07MIAR_Proyecto_Programación - 01 Preparacion del dataset

0. Librerías y funciones auxiliares

```
from google.colab import drive
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage.io import imread, imshow
from skimage.color import rgb2hsv
import json
import pandas as pd
from PIL import Image
import os
from tqdm import tqdm
from tqdm import tqdm
import os
from PIL import Image
import threading
from concurrent.futures import ThreadPoolExecutor, as_completed

def view(img):
    plt.imshow(img, cmap = 'gray')
    plt.axis('off')
    plt.show()

labels=["Loose Silky-bent",
        "Common Chickweed",
        "Scentless Mayweed",
        "Small-flowered Cranesbill",
        "Fat Hen",
        "Charlock",
        "Sugar beet",
        "Cleavers",
        "Black-grass",
        "Shepherds Purse",
        "Common wheat",
        "Maize"]
```

1. Creacion del dataset

1.1. Conexión remota con Google Drive

```
# Conectamos con nuestro Google Drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

1.2. Establecer ruta directa al directorio y lectura de datos

```
# Establezco una ruta absoluta a un directorio existente de mi Google Drive
BASE_FOLDER = "/content/drive/MyDrive/BASE_FOLDER/"
```

1.3. Estandarizacion del dataset a un JSON

```
ruta_directorio = BASE_FOLDER + "my_dataset/train"

diccionario_imagenes = {}


for nombre_clase in os.listdir(ruta_directorio):
    ruta_carpeta_clase = os.path.join(ruta_directorio, nombre_clase)

    if os.path.isdir(ruta_carpeta_clase):
        for nombre_imagen in os.listdir(ruta_carpeta_clase):
            ruta_imagen = os.path.join(ruta_carpeta_clase, nombre_imagen)
            diccionario_imagenes[ruta_imagen] = nombre_clase


with open(BASE_FOLDER+ "my_dataset/etiquetas.json", "w") as f:
    json.dump(diccionario_imagenes, f)
```

2. Exploracion del dataset

2.1 cargado del dataset

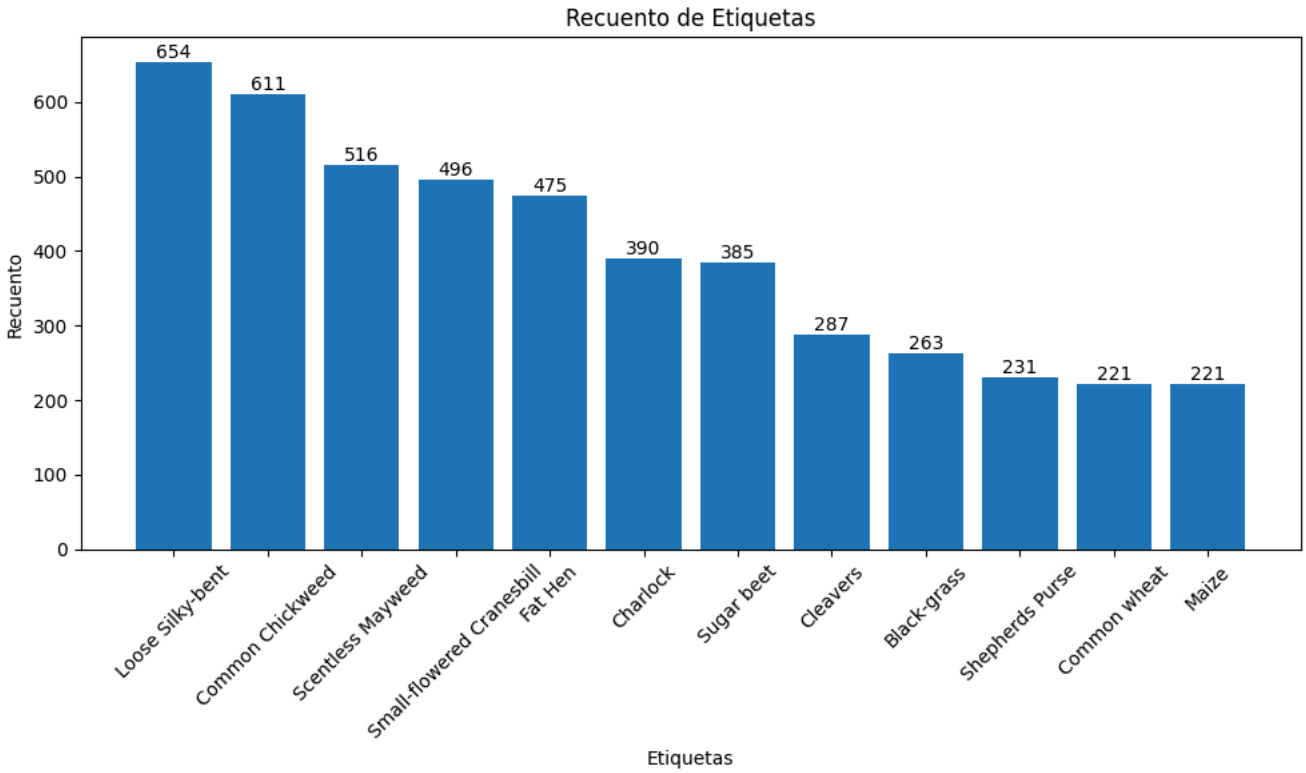
```
with open("/content/drive/MyDrive/BASE_FOLDER/my_dataset/etiquetas.json", 'r') as file:
    data = json.load(file)
df = pd.DataFrame(list(data.items()), columns=['Img', 'Label'])

