1.0-raa-eda

May 31, 2025

[2]:
```
%load_ext autoreload
%autoreload 2
```

[45]:
```
import json
import os
import pandas as pd
```

[46]:
```
with open('train_annotations.json', 'r') as f:
    data = json.load(f)
```

[47]:
```
print(data.keys())
```

```
dict_keys(['info', 'categories', 'images', 'annotations'])
```

[48]:
```
images_df = pd.DataFrame(data['images'])
categories_df = pd.DataFrame(data['categories'])
annotations_df = pd.DataFrame(data['annotations'])
```

[49]:
```
images_df.head()
```

[49]:

	file_name	rights_holder	height	width	\
0	5a2176e7-23d2-11e8-a6a3-ec086b02610b.jpg	Justin Brown	1494	2048	
1	5a2177bf-23d2-11e8-a6a3-ec086b02610b.jpg	Justin Brown	1494	2048	
2	59af966d-23d2-11e8-a6a3-ec086b02610b.jpg	Justin Brown	1494	2048	
3	59a30d59-23d2-11e8-a6a3-ec086b02610b.jpg	Justin Brown	1494	2048	
4	59ffbb81-23d2-11e8-a6a3-ec086b02610b.jpg	Justin Brown	1494	2048	

	frame_num	date_captured	location	seq_num_frames	\
0	2	2011-11-29 17:28:26	38	3	
1	3	2011-10-06 11:49:18	38	3	
2	3	2011-12-15 09:50:03	43	3	
3	1	2013-01-22 09:25:16	61	3	
4	1	2011-09-18 23:14:54	38	3	

	seq_id	id
0	6f011019-5567-11e8-a650-dca9047ef277	5a2176e7-23d2-11e8-a6a3-ec086b02610b
1	6f003940-5567-11e8-afb4-dca9047ef277	5a2177bf-23d2-11e8-a6a3-ec086b02610b
2	6f0be74a-5567-11e8-8662-dca9047ef277	59af966d-23d2-11e8-a6a3-ec086b02610b

```
3  70134747-5567-11e8-8313-dca9047ef277  59a30d59-23d2-11e8-a6a3-ec086b02610b
4  6f000b51-5567-11e8-9c1b-dca9047ef277  59ffbb81-23d2-11e8-a6a3-ec086b02610b
```

[50]:
```
categories_df
```

[50]:
```
    id      name
0    6    bobcat
1    1   opossum
2   30     empty
3    9    coyote
4    3   raccoon
5   11      bird
6    8       dog
7   16       cat
8    5  squirrel
9   10    rabbit
10   7     skunk
11  99    rodent
12  21    badger
13  34      deer
14  33       car
15  51       fox
```

[51]:
```
annotations_df
```

[51]:
```
                                    image_id  category_id  \
0      5a197af2-23d2-11e8-a6a3-ec086b02610b           10
1      59fc7e52-23d2-11e8-a6a3-ec086b02610b           16
2      5a2e130d-23d2-11e8-a6a3-ec086b02610b           10
3      598ad0cf-23d2-11e8-a6a3-ec086b02610b            8
4      598ad0cf-23d2-11e8-a6a3-ec086b02610b            8
…                                        …           …
14066  5a015e75-23d2-11e8-a6a3-ec086b02610b           11
14067  5968c14d-23d2-11e8-a6a3-ec086b02610b           10
14068  59d8ebb0-23d2-11e8-a6a3-ec086b02610b           34
14069  5a27d89c-23d2-11e8-a6a3-ec086b02610b           11
14070  59fe2386-23d2-11e8-a6a3-ec086b02610b           11

                                                    bbox  \
0      [317.44, 747.52, 261.11999999999995, 284.15999…
1               [0.0, 660.48, 381.44, 273.9200000000001]
2                        [1397.76, 458.24, 192.0, 192.0]
3      [989.44, 759.04, 262.4000000000001, 390.400000…
4           [33.28, 1118.72, 583.68, 375.03999999999996]
…                                                     …
14066  [448.0, 739.84, 302.0799999999999, 204.8000000…
14067                                              NaN
```

```
14068  [1277.44, 43.519999999999996, 435.200000000000…
14069  [1186.1840178571429, 812.0115178571428, 505.70…
14070  [974.08, 695.04, 227.84000000000003, 148.47999…
```

```
                                      id
0                                  19688
1                                  43259
2                                  10268
3                                   2721
4                                   2722
…                                      …
14066                               9729
14067  98613bde-7de8-11e7-884d-7845c41c2c67
14068                              11991
14069  dee84346-d0b1-4ec8-a497-abf92e400940
14070                              29856

[14071 rows x 4 columns]
```

