DeepCAD论文复现

1.论文概述

1.1 论文背景

现有 的3D 生成模型多聚焦体素、点云、网格等离散表示，本文首次提出针对 CAD 操作序列的生成模型 DeepCAD，基于 Transformer 架构将 CAD 操作类比自然语言处理，实现形状自动编码与随机生成。为此作者还构建了包含 17.8 万 CAD 模型的公开数据集，并且实验表明模型生成的设计具备多样性与几何准确性，支持工业级编辑与格式转换。

1.2 论文实现方法

首先是将CAD 模型表示成方便神经网络读取的形式，定义草图（线 / 弧 / 圆）与拉伸命令的参数规范，通过归一化与量化将连续参数转为 8 位整数，固定序列长度为 60 以适配神经网络。然后是建立自动编码器和解码器，编码器通过 Transformer 提取序列特征，解码器基于 latent 向量生成操作序列，嵌入层分离命令类型、参数与位置信息。然后是创建数据集，作者从 Onshape 解析大量CAD 模型，筛选仅含草图 - 拉伸序列的模型17.8万个。最后是训练模型与生成模型，作者利用交叉熵损失优化模型，结合 Latent-GAN 实现随机生成，支持从高斯噪声采样生成多样化设计。

1.3 论文的实验结果

首先是验证自动编码的性能，评估的指标有：命令准确率（**ACC_cmd**），代表预测命令类型的正确性；参数准确率（**ACC_param**），代表正确命令的参数误差容忍度（阈值 η=3）；Chamfer 距离（**CD**），代表生成形状与真实形状的点云差异；无效率（**Invalid Ratio**），代表生成无效拓扑的比例。通过进行对比实验，作者发现采用参数量化与数据增强（Ours+Aug）的方法能显著提升准确率（ACC_cmd=99.5%）与几何精度（CD=0.752×10^{-3}），采用相对坐标表示（Alt-Rel）易导致拓扑错误，采用连续参数回归（Alt-Regr）会破坏几何关系。这也证明了论文里的自动编码器模型拥有极高的准确率与精度。

再者是验证生成功能的性能，评估的指标有：**COV (Coverage)**，用于评估生成点云与参考点云的覆盖程度，数值范围为[0, 1]，数值越高，表示生成点云覆盖的参考点云空间越全面；**MMD (Minimum Matching Distance)**，用于衡量生成点云与参考点云之间的最小距离，数值越低，表示生成点云与参考点云之间的匹配程度越高；**JSD (Jensen-Shannon Divergence)**，用于衡量生成点云与参考点云在分布上的差异，数值越低越好，表示生成分布与参考分布更接近。对比 l-GAN模型，在点云指标（COV、MMD、JSD）上性能相当，但 CAD 模型具有更锐利的几何特征和可编辑性。而且随机生成的 CAD 模型涵盖多样结构，支持用户在工业软件中直接编辑（如移动面、修改参数）。

1.4 论文最终的总结与讨论

论文首次实现 CAD 操作序列的生成模型 DeepCAD，并提供大规模数据集，为工业设计自动化奠定基础。但仍有一定的局限性，比如仅支持草图、拉伸等基础操作，未包含倒圆角等依赖 B-rep （边界表示）的操作；长序列生成时拓扑有效性下降，数据分布偏向短序列。未来作者会更注重研究模型，使之扩展支持更多 CAD 操作，改进长序列生成的稳定性。

2.论文复现

2.1 环境配置

在复现论文时，我选择了云平台AutoDL搭配VS Code来作为运行的环境，具体配置如上图，整体配置高于官方要求的最低配置，保证程序能够运行。在官方的github仓库里下载源码与数据集，并通过AutoDL提供的工具，将源码和数据集上传至服务器，并解压。然后根据官方提供的依赖文档，利用指令安装对应的库。指令如下：

```
pip install -r requirements.txt #下载对应的python库
conda install -c conda-forge pythonocc-core=7.5.1 #下载对应的OpenCASCADE
```

