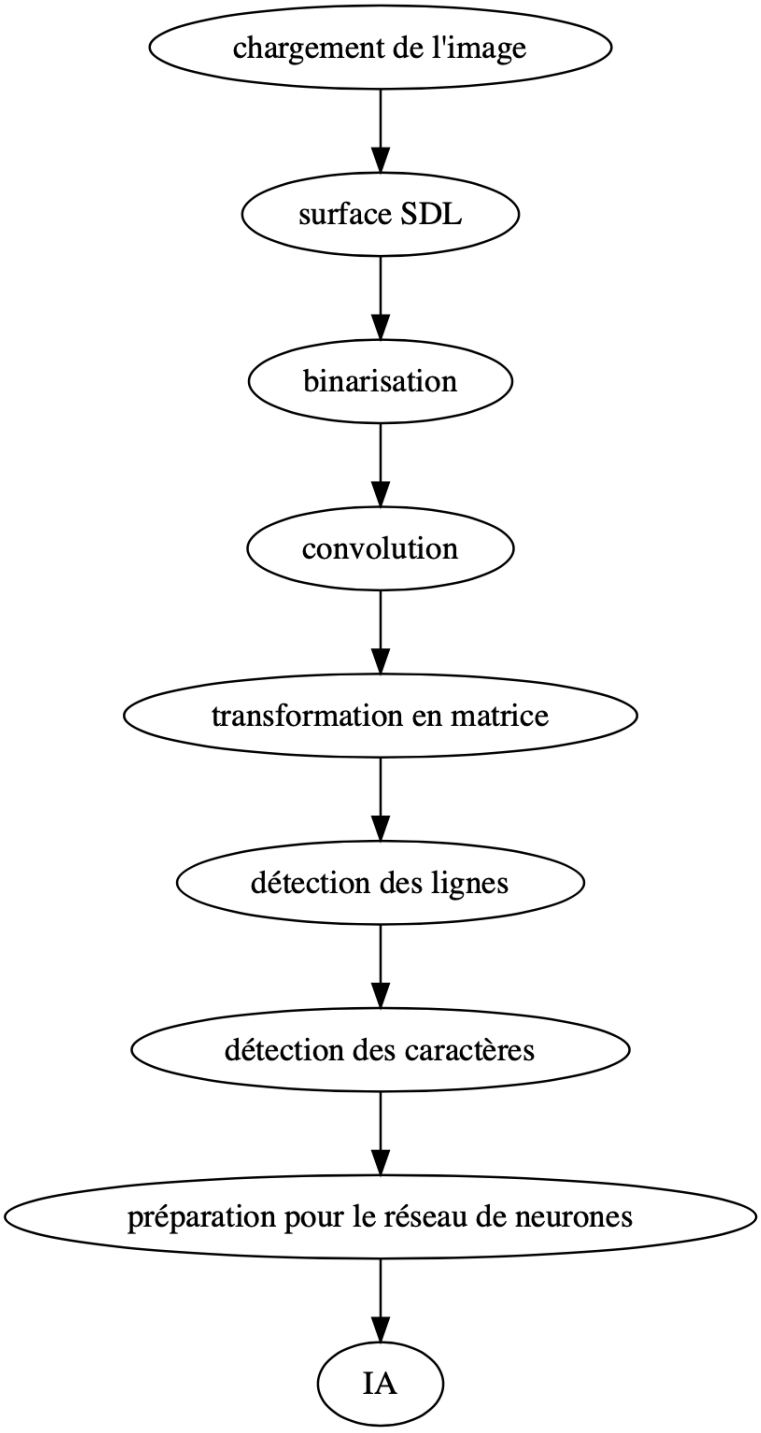
Après la première soutenance, nous nous sommes aperçu qu'avoir un travail des membres plus transversal serait plus bénéfique. Au lieu d'avoir un membre attribué à une tâche spécifique, chacun travaillait sur l'ensemble du projet, bien tout le monde soit responsable d'une partie spécifique.