[52]:
```
merged_df = annotations_df.merge(images_df, left_on='image_id', right_on='id',␣
↪suffixes=('_ann', '_img'))
merged_df.head()
```

[52]:
```
                              image_id  category_id  \
0  5a197af2-23d2-11e8-a6a3-ec086b02610b           10
1  59fc7e52-23d2-11e8-a6a3-ec086b02610b           16
2  5a2e130d-23d2-11e8-a6a3-ec086b02610b           10
3  598ad0cf-23d2-11e8-a6a3-ec086b02610b            8
4  598ad0cf-23d2-11e8-a6a3-ec086b02610b            8

                                              bbox id_ann  \
0  [317.44, 747.52, 261.11999999999995, 284.15999…  19688
1            [0.0, 660.48, 381.44, 273.9200000000001]  43259
2                     [1397.76, 458.24, 192.0, 192.0]  10268
3  [989.44, 759.04, 262.4000000000001, 390.400000…   2721
4        [33.28, 1118.72, 583.68, 375.03999999999996]   2722

                                 file_name rights_holder  height  width  \
0  5a197af2-23d2-11e8-a6a3-ec086b02610b.jpg  Justin Brown    1494   2048
1  59fc7e52-23d2-11e8-a6a3-ec086b02610b.jpg  Justin Brown    1494   2048
2  5a2e130d-23d2-11e8-a6a3-ec086b02610b.jpg  Justin Brown    1494   2048
3  598ad0cf-23d2-11e8-a6a3-ec086b02610b.jpg  Justin Brown    1494   2048
4  598ad0cf-23d2-11e8-a6a3-ec086b02610b.jpg  Justin Brown    1494   2048

   frame_num        date_captured  location  seq_num_frames  \
0          1  2011-11-30 18:05:28        38               3
1          2  2011-11-07 19:10:18        38               3
```

```
2          2  2011-12-16 07:26:30        38               3
3          3  2011-11-14 12:28:10        38               3
4          3  2011-11-14 12:28:10        38               3

                                  seq_id                                id_img
0  6f0112f3-5567-11e8-8b80-dca9047ef277  5a197af2-23d2-11e8-a6a3-ec086b02610b
1  6f00abae-5567-11e8-91d6-dca9047ef277  59fc7e52-23d2-11e8-a6a3-ec086b02610b
2  6f017aa8-5567-11e8-aec9-dca9047ef277  5a2e130d-23d2-11e8-a6a3-ec086b02610b
3  6f00cd8a-5567-11e8-b5bd-dca9047ef277  598ad0cf-23d2-11e8-a6a3-ec086b02610b
4  6f00cd8a-5567-11e8-b5bd-dca9047ef277  598ad0cf-23d2-11e8-a6a3-ec086b02610b
```

[53]:
```
merged_df = merged_df.merge(categories_df, left_on='category_id',␣
 ↪right_on='id', suffixes=('', '_cat'))
merged_df.head()
```

