TECNOLÓGICO NACIONAL DE MÉXICO®

Ing. En Sistemas Computacionales

Inteligencia Artificial

Maestro:

Zuriel Dathan Mora Felix

Tarea 1:

“Pre-procesado de imagenes”

Alumnos:

Grande Espinoza Víctor Ramon	20170684
Montero López Yahel Alejandro	21170401

20 / 05 / 2025

Preprocesador de imágenes:

Importamos las clases y módulos específicos de la librería Pillow. “Image” que es la clase principal para abrir, manipular y guardar imágenes. “ImageEnhance”que nos permite ajustar propiedades como el brillo, contraste, nitidez y color de una imagen.

Importamos la librería NumPy, que es fundamental para realizar operaciones numéricas eficientes, especialmente con arrays.

Importamos el módulo os, que proporciona una forma de interactuar con el sistema operativo. Lo usamos para crear carpetas.

```
from PIL import Image, ImageEnhance
import numpy as np
import os
```

El constructor se ejecuta al crear un objeto ImagePreprocessor. Guarda las rutas de las carpetas de entrada y salida, y se asegura de que la carpeta de salida principal exista.

```
class ImagePreprocessor:
    def __init__(self, input_dataset_root,
output_dataset_root="processed_dataset"):
        self.input_dataset_root = input_dataset_root
        self.output_dataset_root = output_dataset_root
        os.makedirs(output_dataset_root, exist_ok=True)
```

_load_image(): Abre una imagen desde una ruta específica, y la convierte a formato RGB para asegurar consistencia y manejar errores de carga.

```
def _load_image(self, image_full_path):
        try:
            return Image.open(image_full_path).convert("RGB")
        except Exception as e:
            print(f"Error al cargar la imagen {image_full_path}: {e}")
            return None
```

_save_image(): Guarda una imagen procesada en la ruta de salida correcta, crea subcarpetas si es necesario y añade un sufijo al nombre del archivo para indicar la transformación.

```
def _save_image(self, image, output_dir, original_filename, suffix=""):
        os.makedirs(output_dir, exist_ok=True)
        name, ext = os.path.splitext(original_filename)
        output_filename = f"{name}{suffix}{ext}"
        output_filepath = os.path.join(output_dir, output_filename)
        try:
            image.save(output_filepath)
```

```
        except Exception as e:
            print(f"Error al guardar la imagen {output_filepath}: {e}")
```

adjust_brightness, rotate_image, resize_image: Cada uno toma una imagen, aplica una transformación específica (brillo, rotación, escala) y luego usa _save_image para guardar la nueva versión de la imagen en el lugar correcto.

```
def adjust_brightness(self, image_full_path, output_dir, image_file,
factor=1.5):
        img = self._load_image(image_full_path)
        if img:
            enhancer = ImageEnhance.Brightness(img)
            img_bright = enhancer.enhance(factor)
            self._save_image(img_bright, output_dir, image_file,
f"_brillo_{factor:.1f}")

    def rotate_image(self, image_full_path, output_dir, image_file,
angle=90, expand=True):
        img = self._load_image(image_full_path)
        if img:
            img_rotated = img.rotate(angle, expand=expand)
            self._save_image(img_rotated, output_dir, image_file,
f"_rotada_{angle}deg")

    def resize_image(self, image_full_path, output_dir, image_file,
scale_factor=0.5):
        img = self._load_image(image_full_path)
        if img:
            new_width = int(img.width * scale_factor)
            new_height = int(img.height * scale_factor)
            resample_filter = Image.Resampling.LANCZOS if scale_factor < 1.0
else Image.Resampling.BICUBIC
            img_resized = img.resize((new_width, new_height),
resample=resample_filter)
            self._save_image(img_resized, output_dir, image_file,
f"_escalada_{scale_factor:.1f}x")
```

process_dataset(): Este es el método central que recorre el dataset de emociones.

