MM report

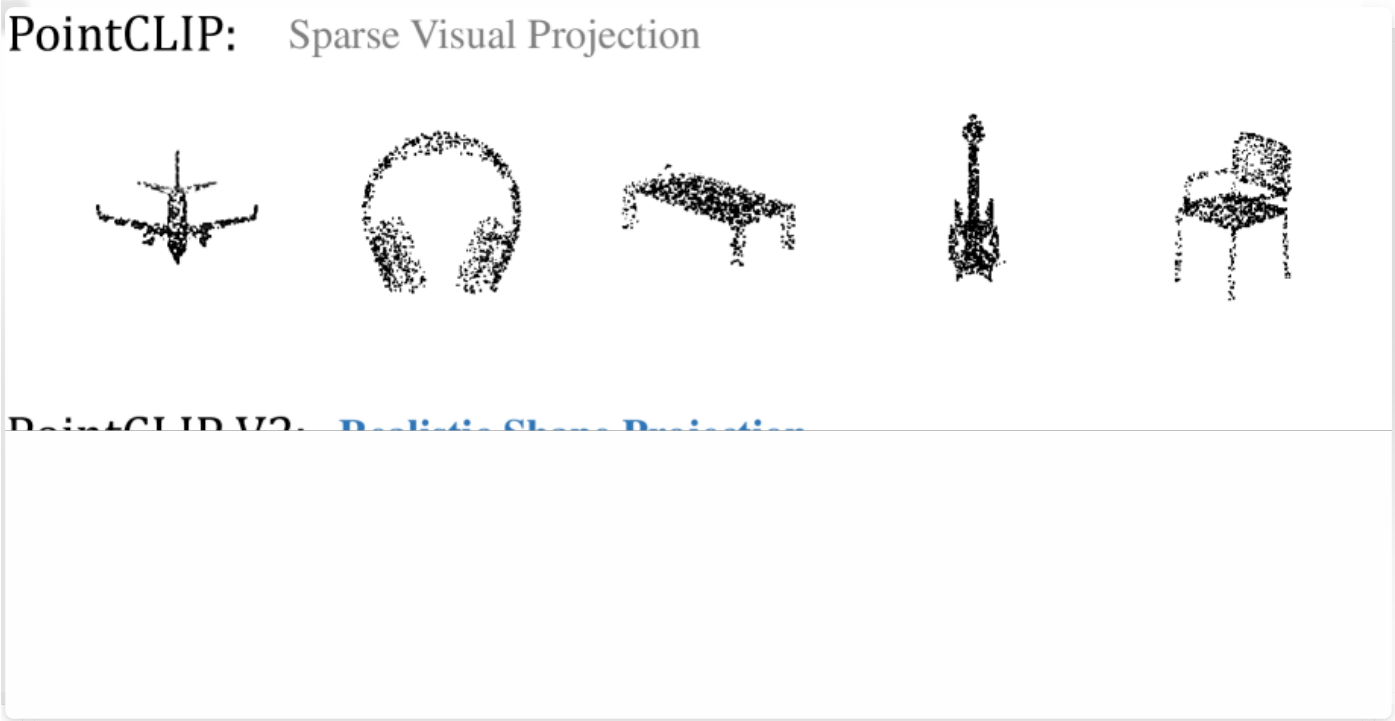
环境依赖和使用方法在zeroshot_cls fewshot_cls zeroshot_seg zeroshot_obj/3detr-main 这四个文件夹中（README.md/requirment.txt）

论文简介

PointCLIP V2: Prompting CLIP and GPT for Powerful 3D Open-world Learning改进了CLIP这一对比语言-图像预训练模型在3D点云上的表现。首先，PointCLIP V2引入了一个逼真的形状投影模块，为CLIP的视觉编码器生成更逼真的深度图，缩小了投影点云与自然图像之间的域差距。其次，PointCLIP V2利用大规模语言模型，为CLIP的文本编码器自动设计更具描述性的3D语义提示，而不是PointCLIP所采用的手工制作的提示。在不引入任何 3D 领域训练的情况下，PointCLIP V2的零样本 3D 分类在三个数据集上以 +42.90%、+40.44% 和 +28.75% 的准确率显着超过了 PointCLIP。此外，PointCLIP V2 可以以简单的方式扩展到少样本分类、零样本分割和零样本 3D 目标检测。代码由 https://github.com/yangyangyang127/PointCLIP_V2 提供。

