Groupe : Référent : Francesco PEZZICOLI
Florian Pineau
Julien Le Goff
Baptiste Samoyault
Austé Minkauskaite

Projet : Reconnaissance optique de caractères

Introduction :
Dans le contexte du cours “Introduction à l'Apprentissage Statistique” (IAS) nous devons réaliser un projet de machine learning en groupe. Nous avons choisi de travailler sur la reconnaissance de lettres/mots manuscrits (ou *optical character recognition* OCR en anglais). Ceci est un sujet ayant beaucoup d’applications dans la vie de tous les jours : reconnaissance de plaques d'immatriculation par les caméras, numérisation de documents imprimés, etc.

Plan :
I) Récapitulatif du dataset ainsi que la description de la tâche à faire
II) Etape 0 : explication des choix faits au préalable
III) Etape 1 : extraction du dataset
IV) Etape 2 : data augmentation
V) Etape 3 : le modèle
VI) Etape 4a : séparation de mots en caractères
VII) Etape 4b : l’interface graphique pour tester notre écriture

I) Récapitulatif du dataset ainsi que la description de la tâche à faire

Dataset :

Pour le dataset nous avons choisi celui appelé EMNIST, il se démarque par la présence de près de 800 000 photos (28 par 28 pixels) de lettres de l’alphabet, ainsi que de chiffres manuscrits.

Le lien du dataset EMNIST :
https://www.nist.gov/itl/products-and-services/emnist-dataset
https://www.kaggle.com/datasets/crawford/emnist

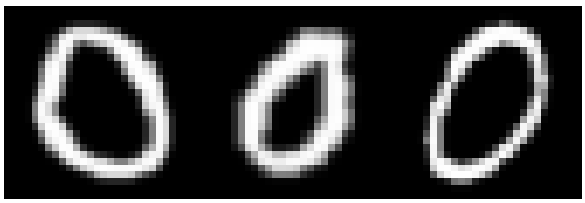
Description de la tâche :

-> Dans un premier temps nous allons écrire et entraîner un algorithme de reconnaissance de caractères simples (lettres ou chiffres). Dans notre cas, une méthode d'apprentissage de type réseau de neurones artificiels (neural network) serait adaptée à la tâche. Pour le tester, nous allons faire un programme de type “Paint” pour pouvoir écrire nous même des caractères qui seront ensuite interprétés par notre algorithme.

-> Dans un deuxième temps, nous ferons la reconnaissance de mots. Il faudra faire un algorithme capable de détecter et séparer chaque mot en caractères distincts et ensuite le modèle de reconnaissance pourra être utilisé pour identifier la succession de caractères.

II) Etape 0 : explication des choix faits au préalable

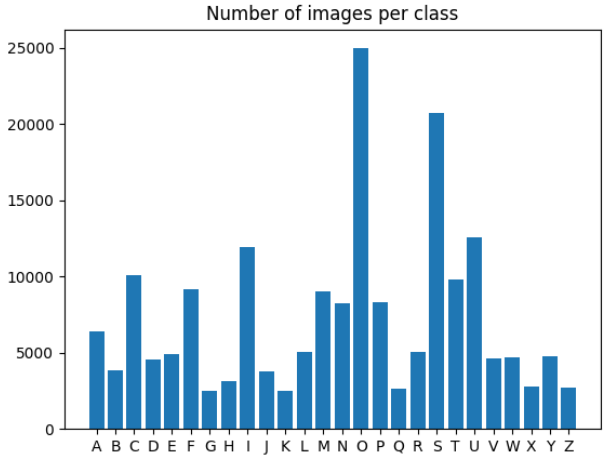
Le dataset EMNIST nous met à disposition une grande quantité d’images de lettres et chiffres manuscrits, individuellement recadrées et catégorisées. Pour restreindre le domaine de notre projet, nous avons choisi de travailler sur les lettres manuscrites et uniquement sur les majuscules. En effet, il est parfois difficile de juger si une lettre est majuscule ou minuscule (cf l’exemple des O/o ci-dessous). Un zéro et un O sont parfois également difficiles à distinguer, surtout lorsqu’ils sont manuscrits. Nous avons donc extrait que les 26 lettres majuscules du dataset EMNIST.



