基于Transformer的断层识别

代码库简介

- 全部使用PyTorch深度学习框架, 因此必须首先安装PyTorch
- 2D等变网络基于e2cnn开发
- 3D等变网络基于PDO_e3DConv开发
- 基于Swin-Transformer的2D模型基于mmcv, mmsegmentation, 以及mmpretrain开发.
- 基于Swin-Transforemr的3D模型基于pytorch-lightning开发
- 2D等变CNN代码库位于pytorch2dunet
- 3D等变CNN代码库位于pytorch3dunet
- 基于2D Swin-Transformer分割模型代码: mmsegmentation, 项目相关配置文件位于 mmsegmentation/projects/Fault_recong
- 2D Swin-Transforemr自监督预训练代码: mmpretrain, 项目相关配置文件位于 mmpretrain/projects/Fault_Recong
- 3D分割, 自监督预训练代码, 项目相关配置文件位于./MIM-Med3D/code/configs
- SAM Adapter分割模型

环境安装

2D Swin-Transformer分割, 预训练, 3D Swin-Transformer分割、预训练代码相互独立, 如只需使用2D模型, 仅需要安装2D模型需要的环境即可. 由于整个项目是由pytorch开发, 首先需要安装pytorch, 我使用的是PyTorch: 1.12.1+cu113版本的torch, 由于CUDA版本不同, 可能需要安装的torch略有差异, 可直接去官网安装torch. 推荐安装torch-1.12.1版本. **在安装完torch之后, 才可以进行2D或者3D模型的环境配置**

例如在cuda版本为11.3的机器上安装torch 1.12.1, torch官网给出的安装命令为

```
pip install torch==1.12.1+cu113 torchvision==0.13.1+cu113
torchaudio==0.12.1 --extra-index-url https://download.pytorch.org/whl/cu113
```

2D等变网络环境安装

进入2D等变代码库, 在安装完PyTorch之后, 只需要额外安装e2cnn库即可

```
pip install e2cnn
```

3D等变网络环境安装

进入3D等变代码库, 除了PyTorch外, 缺失的包直接pip install安装即可

2D Swin-Transformer分割模型环境安装

进入2D Swin-Transformer分割代码库, 原本代码库的说明文档位于./mmsegmentation/README_zh-CN.md, 这里简单说明一下安装步骤, 如遇问题可参考mmsegmentation的官方文档.

步骤0: 使用MIM安装MMCV

```
pip install -U openmim
mim install mmengine
mim install "mmcv>=2.0.0"
```

如果无法使用MIM安装, 可去MMCV官网, 选择合适的torch和cuda版本, 使用pip安装. 例如安装基于cuda11.3, torch1.12.x的mmcv命令为

```
pip install mmcv==2.0.0 -f
https://download.openmmlab.com/mmcv/dist/cu113/torch1.12/index.html
```

步骤1: 安装MMSegmentation

```
cd mmsegmentation
pip install -v -e .
# '-v' 表示详细模式，更多的输出
# '-e' 表示以可编辑模式安装工程，
# 因此对代码所做的任何修改都生效，无需重新安装
```

2D Swin-Transformer自监督预训练环境安装

进入2D Swin-Transformer自监督预训练代码库, 原本代码库的说明文档位于./mmpretrain/README_zh-CN.md, 这里简单说明一下安装步骤, 如遇问题可参考mmpretrain的官方文档.

```
cd mmpretrain
pip3 install openmim
mim install -e .
```

3D Swin-Transformr分割模型环境安装

进入3D Swin-Transformr分割代码库:

```
cd MIM-Med3D
pip install -r requirements.txt
```

模型的预测和训练接口

2D等变网络的训练和预测

项目位于在2D等变网络文件夹pytorch2dunet下 数据格式要求为

```
.
├── train
│   ├── ann
│   └── image
└── val
    ├── ann
    └── image
```

其中ann文件夹下为*.png的(0,1)体断层标注, image文件夹下为*.npy的数据体, image和ann的文件名需要对应上.