PointCLIP V2 的整体框架。对于视觉编码，我们通过逼真的形状投影生成高质量的深度图。对于文本编码，利用预先训练的大规模语言模型进行 3D 特定的提示。PointCLIP V2 可以通过反向投影密集对齐来扩展为零样本分割，并在给定 3D 区域建议的情况下进行零样本 3D 检测。

跟2D的zero/few shot工作类似，PointCLIP V2使用GPT辅助生成text；类似PointCLIP，Visual部分则使用点云投影，PointCLIP V2为了使Visual features和Textual features尽可能对齐，将投影图进行了优化，如下图所示。

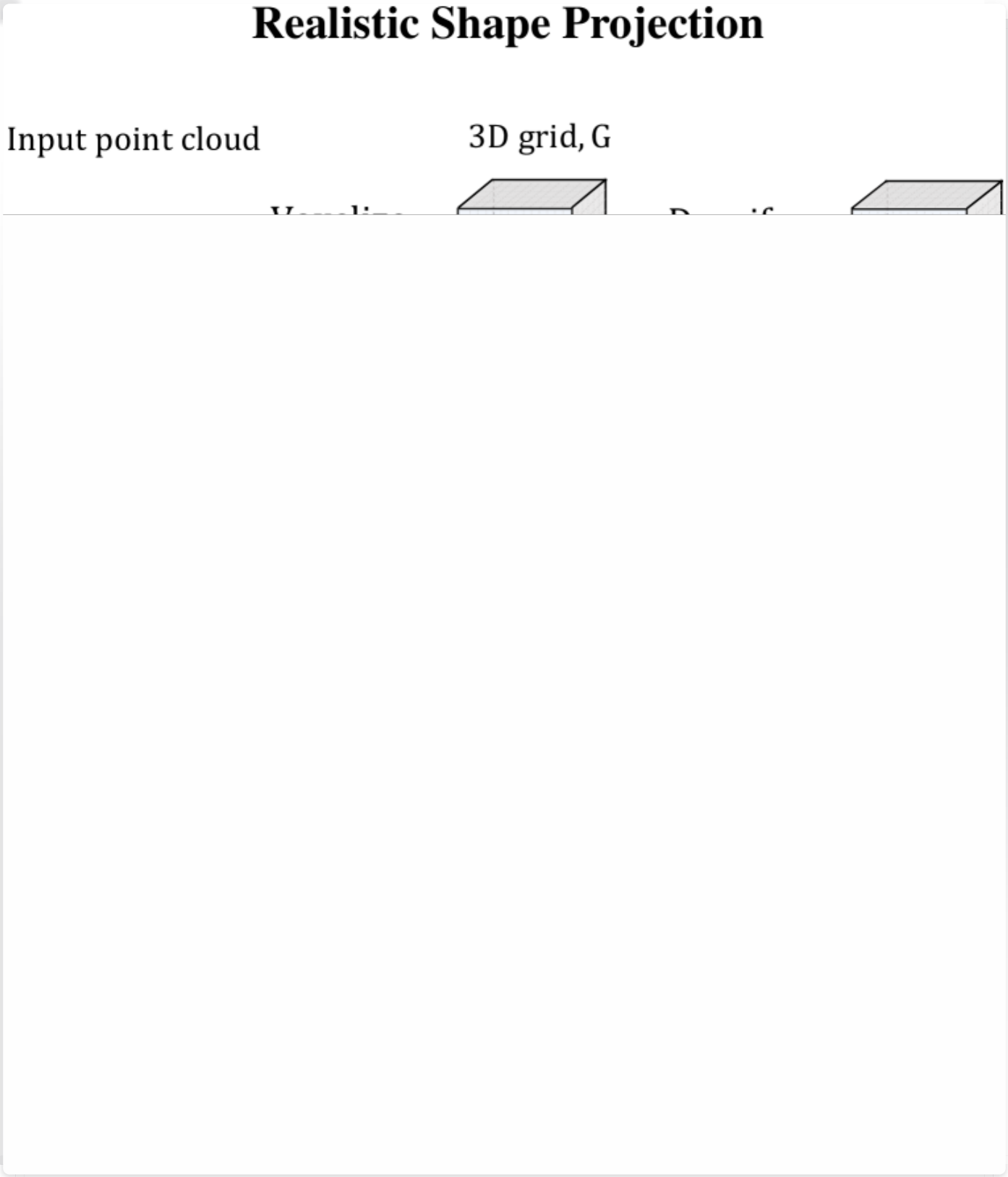


视觉投影的比较。PointCLIP V2可生成更逼真的深度图，具有更密集的点分布和更平滑的深度值。

具体的投影方法是如下图所示，这个步骤是决定zero shot和few shot的步骤。PointCLIP V2通过四个主要步骤来实现：体素化、致密化、平滑和挤压。

1. **体素化**：首先，针对不同视图，创建一个零初始化的三维网格。这个网格的每个维度代表一个空间分辨率，其中一个维度专门代表垂直于视图平面的深度。点云的 3D 坐标被归一化并投影到这个网格中，生成包含稀疏深度值的网格。如果多个点投影到同一个体素，只保留最小深度值，因为深度较小的点会遮挡深度较大的点。
