Rapport vacance de décembre-07/01/2019 Antoine

Vacances :

J'ai décidé pendant les vacances de décembre de développer la partie code de notre projet. Je devais donc être capable de reproduire sur une image numérique des phrases manuscrites identiques à celles de Raphael mais toutes générer aléatoirement.

La première étape fût donc de récupérer des pages d'écritures. On a commencé par 80 pages de cours de maths.

Initialement nous voulions utiliser la base de donnée IAM afin de reconnaitre des mots et de les analyser, seulement on n'avait pas pensé que la langue dans laquelle était la base de donnée allait nous l'empêcher. En effet en utilisant IAM on aurait pu lire directement le mot sans lire les lettres une par une (c'est ce que fait un humain) cependant on aurait pu lire des mots anglais mais Raphael n'est pas anglais

Nota : La base de l'IAM est une base donnée regroupant 115 000 mots, pouvant être séparer en lettres. En fait on dispose IAM d'une grande quantité de fichier XML regroupant des lignes de textes, séparés en mots puis en lettre. On dispose aussi d'une multitude de suite de points (coordonnées) qui sont des séquences pour écrire de la lettre a lettre

Lien : *http://www.fki.inf.unibe.ch/databases/iam-handwriting-database*

Il a donc fallu trouver une autre méthode

J'ai donc créer un algorithme séparant chaque mot de l'image, afin de pouvoir travailler mot à mot. Cela n'a pas été extrêmement difficile, en une journée c'était fait.

Un gros problème (*non résolu*) fût de splitter chaque lettre du mot en petites images, évidement lorsque la personne n'écrit pas en cursive c'est facile, mais Raphael fait un mélange des deux et n'a pas une écriture stable et bien définie.

J'ai donc tenter d'analyser la densité de pixel :

Cette courbe représente la densité de pixel blanc par rapport à la position horizontal dans l'image. Comme vous pouvez le voir on peut rien en déduire.

L'analyse du gradient de cette courbe ne mène aussi à rien

Deuxième technique : détection de formes :

(Ne mène aussi à rien)

DONC :

Les diverses techniques employés (même en les combinant, en calculant la moyenne de la densité de pixel etc…) ne mènent à rien.

Après avoir longuement réfléchis je me suis ne dit pourquoi pas utiliser la détection de lettre par réseaux de neurones en balayant chaque mot directement, sans faire de la lettre à lettre.

J'ai donc entrainer un réseau de neurones (code dans « train.py ») permettant de reconnaitre des chiffres, lettres minuscule et lettres majuscules. En l'entraînant environ 1h, le résultat stagné a 85-86% pourcent de bonnes réponses sur la base de données de l'Emnist.

Cependant sur l'écriture de Raphael, les réseaux de neurones se trompait dans environ 40% des cas, notamment au niveaux des « 1 » qui étaient toujours détectés comme des « A ».

Il a donc fallu « nourrir » le réseau de neurones avec un nouveau jeu d'entrainement, celui de Raphael. J'ai donc demander à Raphael de beaucoup écrire, notamment des lettres uniques, afin des créer de nouvelles données catégorisées dans des fichiers.

Après avoir accumuler et triées plus de 2500 caractères, les tests s'avèrent plus généreux avec environ 8% erreurs.

Cependant j'avais oublié quelque chose, lorsque le balayage allait se faire, même si le réseau de neurones prenait un mot en entier, il allait choisit parmi les classes prédéfinies, et donc par exemple, si on lui donne en entrée le mot « elle » il donnerai en sorti une lettre ou un chiffre, ce qui ne serait pas bon du tout ! Il a donc fallu rajouter la classe « autre », si le réseau de neurone détecté autre chose qu'un caractère, il choisissait « autre ».

J'ai donc essayé d'utiliser le balayage afin de détecter chaque lettres d'un mot, mais le résultat s'est avérer bien mauvais, et je n'avoue ne pas savoir pourquoi.

Peut-être mon réseau de neurones est trop petit, ou bien peut-être faut-il une quantité folle de données pour d'approcher d'un résultat correct

Le plan était initialement d'avoir ses mots référencés afin de les infecter dans un autre réseau de neurones pouvant apprendre à générer du texte, cependant j'ai appris qu'il fallait écrire sur un tableau interactif afin d'enregistrer une succession de points lors de l'écriture et qu'une image ne sert à rien, on est pas capable d'apprendre à écrire à une machine avec une image, mais seulement avec une série de points.

Afin de contourner l'impasse j'ai voulu utiliser le premier réseau de neurones afin d'obtenir l'espacement entre les lettres.

Par exemple combien de pixel entre un « p » et un « u », un « n » et un « e ».

Le code contenu dans « character-extrator.py » était là pour ça mais comme dit précédemment le réseau de neurones n'est pas assez performant.

Mais je me suis contenter de son résultat. Les lettres étaient à peu près bien détecter (le mot était bien découpé), mais les prédictions étaient souvent fausses.