# Contar el número de claves (filas) para cada valor único de 'Label'
conteo_etiquetas = df['Label'].value_counts().reset_index()
conteo_etiquetas.columns = ['Label', 'Count']

df
```

	Img	Label
0	/content/drive/MyDrive/BASE_FOLDER/my_dataset/...	Black-grass
1	/content/drive/MyDrive/BASE_FOLDER/my_dataset/...	Black-grass
2	/content/drive/MyDrive/BASE_FOLDER/my_dataset/...	Black-grass
3	/content/drive/MyDrive/BASE_FOLDER/my_dataset/...	Black-grass
4	/content/drive/MyDrive/BASE_FOLDER/my_dataset/...	Black-grass
...	...	...
4745	/content/drive/MyDrive/BASE_FOLDER/my_dataset/...	Sugar beet
4746	/content/drive/MyDrive/BASE_FOLDER/my_dataset/...	Sugar beet
4747	/content/drive/MyDrive/BASE_FOLDER/my_dataset/...	Sugar beet
4748	/content/drive/MyDrive/BASE_FOLDER/my_dataset/...	Sugar beet
4749	/content/drive/MyDrive/BASE_FOLDER/my_dataset/...	Sugar beet

4750 rows × 2 columns

2.2 Conteo de las clases de dataset

```
# Contar el número de claves (filas) para cada valor único de 'Label'
conteo_etiquetas = df['Label'].value_counts().reset_index()
conteo_etiquetas.columns = ['Label', 'Count']

plt.figure(figsize=(10, 6))
bars = plt.bar(conteo_etiquetas['Label'], conteo_etiquetas['Count'])

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, int(yval), va='bottom', ha='center')

plt.xlabel('Etiquetas')
plt.ylabel('Recuento')
plt.title('Recuento de Etiquetas')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