2. **致密化**：为了解决稀疏点云导致的视觉不连贯问题，通过局部最小值池化操作来致密化网格，以保证视觉连续性。这个步骤通过保留最小深度值，使稀疏点之间原本空白的体素得到合理的深度值填充，而背景体素保持空白，从而实现更密集和平滑的空间表示。
3. **平滑**：局部池化可能在一些 3D 表面引入伪影，因此采用非参数高斯核进行形状平滑和噪声过滤。合适的核大小和方差可以消除致密化引起的空间噪声，同时保留原始 3D 形状中的边缘和角落清晰度，得到更紧凑、更平滑的形状。
4. **挤压**：最后一步是将深度维度挤压，以获得投影的深度图。这是通过提取每个深度通道的最小值作为每个像素位置的值，并将其重复三次作为 RGB 强度来实现的。这种基于网格的正交投影方法对硬件实现更友好，与 PointCLIP 中的 3D 到 2D 透视变换相比更高效。

Few shot过程中，给定一小组 3D 训练数据，PointCLIP V2可以将真实形状投影的平滑操作修改为可学习。具体来说，由于不规则点云已经转换为基于网格的体素，我们在高斯滤波之后采用了两个三维卷积层。这些可学习的模块学习从少量数据集中总结 3D 域知识，并进一步调整 3D 形状以使其对 CLIP 更友好。



