ENHANCING UNDERWATER IMAGERY WITH AI-POWERED DEEP LEARNING TECHNIQUES

This project aims to enhance underwater images by employing a Lightweight Generative Adversarial Network (GAN) model. It is constructed and trained on Google Colab with TensorFlow, and the preprocessing, training, and evaluation procedures are properly organized for marine image enhancement applications.

Project Data Structure

```
ImageEnhancement/
├── data/
│   ├── train/
│   │   ├── input/       # Raw underwater images
│   │   └── target/      # Target enhanced images
│   └── test/
│       ├── input/       # Test underwater images
│       └── target/      # Corresponding Target images
```

Project Objective

Underwater photos tend to have poor visibility, low contrast, and color distortion because of light absorption and scattering. The objective of this project is to:

- Restore visual quality of underwater photos.
- Enhance generalization using both paired (supervised) and unpaired (unsupervised) learning methods.

Dataset Details

Dataset link:

(https://drive.google.com/drive/folders/1iDmIEZq1DS1UeQU2Yz2SP7TObzvHFhQu?usp=sharing)

- UIEB (Underwater Image Enhancement Benchmark): Paired dataset with input-degraded and ground-truth images.
- EUVP (Enhancing Underwater Visual Perception): Includes paired and unpaired images.

Preprocessing Pipeline

- Resize all images to a standard size (256×256).
- Histogram Equalization on luminance channel (Y) for improved contrast.
- Gaussian Blurring to eliminate noise.
- Normalization of pixel values to range [-1, 1].

Model Architecture

Generator

- Has a U-Net-like encoder-decoder architecture.
- Utilizes downsampling and upsampling blocks with skip connections.
- Is composed of lightweight MobileNet-inspired layers to minimize complexity.

Discriminator

- PatchGAN classifier that judges local patches for realism.
- Consumes both input and target/generated images to decide authenticity.
- Has standard convolutional layers with batch normalization and LeakyReLU.

Loss Functions

- Generator Loss = Adversarial Loss + L1 Content Loss (Weighted sum with a λ constant to emphasize visual similarity).
- Discriminator Loss = Binary Cross-Entropy (Real images = 1, Generated = 0).

Training Procedure

- Load and preprocess training data (input-target pairs).
- Train with an adversarial setup:
    - Generator attempts to deceive the discriminator.
    - Discriminator can tell real from fake images.
- Save model checkpoints after every epoch.
- Optionally resume training from the last checkpoint.

Evaluation

Images are ranked based on such metrics as:

- SSIM (Structural Similarity Index)
- PSNR (Peak Signal-to-Noise Ratio)
- Visual Quality (Human Perceptual Analysis)

You are able to compare visually improved results with targets employing stored results.

Requirements

- Python 3.x
- TensorFlow (2.x)

- NumPy, OpenCV, PIL, Matplotlib, tqdm, scikit-image
- Google Colab (for training)

Install required packages in Colab with:

bash

CopyEdit

!pip install pydot graphviz

!apt-get install graphviz

How to Use

- Place your dataset in the provided folder organization on Google Drive.
- Execute the notebook on Colab.
- Train the model or load from checkpoint.
- Test and save improved image outputs.
- Investigate performance by SSIM and visual examination.