Exemples d’un O majuscule, o minuscule et un 0 (zéro) respectivement

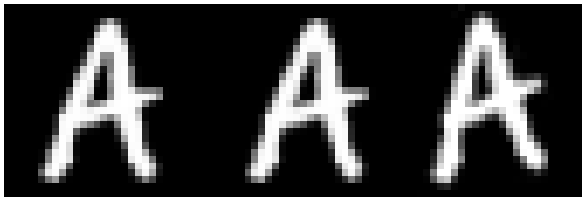
III) Etape 1 : extraction du dataset

Après avoir téléchargé le dossier, nous avons extrait les lettres selon chaque catégorie dans un fichier séparé. Ensuite, pour chaque image dans chaque fichier, nous l'avons ajouté à notre dataset créé pour l'occasion avec le nom correspondant à la lettre, la bonne taille et couleur. Finalement, le dataset est mélangé afin de ne pas avoir des "blocs" de la même lettre.
Nous voyons cependant que la distribution des lettres n'est pas égale :

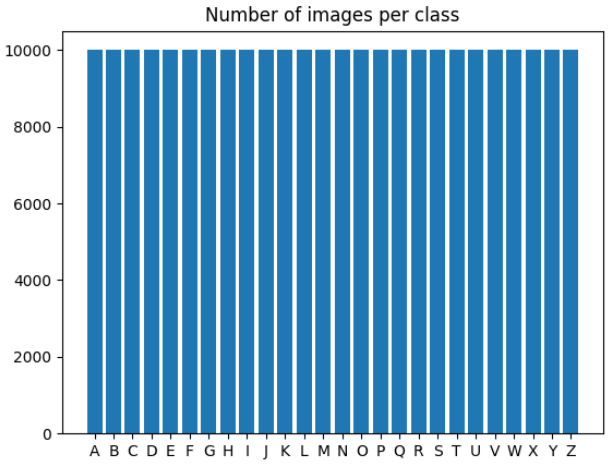


IV) Etape 2 : data augmentation

Sur la figure d'avant nous voyons clairement que certaines lettres sont surreprésentées par rapport aux autres. Cela n'est pas optimal car *idéalement* pour que l'algorithme fonctionne bien, il faudrait avoir une distribution similaire de chaque classe.
Voici une explication simple en quoi une surabondance d'une certaine classe n'est pas optimale. Imaginons que nous n'avons que 2 classes, la première classe "a" représente 99% du dataset et la deuxième "b" le reste. Un algorithme entraîné sur ce dataset pourrait simplement toujours rendre comme réponse "a" et avoir un taux de succès de 99%. Pourtant cet algorithme n'est pas correct car il aura un taux d'échec de 100% pour la classe "b". Il faudrait avoir un jeu de données équilibré pour éviter ce type d'erreurs de classification.
Pour équilibrer le jeu de données nous avons fait de la "data augmentation". Plus simplement, nous avons créé d'autres images à partir des images existantes en faisant des translations et des rotations.



Exemple d'un A transposé vers la droite et légèrement tourné vers la gauche (observez que les modifications sont censées être très légères pour éviter de trop s'éloigner des images de base)

