
Cahier de Charge du Projet:

# Reconnaissance des plantes à l'aide d'un réseau de Neurones

Encadrant: **Hugo Bolloré**

Date: 27/12/2021

**Problématique**
Le but de ce projet est dans un premier temps de coder un réseau de Neurones en $C++$ pour la reconnaissance des fleurs existantes dans une base de données [2].

**Étapes du Project**
1. Base de données:
La base de données regroupe un ensemble de cinq classes d'un total de 3665 fleurs à partir de [2].
Elles sont regroupées comme suit:

- Daisy: 733 images,
- Dandelion: 733 images,
- Rose: 733 images,
- Sunflower: 733 images,
- Tulip: 733 images.

A titre d'exemple, la figure (1) illustre une fleur de classe Daisy.

Figure 1: Fleur de type Daisy.

Trois fichiers CSV sont ajoutés à l'ensemble de données. Ils incluent:

- L'ensemble d'images d'entraînement: 70%,
- L'ensemble d'images de validation: 20%,
- L'ensemble d'images de test: 10%.

Le fichier CSV est organisé de la façon suivante:

| Data Number | File Name | Label | Class Name |
|---|---|---|---|

Table 1: Organisation du fichier CSV.

avec:

- **Data Number:** contient le numéro de l'image dans le fichier CSV (à partir de 0),
- **File Name:** contient le nom de l'image sous la forme *classe/nom_image.png*,
- **Label:** contient le numéro de classe dans le fichier (par exemple; 0 indique la première classe dans le fichier CSV),
- **Class Name:** contient le nom du classe parmi les 5 classes.

2. Structure du Code:
**Préliminaire:**
Architecture du CNN: On considère l'image de la figure (1). Le réseaux de neurones évolutif est composé des couches suivantes :

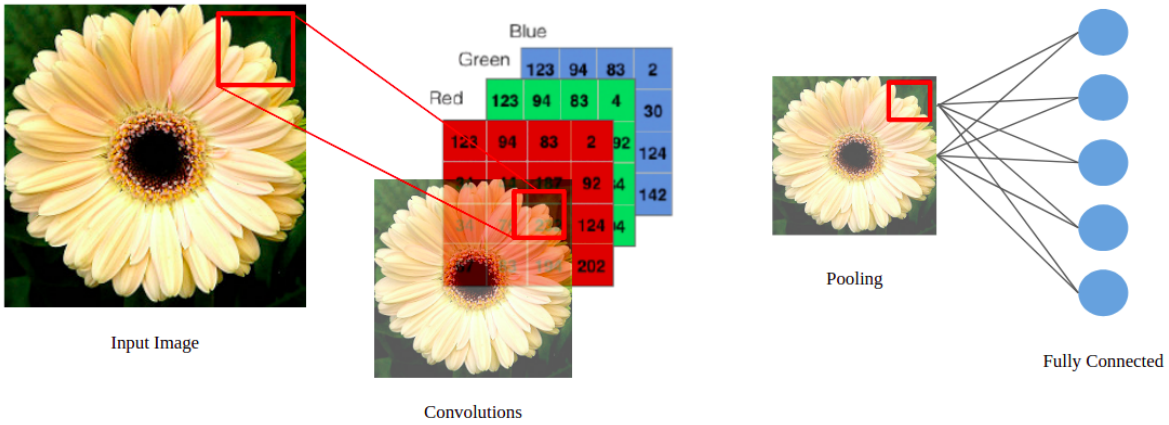


Figure 2: Le réseau de neurones convolutif considéré.

Dans ce projet, nous allons créer un modèle de deep learning capable de reconnaître l'image de fleur donnée comme entrée. Comme nous avons cinq classes, le vecteur de sortie affiche la classe de l'image donnée. La figure (3) illustre que la classe choisie prend le chiffre 1 dans le vecteur de sortie, tandis que les autres classes prennent des 0.