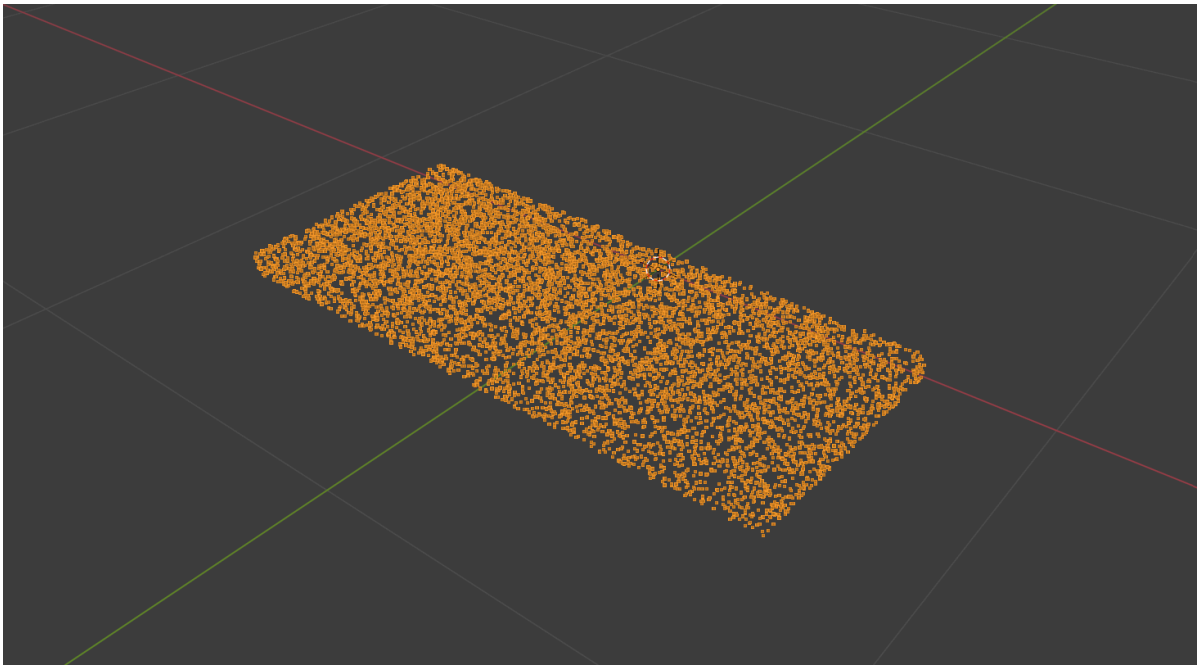
2.2 解压数据集并验证

解压完数据集后，里面有两个文件夹，分别是cad_json、cad_vec。cad_json包含的是描述CAD指令的源json文件，cad_vec包含的是向量化的CAD指令。数据集里总共包含了100组CAD指令，对应100个几何模型。我们可以用指令，运行json2pc.py文件，来使json文件转化成pc端常用的文件形式。指令如下：

```
python json2pc.py --only_test
```

```
face_normals didn't match triangles, ignoring!
convert point cloud failed: 0032/00323035
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
convert point cloud failed: 0095/00958838
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
Warning: 2 faces have been skipped due to null triangulation
face_normals didn't match triangles, ignoring!
Warning: 2 faces have been skipped due to null triangulation
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
[Parallel(n_jobs=10)]: Done 7990 tasks      | elapsed:  2.0min
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
face_normals didn't match triangles, ignoring!
[Parallel(n_jobs=10)]: Done 8052 out of 8052 | elapsed:  2.0min finished
root@autodl-container-2a6e44bb3e-1f06ce1f:~/autodl-tmp/DeepCAD-master/dataset#
```

上图是指令的运行结果，运行过程中存在面片丢失的情况，但不影响最终结果，总共用时2min，处理了8052个任务。最终生成的是ply格式的文件，是常用的点云格式。我将生成的结果导入至Blender中，得到以下结果：



