

3D-RETR 项目复现指南

zrj-cn

221900357

221900357@smail.nju.edu.cn

南京大学 智能科学与技术

1. 项目介绍:

(1) 项目名称: 3D-RETR

(2) 项目链接: <https://github.com/fomalhautb/3D-RETR>

(3) 项目框架:

① cuda10.1

② torchvision=0.9.1

③ python=3.6.13

(4) 需要的数据集:

① ShapeNet-Rendering (大小约 12G)

1) [项目提供的下载链接](#) (来自 stanford 镜像)

② ShapeNetVox32

1) [项目提供的下载链接](#) (来自 stanford 镜像)

③ 说明: 因为项目提供的数据集下载链接下载速度出奇的慢, 所以这里提供我下载好的数据集的南大镜像链接:

1) [南大网盘链接](#) (有效期至 2025 年 1 月 31 日)

2. 项目搭建流程:

(1) 注意事项:

① 请确保所用 GPU 架构适配 cuda 版本和 torch 版本

② 训练该模型需要较大算力, 我选择了在 AutoDL 租了一块 32GB 的 RTX Tesla V100, 这是因为其相对强大并且适配 cuda10.1 (之前租 3090 和 4090 都很难调到适配)

(2) 下载项目:

① 使用以下任一命令(第2个命令很难成功执行, 尽量选择命令1):

- 1) `git clone https://github.com/fomalhautb/3D-RETR.git`
- 2) `git clone git@github.com:FomalhautB/3D-RETR.git`

② 或者直接下载项目并用 `unzip` 解压缩:

(3) 进入 3D-RETR 目录, 创建 3d-retr 环境:

① 创建环境:

- 1) `cd 3D-RETR`
- 2) `conda env create -f config/environment.yaml`

② 激活环境:

- 1) `conda activate 3d-retr`

③ 注意事项:

- 1) 如果环境打不开: 请尝试重启 kernel
- 2) 若依照上述流程创建环境, 请确保 GPU 型号适配
cuda10.1 的版本。

(4) 解压缩数据集并创建环境变量:

① 进入 data 目录, 解压缩数据集:

- 1) `cd data`
- 2) `tar -zxvf ShapeNetRendering.tgz`
- 3) `tar -zxvf ShapeNetVox32.tgz`

② 创建环境变量:

- 1) `vim ~/.bashrc`
- 2) `export SHAPENET_IMAGE=/autodl-tmp/ShapeNetRendering`
- 3) `export SHAPENET_VOXEL=/autodl-tmp/ShapeNetVox32`
- 4) `source ~/.bashrc`

③ 检查环境变量是否正确:

- 1) `echo $SHAPENET_IMAGE`
- 2) `echo $SHAPENET_VOXEL`

(5) 训练尝试: 项目官方提供了一个指令来尝试训练单视图的三维重建:

```
python train.py \  
  
  --model image2voxel \  
  
  --transformer_config config/3d-retr-b.yaml \  
  
  --annot_path data/ShapeNet.json \  
  
  --model_path $SHAPENET_VOX \  
  
  --image_path $SHAPENET_IMAGES \  
  
  --gpus 1 \  
  
  --precision 16 \  
  
  --deterministic \  
  
  --train_batch_size 16 \  
  
  --val_batch_size 16 \  
  
  --num_workers 4 \  
  
  --check_val_every_n_epoch 1 \  
  
  --accumulate_grad_batches 1 \  
  
  --view_num 1 \  
  
  --sample_batch_num 0 \  
  
  --loss_type dice \  

```

- ① 勘误: 官方给的命令是有错误的, 错误在于 “--model_path \$SHAPENET_VOX” 和 “--image_path \$SHAPENET_IMAGES”, \$SHAPENET_VOX 和 \$SHAPENET_IMAGES 是不存在的, 因为在项目的

markdown 文件中，我们设定的环境变量为

\$SHAPENET_VOXEL 和\$SHAPENET_IMAGE，而非

\$SHAPENET_VOX 和\$SHAPENET_IMAGES。这里需要将命令更正。

② 以下给出更正后的指令：

```
python train.py --model image2voxel
--transformer_config config/3d-retr-b.yaml --annot_path
data/ShapeNet.json --model_path $SHAPENET_VOXEL
--image_path $SHAPENET_IMAGE --gpus 1 --precision
16 --deterministic --train_batch_size 16 --val_batch_size
16 --num_workers 4 --check_val_every_n_epoch 1
--accumulate_grad_batches 1 --view_num 1
--sample_batch_num 0 --loss_type dice
```

③ 除此之外需要注意：

1) 如果你的文件目录是如下形式，运行时仍会报错，会显示找不到数据集的相关文件：

```
root
```

```
----3D-RETR
```

```
-----train.py ( and so on )
```

```
----autodl-tmp ( or other file especially for data )
```

```
-----ShapeNetRendering
```

```
-----ShapeNetVox32
```