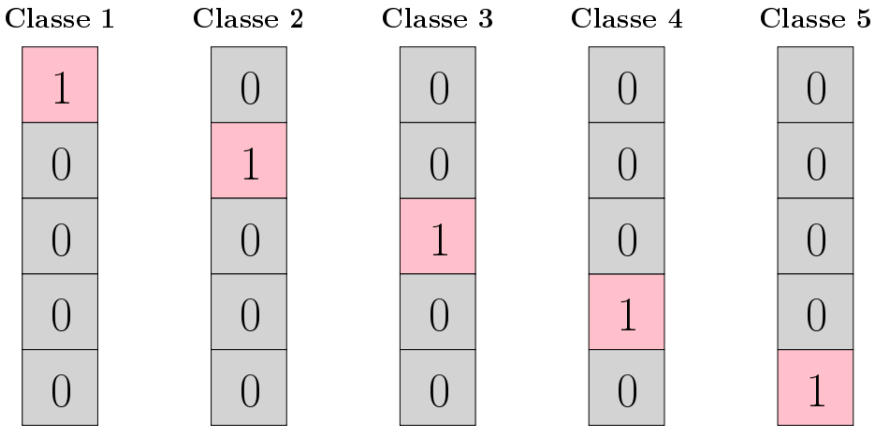


Figure 3: Vecteurs de sortie dépendants de la classe choisie.

Les images de la base de données sont en trois dimensions $240 \times 240 \times 3$, qui représentent:

- **Dimension 1:** 240 correspond à la hauteur de l'image exprimée en pixels,
- **Dimension 2:** 240 correspond à la largeur de l'image exprimée en pixels,
- **Dimension 3:** 3 correspond aux intensités de couleurs rouge, vert et bleu (RGB), nécessaires à la formation des couleurs du pixel, comme presenté dans la figure 4.

  Ainsi, chaque couleur sera une combinaison d'intensité de ces trois couleurs [3].

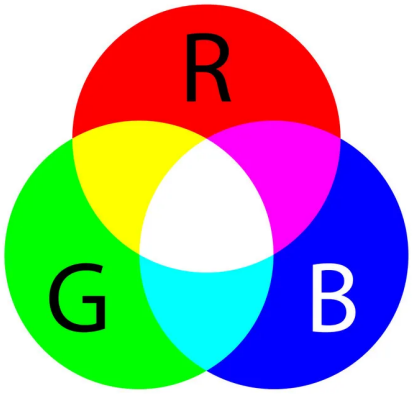


Figure 4: Modèle de couleurs RGB [4].

La couche de convolution consiste à appliquer un filtre (composé d'un ensemble de poids) sur les images afin d'extraire ses caractéristiques, comme illustré dans la figure (5). Ici, on multiplie chaque poids par un pixel de l'image et on somme pour avoir une valeur. Puis, on avance d'un *stride* spécifique pour obtenir une nouvelle représentation de l'image. La fonction d'activation choisie dans notre projet est celle de Softmax.

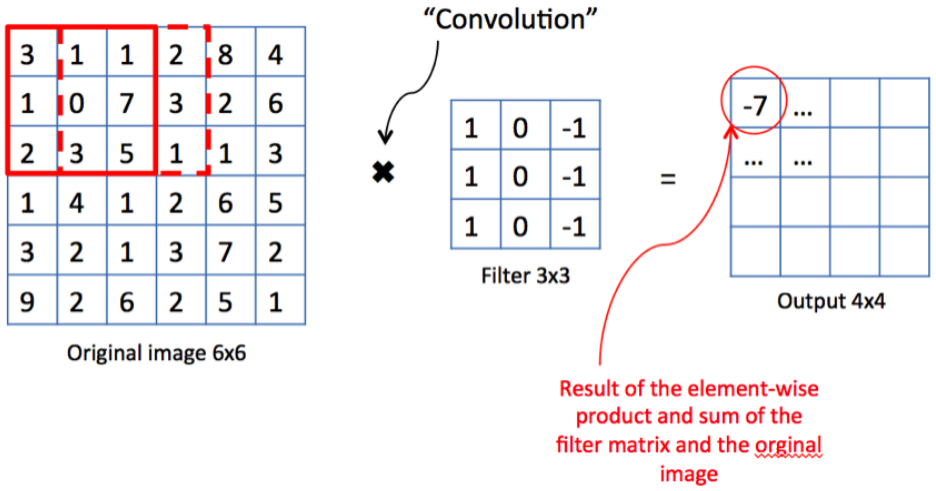


Figure 5: Couche de convolution [1].