Tâches	Hamza	Martin	Bastien	Loic
Segmentation		X		X
IA	X			X
Interface graphique			X	

6 Traitement de l’image

6.1 Refaire les choses proprement...

Lors de la première soutenance, nous nous sommes aperçu que les étapes de la segmentation n’étaient pas parfaitement claires. Nous avons donc tout refait, en repartant de 0. Une image valant 1000 mots, voici le schéma explicatif du traitement de l’image.



Nous utiliserons SDL pour chager, binariser et traiter l'image. Cette

7 Réseau de neurones

7.1 Définition

Le Neural Network, réseau de neurone en français, est une intelligence artificielle. Le principe de ce dernier est le fait d'apprendre tel le cerveau humain. En effet, l'apprentissage de l'Homme se fait par des connections de nombreuses synapses à travers un système biologie afin de favoriser l'utilisation de la mémoire et donc l'apprentissage. Il est utilisé afin de faciliter la tâche aux utilisateurs ou aux entreprises. Ce genre d'intelligence arificielle est implémentée afin d'éxecuter des tâches dites "simples et répétitives" ; Le domaine de l'intelligence artificielle est en pleine expansion. En effet, à l'heure d'aujourd'hui, les intelligences artificielles font parties intégrantes de toutes les grandes sociétés et sont devenues indispensables. Pour ce réseau de neurone, nous devions faire un Neural Network afin de pouvoir reconnaître descaractères dans une image. Il fut important de garder celà en tête pendant laconceptualisation et nous avons bien réussi à le faire.

7.2 Organisation

Comment est organisé le réseau de neurones ?Le réseau de neurone se décompose en plusieurs parties dont une partie étant la propagation et la propagation arrière(dits respectivement Forward Propagation et Back Propagation). Pour l'OCR que nous avons aujourd'hui, nous avons un Réseau de neurones comportant 3 couches distinctes : une couche d'entrée, une couche cachée et une couche de sortie. La couche d'entrée est composée de 256 noeuds(16*16) car les images récupérées sont toujours de cette taille-ci. La couche cachée est de 300 noeuds. Après de nombreux tests, nous avons conclus que notre OCR était plus fonctionnel avec ce nombre-ci à l'instar des autres. La couche de sortie est de 69 noeuds car elle comprend toutes les valeurs possibles de sortie. Elle comprend toutes les minuscules (a - z), les Majuscules (A - Z), les chiffres (0 - 9) ainsi que lessymboles les plus utiles tels que '.', ',' ou encore les parenthèses. Chacun des noeuds sont reliés aux noeuds des couches adjacentes. Ces liens sont valués tout comme les noeuds. On appelle les valeurs des liens "valeur des poids" et la valeur des noeuds est dit "valeur de biais" ;

7.3 Fonctionnement

Comme donné plus tôt, le réseau de neurones se découpe en deux parties : la phase de propagation avant et la phase de propagation arrière. Les deux phases sont executées avec de nombreuses fonctions mathématiques. La propagation avant sert à trouver des valeurs de tests grâce aux valeurs des entrées, les valeurs des poids ainsi que les biais. Ici l'entrée est composée de 0 et de 1 qui sont les valeurs du caractère après segmentation. Les valeurs de poids et lesbiais sont organisées aléatoirement entre -0.5 et 0.5. La combinaison mathématiquede ces valeurs nous donnent les valeurs de sortie qui sont aussi composées de 0 et de 1.La propagation arrière permet de corriger les erreurs de poids et ainsi accéder à des valeurs correctes suite à de nombreuses itérations. Elle prend en compte les valeurs visées et les valeurs calculées précedemment pendant la propagation avant. Grâce aux formules de delta et de sigmoid nous arrivons à obtenir les valeurs escomptées malgré une petite marge d'erreur. Il est composé d'un mode entrainement comportant ces deux phases, pendant cette phase un fichier est créeet sauvegardé avec tous les poids et les biais des matrices. Ce fichier pourra être réutilisé ultérieurement afin de réduire un autre entrainement ou bien de lancer l'OCR. En effet, le Neural Network contient un mode de transmission direct (qui équivaut à une seule propagation avant), ce mode-ci prend en compte deux fichier :- un fichier Input pour pouvoir remplir la matrice de départ contenant 256 caractères. Ce fichier peut contenir plusieurs caractères mots car il est utilisé dans une boucle afin de pouvoir analyser tout un texte. On a mit en place un "code" d'écrture qui est un caractère par ligne avec ” pour passage à la ligne et ’’ pour un changementde mot.- un fichier de poids qui sera parsé de telle sorte que les matrices de poids peuventêtre toutes initialisées ainsi que leur biais.