可以使用如下命令进行2D等变网络的训练, 注意需要指定训练和验证集的文件夹, 结果保存在./pytorch2dunet/e2UNet_CKPTS文件夹下.

```
cd pytorch2dunet
CUDA_VISIBLE_DEVICES=0 python train.py --model_type e2UNet \
                                       --train_dir {训练数据文件夹} \#  eg:
./Fault_data/2d-simulate-data/train/
                                       --val_dir {验证数据文件夹} \ # eg:
./Fault_data/2d-simulate-data/val/
                                       --ckpt_save_dir e2UNet_CKPTS \
                                       --batch-size 4 --amp --classes 1 --
epochs 20 --bilinear
```

训练完成后可以使用如下命令对验证/测试集内的seismic切片进行预测

```
python predict.py --model_type e2UNet \
                  --model_ckpts ./e2UNet_CKPTS/checkpoint.pth \ # 这里需要
指定模型ckpt的位置
                  --input {需要预测的断层切片文件夹} \ # eg: ./Fault_data/2d-
simulate-data/val/image
                  --output ./e2UNet_CKPTS/preds --mask-threshold 0.5 --
bilinear
```

3D等变网络的训练和预测

项目位于3D等变网络文件夹pytorch3dunet下, 数据格式为data_root下包含的文件结构为

```
.
├── train
└── val
```

train/val文件夹下为一系列*.h5文件, 每个文件内包含"raw", "label"分别对应切割好的256 * 256 *256 大小的数据体，断层体

可以使用如下命令进行3D等变网络的训练, 注意记得修改./config/config_train_pdo.yml中第40行和第63行的训练和验证数据文件夹地址, 训练的ckpt会保存在./pytorch3dunet/3dunet-pdo/best_checkpoint.pth内

```
cd pytorch3dunet
CUDA_VISIBLE_DEVICES=0 python train.py --config
./config/config_train_pdo.yml
```

训练完成后可以使用如下命令对切好的验证/测试集内的seismic cube进行预测, 可通过修改pytorch3dunet/config/config_test_pdo.yml中的第2行来明确具体使用的ckpt, 第37行来指定输出文件夹的位置, 第43行指定需要预测的cube(H5 files)

```
CUDA_VISIBLE_DEVICES=0 python predict.py --config
pytorch3dunet/config/config_test_pdo.yml
```

2D Swin-Transformer 模型预测接口

在2D分割文件夹下

通用格式, 调用./mmsegmentation/projects/Fault_recong/predict.py中的predict_3d或predict_2d函数.

其中predict_3d函数接受的输入为.npy或者.sgy文件

predict_2d函数接受的输入为包含所有需要预测的2d图片的文件夹, 里面的文件为.npy或者.png的单通道图片.

```
cd mmsegmentation
python ./projects/Fault_recong/predict.py --config {Path to model config} \
                                          --checkpoint {Model checkpoint
path} \
                                          --input {Input image root dir/cube
path} \
                                          --save_path {Path to save predict
result} \
                                          --predict_type {Predict 2d/3d
fault} \
                                          --convert_25d {Whether convert to
2.5d} \
                                          --step {step size of 2.5d data} \
                                          --force_3_chan {Whether convert to
3 channel} \
```

```
                                                --device {Set cuda device} \
                                                --direction {inline/xline} \
```