[53]:
```
                               image_id  category_id  \
0  5a197af2-23d2-11e8-a6a3-ec086b02610b           10
1  59fc7e52-23d2-11e8-a6a3-ec086b02610b           16
2  5a2e130d-23d2-11e8-a6a3-ec086b02610b           10
3  598ad0cf-23d2-11e8-a6a3-ec086b02610b            8
4  598ad0cf-23d2-11e8-a6a3-ec086b02610b            8

                                              bbox id_ann  \
0  [317.44, 747.52, 261.11999999999995, 284.15999…  19688
1          [0.0, 660.48, 381.44, 273.9200000000001]  43259
2                  [1397.76, 458.24, 192.0, 192.0]  10268
3  [989.44, 759.04, 262.4000000000001, 390.400000…   2721
4       [33.28, 1118.72, 583.68, 375.03999999999996]   2722

                                file_name rights_holder  height  width  \
0  5a197af2-23d2-11e8-a6a3-ec086b02610b.jpg  Justin Brown    1494   2048
1  59fc7e52-23d2-11e8-a6a3-ec086b02610b.jpg  Justin Brown    1494   2048
2  5a2e130d-23d2-11e8-a6a3-ec086b02610b.jpg  Justin Brown    1494   2048
3  598ad0cf-23d2-11e8-a6a3-ec086b02610b.jpg  Justin Brown    1494   2048
4  598ad0cf-23d2-11e8-a6a3-ec086b02610b.jpg  Justin Brown    1494   2048

   frame_num        date_captured  location  seq_num_frames  \
0          1  2011-11-30 18:05:28        38               3
1          2  2011-11-07 19:10:18        38               3
2          2  2011-12-16 07:26:30        38               3
3          3  2011-11-14 12:28:10        38               3
4          3  2011-11-14 12:28:10        38               3

                                  seq_id                                id_img  \
0  6f0112f3-5567-11e8-8b80-dca9047ef277  5a197af2-23d2-11e8-a6a3-ec086b02610b
1  6f00abae-5567-11e8-91d6-dca9047ef277  59fc7e52-23d2-11e8-a6a3-ec086b02610b
2  6f017aa8-5567-11e8-aec9-dca9047ef277  5a2e130d-23d2-11e8-a6a3-ec086b02610b
```

```
3  6f00cd8a-5567-11e8-b5bd-dca9047ef277  598ad0cf-23d2-11e8-a6a3-ec086b02610b
4  6f00cd8a-5567-11e8-b5bd-dca9047ef277  598ad0cf-23d2-11e8-a6a3-ec086b02610b

   id    name
0  10  rabbit
1  16     cat
2  10  rabbit
3   8     dog
4   8     dog
```

[54]:
```
import matplotlib.pyplot as plt
from PIL import Image, ImageDraw
```

[168]:
```
IMAGES_FOLDER = 'eccv_18_all_images_sm'
```

[ ]:
```
sample = merged_df.iloc[10]
image_path = os.path.join(IMAGES_FOLDER, sample['file_name'])

img = Image.open(image_path)
draw = ImageDraw.Draw(img)
```

[71]:
```
original_width = sample['width']
original_height = sample['height']

scale_x = img.width / original_width
scale_y = img.height / original_height

bbox = sample['bbox']
x0, y0, width, height = map(float, bbox)
x1, y1 = x0 + width, y0 + height

x0_scaled = x0 * scale_x
y0_scaled = y0 * scale_y
x1_scaled = x1 * scale_x
y1_scaled = y1 * scale_y

draw.rectangle([x0_scaled, y0_scaled, x1_scaled, y1_scaled], outline='red',␣
 ↪width=3)

plt.figure(figsize=(10, 8))
plt.imshow(img)
plt.title(f"Category: {sample['name']}")
plt.axis('off')
plt.show()
```

Category: cat

[80]:
```
import numpy as np
```

[81]:
```
def rescale_bounding_boxes(df, target_width, target_height, bbox_column='bbox',
                           width_column='width', height_column='height',
                           output_column='bbox_scaled'):
    """
    Reescala bounding boxes a nuevas dimensiones objetivo.

    Parameters:
    - df: pandas.DataFrame con columnas bbox, width, height.
    - target_width: nuevo ancho de la imagen.
    - target_height: nuevo alto de la imagen.
    - bbox_column: nombre de la columna que contiene los bboxes originales.
    - width_column: nombre de la columna que contiene el ancho original.
    - height_column: nombre de la columna que contiene el alto original.
    - output_column: nombre de la columna de salida que guardará las cajas␣
 ↪reescaladas.

    Returns:
    - DataFrame con columna adicional de bboxes reescalados.
```

```
    """
    original_width = df[width_column].iloc[0]
    original_height = df[height_column].iloc[0]
    scale_x = target_width / original_width
    scale_y = target_height / original_height

    def scale_bbox(bbox):
        if bbox is None or isinstance(bbox, float) and np.isnan(bbox):
            return None
        if not isinstance(bbox, (list, tuple)) or len(bbox) != 4:
            return None
        x0, y0, width, height = map(float, bbox)
        x1, y1 = x0 + width, y0 + height
        x0_scaled = x0 * scale_x
        y0_scaled = y0 * scale_y
        width_scaled = (x1 - x0) * scale_x
        height_scaled = (y1 - y0) * scale_y
        return [x0_scaled, y0_scaled, width_scaled, height_scaled]

    df[output_column] = df[bbox_column].apply(scale_bbox)
    return df
```

