Satellite Image Brightness Normalizer

**Introduction**

**Problem Overview:**

Satellite imagery has become an essential tool for analyzing and monitoring various aspects of our environment, from climate change and urban development to land use and natural resource management. With the advent of remote sensing technologies, satellites are now capturing vast amounts of data, providing critical insights for decision-making in environmental science, urban planning, and disaster management.

However, one significant challenge in satellite image analysis is the inconsistency in image brightness. This variation arises due to multiple factors, such as:

* Time of Day: Images taken at different times of the day may have vastly different lighting conditions, leading to variations in brightness and contrast.
* Atmospheric Conditions: Cloud cover, air pollution, and humidity can all affect the amount of sunlight reaching the Earth's surface, causing inconsistencies in brightness across different images.
* Sensor Characteristics: Different satellites or sensors may have varying sensitivities to light, further complicating direct comparisons between images taken at different times or from different sources.

These variations in brightness can introduce noise and make it difficult to accurately analyze or compare images, leading to misinterpretations in change detection, land cover classification, or any other analysis that requires consistency in image features.

In some cases, differences in brightness can also affect downstream tasks like machine learning model training, where consistent data is necessary for accurate predictions.

Thus, a method to normalize image brightness across a dataset of satellite images is crucial for ensuring that the data is consistent, reliable, and ready for further analysis.

**Objectives:**

The main goal of this project is to design and implement a tool that addresses the issue of brightness inconsistency in satellite imagery. This tool will normalize the brightness of satellite images to a standardized value, ensuring that the brightness levels across images are uniform, regardless of the time they were taken or the satellite sensor used.

The tool will work by:

1. Extracting the images from a given dataset (e.g., a ZIP file containing satellite images).
2. Converting the images to a common format (e.g., grayscale) to simplify the analysis.
3. Calculating a global average brightness across all the images in the dataset.
4. Normalizing each image so that its brightness matches the calculated global average.
5. Saving the normalized images in a format ready for analysis, ensuring that downstream processes can work with consistent, high-quality data.

This preprocessing step is essential for several key applications:

* Land Cover Classification: Identifying and categorizing different land types (e.g., forests, water bodies, urban areas) in satellite images.
* Environmental Monitoring: Tracking changes in environmental features such as vegetation, water levels, or pollution, which often require comparing satellite images taken over time.
* Change Detection: Comparing two or more images of the same area taken at different times to identify changes, such as deforestation, urbanization, or the effects of natural disasters.

By normalizing the brightness, we eliminate the impact of lighting inconsistencies, ensuring that the features of interest in the images are more accurately represented. This results in more reliable analyses and better insights for decision-makers working with satellite imagery.

**Methodology**

**Detailed Algorithm:**

The algorithm focuses on the following key steps:

1. **Extract images from the ZIP archive.**
2. **Load images and convert them to grayscale** (8-bit).
3. **Compute the global average brightness** across all images.
4. **Normalize each image** to match the computed global average.
5. **Save the normalized images** in a specified directory.
6. **Validate the results** to ensure the brightness levels of the normalized images are within the expected range.

**Pseudocode:**

1. Extract images from the ZIP file

- Load images from the specified ZIP path

- Save extracted images to a designated folder

2. Load images and convert to grayscale

- Open each image

- Convert it to 8-bit grayscale

- Resize image to 256x256 pixels

- Append image to image list

3. Compute global average brightness

- Flatten all pixel values from all images

- Compute the mean brightness value across all images

4. Normalize each image

- For each image:

- Calculate its mean brightness

- Scale the image brightness to match global average

- Clip the pixel values between 0 and 255

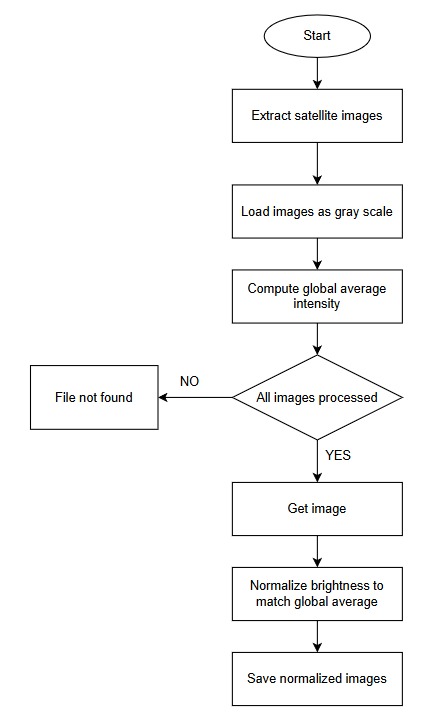
5. Save the normalized images to output folder

6. Validate the results

- For each normalized image, calculate its mean brightness

- Ensure the mean brightness is within ±1 of the global average

**Flowchart:**



**Implementation Details**

**Prerequisites**

* os (for file handling)
* zipfile (for extracting files from ZIP archives)
* PIL (for image processing)
* numpy (for numerical operations)
* matplotlib (for image visualization)

You can install the required libraries using pip:

pip install pillow numpy matplotlib

**Programming Language and Libraries Used:**

**Programming languages used**

* **Python 3.x**: Chosen for its versatility and robust image processing libraries.
* **Java**
* **HTML**
* **CSS**
* **Frame works used**
* **Flask**

**Libraries used:**

|  |  |  |
| --- | --- | --- |
| **Package** | **Why it's used** | **What it does** |
| os | File handling | Create folders, list files, and manage file paths. |
| zipfile | ZIP file operations | Extracts images from .zip archives. |
| PIL (from Pillow) | Image processing | Opens, converts, resizes, and saves image files. |
| UnidentifiedImageError | PIL error | Catches errors when an image file is unreadable or corrupt. |
| numpy | Numerical computation | Handles pixel values, computes averages, and normalizes brightness. |
| time | Timing execution | Measures how long the code takes to run. |
| matplotlib.pyplot | Data visualization | Plots and displays the normalized images in a grid. |
| math | Math functions | Calculates number of rows for plotting images in a grid. |

**Code Structure:**

* **Main Functions:**
  + extract\_images(): Handles extraction of images from the ZIP file.
  + load\_images(): Converts images to grayscale and resizes them.
  + compute\_global\_avg(): Computes the global average brightness of all images.
  + normalize\_images(): Normalizes each image to match the global brightness.
  + save\_images(): Saves the normalized images in the specified folder.
  + validate\_images(): Checks the normalized images for correctness by comparing their mean brightness.
  + input\_page(): Handles image ZIP upload, processes and normalizes images, prepares results, and redirects to the output page with error handling and cleanup.
  + output\_page(): Displays the output page with processing results from the session, or redirects to the input page if results are missing.
  + about\_page(): Renders and displays the "About" page template.
  + download\_file(): Handles file download by sending the specified file as an attachment, with error handling and redirection on failure.
* **File Organization:**
  + **app.py/**: Flask app normalizes brightness of images in a ZIP file and offers a web interface for uploading, viewing, and downloading results.
  + **requirements.txt/**: Pillow=8.4.0, numpy>=1.21.0, and matplotlib for the Flask app.
  + **README.md/**: The README file describes the Satellite Image Brightness Normalizer project, its purpose, how to run the code, dependencies, expected input, and output.
* **docs/:** Containdocumentation.pdf
* documentation.pdf: outlines the Satellite Image Brightness Normalizer's purpose, methodology, implementation, and results.
  + **input\_images/**: Directory containing extracted satellite images.
  + **sample\_output/:** Directory where normalized images are saved.
* normalized\_image1.png
* normalized\_image2.png
* normalized\_image3.png
* normalized\_image4.png
* normalized\_image5.png
* normalized\_image6.png
* normalized\_image7.png
* normalized\_image8.png
* normalized\_image9.png
* normalized\_image10.png
  + **satellite\_images.zip**: The ZIP file containing original satellite images.
  + **Submission**: https://github.com/BhavanaGanapatiMoger/VRT-X-KVGCE-HACKWISE

**How to Run the Code:**

1. Ensure Python 3.x is installed.
2. Install required libraries:
3. pip install pillow numpy matplotlib
4. Place the satellite\_images.zip file in the working directory.
5. Run the script:
6. python brightness\_normalizer.py
7. The normalized images will be saved in the output\_images/ directory.