7.4 Formules

Pour la propagation avant, nous avons utilisé des formules simples de calculsmatriciels tel que f(x) = Weight[i][j]*Input[i] + bias[j] avec Weight : la matrice depoids entre deux couches, bias : la matrice contenant les poids des neurones d'arrivéset Input : la matrice contenant les valeurs d'entrée. Pour la propagation arrière ditbackpropagation, nous utilisons la sigmoid : Output[i] = 1 / 1 + exp(-Output[i]) etle delta de sigmoid : Output[i] = Output[i] * (1 - Output[i]). Les valeurs sont doncmodifiées à la remontée et sont donc mathématiquement calculées afin d'arriver àun résultat optimal.

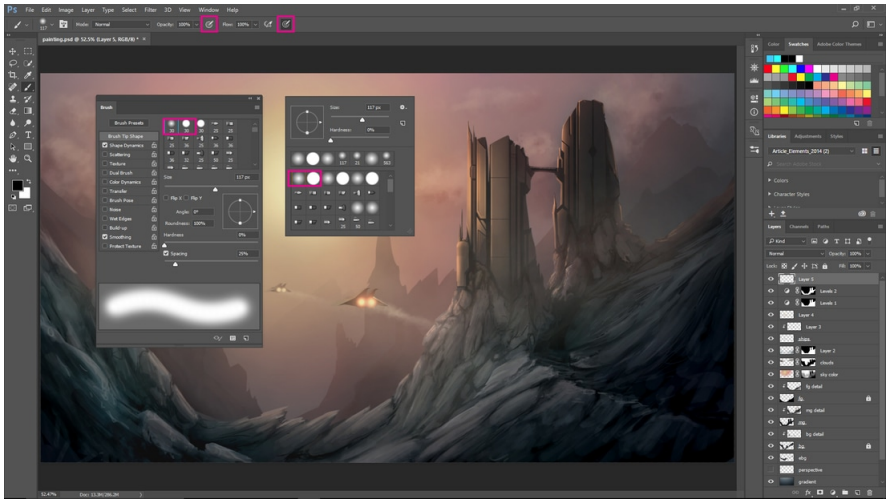
7.5 Conclusion sur l'IA

La compréhension de l'algorithme du réseau de neurones fut assez compliqué. J'aidû assimiler les connaissances nécessaires afin de confectionner le réseau de neurones.Le langage C n'étant pas adapté à la conceptualisation d'un réseau de neurones, lenombre de documents existants sur le sujet est moindre. Le passage du XOR àl'OCR était aussi compliqué car ils ont beau utiliser la même manière de faire, lesalgorithmes sont complètement différents

8 Interface graphique

8.1 Principe d'une interface

L'interface graphique, souvent appelée GUI (de l'anlais Graphical User Interface) est une partie importante d'un logiciel de reconnaissance optique des caractères. Elle permet la manipulation de fichiers de manière très intuitive, de telle sorte que n'importe quel utilisateur puisse s'y habituer rapidement et facilement. C'est dans une optique de distribution à un large public que l'on utilise une interface graphique.



