Mini Project 1: Dataset Curation & Preprocessing

Low-Light Image Enhancement Dataset Preparation

Benjamin Tran
CS 5190 Computer Vision

November 4th, 2025

1 Problem Statement

This mini project addresses the dataset preparation phase for a final group project on Hybrid Low-Light Image Enhancement. The goal is to collect, preprocess, and organize low-light image datasets to support both classical enhancement methods (histogram equalization, Retinex) and machine learning approaches. The primary challenge is creating a comprehensive, well-structured dataset with proper preprocessing, augmentation, and train/test splits that enables fair comparison between different enhancement techniques.

The preprocessing pipeline must handle diverse image formats, normalize pixel values, apply synthetic darkening for data augmentation, and organize images for efficient training and evaluation in subsequent project phases.

2 Dataset

2.1 Dataset Source

The **LOL (Low-Light) Dataset** was selected as the primary dataset for this project. The dataset structure includes:

- **Training Set (`our485/`)**: 485 image pairs (low-light and corresponding high-light ground truth)
- **Evaluation Set (`eval15/`)**: 15 image pairs for final evaluation

The dataset contains real-world low-light images captured under various lighting conditions, making it suitable for testing enhancement algorithms. Each image pair consists of a low-light version and its corresponding well-lit ground truth, enabling supervised learning approaches.

2.2 Dataset Organization

The dataset was organized with the following structure:

```
lol_dataset/
          our485/
                low/   (485 low-light images)
                high/  (485 ground truth images)
          eval15/
              low/   (15 low-light images)
              high/  (15 ground truth images)
```

For this preprocessing phase, we focused on the low-light images from the `low/` directories, which serve as input for enhancement algorithms.

3 Challenges

Several challenges were encountered during dataset curation and preprocessing:

1. **Large Dataset Size**: The original dataset contains 1000 images (485 training + 485 ground truth + 15 eval + 15 ground truth). Processing and augmenting these images required efficient memory management and batch processing.
2. **Inconsistent Image Dimensions**: Original images had varying resolutions, requiring standardization to a uniform size (512×512) while maintaining aspect ratio through proper interpolation.
3. **Data Augmentation Strategy**: Creating realistic synthetic low-light variations required careful parameter tuning (darkening factors, gamma correction, noise levels) to maintain image quality while increasing dataset diversity.
4. **Memory Constraints**: With augmentation creating 9 additional versions per image, the processed dataset grows to 7,969 images, requiring careful disk space management.
5. **Train/Test Split**: Ensuring proper data distribution while maintaining the original evaluation set (`eval15`) as a held-out test set required careful splitting strategy.
6. **Normalization**: Choosing appropriate normalization method (standard normalization vs. min-max) that preserves image characteristics while improving model convergence.

4 Preprocessing

A comprehensive preprocessing pipeline was implemented using Python with OpenCV, NumPy, and scikit-learn. The pipeline consists of four main stages:

4.1 Image Resizing

All images were resized to a uniform size of **512×512 pixels** using bilinear interpolation. This standardization ensures:

- Consistent input dimensions for all models
- Reduced computational requirements
- Maintained aspect ratio through proper interpolation

4.2 Normalization

Images were normalized using **standard normalization** (zero mean, unit variance):

- Formula: $(x - \mu)/\sigma$
- Results scaled to [0, 255] range for compatibility with standard image processing libraries
- This normalization improves model convergence and stabilizes training

4.3 Data Augmentation

Synthetic darkening augmentation was applied to increase dataset diversity and robustness. The augmentation process:

- Creates **9 additional variations** per original image
- Uses multiple darkening factors (0.2, 0.3, 0.4)
- Applies gamma correction with values (1.8, 2.0, 2.2)
- Adds controlled noise to simulate real-world low-light conditions

This augmentation strategy significantly expands the training dataset while maintaining realistic low-light characteristics.

4.4 Train/Test Split

The dataset was split using an **80/20 ratio**:

- **Training set**: 80% of images (with augmentation applied)
- **Validation set**: 20% of images (no augmentation, for unbiased evaluation)
- **Test set**: 15 images from `eval15` (held-out evaluation set)

The split was performed using scikit-learn's `train_test_split` with a fixed random seed (42) for reproducibility.