逼真的形状投影。该开关选择直接零样本分类或具有可学习平滑的少样本分类。

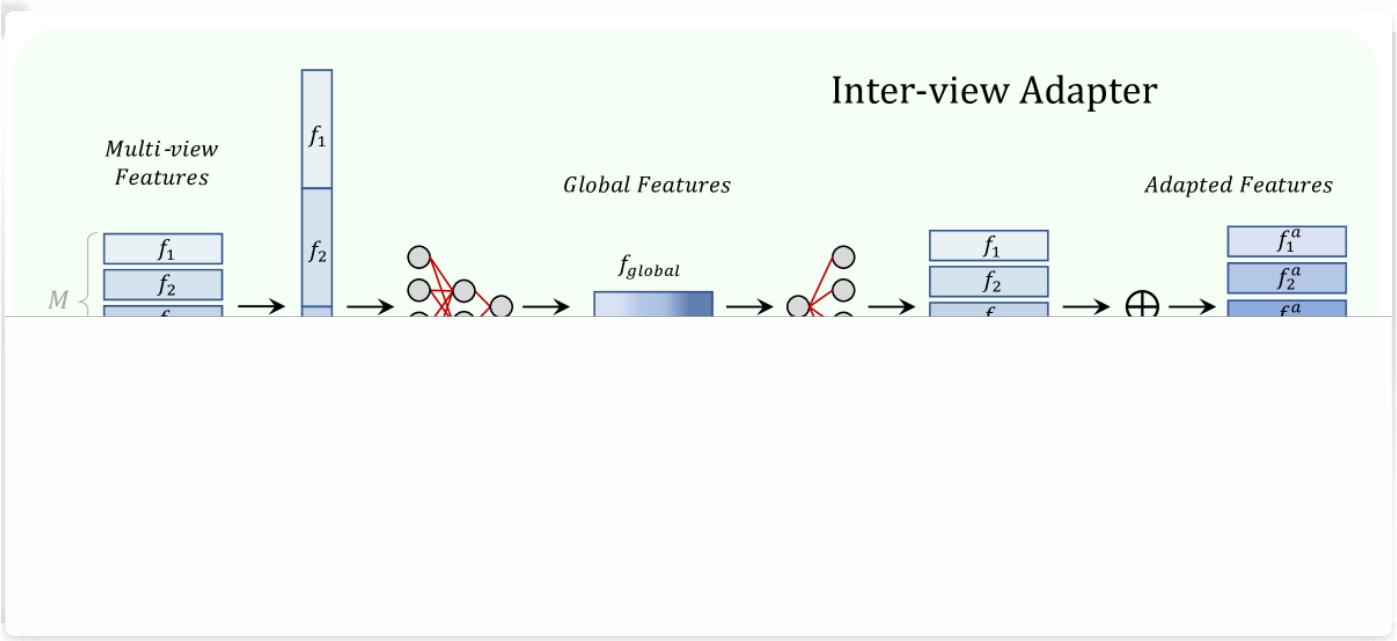
训练过程中，smooth并且经过encoder之后，PointCLIP V2使用了一个Adapter，这是PointCLIP中采用的**Inter-View Adapter**，先把多视角的特征结合成一维，通过两个全连接层得到全局特征，

$$f_{global} = ReLU(Concate(f_{1\sim M})W_1^T)W_2^T \tag{1}$$

其中W_1和W_2为Adapter中的两层权重。之后用全局特征f_{global}生成视图Adapter特征，并通过残差连接将其添加到其原始 CLIP 编码特征中，作为