除此之外, 在./mmsegmentation/projects/Fault_recong/predict.py中还提供了predict_2d_single_image函数, 支持numpy数组作为输入, 其余参数与predict_2d函数相同, 可用于单张图片的预测, 函数返回的是输入图片的断层预测得分

```
# 使用示例
input = np.load(f'{image_path}') # 单通道图片
config_file = './output/swin-base-patch4-window7_upernet_8xb2-
160k_mix_data_v3_force_3_chan-512x512_per_image_normal_simmim_2000e/swin-
base-patch4-window7_upernet_8xb2-160k_mix_data_v3_force_3_chan-
512x512_per_image_normal_simmim_2000e.py'
checkpoint_file = './output/swin-base-patch4-window7_upernet_8xb2-
160k_mix_data_v3_force_3_chan-
512x512_per_image_normal_simmim_2000e/best.pth'
device = 'cuda:0'
score = predict_2d_single_image(config_file, checkpoint_file, input,
device, force_3_chan=True) # score为该图片的断层预测得分[0,1]
```

可以使用如下命令调用混合数据训练的2D网络模型对新的数据进行预测

```
cd mmsegmentation
sh predict.sh {Input cube(*.npy/*.sgy)} {Save path}
```

2D Swin-Transformer模型训练接口

调用的配置文件为./mmsegmentation/projects/Fault_recong/config/swin-base-simmim.py, 注意指定第71行的data_root, 数据格式要求为./data_root下包含的文件结构为

```
.
├── train
│   ├── ann
│   └── image
└── val
    ├── ann
    └── image
```

其中ann文件夹下为*.png的(0,1)体断层标注, image文件夹下为*.npy的数据体, image和ann的文件名需要对应上.

```
cd mmsegmentation
# bash run.sh {GPU Visible Device Num} {GPU数目}
# eg 使用单卡, 从自监督预训练ckpt开始, 训练16000个iter, 训练的ckpts保存
在./output/swin-base-simmim文件夹下
bash run.sh 0 1
# eg 使用8卡
bash run.sh 0,1,2,3,4,5,6,7 8
```

2D Swin-Transformer自监督预训练接口

2D Swin-Transformer的自监督预训练代码库位于mmpretrain, 与断层识别相关的模型配置文件全部位于mmpretrain/projects/Fault_Recong/config。**自监督预训练需要大量的无标注数据以及显卡资源, 建议直接使用集群进行训练！！！！！！！**

数据格式要求为./data_root下包含的文件结构为

```
.
├── train
│   └── image
└── val
    └── image
```

其中image文件夹下为*.npy的数据切片, 注意修改mmpretrain/projects/Fault_Recong/config/simmim_swin-base-w7_1000e_512x512_mix_v2_force_3_chan_per_image_norm.py配置文件中第2行data_root_lst进行混合数据自监督预训练。

自监督预训练命令, 单节点8卡

```
cd mmpretrain
bash run.sh train 0,1,2,3,4,5,6,7
```

也可以**使用集群**进行多节点多卡训练, 之前的mix_v2数据使用了4节点，每节点8卡共32卡进行的自监督预训练

```
cd mmprtrain
sbatch slurm_train.sh {conda环境名} {模型配置文件名} {输出保存CKPT地址}
# 例如, 使用混合数据进行1000 epoch的SimMIM自监督预训练，申请4节点共32卡计算资源进行训
练
sbatch slurm_train.sh Fault_Recong
./projects/Fault_Recong/config/simmim_swin-base-
w7_1000e_512x512_mix_v2_force_3_chan_per_image_norm.py
./output/simmim_swin-base-
w7_1000e_512x512_mix_v2_force_3_chan_per_image_norm.py
```

3D Swin-Transformer模型预测接口

在3D模型代码库下, 调用./MIM-Med3D/code/experiments/sl/predict.py中的predict_sliding_window函数, 模型会按照128x128x128的大小对输入的3D断层进行slice inferrence. 该函数接受的输入为.npy或者.sgy文件, 调用的通用格式如下

```
python ./code/experiments/sl/prediect.py --config {Path to model config} \
                                         --checkpoint {Model checkpoint
path} \
                                         --input {Input cube path} \
                                         --save_path {Path to save predict
result} \
                                         --device {Set cuda device} \
```

预测的结果以及每个像素点的得分会保存在save_path文件夹下..

