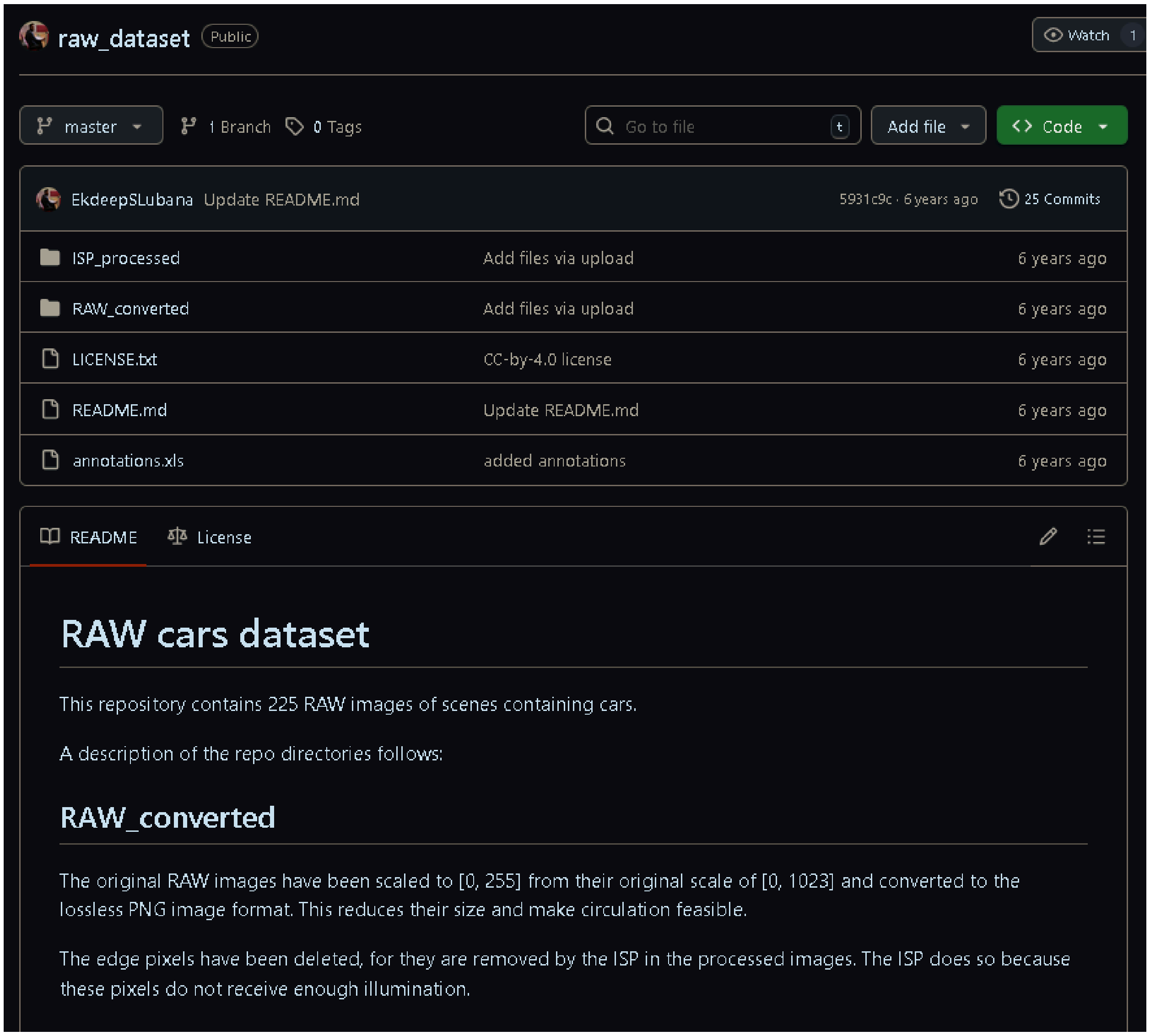
EDGE INTELLIGENCE LAB – 1

DANIEL ROBIN
M.Tech. AI&ML(LTI)
25MML0012

Dataset Used:

https://github.com/EkdeepSLubana/raw_dataset/tree/master



EkdeepSLubana Update README.md 5931c9c · 6 years ago 25 Commits

ISP_processed	Add files via upload	6 years ago
RAW_converted	Add files via upload	6 years ago
LICENSE.txt	CC-by-4.0 license	6 years ago
README.md	Update README.md	6 years ago
annotations.xls	added annotations	6 years ago

RAW cars dataset

This repository contains 225 RAW images of scenes containing cars.

