
Université Claude Bernard Lyon 1

L3 Mathématiques Générales et Applications

---

# Reconnaissance d'images par réseau de neuronnes convolutifs

---

# 1 Introduction

Ce projet a pour sujet la reconnaissance d'image par réseaux de neurones convolutifs.
Les algorithmes ont donc été entraînés à déterminer si un champignon est comestible ou non ainsi qu'à identifier des fruits, des légumes, des animaux et des fleurs.

Le site ainsi que les algorithmes de reconnaissance d'images ont été programmés par les étudiants de L3 Mathématiques Générales et Applications de l'UCBL durant le mois de juin 2022. Nous avons eu une semaine pour nous organiser en groupe afin de récolter les données, préparer le rendu ainsi que le programme en le rendant le plus efficace possible.

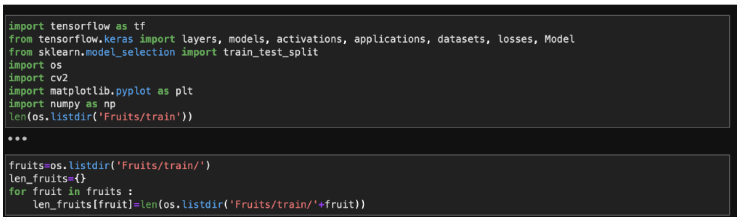
Ce document est dédié aux explications du fonctionnement de l'algorithme sur un exemple simple : la reconnaissance de fruits.

## 2 Présentation de l'algorithme

Nous nous sommes basés sur les travaux de Aditya Khosla, Nityananda Jayadevaprakash,Bangpeng Yao et Li Fei-Fei de l'université de Stanford dans lesquels ils entrainaient un réseau neuronal convolutif à reconnaitre la race d'un chien.

Pour cela, ils ont considérés 120 races différentes et utilisés environ 150 images par race pour entrainer leur algorithme en ayant, au total 20 580 images. Nous nous sommes basés sur le même principe, en utilisant des DataSet libres de droits disponibles sur Kaggle.

Nous avons tout d'abord commencé par faire le code sur une base de données de 33 fruits.



```
import tensorflow as tf
from tensorflow.keras import layers, models, activations, applications, datasets, losses, Model
from sklearn.model_selection import train_test_split
import os
import cv2
import matplotlib.pyplot as plt
import numpy as np
len(os.listdir('Fruits/train'))
```

```
•••
```

```
fruits=os.listdir('Fruits/train/')
len_fruits={}
for fruit in fruits :
    len_fruits[fruit]=len(os.listdir('Fruits/train/'+fruit))
```

FIGURE 1 – liste des bibliothèques utilisées

Nous avons utilisé les bibliothèques ci-dessus et commencé par faire une étude préliminaire de nos données. Pour chaque fruits nous avons considéré entre 400 et 700 images par variétés.

Pour que les données soient exploitables, nous avons dû redimensionner sans couper nos images afin qu'elles soient toutes de la même taille. Ici en 100x100.

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(100, 100, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(33))

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

model.summary()
```

FIGURE 2 – Création des modèles utilisés

Dans un premier temps nous avons utilisé des filtres convolutifs (ici 32) afin de décomposer l'image puis utilisé la fonction MaxPooling() pour réduire la taille de l'image par 2 qui nous permet sur 4 pixels de ne garder que celui ayant la valeur la plus élevée.

Nous recommençons plusieurs fois afin de récupérer les données les plus importantes de l'image.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 98, 98, 32)        896

 max_pooling2d_2 (MaxPooling  (None, 49, 49, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 47, 47, 64)        18496

 max_pooling2d_3 (MaxPooling  (None, 23, 23, 64)       0
 2D)

 conv2d_5 (Conv2D)           (None, 21, 21, 64)        36928

 flatten_1 (Flatten)         (None, 28224)             0

 dense_2 (Dense)             (None, 64)                1806400

 dense_3 (Dense)             (None, 33)                2145

=================================================================
Total params: 1,864,865
Trainable params: 1,864,865
Non-trainable params: 0
```

FIGURE 3 – Sortie de model.summary()

```
history = model.fit(X_train, y_train, epochs=10, batch_size = 20, validation_data=(X_test, y_test))
```

FIGURE 4 – Entrainement et test du model

A chaque image on associe une valeur puis on teste si cette valeur correspond et à quel degrés.

Pour cela le model fit utilise 10 étapes sur un échantillon aléatoire d'images afin d'améliorer la performance de l'algorithme tout en évitant le sur-apprentissage qui pourrait faire perdre de la précision.

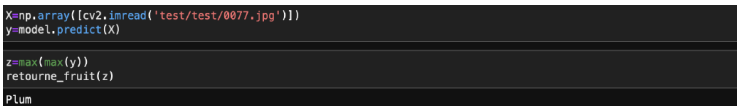



FIGURE 5 – Exemple sur une image aléatoire

Nous avons fourni une image au model et il nous a renvoyé le nom du fruit dont il s'agissait : ici une prune.

La photo fournie était bien celle d'une prune.