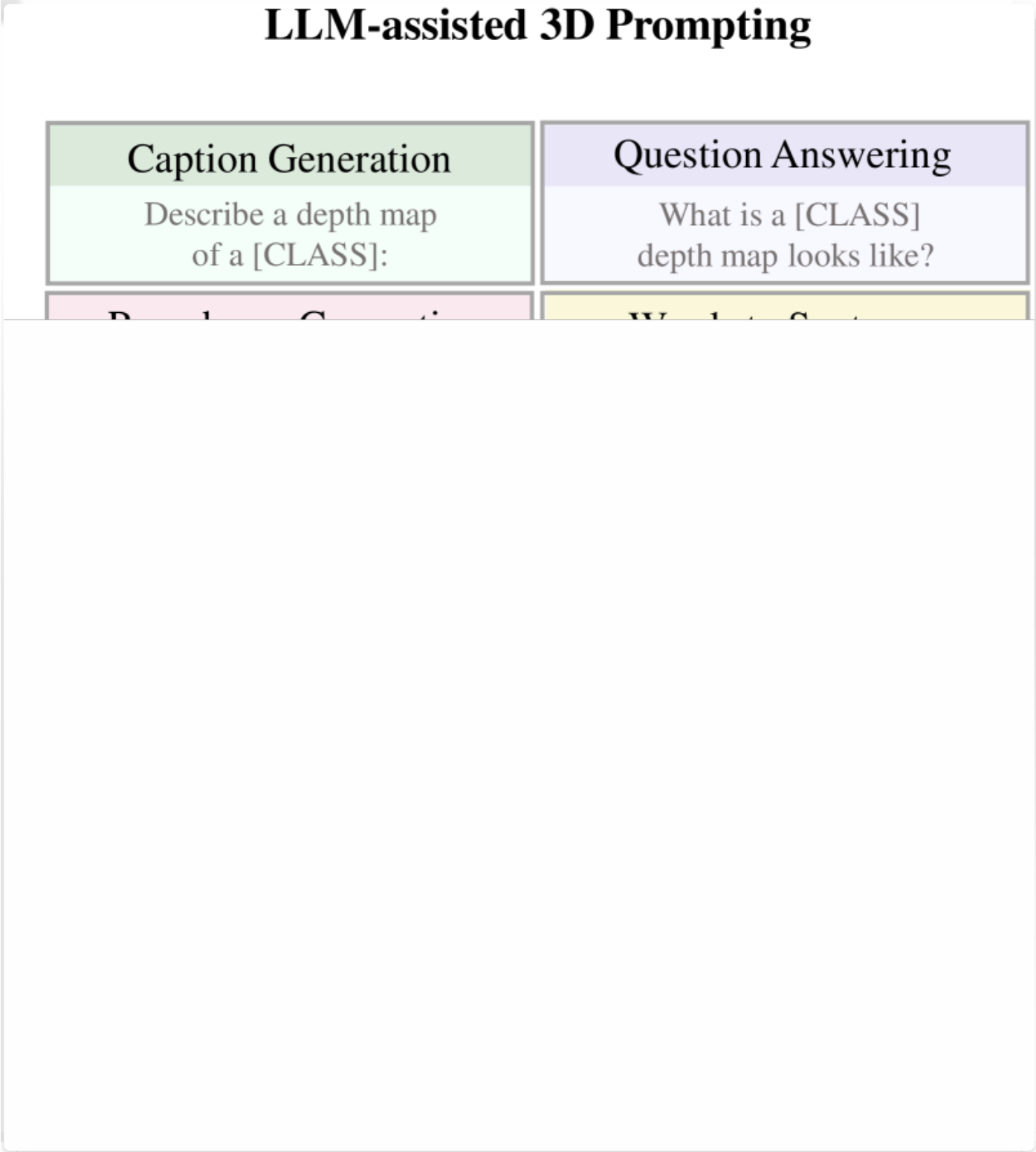
$$f_i^a = f_i + ReLU(f_{global}W_{3i}^T) \tag{2}$$

这个轻量级的三层MLP Adapter将新学到的 3D 小样本知识与 2D 预训练 CLIP 的知识融合在一起，进一步促进了跨模态知识的迁移。



Inter-view Adapter的详细结构。给定点云的多视图特征，适配器提取其全局表示并生成按视图自适应的特征。通过残差连接，新学到的 3D 知识被融合到预先训练的 CLIP 中。

这篇工作的另外一个改进则是对text input进行了改进，PointCLIP使用的是"Point Cloud Depth Map of a [class]"，而PointCLIP V2使用了GPT生成的text，具体的生成方法是根据四种Prompt生成多样的text，论文中没有提到具体的数量，我们在复现以及阅读代码的过程中发现，PointCLIP V2重复Prompt了很多次，对于每一个类别都有1200条左右的text，最终通过在测试/验证集上进行搜索和筛选采取最对齐的每个类别的text。



将四种类型的语言命令输入到预训练LLMs中，这些命令为 CLIP 的文本编码器生成特定于 3D 的提示。实际上这里的Prompt有很多，大致分为这四个系列，最终生成1200左右条text-inputs

复现

代码来源

zeroshot_cls和zeroshot_seg是github链接https://github.com/yangyangyang127/pointclip_v2 的直接复现，fewshot_cls是结合了zeroshot_cls的代码与github链接https://github.com/ZrrSkywalker/PointCLIP 组合得到的代码，zeroshot_detection是根据作者文中意思在论文https://arxiv.org/abs/2109.08141 对应的3detr的github链接 https://github.com/facebookresearch/3detr 加上三个适配PointclipV2的py脚本实现的。

Zeroshot_seg

采用vit_b16作为Visual Encoder

Mertics	airplane	bag	cap	car	chair	earphone	guitar	knife	lamp	laptop	motorbike	mug	pistol	rocket	skateboard	table
ACC	52.97	86.12	76.51	57.44	67.31	84.77	87.36	86.73	57.15	76.28	71.69	83.15	79.87	70.60	82.40	75.48
IOU	33.44	60.41	52.88	27.21	51.54	56.47	71.36	76.72	44.68	61.51	31.49	48.01	46.07	49.67	43.91	61.14