Example d'une interface graphique, Photoshop

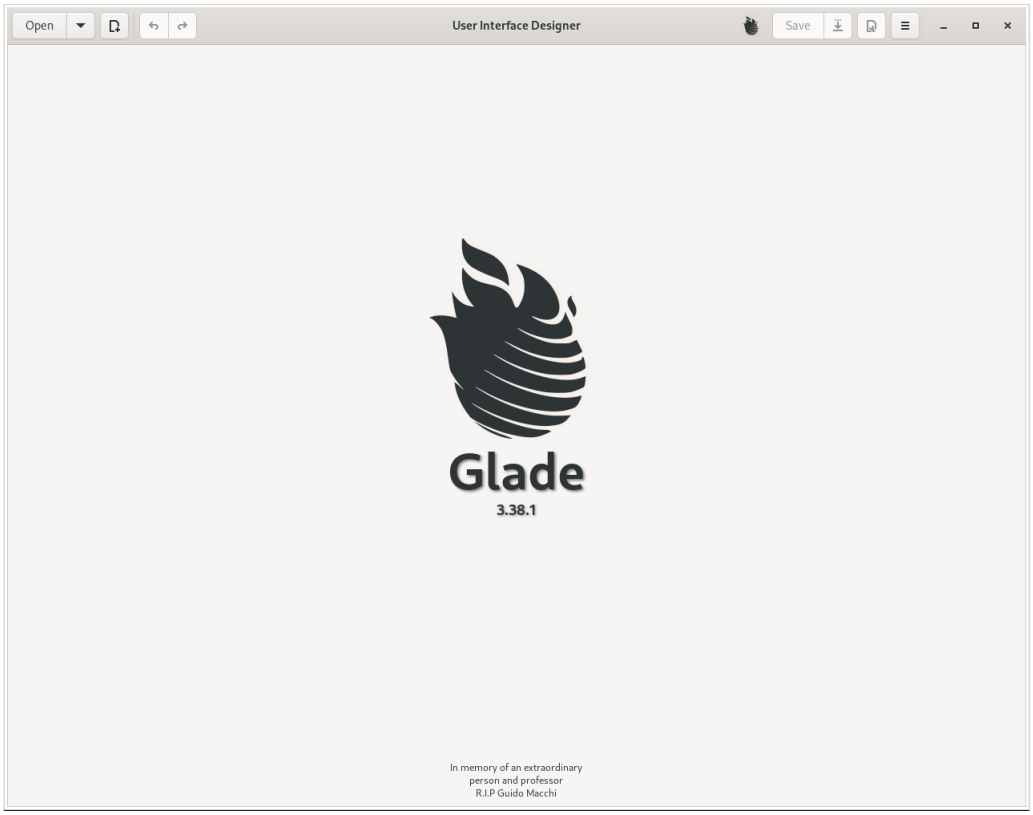
8.2 Gtk+-3.0

GTK (The Gimp Toolkit) est un ensemble de bibliothèques logicielles écrites en langage C et permettant de réaliser des interfaces graphiques, avec l'aide du logiciel Glade. Pour créer une interface, il faut d'abord créer une fenêtre, qui contiendra des boites dans lesquelles se trouveront des gadgets (widgets). Ces widgets ont beaucoup d'utilisations possibles. Ils peuvent être des boutons utilisateurs, des fenêtres dans lesquelles on peut afficher des images, des boites de texte... Cela permet de minimiser la dose de code Gtk.

8.3 Glade

Glade est un outil permettant de créer des interfaces utilisateurs pour logiciels. C'est un outil compliqué à prendre en main. De prime abord, il parait simple ; réaliser un bouton sur lequel on peut cliquer est par exemple assez facile à faire. Cependant, Glade propose beaucoup d'options réparties sur beaucoup de menus différents. Ainsi, lorsque l'on souhaite réaliser une interface plus complète. Il a donc fallu aller parcourir des tutoriels sur divers sites afin de comprendre un peu mieux les différentes fonctionnalités de ce logiciel.

Glade permet de créer automatiquement le code nécessaire à l'affichage même de l'interface. Cependant, cela ne permet pas d'en rendre les objets fonctionnels. Il faut pour cela créer le code source, par exemple pour permettre à l'utilisateur une recherche parmi les dossiers de l'ordinateur.



Glade

8.4 Les Box

Les Box représentent la logique des bibliothèques Gtk. Ce sont les éléments principaux et les plus importants de l'interface grpahique, bien qu'elles n'apparaissent pas aux yeux de l'utilisateur.
Il existe deux sous-catégories de box : les hbox permettant d'empiler des éléments de manière horizontale, et les vbox, pour empiler de manière verticale. Ces box contiendront tous les éléments dont l'utilisateur se servira pour lancer le processus de reconnaissance des caractères.