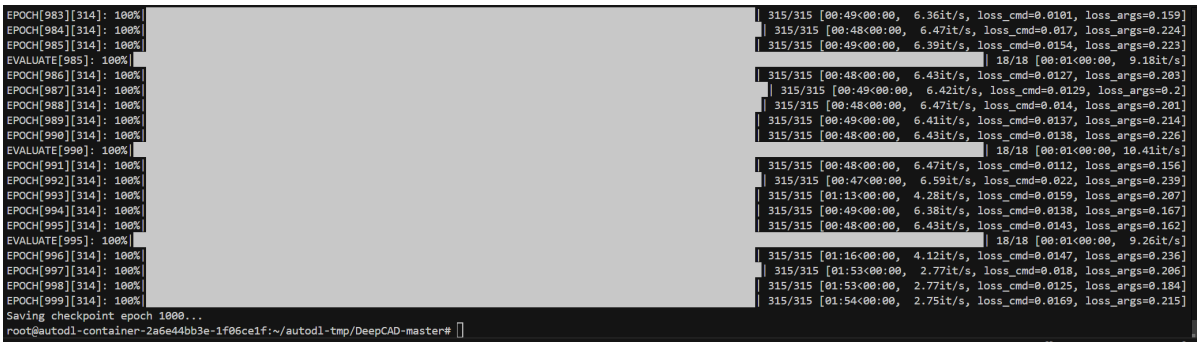
由此看以看到点云生成的效果不错，比较均匀，并且能明显地看出物体的几何轮廓。通过这个方式，验证了数据集的可用性。

2.3 模型训练

之后开始训练模型，首先自动编码器，运行指令，训练一个1000步的模型。指令如下：

```
python test.py --exp_name newDeepCAD --mode rec --ckpt 1000 -g 0
```

这个指令将基于实验名字叫newDeepCAD的文件夹，利用GPU 0来进行运算，训练一个1000步的自动编码器模型。日志里记录了每个步长所消耗的时间与损失，以下是训练结果的截图。



最终训练花费了16个小时。

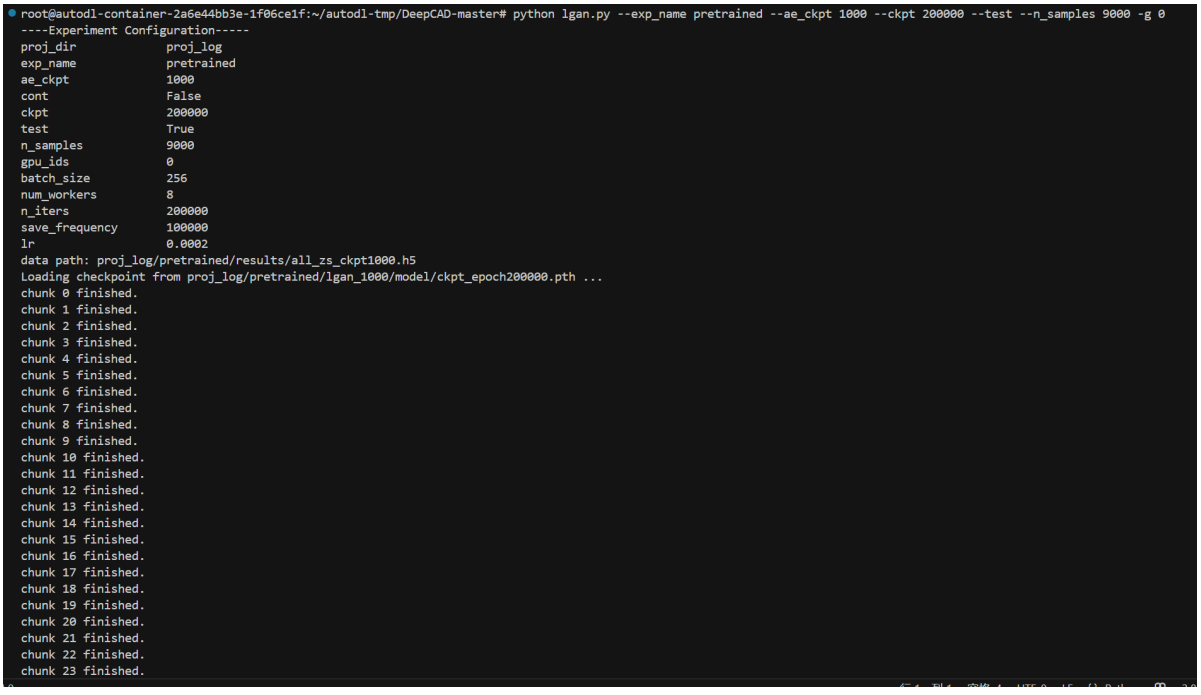
为了验证生成功能，还需要训练lGAN模型，但鉴于训练模型的时间过长，本次复现使用的是官方提供的预训练模型。

