Aim

The aim of this project is to implement and run the Enhanced Super-Resolution Generative Adversarial Network (ESRGAN) for image super-resolution on a local machine with GPU support.

Objective

The primary objective is to upscale low-resolution images to high-resolution images using the ESRGAN model, improving the image quality and details.

Summary

This project involves running the ESRGAN model to enhance the resolution of images. ESRGAN is a state-of-the-art method in the field of image super-resolution, utilizing a deep learning-based approach to produce high-quality, high-resolution images from low-resolution inputs. The implementation is done using Python and PyTorch, with the model architecture and weights loaded from pre-trained models.

Tools and Libraries Used

- Python
- PyTorch
- OpenCV
- NumPy
- Glob
- OS

Procedure

1. Load the ESRGAN Model and Weights

CODE:

```
import os.path as osp

import glob

import cv2

import numpy as np

import torch

import RRDBNet_arch as arch


model_path = 'models/RRDB_ESRGAN_x4.pth'  # Path to the pre-trained model

device = torch.device('cuda')  # Using GPU for inference
```

```
model = arch.RRDBNet(3, 3, 64, 23, gc=32)

model.load_state_dict(torch.load(model_path), strict=True)

model.eval()

model = model.to(device)
```

This code block sets up the ESRGAN model by loading the pre-trained weights and setting the device to GPU.

2. Prepare the Input Image

CODE:

```
test_img_folder = 'LR/*'


for path in glob.glob(test_img_folder):
    base = osp.splitext(osp.basename(path))[0]
    img = cv2.imread(path, cv2.IMREAD_COLOR)
    img = img * 1.0 / 255
    img = torch.from_numpy(np.transpose(img[:, :, [2, 1, 0]], (2, 0, 1))).float()
    img_LR = img.unsqueeze(0).to(device)
```

This code reads the low-resolution images, normalizes them, and prepares them for the model by converting them to tensors.

3. Generate High-Resolution Images

CODE:

```
with torch.no_grad():
    output = model(img_LR).data.squeeze().float().cpu().clamp_(0, 1).numpy()
output = np.transpose(output[[2, 1, 0], :, :], (1, 2, 0))
output = (output * 255.0).round()
cv2.imwrite('results/{:s}_rlt.png'.format(base), output)
```

The model is used to generate the high-resolution output, which is then saved as an image file.

Highlights

- **Model Architecture**: The RRDBNet architecture used in ESRGAN is designed to improve image quality by employing residual-in-residual dense blocks (RRDBs), which help in capturing fine details.
- **Upsampling**: The model includes upsampling layers that progressively increase the image resolution.
- **Use of Pre-trained Models**: The project leverages pre-trained weights, which significantly reduce the training time and resources needed for achieving high-quality results.

Conclusion

This project successfully demonstrates the use of ESRGAN for image super-resolution, converting low-resolution images into high-resolution images with enhanced details. The use of GPU acceleration significantly speeds up the inference process, making it feasible to apply this method to a large number of images efficiently. The ESRGAN model's architecture, particularly the use of RRDBs, proves effective in capturing and enhancing fine image details.