

Classification des Images à l'aide du Conventional Neural Network CNN:

1.1 DataBase:

La base de données citée dans [2] est riche d'images de plantes provenant de différentes classes. Prenant un cas de classe illustrée dans la figure 1.



Figure 1: Dilsea carnosa (Schmidel) Kuntze [2].

L'objectif de cet algorithme est de construire un classificateur d'image capable de déterminer les plantes à partir d'une photo donnée. On peut considérer par titre d'exemple 10 classes de plantes de [2]:

- Saxifraga cotyledon* L. : <https://www.gbif.org/occurrence/3013500444>
- Polypodium Vuglare* L. : <https://www.gbif.org/occurrence/3013501303>
- Thuidium tamariscinum* W.P.Schimper, 1852: <https://www.gbif.org/occurrence/3013504463>
- Dilsea Carnosa* Schmidel Kuntze : <https://www.gbif.org/occurrence/3013508598>
- Delesseria Sanguinea* Hudson J.V.Lamouroux: <https://www.gbif.org/occurrence/3013511508>
- Anomodon Longifolius* C.J. Hartman 1838 : <https://www.gbif.org/occurrence/3013535464>
- Kindbergia praelonga* Hedw Ochyra : <https://www.gbif.org/occurrence/3013585373>
- Saxifraga Cotyledon* L. : <https://www.gbif.org/occurrence/3013600364>
- Fraxinus excelsior* L. : <https://www.gbif.org/occurrence/3013679581>
- Hypericum tetrapetalum* Lam. : <https://www.gbif.org/occurrence/3017940855>

1.2 Réseau de Neurones:

L'algorithme doit contenir des entrées, des training data utilisées pour entraîner l' algorithme à connaître la correspondance des images à des classes spécifiques et des test data pour tester notre algorithme à la fin.

Les **training data** contiennent des images multiples de chaque classe. La précision dépend du nombre d'images. En augmentant ces derniers, la précision augmente mais aussi le temps de simulation augmente.

Les **test data** contiennent des images variées et mixtes des différentes classes. Notre but final est que l'algorithme arrive à préciser la classe de chaque image donnée. Donc la sortie doit être la sortie désirée. Par conséquent, on va chercher à minimiser l'erreur entre la sortie réelle et la sortie désirée.

Dans [3], la classification d'image est faite selon le nombre de pixels p. Ce dernier doit être égal à la longueur multipliée par la largeur de l'image.

Prenant par exemple le cas de la figure 1. Nous avons 10 classes qui vont être considérées comme des filtres f. La sortie Y obtenu pour une image est [3]:

$$resultat\ Y = \sum_{k=1}^{1788864} p_k \times f_k = P \times F$$

avec:

$$P = (p_1 p_2 \dots p_{1788864})$$

$$F = Transpose(f_1 f_2 \dots f_{1788864})$$

Comme nous avons 10 classes (filtres) donc 10 sorties, on peut intégrer les classes dans une matrice 1788864*10, tel que:

$$(p_1 p_2 \dots p_{1788864}) \times (f_{i,j}) = (y_1 y_2 \dots y_{10})$$

avec $i = 1 : 1788864$ et $j = 1 : 10$

Ainsi, le réseau de neurones schématisé dans la figure 2 de la plante Dilsea carnosa (Schmidel) Kuntze a comme entrée le nombre de pixels de l'image. La sortie va être une des 10 classes.