conteo_etiquetas
```

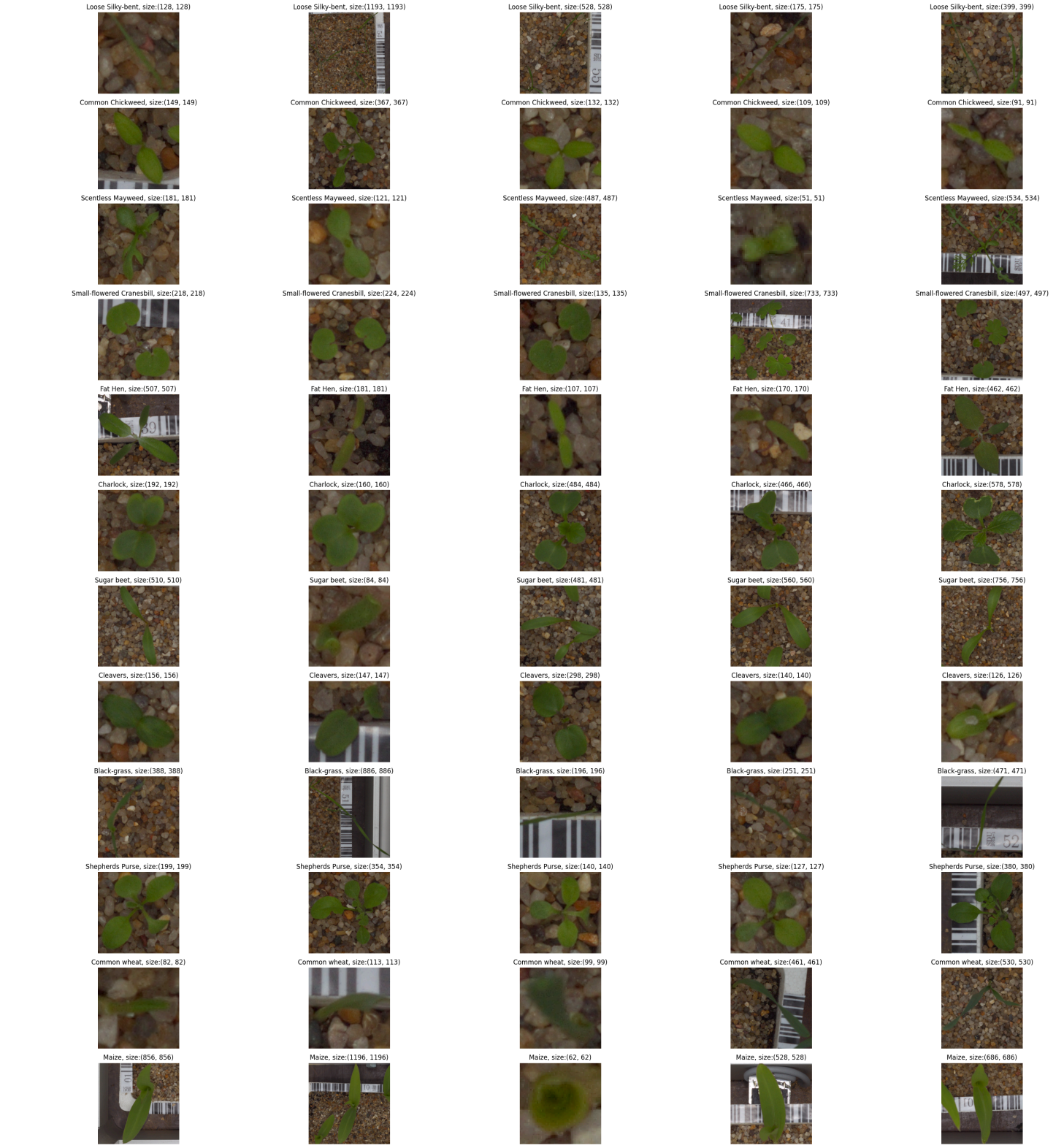


	Label	Count
0	Loose Silky-bent	654
1	Common Chickweed	611
2	Scentless Mayweed	516
3	Small-flowered Cranesbill	496
4	Fat Hen	475
5	Charlock	390
6	Sugar beet	385
7	Cleavers	287
8	Black-grass	263
9	Shepherds Purse	231
10	Common wheat	221
11	Maize	221

2.3 visualizar imagenes

```
def obtener_n_esima_imagen_por_etiqueta(etiqueta, n, dataframe):
    # Filtrar el DataFrame por la etiqueta dada
    df_filtrado = dataframe[dataframe['Label'] == etiqueta]

    # Restablecer el índice para poder acceder por índice numérico
    df_filtrado = df_filtrado.reset_index(drop=True)

    # Verificar si 'n' está dentro del rango del número de imágenes para esa etiqueta
    return df_filtrado.iloc[n]['Img']

def discover_dataset(df,num_cols):

    num_rows = len(labels)

