Advanced Image Preprocessing and Augmentation for Deep Learning (DL)

Assignment Objective

The goal of this assignment is to implement a robust image preprocessing and augmentation pipeline that mimics the workflow necessary to prepare raw images for training a Convolutional Neural Network (CNN). You will explore fundamental techniques across filtering, color space transformation, noise management, and data augmentation.

Required Setup and Tools

- **Programming Language:** Python 3.x
- **Libraries:**
  - `numpy` (for array manipulation)
  - `opencv-python` (`cv2`) or `Pillow` (`PIL`) (for core image operations)
  - `matplotlib` (for visualization)
- **Input Data:** Use a sample set of 5–10 diverse images (e.g., common images like flowers, animals, or general scenes).

Task 1: Essential Preprocessing and Normalization

These steps ensure all input images conform to the requirements of a neural network model architecture (e.g., VGG, ResNet).

1. **Uniform Resizing:**
   - Implement a function that takes an input image and resizes it to a fixed dimension, W×H. Use 224×224 pixels as the target size.
   - Ensure the function can handle images with different initial aspect ratios (you may choose to stretch the image or pad it, but stretching is often the simplest initial approach).
2. **Color Channel Format Conversion:**
   - Verify and, if necessary, convert the image color ordering to the standard format required by most DL frameworks (RGB: Red, Green, Blue). Note: OpenCV often uses BGR by default, so conversion might be necessary.
3. **Data Type and Normalization:**

    - Convert the image array data type to floating-point (e.g., `float32`).
    - Implement **Min-Max Normalization**: Scale the pixel values from the standard range to the floating-point range of [0.0,1.0].
    - *Challenge (Optional):* Implement **Standardization** (Z-score normalization) using pre-calculated global mean (μ) and standard deviation (σ) across all training data (for this assignment, you can calculate the μ and σ of your 5–10 sample images).

Task 2: Filtering and Noise Management

Noise is common in real-world data. Filtering is critical for both removing noise and highlighting certain image features.

1. **Noise Addition:**
    - Implement a function to deliberately corrupt an image by adding **Salt-and-Pepper Noise**.
    - Implement a function to corrupt an image by adding **Gaussian Noise** (additive white noise).
    - Apply both noise types to the same original image, creating two distinct corrupted versions.
2. **Noise Removal (Denoising):**
    - Apply a **Median Filter** (e.g., 3×3 or 5×5 kernel) to the **Salt-and-Pepper Noise** corrupted image.
    - Apply a **Gaussian Blur/Filter** (e.g., σ=1.0 or 2.0) to the **Gaussian Noise** corrupted image.
3. **Quantitative Evaluation:**
    - Calculate the **Peak Signal-to-Noise Ratio (PSNR)** and **Mean Squared Error (MSE)** between: a) The original image and the noisy image. b) The original image and the denoised image (after filtering).
    - Compare the metrics to demonstrate the effectiveness of the chosen filters.

Task 3: Color Transformation and Data Augmentation

Data augmentation is a crucial preprocessing technique to prevent overfitting in DL models.

1. **Color Space Transformation:**
    - Implement a function to convert the image from **RGB to Grayscale**. (Grayscale images are often used in certain CNN tasks, or as a channel for color augmentation).
2. **Hue and Saturation Adjustment:**

    - Convert an image from **RGB to the HSV (Hue, Saturation, Value) color space**.
    - Implement an operation to **randomly adjust the Saturation** of the image by a factor between 0.5 and 1.5.
    - Convert the image back from HSV to RGB for display.
3. **Geometric Augmentation Pipeline:**
    - Create a single augmentation function that randomly applies the following techniques with a 50% probability for each: a) **Horizontal Flip:** Mirror the image along the vertical axis. b) **Random Rotation:** Rotate the image by a small random angle between -10° and $+10^{\circ}$. Fill any resulting empty pixels with a neutral color (e.g., the image mean).

Deliverables

1. **Python Script:** A well-commented Python file (`preprocessing_pipeline.py`) containing all implemented functions and a demonstration of each task.
2. **Visual Report:** A set of visualizations (using `matplotlib`) showing the results for at least one sample image, including:
    - Original Image.
    - Resized and Normalized Image.
    - Image with Salt-and-Pepper Noise and the Median Filtered Result (with PSNR/MSE comparison).
    - Image with Gaussian Noise and the Gaussian Filtered Result (with PSNR/MSE comparison).
    - Image after Saturation Adjustment.
    - Image after Random Geometric Augmentation.