A description of the repo directories follows:

RAW_converted

The original RAW images have been scaled to [0, 255] from their original scale of [0, 1023] and converted to the lossless PNG image format. This reduces their size and make circulation feasible.

The edge pixels have been deleted, for they are removed by the ISP in the processed images. The ISP does so because these pixels do not receive enough illumination.

RAW cars dataset (uncleaned)

- This repository contains 225 RAW images of scenes containing cars.

RAW_converted (the folder that contains the uncleaned images we're using)

- The original RAW images have been scaled to [0, 255] from their original scale of [0, 1023] and converted to the lossless PNG image format. This reduces their size and make circulation feasible.
- For the sake of simplicity, I'm only reading 5 images for processing
- We can see that the 5 images I selected have different shapes

This is just a simple program to resize an image to (150, 150), the final output images may be stretched or compressed

```python
import requests
from PIL import Image
import numpy as np
import io
import time

# --- Configuration ---
# Base URL for the raw image files (RAW_converted directory)
BASE_RAW_URL = "https://raw.githubusercontent.com/EkdeepSLubana/raw_dataset/master/RAW_converted/"
# Define the range of images: 1.png to 5.png
IMAGE_IDS = range(1, 6)

# The uniform size we want for the "Cleaned" dataset
TARGET_SIZE = (150, 150)
cleaned_dataset = []
successful_loads = 0

print(f"Starting download and resize of {len(IMAGE_IDS)} images (.png format)...")

# --- Iterate through URLs, Load, and Resize ---
for img_id in IMAGE_IDS:
    file_name = f"{img_id}.png"
    full_url = BASE_RAW_URL + file_name

    # --- OUTPUT FOR ALL 5 IMAGES ---
    print(f"\nProcessing {file_name}...")

    try:
        response = requests.get(full_url, stream=True, timeout=10)
        response.raise_for_status()

        image_bytes = io.BytesIO(response.content)

        # Convert to 'RGB' to ensure 3 channels
        img = Image.open(image_bytes).convert('RGB')

        # --- OUTPUT FOR ALL 5 IMAGES ---
        # --- OUTPUT FOR ALL 5 IMAGES ---
        print(f"  Original size (Uncleaned): {img.size}")

        resized_img = img.resize(TARGET_SIZE)

        image_array = np.array(resized_img)
        cleaned_dataset.append(image_array)
        successful_loads += 1

        # --- OUTPUT FOR ALL 5 IMAGES ---
        print(f"  Resized size (Cleaned): {resized_img.size}")

    except requests.exceptions.RequestException as e:
        print(f"Skipping {file_name}: Download error: {e}")
    except IOError:
        print(f"Skipping {file_name}: Failed to open or process image.")
    except Exception as e:
        print(f"Skipping {file_name}: An unexpected error occurred: {e}")

# ... (Final Summary) ...
end_time = time.time()
if cleaned_dataset:
    final_data = np.stack(cleaned_dataset)
    print("\n--- Final Dataset Summary ---")
    print(f"Successfully loaded and resized {successful_loads} images in {end_time - start_time:.2f} seconds.")
    print(f"Final Cleaned Dataset Shape (N, H, W, C): {final_data.shape}")
else:
    print("\n--- Final Dataset Summary ---")
    print("No images were successfully loaded due to network issues. The final shape explanation assumes successful execution.")
```

```
Starting download and resize of 5 images (.png format)...

Processing 1.png...
  Original size (Uncleaned): (3480, 4640)
  Resized size (Cleaned): (150, 150)

Processing 2.png...
  Original size (Uncleaned): (3480, 4640)
  Resized size (Cleaned): (150, 150)

Processing 3.png...
  Original size (Uncleaned): (4640, 3480)
  Resized size (Cleaned): (150, 150)

Processing 4.png...
  Original size (Uncleaned): (4640, 3480)
  Resized size (Cleaned): (150, 150)

Processing 5.png...
  Original size (Uncleaned): (4640, 3480)
  Resized size (Cleaned): (150, 150)

--- Final Dataset Summary ---
Successfully loaded and resized 5 images in 4241.64 seconds.
Final Cleaned Dataset Shape (N, H, W, C): (5, 150, 150, 3)
```

- N - This is the batch size, representing the total count of images that were successfully loaded and stacked (1.png through 10.png).
- H - The fixed vertical dimension (height) of every single image, as enforced by the TARGET_SIZE = (150, 150) parameter.
- W - The fixed horizontal dimension (width) of every single image, also enforced by the TARGET_SIZE parameter.
- C - The number of color channels. PNG images typically store color data as RGB (Red, Green, Blue), which requires 3 channels.