[82]:
```
rescaled_df = rescale_bounding_boxes(merged_df, target_width=1024,␣
↪target_height=747)
print(rescaled_df[['file_name', 'bbox_scaled']].head())
```

```
                                file_name  \
0  5a197af2-23d2-11e8-a6a3-ec086b02610b.jpg
1  59fc7e52-23d2-11e8-a6a3-ec086b02610b.jpg
2  5a2e130d-23d2-11e8-a6a3-ec086b02610b.jpg
3  598ad0cf-23d2-11e8-a6a3-ec086b02610b.jpg
4  598ad0cf-23d2-11e8-a6a3-ec086b02610b.jpg

                                         bbox_scaled
0  [158.72, 373.76, 130.55999999999997, 142.07999…
1          [0.0, 330.24, 190.72, 136.96000000000004]
2                      [698.88, 229.12, 96.0, 96.0]
3  [494.72, 379.52, 131.20000000000005, 195.20000…
4        [16.64, 559.36, 291.84, 187.51999999999998]
```

[ ]:
```
def show_random_image_with_bbox(df, images_folder, bbox_column='bbox_scaled',
                                file_column='file_name', label_column='name'):
    """
    Muestra aleatoriamente una imagen del dataframe con su bbox dibujado.

    Parameters:
    - df: pandas.DataFrame que debe tener bbox escalados.
```

```
    - images_folder: ruta a la carpeta donde están las imágenes.
    - bbox_column: columna donde está el bbox escalado.
    - file_column: columna con el nombre del archivo de imagen.
    - label_column: columna con la etiqueta/clase.
    """
    valid_df = df[df[bbox_column].notnull()]

    sample = valid_df.sample(1).iloc[0]
    image_path = os.path.join(images_folder, sample[file_column])

    img = Image.open(image_path)
    draw = ImageDraw.Draw(img)

    x0, y0, width, height = map(float, sample[bbox_column])
    x1, y1 = x0 + width, y0 + height

    draw.rectangle([x0, y0, x1, y1], outline='red', width=3)

    plt.figure(figsize=(8, 6))
    plt.imshow(img)
    plt.title(f"Category: {sample[label_column]}")
    plt.axis('off')
    plt.show()
```

[ ]:
```
show_random_image_with_bbox(rescaled_df, images_folder=IMAGES_FOLDER)
```

[118]:
```
rescaled_df[rescaled_df['bbox_scaled'].notnull()]['file_name'].nunique()
```

[118]:
```
12099
```

[152]:
```
bbox_counts = rescaled_df[rescaled_df['bbox_scaled'].notnull()].
↪groupby('file_name').size()
bbox_counts.shape
```

[152]:
```
(12099,)
```

[123]:
```
multi_bbox_images = bbox_counts[bbox_counts > 1]
num_multi_bbox_images = len(multi_bbox_images)

print(f"Número de imágenes con más de un bbox: {num_multi_bbox_images}")
```

```
Número de imágenes con más de un bbox: 414
```

[124]:
```
multi_bbox_images.head()
```

[124]:
```
file_name
585c0451-23d2-11e8-a6a3-ec086b02610b.jpg    2
585c0558-23d2-11e8-a6a3-ec086b02610b.jpg    2
585da972-23d2-11e8-a6a3-ec086b02610b.jpg    2
585f4cf6-23d2-11e8-a6a3-ec086b02610b.jpg    2
585f4e71-23d2-11e8-a6a3-ec086b02610b.jpg    3
dtype: int64
```