2.4 验证论文里提到的生成功能

模型准备好之后，就可以开始验证论文里的功能了。运行指令，加载训练好的模型，并运行生成功能。指令如下：

```
python lgan.py --exp_name pretrained --ae_ckpt 1000 --ckpt 200000 --test --
n_samples 9000 -g 0
# 实验名字pretrained，加载1000次checkpoint的自动编码器权重，加载200000次迭代的主模型权重，
模式为测试模式，生成9000个样本，用0号GPU计算
```

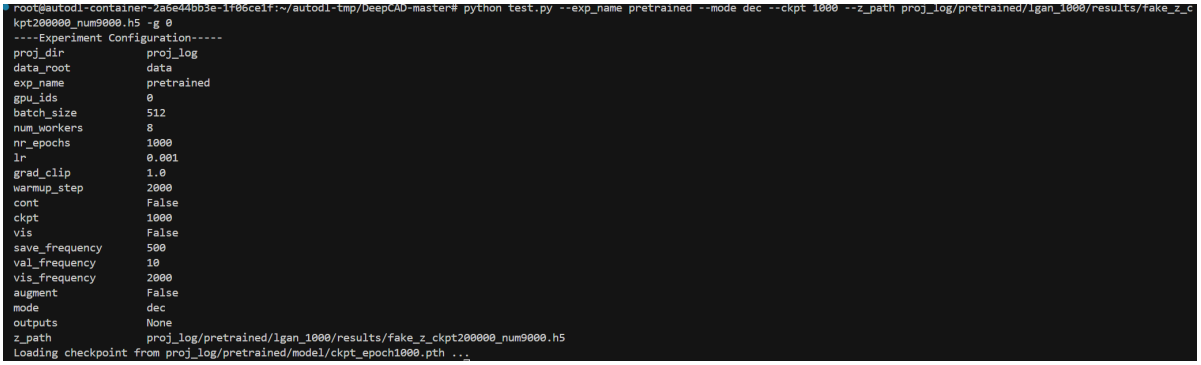
以下是计算的结果，可以观察到每个chunk都顺利完成了。



生成完点云数据之后，开始验证解码功能。运行以下指令：

```
python test.py --exp_name pretrained --mode dec --ckpt 1000 --z_path
proj_log/pretrained/lgan_1000/results/fake_z_ckpt200000_num9000.h5 -g 0
# 实验名字为pretrained，将模式改为解码模式，默认为编码，模型权重为1000，通过z_path设置需要解
码的隐变量路径为proj_log/pretrained/lgan_1000/results/fake_z_ckpt200000_num9000.h5，
使用0号GPU计算
```