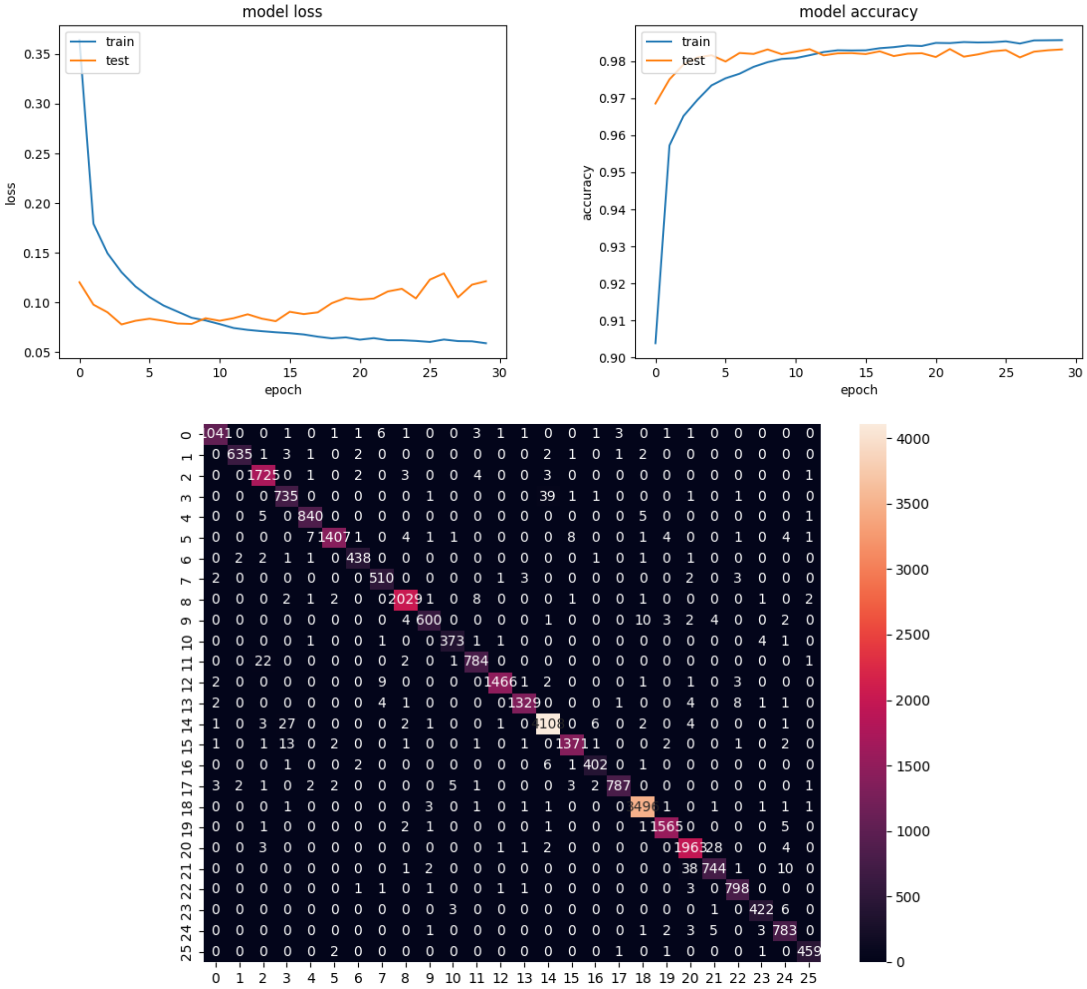


Nombre d'images par classe après la data augmentation

V) Etape 3 : le modèle

Utiliser un réseau de neurones (neural network) pour la classification de chiffres avec le dataset MNIST est une méthode classique du machine learning. Nous avons donc également choisi d'utiliser un neural network pour notre projet. C'est un algorithme d'apprentissage profond et puissant qui est capable de traiter des données complexes et d'effectuer des tâches de classification, de prédiction et de reconnaissance de motifs. L'algorithme de réseau de neurones apprend à partir des données en ajustant les poids et les biais des neurones dans les différentes couches du réseau. Les données sont présentées au réseau sous forme de vecteurs d'entrée, qui sont transformés par les poids et les biais des neurones de la première couche pour produire une sortie. Comme l'explication de cet algorithme n'est pas l'objet de ce rapport, nous n'allons pas fournir ici une explication plus précise du fonctionnement d'un réseau de neurones. Comme les neural networks ne sont pas traités en cours, une explication simplifiée et moins détaillée sera donnée durant la présentation orale pour faciliter la compréhension des interlocuteurs.

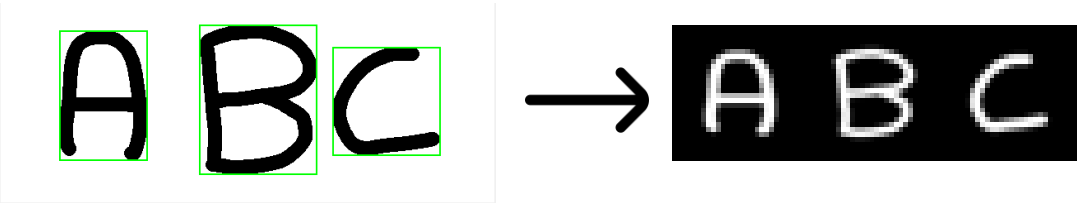
Après avoir fait des recherches et lu la documentation sur les types de fonctions d'activation et les types de couches nous avons finalisé notre modèle. Notre modèle comporte trois couches de convolution avec des filtres de tailles (3, 3) et une activation ReLU, suivies par une normalisation par lots et un pooling max. Après la convolution, le modèle est aplati et passe par deux couches denses avec des activations ReLU et des abandons (dropout) pour éviter le surapprentissage. Finalement, une couche dense de sortie utilise l'activation softmax pour classer les images en 26 catégories différentes.



Graphes de loss, accuracy et confusion matrix de notre modèle

VI) Etape 4a : séparation de mots en caractères

Ayant un modèle qui fonctionne bien sur un caractère à la fois, nous avons voulu être capable l'appliquer à plusieurs caractères à la fois. Cela est simple à accomplir avec OpenCV. Voici un exemple :



VII) Etape 4b : l'interface graphique pour tester notre écriture

Pour pouvoir tester nous même notre modèle nous avons créé une interface graphique :