[143]:
```python
import random
```

[144]:
```python
def show_image_with_multi_bbox(df, images_folder, file_name_column='file_name',
                               bbox_column='bbox_scaled', label_column='name',␣
↪file_name=None):
    """
    Muestra una imagen (aleatoria si no se especifica) con todos sus bboxes␣
↪dibujados.

    Parameters:
    - df: pandas.DataFrame con bboxes escalados.
    - images_folder: ruta a la carpeta de imágenes.
    - file_name_column: columna con el nombre del archivo.
    - bbox_column: columna con las cajas escaladas.
    - label_column: columna con las etiquetas.
    - file_name: (opcional) nombre de archivo específico. Si None, selecciona␣
↪aleatorio.
    """
    valid_df = df[df[bbox_column].notnull()]

    if file_name is None:
        multi_bbox_files = valid_df.groupby(file_name_column).size()
        multi_bbox_files = multi_bbox_files[multi_bbox_files > 1].index
        if len(multi_bbox_files) == 0:
            print("No hay imágenes con múltiples bboxes.")
            return
        file_name = random.choice(multi_bbox_files)

    image_df = valid_df[valid_df[file_name_column] == file_name]

    if image_df.empty:
        print(f"No se encontraron bboxes para la imagen {file_name}")
        return

    image_path = os.path.join(images_folder, file_name)
    img = Image.open(image_path)
    draw = ImageDraw.Draw(img)

    for _, row in image_df.iterrows():
```

```
        x0, y0, width, height = map(float, row[bbox_column])
        x1, y1 = x0 + width, y0 + height
        draw.rectangle([x0, y0, x1, y1], outline='red', width=3)
        draw.text((x0, y0), row[label_column], fill='red')

    plt.figure(figsize=(10, 8))
    plt.imshow(img)
    plt.title(f"File: {file_name}")
    plt.axis('off')
    plt.show()
```

[ ]:
```
show_image_with_multi_bbox(rescaled_df, images_folder=IMAGES_FOLDER,␣
↪file_name='585f4e71-23d2-11e8-a6a3-ec086b02610b.jpg')
```

[ ]:
```
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14071 entries, 0 to 14070
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
```

```
 0   image_id         14071 non-null  object
 1   category_id      14071 non-null  int64
 2   bbox             12617 non-null  object
 3   id_ann           14071 non-null  object
 4   file_name        14071 non-null  object
 5   rights_holder    14071 non-null  object
 6   height           14071 non-null  int64
 7   width            14071 non-null  int64
 8   frame_num        14071 non-null  int64
 9   date_captured    14071 non-null  object
 10  location         14071 non-null  int64
 11  seq_num_frames   14071 non-null  int64
 12  seq_id           14071 non-null  object
 13  id_img           14071 non-null  object
 14  id               14071 non-null  int64
 15  name             14071 non-null  object
dtypes: int64(7), object(9)
memory usage: 1.7+ MB
```

[ ]:
```
show_image_with_multi_bbox(rescaled_df, images_folder=IMAGES_FOLDER)
```

0.1 ¿Cuántos ejemplos por clase tenemos? (bbox = clase detectada)

[156]:
```
valid_df = rescaled_df[rescaled_df['bbox_scaled'].notnull()]
class_counts = valid_df['name'].value_counts()
class_counts
```

[156]:
```
name
opossum     2514
rabbit      2278
coyote      1371
cat         1170
squirrel    1037
raccoon     1030
dog          769
bobcat       684
car          668
bird         560
rodent       264
skunk        214
deer          44
badger         9
fox            5
Name: count, dtype: int64
```

Clase (inglés)	Clase (español)	Cantidad
opossum	zarigüeya	2,514
rabbit	conejo	2,278
coyote	coyote	1,371
cat	gato	1,170
squirrel	ardilla	1,037
raccoon	mapache	1,030
dog	perro	769
bobcat	lince rojo	684
car	auto/coche	668
bird	ave	560
rodent	roedor	264
skunk	mofeta/zorrillo	214
deer	ciervo/venado	44
badger	tejón	9
fox	zorro	5

0.1.1 Análisis de balance de clases

- Se evidencian clases muy minoritarias (fox, badger, deer, etc)
- Se podria utilizar *data augmentation* para clases medianas, y few-shot para ultra minoritarias
- Tambien su pudo haber agrupado clases, si es que tuviera sentido biológico o justificación científica, cosas que se duda entre zorro, tejón y ciervo.