Para cada emoción, lista las imágenes, las carga, guarda una copia de la imagen original en la carpeta de salida, y luego aplica todas las transformaciones configuradas (brillo, rotación, escala) a esa imagen, guardando cada nueva versión.

```
def process_dataset(self, brightness_factors=None, rotation_angles=None,
scale_factors=None):
        for emotion_dir in os.listdir(self.input_dataset_root):
```

```
            current_input_path = os.path.join(self.input_dataset_root,
emotion_dir)
            current_output_path = os.path.join(self.output_dataset_root,
emotion_dir)

            if os.path.isdir(current_input_path):
                print(f"\n--- Procesando emoción: {emotion_dir} ---")
                image_files = [f for f in os.listdir(current_input_path) if
f.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp', '.gif'))]

                if not image_files:
                    print(f"No se encontraron imágenes en
'{current_input_path}'.")
                    continue

                for image_file in image_files:
                    image_full_path = os.path.join(current_input_path,
image_file)
                    original_img = self._load_image(image_full_path)
                    if original_img:
                        self._save_image(original_img, current_output_path,
image_file, "")
                    if brightness_factors:
                        for factor in brightness_factors:
                            self.adjust_brightness(image_full_path,
current_output_path, image_file, factor)
                    if rotation_angles:
                        for angle in rotation_angles:
                            self.rotate_image(image_full_path,
current_output_path, image_file, angle)
                    if scale_factors:
                        for factor in scale_factors:
                            self.resize_image(image_full_path,
current_output_path, image_file, factor)
```

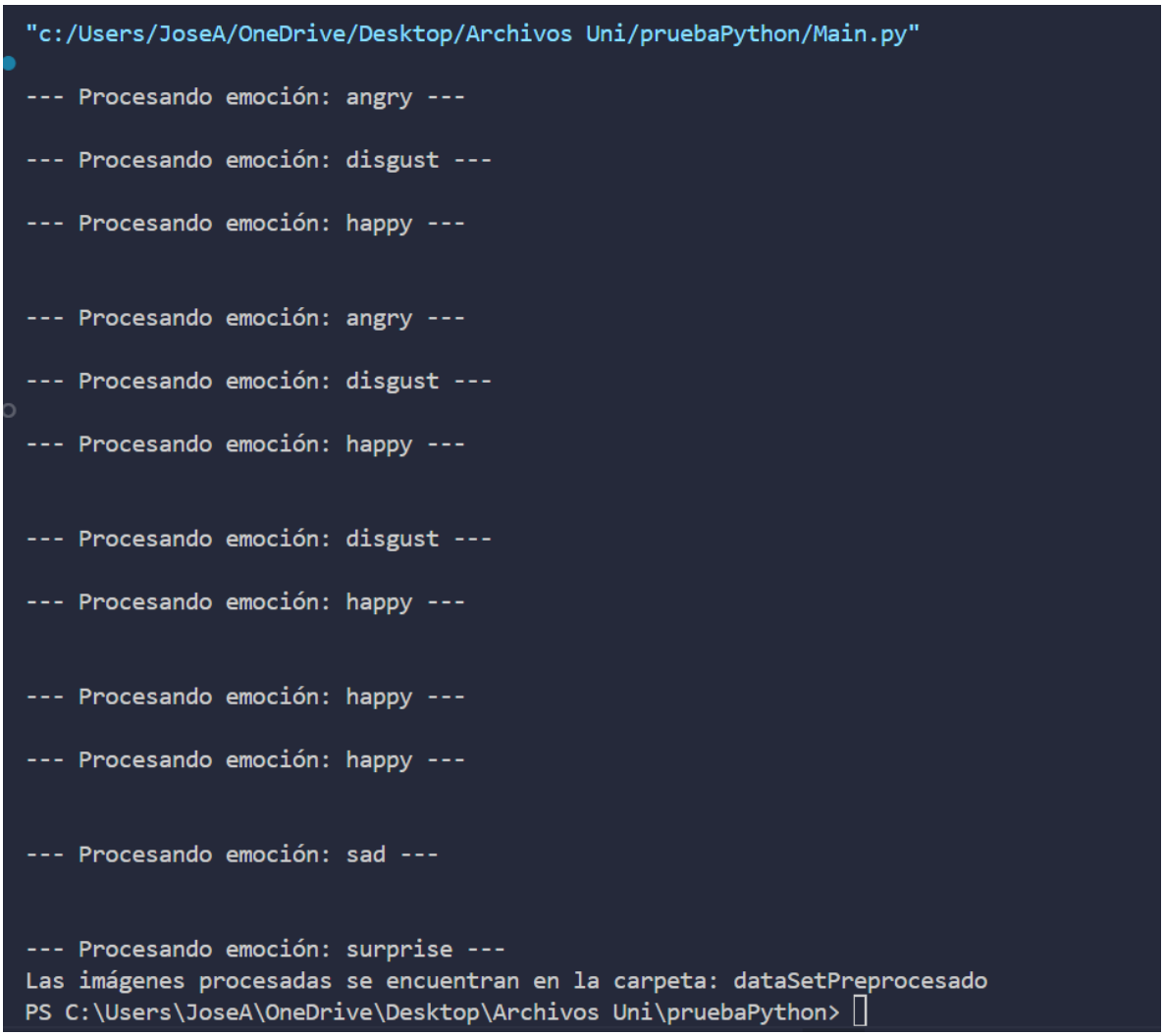
Aquí configuramos las rutas de las carpetas de dataset (dataSetImagenes, dataSetPreprocesado).