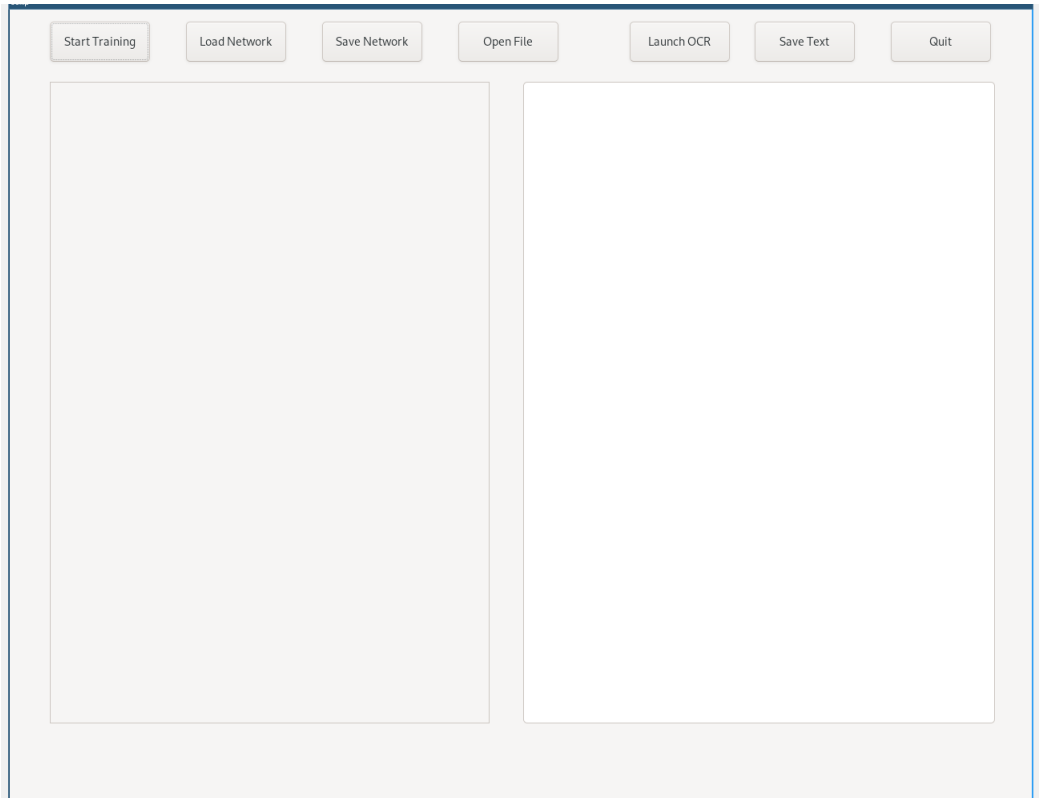
8.5 Les Widgets

Les Widgets, ou Gadgets en français, constituent le véritable coeur graphique d'une interface utilisateur. Il en existe plusieurs sortes : les widgets se présentant sous forme de fenêtres et affichant quelquechose, comme des images ou du texte, les widgets permettant à l'utilisateur d'effectuer directement une action, comme les boutons ou les barres de menus. Il en existe aussi qui sont des barres de progression ou de scrolling, pour pouvoir se déplacer dans les documents. C'est donc réellement au travers des widgets qu'un utilisateur pourra interagir avec l'interface, et donc le logiciel.

8.6 Notre interface

Notre interface graphique se présente sous forme de deux fenêtres, l'une permettant de montrer l'image que l'utilisateur souhaite faire traiter par le logiciel de reconnaissance optique des caractères, la deuxième affichant le texte que le logiciel rendra.

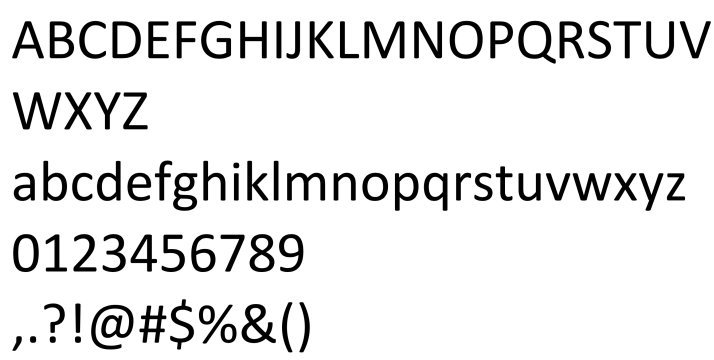
Elle a également pour vocation de réunir toutes les parties du projet réalisées de manière indépendante, car elle permet de traiter la totalité du processus de l'OCR.