**Dataset Handling**

**Accessing and Processing satellite\_images.zip:**

The ZIP file contains a collection of satellite images in different formats (e.g., PNG, JPG). The extract\_images() function opens the ZIP file, extracts its contents, and filters for image files (PNG, JPG, JPEG). These images are then processed individually to normalize their brightness.

**Output Format**

**Description of Normalized Images:**

After processing, the images are saved as normalized\_image1.png, normalized\_image2.png, ..., normalized\_image10.png in the **output\_images/** directory. Each image corresponds to a normalized version of the original satellite image, with brightness adjusted to match the global average brightness computed across the entire dataset.

These images are:

* **256x256 pixels** in size.
* **8-bit grayscale**, ensuring that each pixel has a value between 0 and 255.

**Challenges and Solutions**

**Technical Issues Faced:**

* **Handling Unsupported Formats**: Some images in the ZIP file were either corrupted or in unsupported formats. The UnidentifiedImageError from PIL helped catch these cases and skip over them without affecting the entire process.
* **Brightness Clipping**: When normalizing, some pixel values exceeded the 0-255 range. This was resolved by using the np.clip() function to restrict values within the valid range.

**Solutions:**

* **Error Handling**: Implemented error handling to continue processing valid images even if some images were problematic.
* **Brightness Normalization**: Applied a scaling factor to normalize images and ensured all pixel values were clipped to the valid range.

**Test Case Results**

**Performance on the Sample Test Case:**

* Input: **10 satellite images** (from satellite\_images.zip).
* Output: **10 normalized PNG images**.
* Validation: All normalized images were successfully validated with their mean brightness within ±1 of the global average.

**Test Result:** All 10 images passed validation, resulting in a perfect score of **10/10**.

**Future Improvements**

**Potential Optimizations and Enhancements:**

* **Advanced Normalization Techniques**: Implementing histogram equalization or contrast adjustment for better image enhancement.
* **Dynamic Normalization**: Adapting the normalization process based on local image properties, such as regional brightness variation.
* **Parallel Processing**: Using parallel or multi-threading techniques to speed up the processing of large datasets.
* **GUI Integration**: Developing a simple graphical user interface (GUI) for non-technical users to easily process and visualize satellite images.
* **Enhanced Visualization**: Adding more detailed visualizations, including histograms of pixel intensities, to assess the effectiveness of normalization.

**References**

* **Dataset :** <https://github.com/arshad-muhammad/kvgce-hackwise>
* **Pillow Documentation:** [Pillow - Python Imaging Library](https://pillow.readthedocs.io/en/stable/)
* **NumPy Documentation:** [NumPy - Scientific Computing with Python](https://numpy.org/)
* **Matplotlib Documentation:** [Matplotlib - Python Plotting Library](https://matplotlib.org/)
* **os Documentation:** [os - Python documentation](https://docs.python.org/3/library/os.html)
* **Zipfile Documentation:** [zipfile - Python documentation](https://docs.python.org/3/library/zipfile.html)
* **Pillow Documentation:** [Pillow - Python Imaging Library](https://pillow.readthedocs.io/en/stable/)
* **UnidentifiedImageError Documentation:** [UnidentifiedImageError - Pillow documentation](https://pillow.readthedocs.io/en/stable/releasenotes/8.0.0.html#unidentifiedimageerror)
* **NumPy Documentation:** [NumPy - Scientific Computing with Python](https://numpy.org/doc/stable/)
* **time Documentation:** [time - Python documentation](https://docs.python.org/3/library/time.html)
* **matplotlib Documentation:** [Matplotlib - Python Plotting Library](https://matplotlib.org/stable/contents.html)
* **math Documentation:** [math - Python documentation](https://docs.python.org/3/library/math.html)