Se definen las listas de parámetros para cada transformación (qué brillos, qué ángulos, qué escalas se aplicaran).

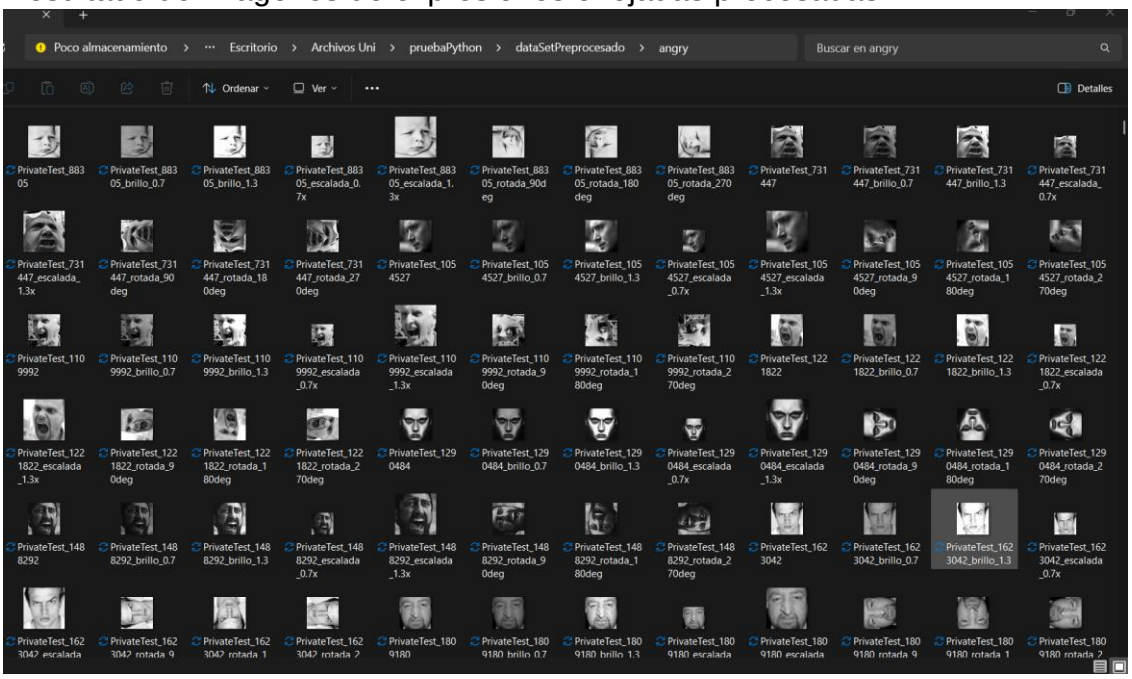
Creamos una instancia de ImagePreprocessor y llamamos a processor.process_dataset() para iniciar todo el trabajo.

```
if __name__ == "__main__":
    input_data_root = "dataSetImagenes"
    output_data_root = "dataSetPreprocesado"

    brightness_factors_to_apply = [0.7, 1.3]
    rotation_angles_to_apply = [90, 180, 270]
    scale_factors_to_apply = [0.7, 1.3]

    processor = ImagePreprocessor(input_data_root, output_data_root)

    processor.process_dataset(
        brightness_factors=brightness_factors_to_apply,
        rotation_angles=rotation_angles_to_apply,
        scale_factors=scale_factors_to_apply
    )

    print(f"Las imágenes procesadas se encuentran en la carpeta:
{output_data_root}")
```