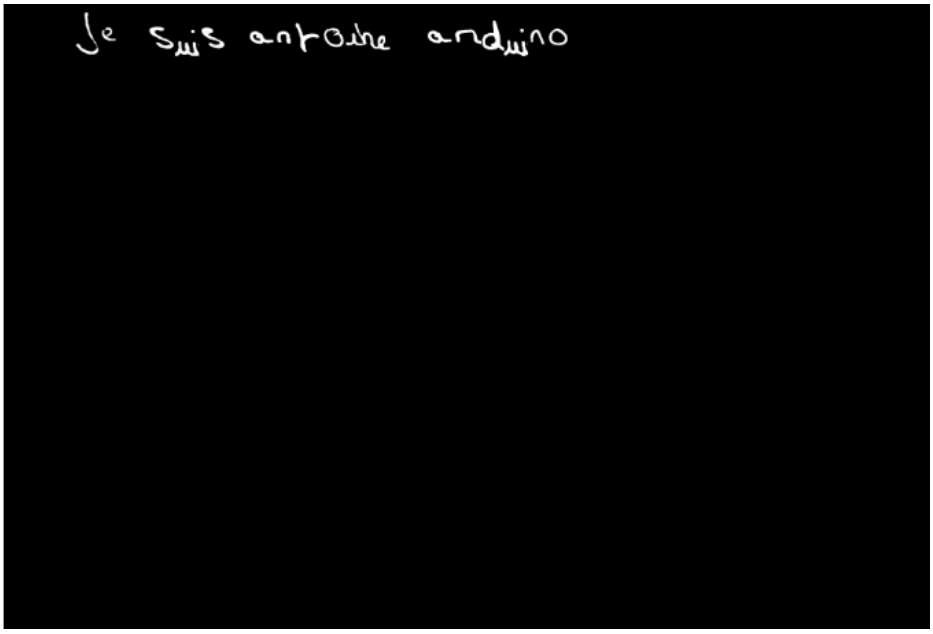
J'ai donc récupérer les mots à peu près bien découpé et ensuite corriger les erreurs de prédictions (n au lieu de r pas exemple).
Je pense que avec 10 000 images on aurait pu avoir un bon taux de prédictions car il faut savoir que les lettres ou on avait donné les plus d'images (ex : « e »), les détections étaient souvent juste.

Maintenant qu'on a nos lettres et mots triés, il nous reste à faire le code permettant de construire la page et de communiquer avec l'imprimante.

J'ai commencé ce code …

07/01/2019 :

Lors de cette séance, j'ai finaliser la version 1 du code permettant de générer des pages de textes. Sans plus attendre voici juste quelques mots écrits par mon algorithme :



Je compte améliorer les liaisons entre caractères, et augmenter considérablement le texte écrit (même si je le peux déjà mais le résultat n'est pas terrible). Cependant il m'est presque impossible de parfaitement aligner les lettres entres elles. Je ne pense pas pouvoir plus me rapprocher de l'écriture de Raphael, ce qui est fort dommage car ça ne s'en rapproche pas trop.

Il faut savoir que pour arriver à ces résultats j'utilise deux choses, des parties de mots toutes référencés et découper, grâce (en partie) a mon réseau de neurones, et plus majoritairement à ma patiente à corriger les fautes de mon réseau de neurones lors du découpage.

J'ai donc ajouter certains fichiers contenant du code, en voici la liste avec une petite description pour chacun d'entre eux :

- **Character-extractor.py** *permet à partir d'un ensemble de mots, d'extraire le plus possible de lettres, ou d'ensembles de lettres*
- **Data.py** *un petit fichier juste pour mettre quelques variables*
- **Images_utils.py** *ensemble de fonctions nécessaires aux autres modules*
- **Line.py** *fichier dans lequel se trouve la classe Line, représentant une ligne d'écriture*
- **Page.py** *fichier dans lequel se trouve la classe Page, représentant une page d'écriture*
- **Printing.py** *fichier initialiseur, lance la construction des pages, la transformation en coordonnées pour l'imprimante et la communication avec la carte arduino*
- **Train.py** *fichier permettant d'entrainer notre réseau de neurone avec la base de donnée Emnist*
- **Train2.py** *fichier permettant d'entrainer notre réseau de neurones avec les données de Raphael*
- **Word.py** *fichier dans lequel se trouve la classe mot (gère sa création, etc …)*
- **Word-extractor.py** *ce code permettant à partir d'un ensemble de pages, d'extraire chaque mot sous forme de petites images*

Ces fichiers se trouvent dans : **src/neuronal-network-handwritting/src/**

Pour plus de précisions aux niveaux du code, vous pouvez directement aller voir dans les fichiers, ils sont tous commentés.

Je dois impérativement finaliser la première version de mon code qui comporte quelques bug (mais qui fonctionne quand même) avant la semaine prochaine. Nous comptons avec Raphael finir la structure de l'imprimante pour la semaine prochaine et bien avancer la partie électronique.

Important : Ce rapport contient beaucoup de choses non expliqués, mais j'ai passé énormément temps à faire tout ça (plus de 60 heures de travail en tout), mais ce n'est pas toujours facile d'écrire au fur et à mesure son rapport, surtout quand ça ne marche pas (ce qui est arrivé très souvent), mais encore moins facile de s'en remémorer ☹.