## 一、实现方式

#### 1. 总体架构

本项目基于LCM模型（[https://huggingface.co/SimianLuo/LCM\_Dreamshaper\_v7），实现了一个Image2Image的推理流水线，分为三个并行运行的进程：](https://huggingface.co/SimianLuo/LCM_Dreamshaper_v7%EF%BC%89%EF%BC%8C%E5%AE%9E%E7%8E%B0%E4%BA%86%E4%B8%80%E4%B8%AAImage2Image%E7%9A%84%E6%8E%A8%E7%90%86%E6%B5%81%E6%B0%B4%E7%BA%BF%EF%BC%8C%E5%88%86%E4%B8%BA%E4%B8%89%E4%B8%AA%E5%B9%B6%E8%A1%8C%E8%BF%90%E8%A1%8C%E7%9A%84%E8%BF%9B%E7%A8%8B%EF%BC%9A" \t "_blank)

* **VAE Encode 加噪进程**：负责VAE编码和添加噪声。
* **UNet Denoise 进程**：负责使用UNet对潜变量进行去噪处理。
* **VAE Decode 进程**：负责将去噪后的潜变量通过VAE解码成输出图像。

这三个进程通过RabbitMQ消息队列进行通信，Redis用于存储共享的配置信息（如prompt）。

#### 2. 技术选型

* **RabbitMQ**：用于在各个进程之间传递消息，实现异步并行处理。
* **Redis**：用于存储共享的参数和状态信息，如prompt、处理进度等。
* **PyTorch & Diffusers**：用于加载和使用预训练的LCM模型。
* **Python 多进程编程**：使用multiprocessing模块管理多个进程的启动和运行。

#### 3. 流程说明

**初始化阶段**：

* + 从用户输入的队列input\_queue中获取多张原始图像。
  + 在程序启动时指定统一的prompt，存储在Redis中。

**VAE Encode 加噪进程**：

* + 从input\_queue中获取原始图像。
  + 使用VAE对图像进行编码，得到潜变量（latent）。
  + 对潜变量添加噪声，噪声强度设置为0.5。
  + 将加噪后的潜变量发送到encode\_to\_denoise\_queue队列。

**UNet Denoise 进程**：

* + 从encode\_to\_denoise\_queue中获取加噪后的潜变量。
  + 进行2步去噪循环，每一步使用UNet对潜变量进行去噪处理。
  + 将去噪后的潜变量发送到denoise\_to\_decode\_queue队列。

**VAE Decode 进程**：

* + 从denoise\_to\_decode\_queue中获取去噪后的潜变量。
  + 使用VAE将潜变量解码为输出图像。
  + 将输出图像发送到output\_queue，供后续处理或保存。

#### 4. 并行与通信

* 各个进程独立运行，互不阻塞。
* 使用RabbitMQ实现进程间的消息传递，确保数据的有序和可靠性。
* Redis用于共享全局配置（如prompt），各个进程可以从Redis中获取所需的信息。

## 二、实现效果

* **功能性**：成功实现了基于LCM模型的Image2Image流水线处理，能够对多张输入图像进行并行处理，生成对应的输出图像。
* **性能**：由于各个阶段在独立的进程中并行运行，并使用RabbitMQ进行高效的消息传递，整体处理效率有所提升。
* **扩展性**：整个架构具有良好的扩展性，可以根据需要增加处理步骤或修改参数（如噪声强度、去噪步数等）。

## 使用说明

* **启动RabbitMQ和Redis：**确保RabbitMQ和Redis服务已在本地运行。如果没有安装，请按照官方文档进行安装和启动。
  + docker run --name some-redis -d -p 6379:6379 redis
  + docker run --name some-rabbit -d -p 5672:5672 -p 15672:15672 rabbitmq:3-management
* **安装依赖库：**pip install pika redis torch diffusers transformers Pillow
* **运行主程序：**python main.py "your prompt here"
* **发送输入图像**：python input\_producer.py image1.jpg image2.png
* **启动输出结果处理程序：**python output\_consumer.py
* **查看输出图像：**处理完成的图像将保存在output\_images文件夹中。

## 注意事项

* **模型下载：**首次运行程序时，diffusers库会自动从Hugging Face下载所需的模型
* **设备支持：**代码中未指定使用GPU，如需要加速处理，可在模型加载和张量计算时添加.to('cuda')