以下是计算结果，展示了模型的所有参数，包括lr（学习率）、batch_size（批处理大小）等。

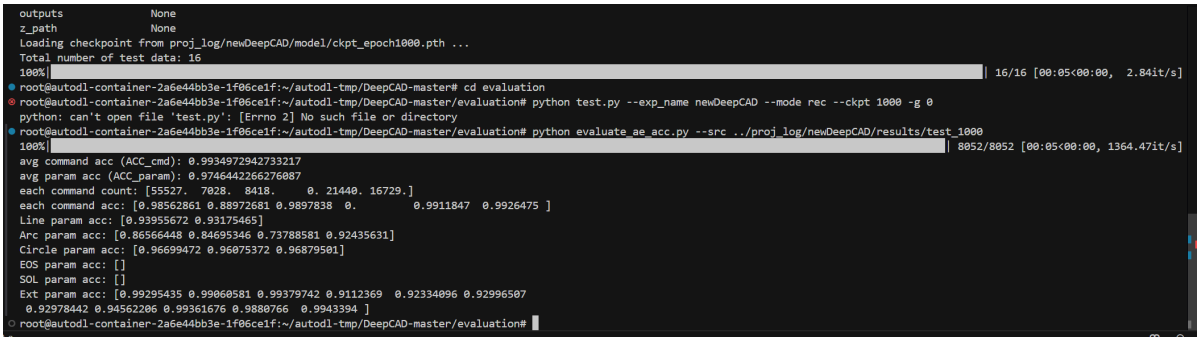


2.5 模型评估

在验证完功能后，对我们的模型进行评估。首先是对我自己训练的自动编码器进行评估。运行以下指令：

```
python evaluate_ae_acc.py --src ../proj_log/newDeepCAD/results/test_1000
```

这个指令能计算编码出来的指令、参数等信息的正确率，即**ACC_cmd**和**ACC_param**，具体数据如下图。我们看到平均指令正确率达到了99.3%，平均参数正确率达到了97.5%。证明了论文中的自动编码器模型具有很高的可行性。



之后是对执行生成功能的模型进行评估。根据论文里提到的评估方法，总共需要对三个指标进行评估：

①**COV (Coverage)**：用于评估生成点云与参考点云的覆盖程度，数值范围为[0, 1]，数值越高，表示生成点云覆盖的参考点云空间越全面。

②**MMD (Minimum Matching Distance)**：用于衡量生成点云与参考点云之间的最小距离，数值越低，表示生成点云与参考点云之间的匹配程度越高。

③**JSD (Jensen-Shannon Divergence)**：用于衡量生成点云与参考点云在分布上的差异，数值越低越好，表示生成分布与参考分布更接近。

运行以下指令，对模型生成的点云数据进行评估。

```
sh run_eval_gen.sh
../proj_log/pretrained/lgan_1000/results/fake_z_ckpt200000_num9000_dec 1000 0
# 指定解码后的点云文件路径
为../proj_log/pretrained/lgan_1000/results/fake_z_ckpt200000_num9000_dec，评估样本为1000，使用0号GPU计算
```

以下是评估的结果，得到了3次迭代的结果。参考点云数量（1000，2000，3），生成点云数量（3000，2000，3），得到了平均的MMD为0.01465，平均的COV为0.7913，平均的JSD为0.04015。与论文里给出数据较为接近，复现了论文里的效果。

```
[Parallel(n_jobs=8)]: Done 9000 out of 9000 | elapsed:  2.0min finished
n_test: 1000, multiplier: 3, repeat times: 3
iteration 0...
reference point clouds: (1000, 2000, 3)
time: 1.59s
generated point clouds: (3000, 2000, 3)
time: 0.65s
100%|| 3000/3000 [10:32<00:00,  4.75it/s, cov=0.788]
{'MMD-CD': 0.014751328155398369, 'COV-CD': 0.7879999876022339, 'JSD': 0.03939593209995884}
iteration 1...
reference point clouds: (1000, 2000, 3)
time: 1.61s
generated point clouds: (3000, 2000, 3)
time: 0.62s
100%|| 3000/3000 [10:32<00:00,  4.75it/s, cov=0.787]
{'MMD-CD': 0.014778675511479378, 'COV-CD': 0.7870000004768372, 'JSD': 0.04113871483679965}
iteration 2...
reference point clouds: (1000, 2000, 3)
time: 1.58s
generated point clouds: (3000, 2000, 3)
time: 0.62s
100%|| 3000/3000 [10:31<00:00,  4.75it/s, cov=0.799]
{'MMD-CD': 0.01442530658096075, 'COV-CD': 0.7990000247955322, 'JSD': 0.03992215201902738}
average result:
{'avg-MMD-CD': 0.014651770082612833, 'avg-COV-CD': 0.7913333376248678, 'avg-JSD': 0.04015226631859529}
```