5 Model Training

5.1 Preprocessing Pipeline Implementation

The preprocessing pipeline was implemented as a modular Python class (`LowLightDatasetPreprocessor`) with the following components:

- **Resizing Module**: Handles image resizing with configurable target dimensions
- **Normalization Module**: Implements standard and min-max normalization methods
- **Augmentation Module**: Generates synthetic darkening variations with configurable parameters
- **Data Loader**: Processes images in batches with progress tracking
- **Split Manager**: Handles train/test/validation splits with reproducible random seeds

5.2 Processing Workflow

The training data preparation workflow follows these steps:

1. Image discovery: Recursively search for image files in input directory
2. Train/test split: Divide images into training and validation sets
3. Training set processing: For each training image:
    - Resize to 512×512
    - Apply standard normalization

- Generate 9 augmented versions (synthetic darkening)
- Save all 10 versions to training directory

4. Validation set processing: For each validation image:
   - Resize to 512×512
   - Apply standard normalization
   - Save single version (no augmentation)
5. Test set processing: Process `eval15` images separately

5.3 Implementation Details

Tools and Libraries:

- **Python 3.11**
- **OpenCV**: Image processing and manipulation
- **NumPy**: Numerical operations
- **scikit-learn**: Train/test splitting
- **Matplotlib**: Visualization
- **tqdm**: Progress tracking

6 Results

6.1 Dataset Statistics

Final Processed Dataset:

- **Original Training Images**: 776 images (from `our485/low`)
- **Processed Training Images**: 7,760 images (with augmentation)
  - Each original image generates 10 versions (1 original + 9 augmented)
- **Validation Images**: 194 images (no augmentation)
- **Test Images**: 15 images (from `eval15`)
- **Total Processed Images**: 7,969 images

Image Specifications:

- Resolution: 512 × 512 pixels
- Format: JPEG
- Color space: RGB (BGR in OpenCV)
- Normalization: Standard normalization applied

6.2 Output Structure

The processed dataset is organized as follows:

```
processed_dataset/
          train/              # 7,760 training images
                image1.jpg
                image1_aug1.jpg
                image1_aug2.jpg
                ...
          test/               # 209 test images
                validation images (194)
                eval_*.png (15)
          visualizations/     # Preprocessing examples
                preprocessing_examples_train.png
                preprocessing_examples_test.png
          dataset_stats.json  # Dataset statistics
```

6.3 Visual Deliverables

The following visualizations were generated:

1. **Preprocessing Examples**: Side-by-side comparisons showing original, resized, normalized, and synthetically darkened images
2. **Dataset Statistics**: Statistical analysis including:
   - Brightness distribution histograms
   - RGB channel statistics
   - Image size distribution
   - Train/test split visualization

6.4 Key Achievements

- Successfully processed 7,969 images with consistent preprocessing
- Created comprehensive dataset with 10× augmentation for training
- Maintained proper train/test/validation splits for unbiased evaluation
- Generated visual documentation of preprocessing steps
- Created reusable, modular preprocessing pipeline
- Dataset ready for both classical and ML-based enhancement methods

7 Conclusion

The dataset curation and preprocessing mini project has been successfully completed. The LOL dataset has been processed, augmented, and split into appropriate train/test sets. The preprocessing pipeline includes resizing, normalization, and synthetic darkening augmentation, resulting in a dataset of 7,969 processed images ready for the final group project on Hybrid Low-Light Image Enhancement.

The dataset is properly organized, documented, and includes visual examples demonstrating each preprocessing step. All code is modular and reusable, allowing for easy extension and modification as needed for the final project implementation. The preprocessed dataset is now

ready for integration with classical enhancement methods (histogram equalization, Retinex) and machine learning approaches in the subsequent project phases.

Project Files:

- `dataset_preprocessing.py`: Main preprocessing script
- `process_lol_dataset.py`: LOL-specific processing script
- `visualize_dataset.py`: Statistical visualization tool
- `README.md`: Project documentation
- `requirements.txt`: Python dependencies