    # Crear el subplot
    fig, axs = plt.subplots(num_rows, num_cols, figsize=(30, 30))

    # Iterar sobre las etiquetas y mostrar las imágenes en el subplot
    for i, l in enumerate(labels):
        for j in range(num_cols):
            image = Image.open(obtener_n_esima_imagen_por_etiqueta(l, j+1, df))  # Se añade 1 para obtener la enésima y la (enésima+1) i
            axs[i, j].imshow(image)
            axs[i, j].set_title(f"{l}, size:{image.size}")
            axs[i, j].axis('off')

    plt.tight_layout()
    plt.show()
```

```
discover_dataset(df,5)
```


Loose Silky-bent, size:(128, 128)
Loose Silky-bent, size:(1193, 1193)
Loose Silky-bent, size:(528, 528)
Loose Silky-bent, size:(175, 175)
Loose Silky-bent, size:(399, 399)
Common Chickweed, size:(149, 149)
Common Chickweed, size:(367, 367)
Common Chickweed, size:(132, 132)
Common Chickweed, size:(109, 109)
Common Chickweed, size:(91, 91)
Scentless Mayweed, size:(181, 181)
Scentless Mayweed, size:(121, 121)
Scentless Mayweed, size:(487, 487)
Scentless Mayweed, size:(51, 51)
Scentless Mayweed, size:(534, 534)
Small-flowered Cranesbill, size:(218, 218)
Small-flowered Cranesbill, size:(224, 224)
Small-flowered Cranesbill, size:(135, 135)
Small-flowered Cranesbill, size:(733, 733)
Small-flowered Cranesbill, size:(497, 497)
Fat Hen, size:(507, 507)
Fat Hen, size:(181, 181)
Fat Hen, size:(107, 107)
Fat Hen, size:(170, 170)
Fat Hen, size:(462, 462)
Charlock, size:(192, 192)
Charlock, size:(160, 160)
Charlock, size:(484, 484)
Charlock, size:(466, 466)
Charlock, size:(578, 578)
Sugar beet, size:(510, 510)
Sugar beet, size:(84, 84)
Sugar beet, size:(481, 481)
Sugar beet, size:(560, 560)
Sugar beet, size:(756, 756)
Cleavers, size:(156, 156)
Cleavers, size:(147, 147)
Cleavers, size:(298, 298)
Cleavers, size:(140, 140)
Cleavers, size:(126, 126)
Black-grass, size:(388, 388)
Black-grass, size:(886, 886)
Black-grass, size:(196, 196)
Black-grass, size:(251, 251)
Black-grass, size:(471, 471)
Shepherds Purse, size:(199, 199)
Shepherds Purse, size:(354, 354)
Shepherds Purse, size:(140, 140)
Shepherds Purse, size:(127, 127)
Shepherds Purse, size:(380, 380)
Common wheat, size:(82, 82)
Common wheat, size:(113, 113)
Common wheat, size:(99, 99)
Common wheat, size:(461, 461)
Common wheat, size:(530, 530)
Maize, size:(856, 856)
Maize, size:(1196, 1196)
Maize, size:(62, 62)
Maize, size:(528, 528)
Maize, size:(686, 686)

3 Preparacion del dataset

3.1 Reescalado de las imagenes

```
def resize_image(image, target_size):
    """reescala las imagenes"""
    width, height = image.size

    scale = max(target_size[0] / width, target_size[1] / height)

    new_width = int(width * scale)
    new_height = int(height * scale)

    resized_image = image.resize((new_width, new_height), Image.ANTIALIAS)

    padded_image = Image.new("RGB", target_size)
    padded_image.paste(resized_image, ((target_size[0] - new_width) // 2, (target_size[1] - new_height) // 2))

    return padded_image
```

```
def resize_image_aspect_ratio(image, target_size):
    """reescala las imagenes respetando el ratio"""
    width, height = image.size

    aspect_ratio = width / height

    target_width, target_height = target_size
    target_aspect_ratio = target_width / target_height

    if aspect_ratio > target_aspect_ratio:
        new_width = target_width
        new_height = int(new_width / aspect_ratio)
        resized_image = image.resize((new_width, new_height), Image.ANTIALIAS)
        padding_height = (target_height - new_height) // 2
        padded_image = Image.new("RGB", target_size)
        padded_image.paste(resized_image, (0, padding_height))
    else:
        new_height = target_height
        new_width = int(new_height * aspect_ratio)
        resized_image = image.resize((new_width, new_height), Image.ANTIALIAS)
        padding_width = (target_width - new_width) // 2
        padded_image = Image.new("RGB", target_size)
        padded_image.paste(resized_image, (padding_width, 0))

    return padded_image
```

```
image = Image.open("/content/drive/MyDrive/BASE_FOLDER/my_dataset/train/Loose Silky-bent/0012f11c4.png")
print(image.size)
resized_image = resize_image_aspect_ratio(image, (225, 200))
view(resized_image)
print(resized_image.size)
```