Decisión: - Para clases de 200 ejemplos, aplicar augmentation - Para clases de <50 se descartan temporalmente, son demasiado escasas

[161]:
```
rescaled_df.head()
```

[161]:

	image_id	category_id	bbox	id_ann	file_name	rights_holder	height	width	frame_num	date_captured	location	seq_num_frames	seq_id	id_img	id	name	bbox_scaled
0	5a197af2-23d2-11e8-a6a3-ec086b02610b	10	[317.44, 747.52, 261.11999999999995, 284.15999…	19688	5a197af2-23d2-11e8-a6a3-ec086b02610b.jpg	Justin Brown	1494	2048	1	2011-11-30 18:05:28	38	3	6f0112f3-5567-11e8-8b80-dca9047ef277	5a197af2-23d2-11e8-a6a3-ec086b02610b	10	rabbit	[158.72, 373.76, 130.55999999999997, 142.07999…
1	59fc7e52-23d2-11e8-a6a3-ec086b02610b	16	[0.0, 660.48, 381.44, 273.9200000000001]	43259	59fc7e52-23d2-11e8-a6a3-ec086b02610b.jpg	Justin Brown	1494	2048	2	2011-11-07 19:10:18	38	3	6f00abae-5567-11e8-91d6-dca9047ef277	59fc7e52-23d2-11e8-a6a3-ec086b02610b	16	cat	[0.0, 330.24, 190.72, 136.96000000000004]
2	5a2e130d-23d2-11e8-a6a3-ec086b02610b	10	[1397.76, 458.24, 192.0, 192.0]	10268	5a2e130d-23d2-11e8-a6a3-ec086b02610b.jpg	Justin Brown	1494	2048	2	2011-12-16 07:26:30	38	3	6f017aa8-5567-11e8-aec9-dca9047ef277	5a2e130d-23d2-11e8-a6a3-ec086b02610b	10	rabbit	[698.88, 229.12, 96.0, 96.0]
3	598ad0cf-23d2-11e8-a6a3-ec086b02610b	8	[989.44, 759.04, 262.4000000000001, 390.400000…	2721	598ad0cf-23d2-11e8-a6a3-ec086b02610b.jpg	Justin Brown	1494	2048	3	2011-11-14 12:28:10	38	3	6f00cd8a-5567-11e8-b5bd-dca9047ef277	598ad0cf-23d2-11e8-a6a3-ec086b02610b	8	dog	[494.72, 379.52, 131.20000000000005, 195.20000…
4	598ad0cf-23d2-11e8-a6a3-ec086b02610b	8	[33.28, 1118.72, 583.68, 375.03999999999996]	2722	598ad0cf-23d2-11e8-a6a3-ec086b02610b.jpg	Justin Brown	1494	2048	3	2011-11-14 12:28:10	38	3	6f00cd8a-5567-11e8-b5bd-dca9047ef277	598ad0cf-23d2-11e8-a6a3-ec086b02610b	8	dog	[16.64, 559.36, 291.84, 187.51999999999998]

[160]:
```
rescaled_df.shape
```

[160]: (14071, 17)

0.1.2 *data augmentation*

[ ]:
```
def save_recortes_by_class(df, images_folder, output_folder,␣
↪bbox_column='bbox_scaled',
                           file_column='file_name', label_column='name'):
    """
    Recorta las imágenes según bbox y guarda en carpetas por clase.
    """
    for _, row in df.iterrows():
        class_name = row[label_column]
        bbox = row[bbox_column]
        if bbox is None:
            continue
        file_name = row[file_column]

        class_folder = os.path.join(output_folder, class_name)
        os.makedirs(class_folder, exist_ok=True)

        img_path = os.path.join(images_folder, file_name)
        img = Image.open(img_path)

        x0, y0, width, height = map(float, bbox)
        x1, y1 = x0 + width, y0 + height
        cropped_img = img.crop((x0, y0, x1, y1))

        save_name = f"{os.path.splitext(file_name)[0]}_{row['id_ann']}.jpg"
        cropped_img.save(os.path.join(class_folder, save_name))
```

[191]:
```
OUTPUT_FOLDER = os.path.join('bboxes', 'bboxes_recortes')
```

[192]:
```
OUTPUT_FOLDER
```

[192]: 'bboxes\\bboxes_recortes'