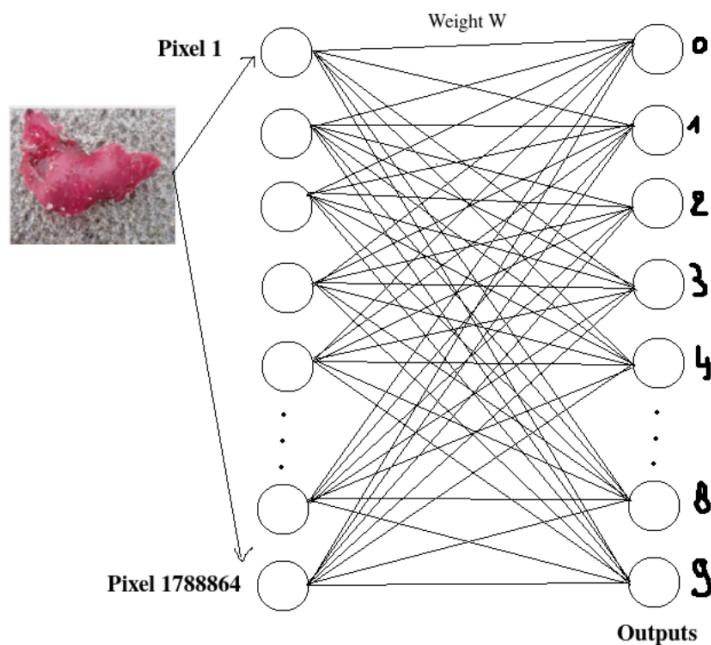


Figure 2: Réseau de Neurones, image d'entrée = Dilsea carnosa (Schmidel) Kuntze [2].

Problématique: La taille de l'image est relativement grande, d'où le nombre important de pixels (plus de mémoire, plus de temps de traitement et d'exécution). Il faut donc penser à un redimensionnement des images utilisées.

Dans [4], on a trouvé un exemple de code en Python de la classification des plantes à l'aide de CNN. LA base de données des images utilisées dans le test et le train est donnée dans [5]. 12 espèces de plantes sont utilisées dans ce cas avec un total de 960 images.

Les étapes de cet algorithme sont les suivantes [4]:

- L'importation de bibliothèques (numpy, pandas, matlib, etc),
- Obtention des données et redimensionnement des images (cela répond au problème posée),
- Nettoyage de l'image et suppression de l'arrière- plan,

Remarque: Cette étape de nettoyage et suppression de l'arrière- plan est importante dans notre cas. Dans la figure 3, la partie importante est la reconnaissance de la plante verte de l'image et non pas l'arrière-plan, qui va consommer de ressources de computation.



Figure 3: Saxifraga cotyledon L. [2].

- Conversion des noms en chiffres: Les noms sont des chaînes de caractères, difficiles à traiter, et vont être converti en classification binaire. Dans notre cas de 10 classes par exemple, la classification va être représentée par un tableau de 10 nombres selon la condition: 0 si l'espèce n'est pas détectée, 1 si l'espèce est détectée. Par exemple, si l'espace Dilsea Carnosa Schmidel Kuntze de la figure 1 est détectée (correspond à la classe 4), notre tableau va être [0,0,0,1,0,0,0,0,0,0]
- Définition du modèle et séparation des jeux de données: Dans cette étape, on divise l'ensemble de données d'entraînement pour la validation (test). On peut diviser les données en données de test et données de validation. (Dans [4], 10 % des données totales sont utilisées comme données de test et les 90 % restants comme données d'entraînement)
- Définition du réseau de neurones conventionnels: Dans ce problème, nous allons utiliser un réseau de neurones convolutifs [4]. Ce réseau de neurones prendra des **images en entrée** et fournira la **sortie finale** en tant que **valeur d'espèce**. Dans [4], 4 couches de convolution et 3 couches entièrement connectées sont utilisées aléatoirement.
- Adaptation du CNN aux jeux de données: Le but est que modèle apprenne de l'ensemble de données d'entraînement et les poids soient mis à jour. (Challenges: réduire le taux d'apprentissage, trouver les meilleurs poids pour le modèle et enregistrer ces poids calculés afin que nous puissions les utiliser davantage pour tester et obtenir des prédictions.)

- Vérification des performances du modèle sur les données, Matrice de Confusion: bon moyen pour analyser les erreurs dans le modèle.
- Obtention des prédictions: prédictions sur l'ensemble de données de test à l'aide du modèle entraîné.

Références:

- [2] Lien, https://www.gbif.org/occurrence/gallery?taxon_key=6
- [3] Nicolas Schmid, “Reconnaissance d’images par réseau de neurones”, Lycée cantonal de Porrentruy, novembre 2018.
- [4] Lien,
https://www.analyticsvidhya.com/blog/2021/05/plant-seedlings-classification-using-cnn-with-python-code/?fbclid=IwAR15rBKrF1GQfy_QSKboG7DMZ1SAXg_tgorIVLBqg26lW3qqAwTqAauf1Eg
- [5] Lien, <https://www.kaggle.com/c/plant-seedlings-classification/data>