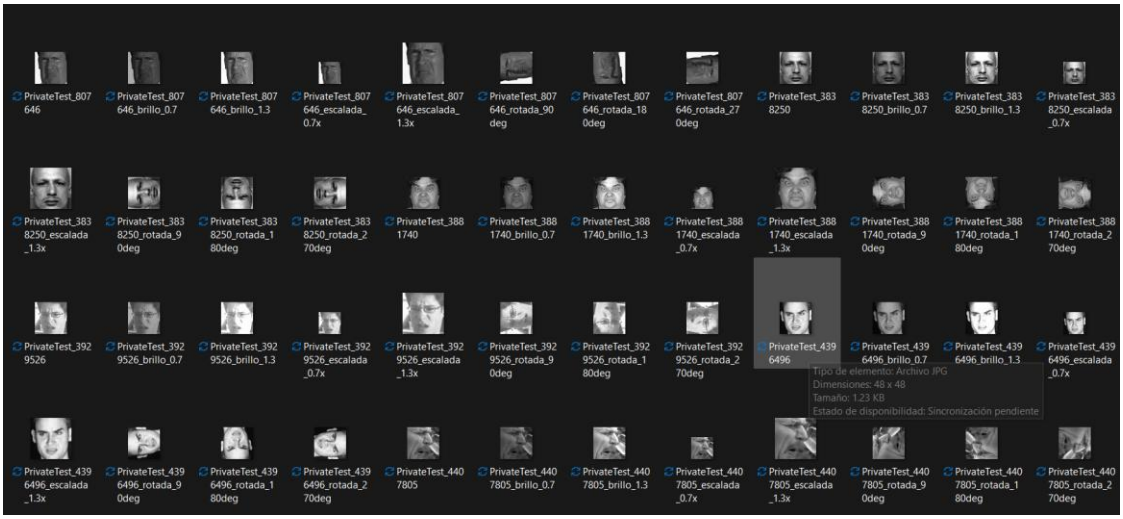
Imágenes de la ejecución:



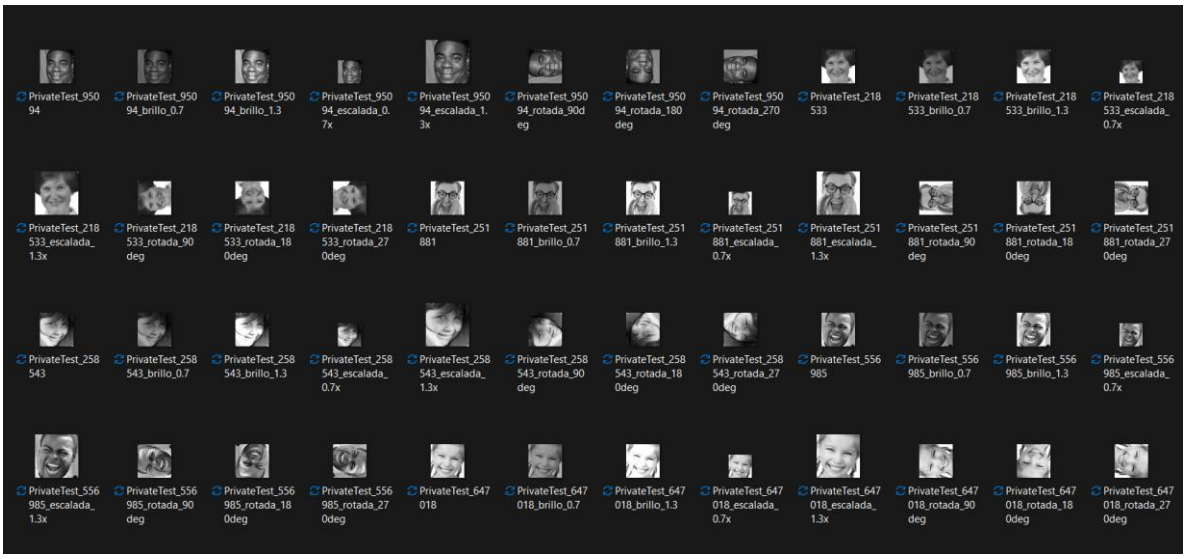
Resultado de imágenes de expresiones enojadas procesadas:



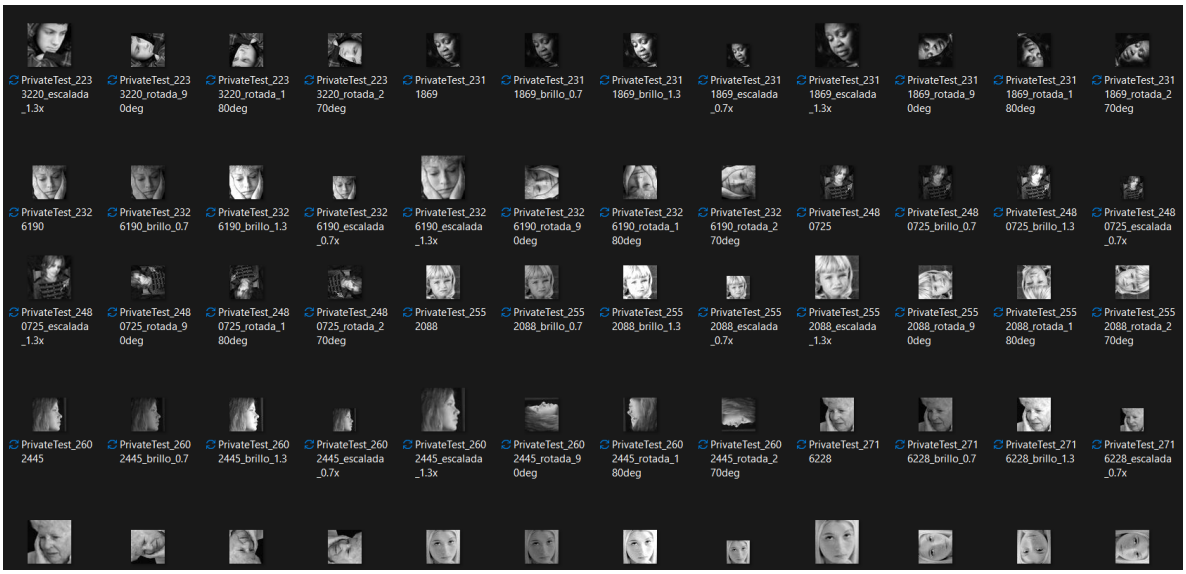
Resultado de imágenes de expresiones disgustadas (usamos está expresión en vez de angustiadas) procesadas:



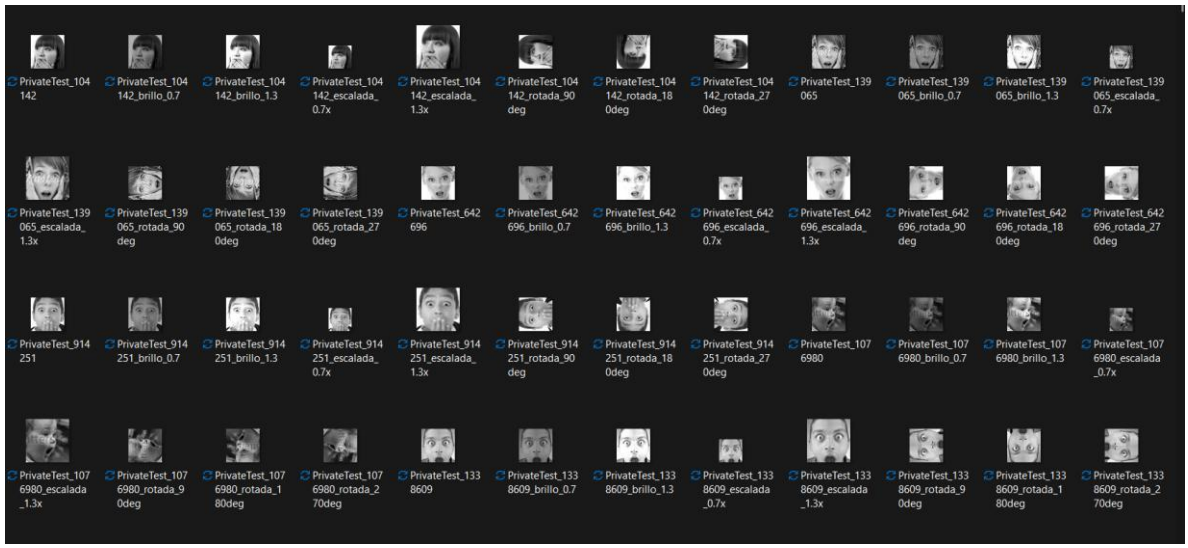
Resultado de imágenes de expresiones felices procesadas:



