Uformer for Diffraction Denoising

1、项目简介

原始论文： https://arxiv.org/abs/2106.03106

论文摘要：

在本文中，我们介绍了 Uformer，这是一种有效且高效的基于 Transformer 的图像恢复架构，我们构建了一个使用 Transformer 块的分层编码器-解码器网络。在 Uformer 中，有两个核心设计。首先，我们引入了一种新颖的局部增强窗口（LeWin）Transformer 块，它执行非重叠窗口基础的自注意力，而不是全局自注意力。这在捕获局部上下文的同时显著降低了高分辨率特征图的计算复杂性。其次，我们提出了一个可学习的多尺度恢复调节器，以多尺度空间偏置的形式调整 Uformer 解码器的多层特征。我们的调节器在引入微小额外参数和计算成本的同时，展示了恢复各种图像恢复任务细节的卓越能力。得益于这两种设计，Uformer 在捕获图像恢复的局部和全局依赖性方面具有高能力。为了评估我们的方法，我们在包括图像去噪、运动去模糊、去焦点模糊和去雨等几个图像恢复任务上进行了广泛的实验。

原论文开源项目地址： https://github.com/ZhendongWang6/Uformer

改进以适应衍射图像降噪任务的代码地址： https://github.com/Graph4HEP/Diffraction_denoising

2、代码实现

2.1 运行环境：

机器租用地址： https://www.autodl.com/market/list

机器配置：autodl，L20，Miniconda / conda3 / 3.10(ubuntu22.04) / 11.8

环境配置：

```
conda init
conda create -n DiffractionDenoising python==3.10
source /etc/network_turbo
conda activate DiffractionDenoising
git clone https://github.com/Graph4HEP/Diffraction_denoising.git
cd Diffraction_denoising
python -m pip install --upgrade pip
pip config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple
vim requirements.txt
pip install -r requirements.txt
```

2.2 数据准备：

2.2.0 tiff 数据准备：

如果衍射图像已经保存为 tiff 格式，则将目录建立为如下形式作为数据目录：

```
|-----data ： 数据目录
  |-----training ： 训练数据目录
  |    |-----LC ： 低通量作为带噪声的输入
  |    |-----HC ： 高通量作为去噪后的目标图像
  |-----validation：验证数据目录
  |    |-----LC ： 低通量作为带噪声的输入
  |    |-----HC ： 高通量作为去噪后的目标图像
```

2.2.1 hdf5 转为 numpy 稀疏矩阵格式：

去噪任务使用的数据为 x 射线晶体衍射图像，图片大小为 256*256 像素，每个像素值被归一化为 0-255 的范围之内

原始数据以 hdf5 的格式进行存储，为了便于数据传输，可以将数据转为稀疏矩阵的形式对数据进行压缩

```
cd data_preprocess/
python hdf5_to_sparse_np.py argv1
```

argv1：待转换的 hdf5 文件，文件格式内部的存储形式为：

```
file['low_count']['data'][:,:,:]
file['high_count']['data'][:,:,:]
```

2.2.2 稀疏矩阵 numpy 格式转回 hdf5

在不同机器之间进行数据传输时，用稀疏矩阵格式的 npz 文件可以大大减少数据传输的时间

不过在传输完 npz 文件之后，需要把 npz 文件转回 hdf5 格式

```
cd data_preprocess/
vim sparse_np_to_hdf5.py
```

修改其中的变量：

```
input_npz_files: [high count 的 npz 文件，low count 的 npz 文件]
hdf5_output_file: 输出文件名
```

之后运行程序：

```
python sparse_np_to_hdf5.py
```

2.2.3 hdf5 转为 tiff 图片

```
cd data_preprocess/
vim hdf5_to_tiff.py
```

修改其中的变量：

```
file: 待转换的 hdf5 文件名
save_folder: 存放 tiff 的目标目录
```

之后运行程序：

```
python hdf5_to_tiff.py
```

2.3 模型训练

```
bash script/train_denoise.sh
```

修改其中的

```
--batch_size: L20 机器设为 64
--train_dir: 训练数据目录，需包含 LC 和 HC 子目录
--val_dir: 验证数据目录，需包含 LC 和 HC 子目录
--env: log 目录后缀
--gpu: 因为原 repo 没有写好 DDP 分布式训练，目前用单卡训练速度更快
```

2.4 模型评估

```
cd test
python test.py argv1 argv2 argv3 argv4
```

其中输入参数如下：

```
argv1: log dirs, which should contains the "models", "results"  folders and "config.json" file.
argv2: data dirs, which should contains the "LC" and "HC" folders
argv3: number of samples. "full" is to test all the data in argv2
argv4: a multiplier factor to time the pixel value to have a better view. Suggestion value is 4
```