PARTIE DATA 1

EXPLORATION ET TRAITEMENT DES DONNES

Dans cette partie nous allons voir l'analyse et le travail réalisé sur le Dataset en expliquant les modification apporté ainsi son manipulation sur le code.

1.1 Jeux des données

1.1.1 La collection des données

Nous avons commencé d'abord notre collecte des données sur le site `https://www.gbif.org/fr/` mais nous avons rencontré des difficultés lors de la collecte des donnes en terme de la qualité des images ainsi le nombres insuffisants d'image pour chaque classe de plantes, ce qui ne nous permet pas de générer un model parfait de reconnaissance d'image. Pour cela nous avons cherché sur d'autre site, et nous avons trouvé le site `https://www.kaggle.com/abdallahalidev/plantvillage-dataset` qui contient des jeux de données partagées par des éditeurs d'ensemble de données et qui sont plus faciles à utiliser et bien organisés.

1.1.2 Exploration des données

Notre jeux de données se compose uniquement de fleurs de 5 classe nommés comme suivant [3] :
1-Daisy
2-Dandelion
3-Rose
4-Sunflower
5-Tulip

Detail sur les jeux de données :

Pour les images des classes que nous avons choisi à entraîné, nous trouverons qu'il y a des informations perturbatrices par exemple, il y a quelque images qui présentes des insectes perturbants l'unicité de la fleur et ces photo ne représentants qu'une partie du jeux donnes, a

part ca notre jeux de donnes contient des images qui sont bien choisi pour générer un model de classification d'image.

Figure .. : Présence d'insectes perturbants l'unicité de la fleur

Modification apporté :

Dans notre jeux de données que nous avons collecté sur le site Kaggle, on trouve qui il y a des classes volumineuses que d'autres avec 984 pour le maximum (Tulip) et 733 pour le minimum (Sunflower), ainsi les photos ont des tailles très variés par exemple une majorité ont 200x200.Il était donc important d'ajuster notre Dataset afin d'avoir un standard avec lequel travailler et générer un model de meilleur performance.

Dans un premier temps Nous avons redimensionner tous les photos des classes à une dimension de 50x50 car il permet de conserver une bonne visualisation tout en nous permettant de travailler sur une matrice carrée bien plus petite, et donc de réduire le temps des calculs.D'autre part nous avons defini le meme nombre d'image pour chaque classe pour avoir un jeux de données uniforme.

Pour cela nous avons générer le fichier *resize.py* en python qui nous permet de modifier notre jeux de données utilisé pour générer le model.

1.1.3 Description des classes

Class Data

(a). Opencv

OpenCV (pour Open Computer Vision) est une bibliothèque graphique libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel.OpenCV C++ est livré avec cet incroyable conteneur d'images Mat qui gère tout pour nous [1].

(a). Descriptif

La classe Data a pour but de stocker des images comprenant 50×50 pixels dans un vecteurs en transformant l'images RVB à 3 canaux à 1 canal.

Les valeurs de pixels sont souvent des nombres entiers non signés compris entre 0 et 255. Bien que ces valeurs de pixels puissent être présentées directement aux modèles de réseaux de neurones dans leur format brut, cela peut entraîner des problèmes lors de l'apprentissage qui sera plus lent que prévu du modèle. Donc il est avantageux de normaliser les valeurs de pixels et de changer la plage $0 - 255$, Dans notre cas nous avons mise à l'echelle les valeurs de pixels dans la plage $[-0.5, 0.5]$ [2].

L'image est une matrice représentée par la classe **cv : :Mat**. Chaque élément de la matrice représente un pixel. Pour les images en niveaux de gris, l'élément de matrice est représenté par un nombre de 8 bits non signé (0 à 255). Pour une image couleur au format RVB, il existe 3 de ces nombres, un pour chaque composante de couleur. La fonction utilisé pour lire une image dans opencv est :

imread() : Cette fonction permet de lire les images et prend les 2 arguments suivants :

— *nom de fichier* : L'adresse complète de l'image à charger est de type string.

— *flag* : C'est un argument optionnel et détermine le mode dans lequel l'image est lue.Pour notre code nous avons utilisé le mode `CV_LOAD_IMAGE_COLOR` pour télécharger l'image en couleur [1].

Les images de notre jeux de données sont des images RVB à 3 canaux. Nous avons transformé les images de 3 canaux à 1 canal faciliter leur travail. Et pour ce faire nous avons utilisé cv : :vect3b dans opencv pour stocker les couleurs des pixels.

Module Direction_file

le but du module Direction est de recuperer les fichiers(notre image) et son label. Le label indique à quelle famille il appartient. C'est cette famille que devra reconnaitre plus tard le réseau de neurones. Le premier chiffre du nom est le label de l'image. Par exemple : *"0img_0.jpg"* représente la classe rose.

Il existe 5 types de fleurs :**Tulip** (label 0), **Daisy** (label 1),**Dandelion** (label 2), **Sunflower** (label 3),**Rose** (label 4). Les images de ces fleurs se trouvent dans le repertoire DataSet, il y a 3500 images.

BIBLIOGRAPHIE

[1] Jeux de donnes, https://www.kaggle.com/apollo2506/flowers-recognition-dataset

[2] "OpenCV" , https://fr.acervolima.com/lire-et-afficher-une-image-dans-opencv-en-ut https://fr.wikipedia.org/wiki/OpenCV

[3] "Normalisation des pixels", https://machinelearningmastery.com/how-to-manually-scale-image-pixel-data-for-deep-learning/