Notre interface graphique au lancement

Comme on peut le voir, l'interface présente deux fenêtres et plusieurs boutons, chacun ayant une fonctionnalité propre.

Le bouton *Start Training* permet de lancer l'entraînement du réseau de neurones. Il va ouvrir une boîte de dialogue laissant l'utilisateur sélectionner un fichier image contenant tous les caractères que le logiciel devra être capable de reconnaître.



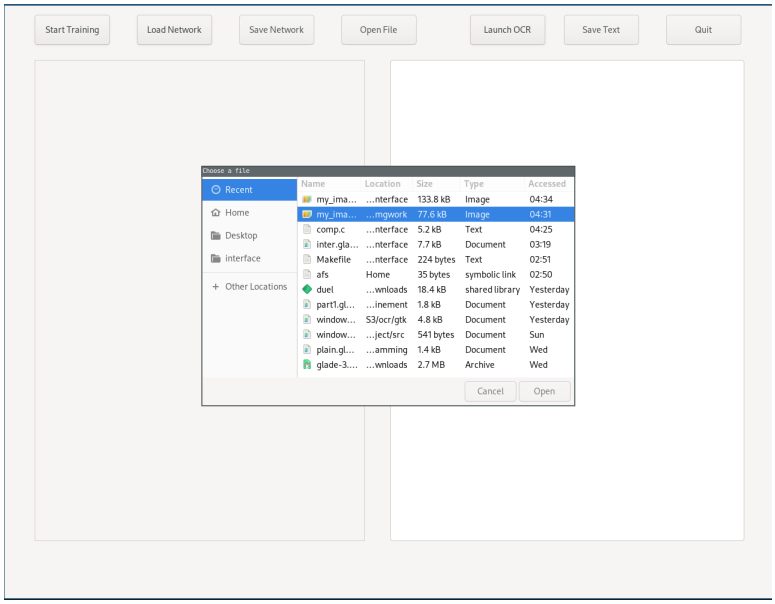
Caractères reconnus par l'OCR

Le bouton *Save Network* permet d'enregistrer l'évolution des poids au cours de l'entraînement du réseau de neurones afin de pouvoir ensuite l'utiliser de manière optimale.

Le bouton *Load Network*, quant à lui, charge les informations préalablement enregistrées.

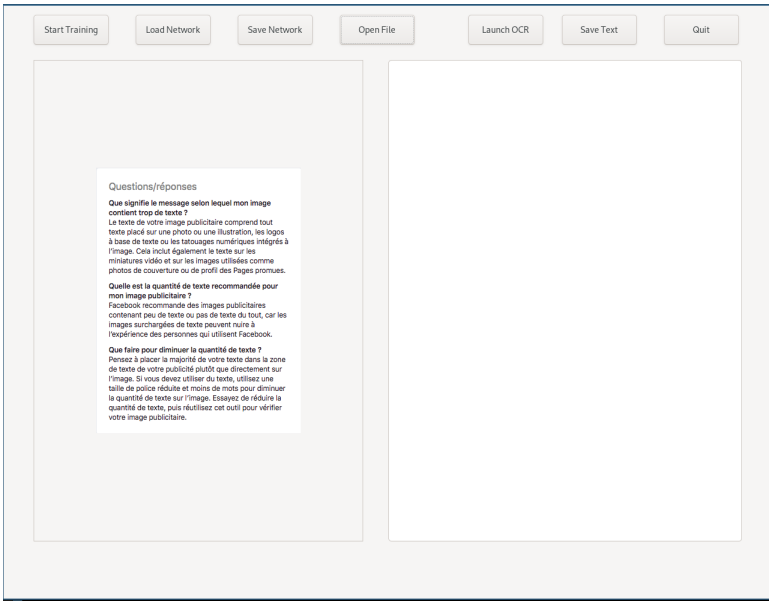
Le but étant d'éviter d'avoir à réentrainer le réseau de neurones à chaque que l'on souhaite utiliser le logiciel de reconnaissance optique des caractères.

