TP sur CNN binaire

0/ Collecter la base de données :

https://github.com/akshatg007/Binary-Classification-Using-Image-Recognition/tree/master/dataset

1/librairies :

```
import os
import pandas as pd
import numpy as np
from pathlib import Path
import matplotlib.pyplot as plt
from sklearn.utils import class_weight as cw
from keras.models import load_model
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, BatchNormalization
from keras.layers import Flatten, Dense, Dropout
from keras.callbacks import Callback, ReduceLROnPlateau, EarlyStopping
```

2/ Définir ImageDataGenerator

```
train = ImageDataGenerator(rescale=1/255)
test = ImageDataGenerator(rescale=1/255)

train_dataset = train.flow_from_directory("…"
                                          target_size=(150,150),
                                          batch_size = 32,
                                          class_mode = 'binary')

  test_dataset = test.flow_from_directory("…",
                                          target_size=(150,150),
                                          batch_size =32,
                                          class_mode = 'binary')
```

3/ Définir CNN

Expliquer le reseau de neurone

```
model = keras.Sequential()

# Convolutional layer and maxpool layer 1
model.add(keras.layers.Conv2D(32,(3,3),activation='relu',input_shape=(150,150,3)))
```

```
model.add(keras.layers.MaxPool2D(2,2))

# Convolutional layer and maxpool layer 2
model.add(keras.layers.Conv2D(64,(3,3),activation='relu'))
model.add(keras.layers.MaxPool2D(2,2))

# Convolutional layer and maxpool layer 3
model.add(keras.layers.Conv2D(128,(3,3),activation='relu'))
model.add(keras.layers.MaxPool2D(2,2))

# Convolutional layer and maxpool layer 4
model.add(keras.layers.Conv2D(128,(3,3),activation='relu'))
model.add(keras.layers.MaxPool2D(2,2))

# This layer flattens the resulting image array to 1D array
model.add(keras.layers.Flatten())

# Hidden layer with 512 neurons and Rectified Linear Unit activation function
model.add(keras.layers.Dense(512,activation='relu'))

# Output layer with single neuron which gives 0 for Cat or 1 for Dog
#Here we use sigmoid activation function which makes our model output to lie between
0 and 1
model.add(keras.layers.Dense(1,activation='sigmoid'))
```

4/ Réaliser l’apprentissage
Expliquer les parametres (optimizer, loss, metrics)

```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

5/ Utiliser d’autres métriques (Précision, F1-score)
N’oubliez pas d’utiliser des librairies