[ ]:
```
save_recortes_by_class(rescaled_df, IMAGES_FOLDER, OUTPUT_FOLDER)
```

[185]:
```
import albumentations as A
import cv2
from tqdm import tqdm
```

[ ]:
```
def apply_augmentations_for_class(class_name, input_root, output_root,␣
↪num_augmentations=2):
    """
    Aplica augmentations a una clase específica.
```

```
    Parameters:
    - class_name: nombre de la clase (carpeta dentro de input_root).
    - input_root: carpeta raíz de los recortes originales.
    - output_root: carpeta raíz para guardar augmentations.
    - num_augmentations: cuántas imágenes augmentadas generar por original.
    """
    input_folder = os.path.join(input_root, class_name)
    output_folder = os.path.join(output_root, class_name)

    transform = A.Compose([
        A.HorizontalFlip(p=0.5),
        A.RandomBrightnessContrast(p=0.2),
        A.Rotate(limit=20, p=0.5),
        A.GaussNoise(p=0.2)
    ])

    os.makedirs(output_folder, exist_ok=True)

    for img_name in tqdm(os.listdir(input_folder), desc=f"Augmenting␣
 ↪{class_name}"):
        img_path = os.path.join(input_folder, img_name)
        img = cv2.imread(img_path)

        if img is None:
            print(f"No se pudo leer la imagen: {img_path}")
            continue

        for i in range(num_augmentations):
            augmented = transform(image=img)['image']
            save_name = f"{os.path.splitext(img_name)[0]}_aug{i}.jpg"
            cv2.imwrite(os.path.join(output_folder, save_name), augmented)
```

[198]:
```
INPUT_FOLDER = os.path.join('bboxes', 'bboxes_recortes')
AUG_OUTPUT_FOLDER = os.path.join('bboxes', 'augmented')
```

[199]:
```
apply_augmentations_for_class('dog', INPUT_FOLDER, AUG_OUTPUT_FOLDER,␣
 ↪num_augmentations=2)
apply_augmentations_for_class('bobcat', INPUT_FOLDER, AUG_OUTPUT_FOLDER,␣
 ↪num_augmentations=2)
apply_augmentations_for_class('car', INPUT_FOLDER, AUG_OUTPUT_FOLDER,␣
 ↪num_augmentations=2)
apply_augmentations_for_class('bird', INPUT_FOLDER, AUG_OUTPUT_FOLDER,␣
 ↪num_augmentations=2)
apply_augmentations_for_class('rodent', INPUT_FOLDER, AUG_OUTPUT_FOLDER,␣
 ↪num_augmentations=4)
apply_augmentations_for_class('skunk', INPUT_FOLDER, AUG_OUTPUT_FOLDER,␣
 ↪num_augmentations=4)
```

```
Augmenting dog: 100%|      | 769/769 [00:10<00:00, 70.71it/s]
Augmenting bobcat: 100%|      | 684/684 [00:09<00:00, 72.32it/s]
Augmenting car: 100%|      | 668/668 [00:33<00:00, 20.21it/s]
Augmenting bird: 100%|      | 560/560 [00:07<00:00, 78.24it/s]
Augmenting rodent: 100%|      | 264/264 [00:03<00:00, 78.79it/s]
Augmenting skunk: 100%|      | 214/214 [00:03<00:00, 59.35it/s]
```

[ ]:
```python
def show_random_augmented_image(class_name,␣
↪augmented_folder='bboxes\\augmented'):
    """
    Muestra aleatoriamente una imagen augmentada de la clase indicada.

    Parameters:
    - class_name: nombre de la carpeta/clase dentro de augmented_folder.
    - augmented_folder: carpeta raíz donde están las carpetas de augmentations.
    """
    class_folder = os.path.join(augmented_folder, class_name)
    if not os.path.exists(class_folder):
        print(f"Carpeta no encontrada: {class_folder}")
        return

    images = os.listdir(class_folder)
    if not images:
        print(f"No hay imágenes en {class_folder}")
        return

    selected_image = random.choice(images)
    img_path = os.path.join(class_folder, selected_image)

    img = Image.open(img_path)
    plt.figure(figsize=(6, 6))
    plt.imshow(img)
    plt.title(f"Class: {class_name}\nFile: {selected_image}")
    plt.axis('off')
    plt.show()
```

[224]:
```python
show_random_augmented_image('dog')
```

0.1.3 Reflexión

Lo ideal sería aplicar augmentations únicamente al objeto (solo al perro o animal en cuestión) y no al fondo.