Le bouton *Open File* va afficher une boite de dialogue permettant à l'utilisateur de choisir l'image qu'il souhaite faire traiter par le logiciel.



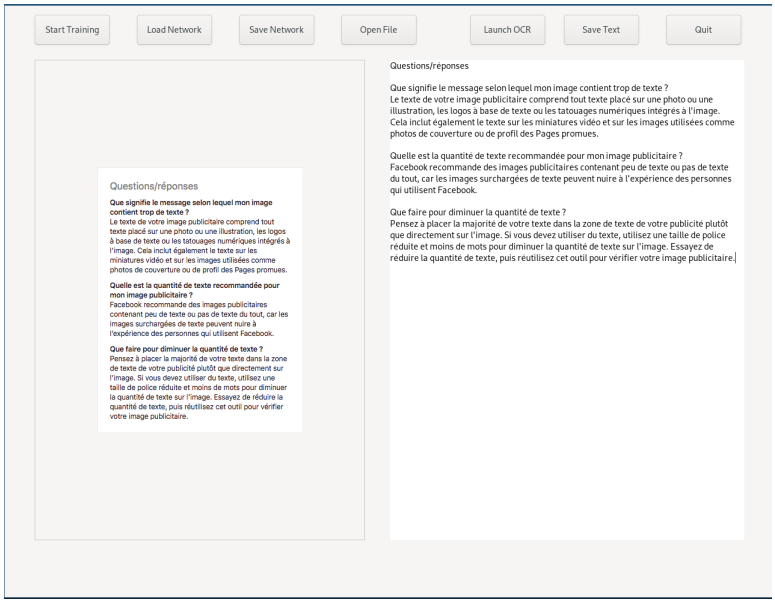
Boite de dialogue

L'image choisie va ensuite être affichée dans la fenêtre de gauche de l'interface utilisateur.



Ouverture de fichier

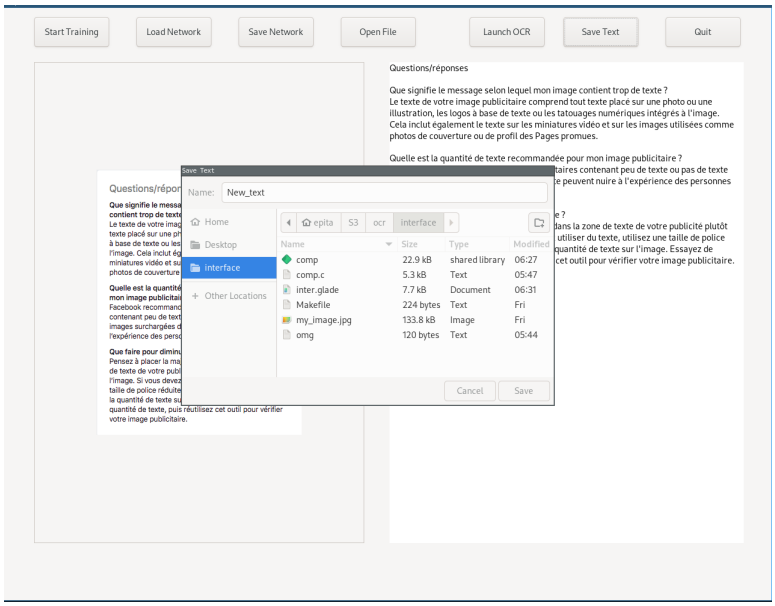
Lorsque le bouton *Launch OCR* est utilisé, le processus du logiciel se lance sur l'image préalablement affichée dans la fenêtre de gauche de l'interface utilisateur. L'image est alors prétraitée, binarisée, puis le réseau de neurones va être lancé avec les paramètres initialisés lors de l'entraînement. Le logiciel va donc "lire" l'image et récupérer le texte qu'elle contient. Ce texte sera ensuite affiché dans la fenêtre de droite de l'interface utilisateur.