2.6 结果导出与可视化

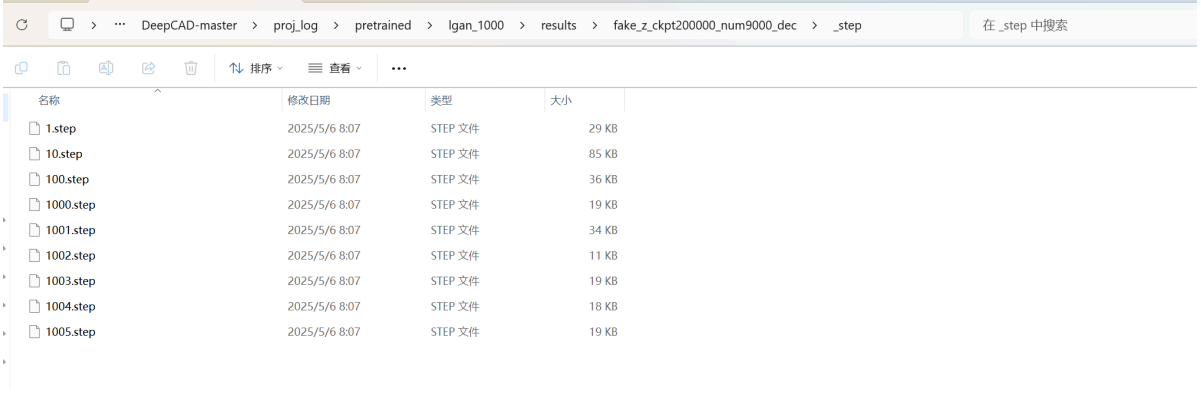
最后导出模型生成的结果，方便导入常用的CAD软件进行观察。运行以下指令：

```
python export2step.py --src
../proj_log/pretrained/lgan_1000/results/fake_z_ckpt200000_num9000_dec/ # export
to step format
```