Para lograr esto, no basta con trabajar sobre bounding boxes…

Este nivel de precisión requiere:

- Máscaras de segmentación pixel a pixel del animal dentro de la imagen.
- Modelos especializados como U-Net u otras arquitecturas de segmentación.

El enfoque es válido, pero es importante reconocer sus limitaciones.

0.2 Armar dataloader

[4]:
```
import shutil
import os
```

[6]:
```
dataset_dir = os.path.join('data', 'dataloader')

bboxes_recortes_dir = os.path.join('bboxes', 'bboxes_recortes')
augmented_dir = os.path.join('bboxes', 'augmented')
```

[232]:
```
allowed_classes = [
    'opossum', 'rabbit', 'coyote', 'cat', 'squirrel', 'raccoon',
    'dog', 'bobcat', 'car', 'bird', 'rodent', 'skunk'
    # NOTA: estamos excluyendo deer, badger, fox
]
```

Encapsulamos únicamente las clases que tienen una muestra representativa

[234]:
```
os.makedirs(dataset_dir, exist_ok=True)

for class_name in allowed_classes:
    dest_class_dir = os.path.join(dataset_dir, class_name)
    os.makedirs(dest_class_dir, exist_ok=True)

    # Copiar desde bboxes_recortes
    src_bbox_class_dir = os.path.join(bboxes_recortes_dir, class_name)
    if os.path.exists(src_bbox_class_dir):
        for filename in tqdm(os.listdir(src_bbox_class_dir),␣
 ↪desc=f"{class_name} - bbox", leave=True):
            src_file = os.path.join(src_bbox_class_dir, filename)
            dest_file = os.path.join(dest_class_dir, filename)
            shutil.copy2(src_file, dest_file)

    # Copiar desde augmented
    src_aug_class_dir = os.path.join(augmented_dir, class_name)
    if os.path.exists(src_aug_class_dir):
        for filename in tqdm(os.listdir(src_aug_class_dir), desc=f"{class_name}␣
 ↪- aug", leave=True):
            src_file = os.path.join(src_aug_class_dir, filename)
            dest_file = os.path.join(dest_class_dir, filename)
            shutil.copy2(src_file, dest_file)
```

```
opossum - bbox: 100%|      | 2514/2514 [00:21<00:00, 116.72it/s]
rabbit - bbox: 100%|      | 2278/2278 [00:22<00:00, 102.61it/s]
coyote - bbox: 100%|      | 1371/1371 [00:16<00:00, 83.62it/s]
cat - bbox: 100%|      | 1170/1170 [00:10<00:00, 110.17it/s]
squirrel - bbox: 100%|      | 1037/1037 [00:09<00:00, 111.84it/s]
raccoon - bbox: 100%|      | 1030/1030 [00:09<00:00, 109.88it/s]
dog - bbox: 100%|      | 769/769 [00:07<00:00, 99.78it/s]
```

```
dog - aug: 100%|      | 1538/1538 [00:15<00:00, 98.87it/s]
bobcat - bbox: 100%|      | 684/684 [00:01<00:00, 496.75it/s]
bobcat - aug: 100%|      | 1368/1368 [00:03<00:00, 383.64it/s]
car - bbox: 100%|      | 668/668 [00:03<00:00, 191.64it/s]
car - aug: 100%|      | 1336/1336 [00:11<00:00, 114.57it/s]
bird - bbox: 100%|      | 560/560 [00:05<00:00, 101.35it/s]
bird - aug: 100%|      | 1120/1120 [00:10<00:00, 102.85it/s]
rodent - bbox: 100%|      | 264/264 [00:02<00:00, 111.79it/s]
rodent - aug: 100%|      | 1056/1056 [00:09<00:00, 108.27it/s]
skunk - bbox: 100%|      | 214/214 [00:02<00:00, 93.12it/s]
skunk - aug: 100%|      | 856/856 [00:08<00:00, 96.73it/s]
```

[9]:
```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    dataset_dir,
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size
)

val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    dataset_dir,
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size
)
```

```
Found 19833 files belonging to 12 classes.
Using 15867 files for training.
Found 19833 files belonging to 12 classes.
Using 3966 files for validation.
```