Rendu après lancement de l'OCR

Le bouton *Quit* va permettre, comme son nom l'indique, de fermer l'interface graphique. Tous les fichiers créés pendant son utilisation seront conservés

Le bouton *Save Text* permettra à l'utilisateur d'enregistrer le texte affiché dans la fenêtre de droite. Une boite de dialogue va s'ouvrir, offrant la possibilité à l'utilisateur de choisir l'emplacement de sauvegarde de ce fichier texte, d'en déterminer le nom, ainsi que de créer de nouveaux dossiers s'il le souhaite. Un fichier texte sera donc créé, contenant le texte extirpé de l'image correspondante par le logiciel de reconnaissance optique des caractères.



Création du fichier texte

Fichier texte enregistré

9 Conclusion

9.1 Conclusions personnelles

9.1.1 Martin

Réaliser ce projet aura demandé de la méticulosité. Le plus dur était d'expliquer et de comprendre des concepts abstraits aux autres membres du groupe. La communication avec les autres membres du groupes a été la tâche la plus ardue. Cela n'a pas été simple mais a été rassurant : même dans un domaine aussi impersonnel et froid que l'informatique, le contact humain est l'aspect central.

9.1.2 Hamza

J'ai aimé travailler ce projet dans l'optique d'obtenir plus d'expérience en programmation. Ce projet était, certes compliqué, mais aussi m'a permis de voir chacun a réussi à faire avancer le groupe pour donner un projet viable. Je me suis personnellement chargé de l'IA qui n'était pas une charge facile en soit.

9.1.3 Loïc

Quand j'ai commencé à regardé le cahier des charges pour la première fois j'avoue avoir pris peur. Les fonctionnalités attendus me paraissaient si complexes. Avant le projet je voyais vaguement à quoi correspondait un réseau de neurones, maintenant j'en ai une vision beaucoup plus claire même si je ne comprend pas encore quelques subitlités mathématiques. Participer à ce travail au delà des compétences en c que cela m'a apporté (je partais de 0 je ne savais même pas compiler un simple hello world), ça m'a aussi permis d'apprendre à mieux communiquer avec les membres du groupe que ce que j'avais pu faire pour mon projet de S2. Quelque chose que je retiens aussi c'est les conditions particulièrement difficiles dans lesquelles nous avons dû faire ce projet. Cette expérience à distance nous a poussé à beaucoup plus communiquer et cela régulièrement car ne pouvant se voir physiquement il fallait pouvoir quand même communiquer régulièrement si on ne voulait pas que l'un ne parte dans une mauvaise direction et que les autres ne s'en rendent compte que trop tard.

r

9.1.4 Bastien

Ce projet aura été pour moi l'occasion d'apprendre à programmer en langage C, et également de découvrir des outils tels que Gtk+-3.0 et Glade, ainsi que d'approfondir les connaissances que j'avais dans la manipulation d'Emacs et de Linux dans son ensemble. J'ai découvert avec le cahier des charges le principe d'un OCR, je n'avais pas la moindre idée du fonctionnement d'une reconnaissance automatique des caractères. Cela m'a également permis d'en apprendre plus sur les réseaux de neurones artificiels.

J'ai rencontré de nombreuses difficultés, notamment avec l'utilisation de Gtk, car il existe une quantité astronomique de fonctions et méthodes lorsque l'on veut créer une interface graphique. Heureusement, Internet est très riche en documentation et en divers tutoriels, ce qui a permis un

9.2 Conclusions générales

Quel projet plein de rebondissements !

Entre les divers confinements, les incertitudes, les dépassements de soi, ce projet nous aura beaucoup apporté.

Entre autres, il nous aura appris qu'un travail fini est un travail qui est réalisé en équipe. Nos difficultés, accentuées par le confinement, ont été celles de la mise en relation du travail des membres du groupe.

Comme fin mot de cette histoire qui restera mémorable, nous vous laissons sur une citation de Steve Jobs :

"Les meilleures choses qui arrivent dans le monde de l'entreprise ne sont pas le résultat du travail d'un seul homme. C'est le travail de toute une équipe."

Merci.