直接使用基于Thebe数据训练的3D分割模型进行推理预测

```
cd MIM-Med3D
sh predict.sh {Input cube(*.npy/*.sgy)} {Save path}
```

3D Swin-Transformer 模型训练接口

模型的训练调用./MIM-Med3D/code/experiments/sl/multi_seg_main.py, 配置文件为./MIM-Med3D/code/configs/sl/fault/swin_unetr_ft.yaml, 需要在109行的labeled_data_root_dir_lst指定数据的位置, 数据格式要求为data_root下包含的文件结构为

```
.
├── train
└── val
```

train/val文件夹下为一系列*.h5文件, 每个文件内包含"raw", "label"分别对应切割好的256 * 256 *256 大小的数据体，断层体

```
# 从自监督预训练ckpt开始, 进行1000个epoch的ft, 训练结果保存
在./output/Fault_Finetuning/swin_unetr_ft文件夹下. 该代码库支持单节点多卡训练, 注
意指定CUDA_VISIBLE_DEVICES
# 例如使用单节点4卡训练
CUDA_VISIBLE_DEVICES=0,1,2,3 sh train.sh
./code/experiments/sl/multi_seg_main.py
./code/configs/sl/fault/swin_unetr_ft.yaml
```

3D Swin-Transformer自监督预训练接口

自监督预训练需耗费较大的计算资源, 使用的是4节点8卡A100-40G GPU进行的DDP训练, 代码库位于MIM-Med3D， 相关的自监督预训练模型config位于MIM-Med3D/code/configs/ssl/fault/swinsimmim_base_m0.75.yaml, 所需的自监督预训练数据格式为

```
.
├── 0.h5
└── 1.h5
```

文件夹下为一系列*.h5文件, 每个文件内包含"raw"对应切割好的128 * 128 * 128 大小的数据体, 当然有标签的监督数据也可以作为自监督预训练数据，只是dataset在读取数据时不读"label"信息，注意修改配置文件中第107行和108行中的unlabeled_data_root_dir_lst以及labeled_data_root_dir_lst。注意该配置文件中第29行num_nodes=4，表示使用4节点进行训练!!!以下是使用4节点16卡A100进行自监督预训练的命令**(需要集群)**

```
cd MIM-Med3D
# 4节点16卡自监督预训练，模型保存在MIM-Med3D/output/Fault_Pretrained/swinsimmim_base_m0.75_250e_mix_v2
sbatch slurm_train.sh Fault_Recong ./code/experiments/ssl/simmim_pretrain_main.py ./code/configs/ssl/fault/swinsimmim_base_m0.75.yaml
```

如果没有集群，可以修改配置文件中第29行num_nodes=1, 并使用如下脚本进行单节点4卡训练

```
# 例如使用单节点4卡训练
CUDA_VISIBLE_DEVICES=0,1,2,3 sh train.sh ./code/experiments/ssl/simmim_pretrain_main.py ./code/configs/ssl/fault/swinsimmim_base_m0.75.yaml
```

使用大模型SAM进行Adapter FT

使用独立的仓库SAM-Adapter, 该仓库的原始说明文档见SAM-Adapter/README.md

创建sam_adapt虚拟环境

```
cd SAM-Adapter
conda env create -f environment.yml
conda activate sam_adapt

# 2D数据FT, ckpts保存在./SAM-Adapter/logs文件夹下
python train.py -net sam -mod sam_adpt -exp_name Fault2D_SAM -sam_ckpt ./checkpoint/sam/sam_vit_b_01ec64.pth -image_size 1024 -b 4 -dataset fault2d -data_path ../Fault_data/public_data/2d_slices -val_freq 1 -vis 50

# predict
```

```
python predict.py --device cuda:0 --input_cube_path {input .npy/.sgy path}
--save_path {Path to save score} --sam_ckpt {ft ckpt path}
```