Libraries Used in the Project

1. NumPy

- **Purpose**: Handles numerical operations efficiently.
- **Use in Dehazing**:
  - Used for array manipulations (e.g., converting images into numerical arrays).
  - Helps in performing matrix operations for haze removal algorithms.
  - Used in pre-processing and normalizing image pixel values.

2. Matplotlib

- **Purpose**: Visualization and debugging tool.
- **Use in Dehazing**:
  - Used for displaying original and dehazed images.
  - Helps visualize loss curves and training progress if using deep learning models.
  - Can be used to generate histograms of image pixel intensities for analysis.

3. PyTorch

- **Purpose**: Deep learning framework for building and training models.
- **Use in Dehazing**:
  - Used for implementing **CNNs or GANs** for learning haze removal.
  - Provides tensor operations and GPU acceleration for training large datasets.
  - Helps in implementing custom **loss functions** for image restoration tasks.

4. PyTorch Lightning (PTL)

- **Purpose**: Simplifies PyTorch training and scaling.
- **Use in Dehazing**:
  - Used for structured and efficient training of deep learning models.
  - Handles **distributed training, automatic checkpointing, and logging**.

- Reduces boilerplate code while maintaining flexibility in training loop modifications.

Image Dataset for AOD-Net Training

Dataset Overview: This dataset is designed for training the AOD-Net model for image dehazing. It consists of two primary folders:

1. **gt/** (Ground Truth): Contains clear, high-quality images without haze.
2. **hazy/**: Contains the corresponding hazy versions of the images in the gt folder.

Dataset Structure:

- Each image in the **hazy/** folder has a corresponding clear image in the **gt/** folder with the same filename.
- Images are aligned to ensure pixel-wise comparison for supervised learning.

Purpose:

- The dataset is used to train AOD-Net, a deep learning model for single-image dehazing.
- The model learns to reconstruct a clear image from a hazy input by minimizing the difference from the ground truth.

Usage:

- Load images from both folders as input-output pairs.
- Apply standard data augmentation techniques if needed.
- Train the AOD-Net model with appropriate loss functions (e.g., MSE, perceptual loss) to optimize performance.

This dataset serves as a benchmark for evaluating dehazing algorithms and improving visibility in hazy environments

AOD-Net Model Overview

Introduction:
AOD-Net (All-in-One Dehazing Network) is a deep learning model designed for

single-image dehazing. This document provides a brief description of its architecture and functionality based on the provided implementation.

Architecture:
The AOD-Net model is implemented using PyTorch and consists of five convolutional layers with ReLU activations. The model follows a multi-scale feature extraction approach, where outputs from different layers are concatenated to enhance feature representation.

Layers and Functionality:

1. **Conv1 (1x1 kernel)** - Extracts initial features from the input image.
2. **Conv2 (3x3 kernel)** - Enhances feature representation.
3. **Conv3 (5x5 kernel)** - Concatenates features from previous layers to capture more context.
4. **Conv4 (7x7 kernel)** - Further refines features by concatenating prior outputs.
5. **Conv5 (3x3 kernel)** - Final feature extraction before computing the clean image.
6. **Final Computation:** The clean image is computed using the formula: `clean_image = ReLU((x5 * x) - x5 + 1)`, where x5 is the final extracted feature and x is the original input image.

Input and Output:

- **Input:** A hazy image (3-channel RGB format).
- **Output:** A dehazed version of the input image.

Conclusion:
This model effectively learns to remove haze by leveraging multi-scale feature extraction and element-wise operations, making it suitable for real-time dehazing applications.

Dataset Loader for AOD-Net Training

This script prepares a dataset for training and validating the AOD-Net dehazing model. It organizes images into training and validation sets, then loads them into a PyTorch `Dataset` class.

Dataset Structure

The dataset consists of two folders:

- **`gt/`** → Contains clear (ground truth) images.
- **`hazy/`** → Contains hazy versions of the corresponding clear images.

Key Components

1. **`populate_train_list()`**
   - Reads images from the `hazy/` folder and groups them based on their corresponding clear image in `gt/`.
   - Splits the dataset into 90% training and 10% validation.
2. **`MyDataset` Class**
   - Loads image pairs (clear and hazy).
   - Resizes images to `(480, 640)`.
   - Normalizes pixel values to `[0,1]`.
   - Converts images to PyTorch tensors in `(C, H, W)` format.
3. **Main Execution (`if __name__ == '__main__':`)**
   - Initializes the dataset with `gt/` and `hazy/` directories.
   - Loads training samples and prints their tensor shapes.

This script enables efficient data loading for training the AOD-Net model on dehazing tasks.

Training Script for AOD-Net

This script trains the **AOD-Net** model for image dehazing using a dataset of paired clear (`gt/`) and hazy (`hazy/`) images.

Key Components

1. **Dataset Loading**
   - Uses `MyDataset` to load training image pairs.
   - Images are loaded in batches and shuffled for better training.
2. **Model & Training Setup**

- Initializes **AOD-Net** and applies custom weight initialization (`weights_init`).
- Uses **Mean Squared Error (MSE) Loss** as the loss function.
- Optimizer: **Adam** with learning rate `1e-3` and weight decay `1e-4`.

3. **Training Process**
   - Runs for a defined number of epochs (`30` by default).
   - Computes loss between predicted clean images and ground truth images.
   - Performs backpropagation and updates model weights.
   - Saves the trained model after each epoch.
4. **Execution (`if __name__ == '__main__':`)**
   - Defines dataset paths (`gt/`, `hazy/`).
   - Sets **batch size = 32** and **epochs = 30**.
   - Calls the `train()` function to start training.

This script ensures efficient model training for dehazing tasks using AOD-Net.

Image Dehazing Script

This script loads a trained **AOD-Net** model and applies it to a hazy image to generate a dehazed output.

Key Components

1. **Image Preprocessing**
   - Loads the hazy image using **PIL** (`Image.open`).
   - Normalizes pixel values (`/ 255.0`).
   - Converts to a **PyTorch tensor**, adjusts dimensions, and adds a batch dimension.
2. **Model Loading & Prediction**
   - Loads the **trained AOD-Net model** (`dehaze_net_epoch_17.pth`).
   - Processes the hazy image through the network.
   - Converts the predicted tensor back to a NumPy array for visualization.
3. **Visualization using Matplotlib**
   - Displays **original hazy image** and **dehazed output** side by side.

4. **Execution (`if __name__ == '__main__':`)**
    - Specifies a test image (`test9.jpg`).
    - Calls `dehaze_image()` to process and display the result.

This script enables **real-time dehazing** of images using a pre-trained deep learning model.