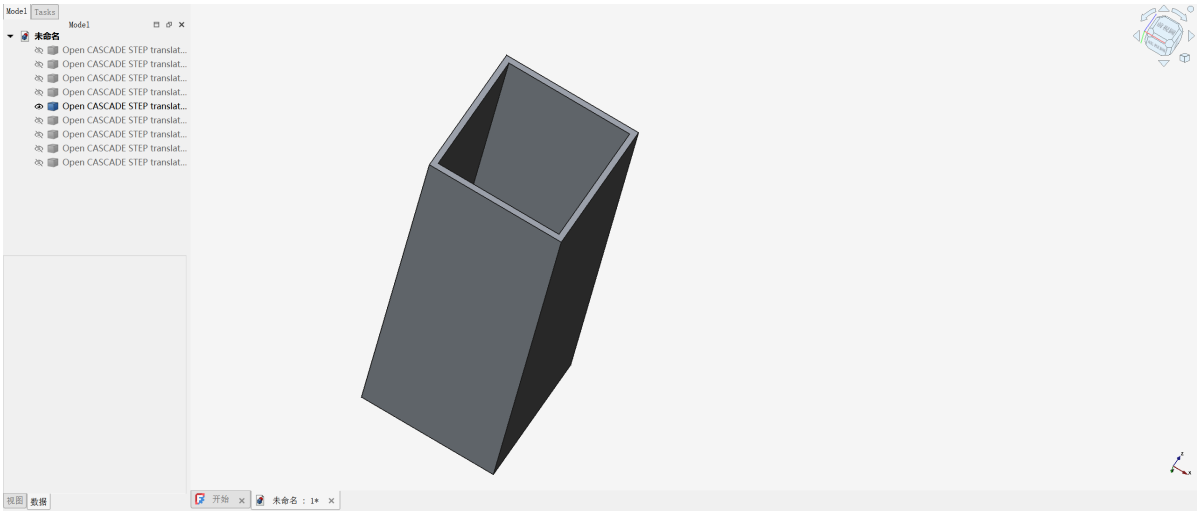
可以导出生成的一系列step文件至文件夹../proj_log/pretrained/lgan_1000/results/fake_z_ckpt200000_num9000_dec/，以下是生成日志。

```
 Step File Name : ../proj_log/pretrained/lgan_1000/results/fake_z_ckpt200000_num9000_dec/_step/1001.step(696 ents)  Write  Done
../proj_log/pretrained/lgan_1000/results/fake_z_ckpt200000_num9000_dec/1002.h5

**************************************************************
******        Statistics on Transfer (Write)                ******

**************************************************************
******        Transfer Mode = 0  I.E.  As Is       ******
******        Transferring Shape, ShapeType = 2                    ******
** WorkSession : Sending all data
 Step File Name : ../proj_log/pretrained/lgan_1000/results/fake_z_ckpt200000_num9000_dec/_step/1002.step(232 ents)  Write  Done
../proj_log/pretrained/lgan_1000/results/fake_z_ckpt200000_num9000_dec/1003.h5

**************************************************************
******        Statistics on Transfer (Write)                ******

**************************************************************
******        Transfer Mode = 0  I.E.  As Is       ******
******        Transferring Shape, ShapeType = 2                    ******
** WorkSession : Sending all data
 Step File Name : ../proj_log/pretrained/lgan_1000/results/fake_z_ckpt200000_num9000_dec/_step/1003.step(380 ents)  Write  Done
../proj_log/pretrained/lgan_1000/results/fake_z_ckpt200000_num9000_dec/1004.h5

**************************************************************
******        Statistics on Transfer (Write)                ******

**************************************************************
******        Transfer Mode = 0  I.E.  As Is       ******
******        Transferring Shape, ShapeType = 2                    ******
** WorkSession : Sending all data
 Step File Name : ../proj_log/pretrained/lgan_1000/results/fake_z_ckpt200000_num9000_dec/_step/1004.step(380 ents)  Write  Done
../proj_log/pretrained/lgan_1000/results/fake_z_ckpt200000_num9000_dec/1005.h5

**************************************************************
******        Statistics on Transfer (Write)                ******

**************************************************************
******        Transfer Mode = 0  I.E.  As Is       ******
******        Transferring Shape, ShapeType = 2                    ******
** WorkSession : Sending all data
 Step File Name : ../proj_log/pretrained/lgan_1000/results/fake_z_ckpt200000_num9000_dec/_step/1005.step(380 ents)  Write  Done
```

利用软件filezilla，将服务器上的文件传输到本地。



然后导入到常用的CAD软件进行可视化展示，这里以FreeCAD为例。



3.总结

我通过对DeepCAD论文进行复现，成功再现了其实验成果，证明了这篇论文对于工业设计领域的创新性。这篇论文提出了一个针对 CAD 操作序列的生成模型 DeepCAD，基于 Transformer 架构实现形状自动编码与随机生成，并构建了公开数据集，相比于以往专注于点云、网格或体素表示的生成方法，DeepCAD 更加贴合工业设计实际需求，为 CAD 模型的自动化生成开辟了新的研究方向。

在复现过程中，我成功搭建了云平台 AutoDL 搭配 VS Code 的运行环境，验证了数据集的可用性。训练了 1000 步的自动编码器模型，自动编码器的平均指令正确率达 99.3%，平均参数正确率达 97.5%。使用官方预训练模型验证了生成功能，对执行生成功能的模型进行评估，COV、MMD、JSD 三个指标结果与论文数据较为接近，成功复现论文效果。最后将生成结果导出为 STEP 文件并在 FreeCAD 中可视化展示。

总的来说，这次复现成功验证了这篇论文的创新性和实用性，为学术界研究CAD生成提供全新的工具，也为工业设计领域提高设计效率和降低开发成本提供了重要支撑。未来可以扩展模型功能，使其支持更多 CAD 操作，如倒圆角等依赖 B - rep 的操作。同时，需解决数据处理的稳定性问题，提升模型性能和可靠性，推动该模型在工业设计自动化领域的应用发展。