All (selected) images have been resized to (150, 150)

POSSIBLE ERRORS

Incorrect File Extension (Should be .png, but requesting .jpg)

```python
url_error_a = BASE_RAW_URL + "1.jpg"
response_a = requests.get(url_error_a)
response_a.raise_for_status() # This will raise an HTTPError (404 Not Found)
print("File A loaded successfully.")
```

```
---------------------------------------------------------------------------
HTTPError                                 Traceback (most recent call last)
Cell In[14], line 3
      1 url_error_a = BASE_RAW_URL + "1.jpg"
      2 response_a = requests.get(url_error_a)
----> 3 response_a.raise_for_status() # This will raise an HTTPError (404 Not Found)
      4 print("File A loaded successfully.")

File ~\anaconda3\envs\mainpro\Lib\site-packages\requests\models.py:1024, in Response.raise_for_status(self)
   1019     http_error_msg = (
   1020         f"{self.status_code} Server Error: {reason} for url: {self.url}"
   1021     )
   1023 if http_error_msg:
-> 1024     raise HTTPError(http_error_msg, response=self)

HTTPError: 404 Client Error: Not Found for url: https://raw.githubusercontent.com/EkdeepSLubana/raw_dataset/master/RAW_converted/1.jpg
```

Attempting to read a non-image file (e.g., README.md)

```python
# NOTE: Need to adjust BASE_RAW_URL to access the root repo directory
base_url_for_readme = "https://raw.githubusercontent.com/EkdeepSLubana/raw_dataset/master/"
url_error_c = base_url_for_readme + "README.md"

response_c = requests.get(url_error_c)
file_bytes_c = io.BytesIO(response_c.content)

# This will raise a PIL.UnidentifiedImageError or similar IOError
img_c = Image.open(file_bytes_c)
print("Image C opened successfully.")
```

```
---------------------------------------------------------------------------
UnidentifiedImageError                    Traceback (most recent call last)
Cell In[15], line 9
      6 file_bytes_c = io.BytesIO(response_c.content)
      8 # This will raise a PIL.UnidentifiedImageError or similar IOError
----> 9 img_c = Image.open(file_bytes_c)
     10 print("Image C opened successfully.")

File ~\anaconda3\envs\mainpro\Lib\site-packages\PIL\Image.py:3498, in open(fp, mode, formats)
   3496     warnings.warn(message)
   3497 msg = "cannot identify image file %r" % (filename if filename else fp)
-> 3498 raise UnidentifiedImageError(msg)

UnidentifiedImageError: cannot identify image file <_io.BytesIO object at 0x0000025D1C25B060>
```

Invalid Resize Target (Needs tuple, giving an int)

- Here we try using 150 for shape instead of (150, 150)

```python
# Assuming 'img' is a successfully loaded image object
if 'img' in locals():
    try:
        # This will raise a TypeError because 150 is not a tuple
        resized_img_d = img.resize(150)
        print("Resize D successful.")
    except TypeError as e:
        print(f"Error D: {e}")
```

```
Error D: argument 1 must be 2-item sequence, not int
```

Here the images we use are different sizes, resulting in error

Attempting to Stack Unresized Images ¶

```python
# Load two images (1.png and 5.png) of different original sizes:
img_1_url = BASE_RAW_URL + "1.png"
img_5_url = BASE_RAW_URL + "5.png"

img_1 = Image.open(io.BytesIO(requests.get(img_1_url).content)).convert('RGB')
img_5 = Image.open(io.BytesIO(requests.get(img_5_url).content)).convert('RGB')

uncleaned_list = [np.array(img_1), np.array(img_5)]

# This will raise a ValueError because the arrays (images) have different shapes
np.stack(uncleaned_list)
print("Stack E successful.") # This line will not be reached
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[17], line 11
      8 uncleaned_list = [np.array(img_1), np.array(img_5)]
     10 # This will raise a ValueError because the arrays (images) have different shapes
---> 11 np.stack(uncleaned_list)
     12 print("Stack E successful.")

File ~\anaconda3\envs\mainpro\Lib\site-packages\numpy\_core\shape_base.py:457, in stack(arrays, axis, out, dtype, casting)
    455 shapes = {arr.shape for arr in arrays}
    456 if len(shapes) != 1:
--> 457     raise ValueError('all input arrays must have the same shape')
    459 result_ndim = arrays[0].ndim + 1
    460 axis = normalize_axis_index(axis, result_ndim)

ValueError: all input arrays must have the same shape
```