2) 出现以上情况是因为:

a. 运行指令时必须处于 3D-RETR 目录下, 这是因为指令中很多命令都默认直接在该文件夹下进行寻找。

b. 而先前我们将环境变量设置成了:

a) `export SHAPENET_IMAGE=/autodl-tmp/ShapeNetRendering`

b) `export SHAPENET_VOXEL=/autodl-tmp/ShapeNetVox32`

c. 这就导致指令实际上会寻找路径

`root/3D-RETR/autodl-tmp/ShapeNetRendering` 和

`root/3D-RETR/autodl-tmp/ShapeNetVox32`, 这肯定是找不到的。

3) 解决方法:

a. 既然已知文件目录形状, 并且执行指令时必须要在 3D-RETR 目录下, 那么改变环境变量;

b. 将环境变量设置为:

a) `export SHAPENET_IMAGE = ../autodl-tmp/ShapeNetRendering`

b) `export SHAPENET_VOXEL = ../autodl-tmp/ShapeNetVox32`

c) 记得保存修改并运行修改

4) 还应该注意: 最好在指令最后添加 `--max_epochs 5` 或者其他数字的指定, 不指定的话, 会跑特别特别久, 中途停下来也不会记录 checkpoint, 所以选择一个小的 epoch 数很重要, 至少会获得 checkpoint, 这对于后续的 eval 很重要。

3. 代码的复现

(1) eval.py 的复现:

- ① 在复现代码中, `eval.py` 的复现被命名为了 `eval2.py`, 其参照了原 `eval` 代码进行复现并添加了中文注释, 方便理解代码中各部分函数的功能和相互之间的逻辑关系。
- ② 其他的一些代码文件也进行了注释工作。

(2) 如何运行测试代码:

- ① 项目官方没有给出运行 `eval` 的指令, 这里给出我测试时使用的指令:

```
python eval.py --annot_path data/ShapeNet.json
--model_path $SHAPENET_VOXEL --image_path
$SHAPENET_IMAGE --batch_size 1 --num_workers 8
--seed 0 --split val --transformer_config
config/3d-retr-b.yaml --background 0 0 0 --beam 1
--view_num 1 --threshold 0.5 --predict --save_path
savepre --resume_from_checkpoint
mlruns/1/3bea0bf41e594498a4bfb60e147007f5/checkp
oints/epoch=04-iouval_iou=0.57091.ckpt
```

- ② 注意:

- 1) 模型必须经过训练, 否则是没有 `checkpoint` 来用于测试与评估的。
- 2) 每人的 `checkpoint` 不一定相等。以自己的 `checkpoint`

为准。

- 3) 我提供的复现代码文件夹中包含我的 checkpoint, 可以直接使用, 请确保路径正确。(eval 指令在 3D-RETR 目录下运行)
- 4) 请在 3D-RETR 目录下新建 savepre 目录用于保存预测的模型。

③ 勘误:

- 1) 在官方代码的基础上直接运行我所给的 eval 指令会生成报错:

```
data = [self.dataset[idx] for idx in possibly_batched_index]
File "/root/3D-RETR/src/data/datasets.py", line 31, in __getitem__
    return self._dataset[self._indices[index].item()]
File "/root/3D-RETR/src/data/datasets.py", line 120, in __getitem__
    image = Image.new("RGB", rgba.size, self._background)
File "/root/miniconda3/envs/3d-retr/lib/python3.6/site-packages/PIL/Image.py", line 2642, in new
    return im._new(core.fill(mode, size, color))
TypeError: color must be int or tuple
```

- 2) 关注报错信息, 这个错误出现在/src/data/datasets.py 文件中, 代码在尝试创建一个新的 Image 对象时, 发现 color 参数的类型不正确。
- 3) 于是对代码进行更正, 确保参数正确: 以下图片的第 91 行代码是后续添加的, 改在 init 方法内。

```
85         self._model_path = model_path
86         self._image_path = image_path
87         self._image_transforms = image_transforms
88         self._mode = mode
89         self._background = background
90         self._view_num = view_num
91         self._background = (255, 255, 255) # 白色背景
```

④ 运行情况: 修改后能正确测试

- ① 从训练过程的 **loss** 变化趋势可以看出，随着训练的进行，模型损失正在逐步缩小。
- ② 但由于 **Transformer** 模型的训练与推理速度较慢，而计算加速效果并不优秀，所以导致 **3D-RETR** 模型的损失降低相对缓慢。
- ③ 模型的评估等板块已经添加在了翻译后的论文当中。论文的翻译版部分模块由我添加，具体情况记录在了论文的脚注中。