```
(175, 175)
<ipython-input-9-8dd5e7f21137>:20: DeprecationWarning: ANTIALIAS is deprecated and will be removed in Pillow 10 (2023-07-01). Use LA
  resized_image = image.resize((new_width, new_height), Image.ANTIALIAS)
```

```
(225, 200)
```

3.2 Reescalar todo el dataset

```
def resize_and_save_image(row,size,folder_name):
    path = row['Img']
    image = Image.open(path)
    resized_image = resize_image_aspect_ratio(image, size)
    new_path = path.replace('train', folder_name)
    os.makedirs(os.path.dirname(new_path), exist_ok=True)
    resized_image.save(new_path)

def resize_all_dataset(df, size, folder_name):
    # Crear una lista de hilos
    threads = []

    # Iterar sobre el DataFrame y crear un hilo para cada fila
    for index, row in df.iterrows():
        thread = threading.Thread(target=resize_and_save_image, args=(row,size, folder_name,))
        thread.start()
        threads.append(thread)

    # Esperar a que todos los hilos terminen
    for thread in threads:
        thread.join()
```

```
resize_all_dataset(df, (224, 224),'train_224')
```

3.3 leyendo dataset reescalado

```
with open("/content/drive/MyDrive/BASE_FOLDER/my_dataset/train_resize_224.json", 'r') as file:
    data = json.load(file)
df = pd.DataFrame(list(data.items()), columns=['Img', 'Label'])

discover_dataset(df,5)
```


Loose Silky-bent, size:(224, 224)
Loose Silky-bent, size:(224, 224)
Loose Silky-bent, size:(224, 224)
Loose Silky-bent, size:(224, 224)
Loose Silky-bent, size:(224, 224)
Common Chickweed, size:(224, 224)
Common Chickweed, size:(224, 224)
Common Chickweed, size:(224, 224)
Common Chickweed, size:(224, 224)
Common Chickweed, size:(224, 224)
Scentless Mayweed, size:(224, 224)
Scentless Mayweed, size:(224, 224)
Scentless Mayweed, size:(224, 224)
Scentless Mayweed, size:(224, 224)
Scentless Mayweed, size:(224, 224)
Small-flowered Cranesbill, size:(224, 224)
Small-flowered Cranesbill, size:(224, 224)
Small-flowered Cranesbill, size:(224, 224)
Small-flowered Cranesbill, size:(224, 224)
Small-flowered Cranesbill, size:(224, 224)
Fat Hen, size:(224, 224)
Fat Hen, size:(224, 224)
Fat Hen, size:(224, 224)
Fat Hen, size:(224, 224)
Fat Hen, size:(224, 224)
Charlock, size:(224, 224)
Charlock, size:(224, 224)
Charlock, size:(224, 224)
Charlock, size:(224, 224)
Charlock, size:(224, 224)
Sugar beet, size:(224, 224)
Sugar beet, size:(224, 224)
Sugar beet, size:(224, 224)
Sugar beet, size:(224, 224)
Sugar beet, size:(224, 224)
Cleavers, size:(224, 224)
Cleavers, size:(224, 224)
Cleavers, size:(224, 224)
Cleavers, size:(224, 224)
Cleavers, size:(224, 224)
Black-grass, size:(224, 224)
Black-grass, size:(224, 224)
Black-grass, size:(224, 224)
Black-grass, size:(224, 224)
Black-grass, size:(224, 224)
Shepherds Purse, size:(224, 224)
Shepherds Purse, size:(224, 224)
Shepherds Purse, size:(224, 224)
Shepherds Purse, size:(224, 224)
Shepherds Purse, size:(224, 224)
Common wheat, size:(224, 224)
Common wheat, size:(224, 224)
Common wheat, size:(224, 224)
Common wheat, size:(224, 224)
Common wheat, size:(224, 224)
Maize, size:(224, 224)
Maize, size:(224, 224)
Maize, size:(224, 224)
Maize, size:(224, 224)
Maize, size:(224, 224)