Afin de réduire la taille de données, on ajoute une couche de max-pooling.Elle consiste à choisir le nombre maximum d'une sous matrice à partir d'une matrice de pixels, comme illustré dans la figure (6).

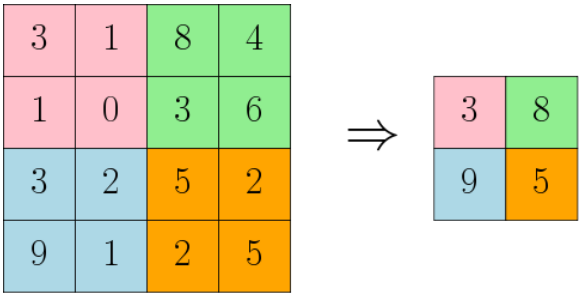


Figure 6: Principe de Max-pooling.

L'étape suivante consiste à extraire les informations a partir de l'image dans un vecteur 1D comme montré dans la figure 7. C'est le principe de *Flattening* des images.

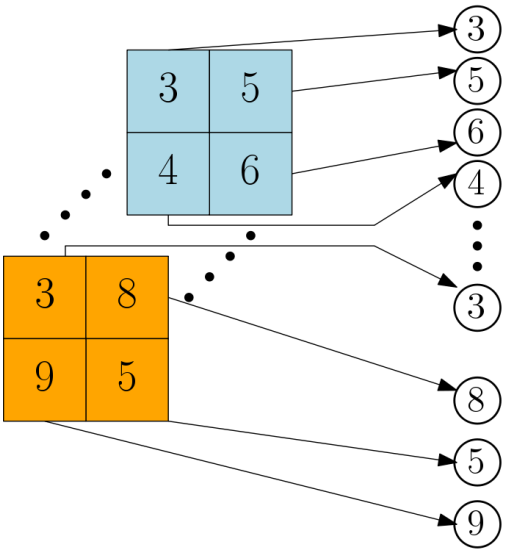


Figure 7: Vecteurs contenant les caractéristiques des images (Flattening).

Pour finir, on utilise des "*Fully connected layers*", ou couches entièrement connectées pour prédire la classe de sortie associée à l'image (sera expliqué dans le rapport).

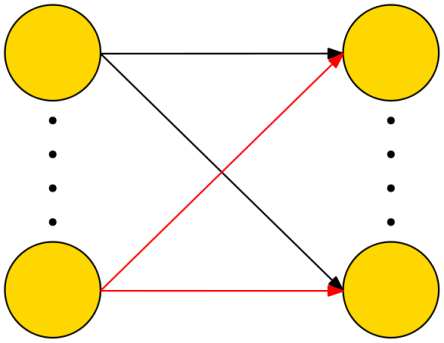

Figure 8: Principe des couches entièrement connectées (Dense Layer).

Il est à noter que les données seront normalisées. Etant donné que l'intensité correspondante à une image est entre 0 et 255, on souhaite changer l'échelle de données en le normalisant (en divisant par 255). Ainsi, on obtient un nouveau échelle entre 0 et 1.

L'entraînement du modèle est possible en définissant la précision "accuracy" du modèle, consideré comme indicateur de preformance du code, pour un certain nombre d'itérations.

# References

[1] *"Convolution Neural Network"*. 2020. URL: https://www.kaggle.com/general/171197.

[2] *"Flowers Recognition"*. kaggle. 2021. URL: https://www.kaggle.com/.

[3] *"Introduction au convolutional neural network"*. 2020. URL: https://github.com/MorganGautherot/Tuto_MRI_ML/blob/master/tp_1/Introduction%5C%20au%5C%20convolutional%5C%20neural%5C%20network.ipynb.

[4] *"RGB Color Model"*. URL: https://www.hisour.com/rgb-color-model-24867/.