Resultado de imágenes de expresiones tristes procesadas:



Resultado de imágenes de expresiones sorprendidas procesadas:



Código completo:

```
from PIL import Image, ImageEnhance
import numpy as np
import os

class ImagePreprocessor:
    def __init__(self, input_dataset_root,
output_dataset_root="processed_dataset"):
        self.input_dataset_root = input_dataset_root
        self.output_dataset_root = output_dataset_root
        os.makedirs(output_dataset_root, exist_ok=True)

    def _load_image(self, image_full_path):
        try:
            return Image.open(image_full_path).convert("RGB")
        except Exception as e:
            print(f"Error al cargar la imagen {image_full_path}: {e}")
            return None

    def _save_image(self, image, output_dir, original_filename, suffix=""):
        os.makedirs(output_dir, exist_ok=True)
        name, ext = os.path.splitext(original_filename)
        output_filename = f"{name}{suffix}{ext}"
        output_filepath = os.path.join(output_dir, output_filename)
        try:
            image.save(output_filepath)
        except Exception as e:
            print(f"Error al guardar la imagen {output_filepath}: {e}")

    def adjust_brightness(self, image_full_path, output_dir, image_file,
factor=1.5):
        img = self._load_image(image_full_path)
        if img:
            enhancer = ImageEnhance.Brightness(img)
            img_bright = enhancer.enhance(factor)
            self._save_image(img_bright, output_dir, image_file,
f"_brillo_{factor:.1f}")

    def rotate_image(self, image_full_path, output_dir, image_file,
angle=90, expand=True):
        img = self._load_image(image_full_path)
        if img:
            img_rotated = img.rotate(angle, expand=expand)
            self._save_image(img_rotated, output_dir, image_file,
f"_rotada_{angle}deg")
```

```
    def resize_image(self, image_full_path, output_dir, image_file,
scale_factor=0.5):
        img = self._load_image(image_full_path)
        if img:
            new_width = int(img.width * scale_factor)
            new_height = int(img.height * scale_factor)
            resample_filter = Image.Resampling.LANCZOS if scale_factor < 1.0
else Image.Resampling.BICUBIC
            img_resized = img.resize((new_width, new_height),
resample=resample_filter)
            self._save_image(img_resized, output_dir, image_file,
f"_escalada_{scale_factor:.1f}x")

    def process_dataset(self, brightness_factors=None, rotation_angles=None,
scale_factors=None):
        for emotion_dir in os.listdir(self.input_dataset_root):
            current_input_path = os.path.join(self.input_dataset_root,
emotion_dir)
            current_output_path = os.path.join(self.output_dataset_root,
emotion_dir)

            if os.path.isdir(current_input_path):
                print(f"\n--- Procesando emoción: {emotion_dir} ---")
                image_files = [f for f in os.listdir(current_input_path) if
f.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp', '.gif'))]

                if not image_files:
                    print(f"No se encontraron imágenes en
'{current_input_path}'.")
                    continue

                for image_file in image_files:
                    image_full_path = os.path.join(current_input_path,
image_file)

                    original_img = self._load_image(image_full_path)
                    if original_img:
                        self._save_image(original_img, current_output_path,
image_file, "")

                    if brightness_factors:
                        for factor in brightness_factors:
                            self.adjust_brightness(image_full_path,
current_output_path, image_file, factor)
```

```
                    if rotation_angles:
                        for angle in rotation_angles:
                            self.rotate_image(image_full_path,
current_output_path, image_file, angle)
                    if scale_factors:
                        for factor in scale_factors:
                            self.resize_image(image_full_path,
current_output_path, image_file, factor)

if __name__ == "__main__":
    input_data_root = "dataSetImagenes"
    output_data_root = "dataSetPreprocesado"

    brightness_factors_to_apply = [0.7, 1.3]
    rotation_angles_to_apply = [90, 180, 270]
    scale_factors_to_apply = [0.7, 1.3]

    processor = ImagePreprocessor(input_data_root, output_data_root)

    processor.process_dataset(
        brightness_factors=brightness_factors_to_apply,
        rotation_angles=rotation_angles_to_apply,
        scale_factors=scale_factors_to_apply
    )

    print(f"Las imágenes procesadas se encuentran en la carpeta:
{output_data_root}")
```