Zeroshot_cls

表格中前一个值为accuracy before search，后一个值为accuracy after search

Visual Encoder	Modelnet40	Scanobjnn
rn50	38.90, 39.26	32.44, 34.87
rn101	44.00, 46.15	28.80, 30.50
vit_b16	63.82, 64.22	34.18, 35.88
vit_b32	54.78, 54.90	35.67, 36.71

通过实验可以看出，vit_b16在Modelnet40任务上表现显著好于其余三个Encoder，而在Scanobjnn上，vit_b32更好一些，数据基本符合文章结论。

Fewshot_cls

Modelnegt40的shot数为16，Scanobjnn的shot数为4，模型Batch由于3090显存限制统一设置为8，优化器采用SGD，"cosine"递减学习率
表格中前一个值为accuracy before search，后一个值为accuracy after search

Visual Encoder	Modelnet40	Scanobjnn
rn50	85.78, 89.95	69.50, 72.24
rn101	85.98, 90.07	70.61, 72.80
vit_b16	86.43, 90.03	71.48, 74.39
vit_b32	86.06, 89.06	70.85, 73.83

通过实验可以看出，四个Encoder在fewshot训练后差别不大，数据基本符合文章结论。

相关的消融实验

消融实验以vit_b32为例测试在Modelnet上不同shot数量的效果，保持参数不变，调整shot数为0,1,2,4,8,16,32，结果如下
表格中前一个值为accuracy before search，后一个值为accuracy after search

shot_number	Modelnet40
0	54.78, 54.90
1	61.55, 72.04
2	69.17, 80.79
4	76.86, 82.98
8	82.66, 88.05
16	86.43, 90.03
32	88.41, 92.14

Zeroshot_detection

本文直接基于3Detr的代码做的检测任务，只是简单分类而非做定位。

作者想法基本如下：
1.用训练好的3detr提取3D框，然后把框里面的points用分类器重新分类。
2.把一些点数少的box设置成没有物体。
3.对于类别，添加了一个额外的“non object”类。

由于这部分数据集过于庞大（Scannet有1.2T，且申请数据集流程复杂，耽误了时间）以及计算资源的限制，我们无法将实验完整复现，只能提供在很小的数据点上可以跑通的代码（也无法保证在全数据集上没有Bug），仅供参考，其中/datasets/dataset/dat.py是数据的下载方式，运行脚本为“CUDA_VISIBLE_DEVICES=0 python main.py --dataset_name scannet --nqueries 256 --test_ckpt pretrained/scannet_masked_ep1080.pth --test_only --enc_type masked”

改进原因

在讨论Few-shot学习过程中，我们首先考虑的是对视觉编码器（Visual Encoder）和文本编码器（Text Encoder）进行微调（finetune）。然而，由于编码器的参数众多而样本量有限，这种微调很容易导致过度拟合，可能会损害模型的性能。例如，PointCLIP论文在附录C中进行的消融研究（ablation study）就表明，同时微调适配器（Adapter）、视觉编码器和文本编码器仅能提高0.2%的准确率，而且在其他情况下，准确率并不比单独微调适配器更高。特别地，如果只对文本编码器进行解冻，可能会损害性能。

进一步地，我们考虑了对深度图（Depth map）的改进。在非参数的其他核稠密化实验中，我们并未观察到显著的效果提升。在Few-shot框架下，采用可学习的平滑分类（Learnable Smooth Classification），每个类别使用16个样本的情况下，准确率达到了89.55%，表明优化空间不大。PointCLIP V2在训练平滑网络的同时，还训练了视图间适配器（Inter-view Adapter），这已经对视觉特征进行了处理。

另一个值得注意的点是，优化文本输入（text input）可以在所有框架和任务中带来提升。在PointCLIP V2中，每个类别大约使用了1200个文本输入，但实际上，每一类最终只需要一个文本输入。PointCLIP V2的实现要求对每种背景网络（backbone network）和每种数据集都进行一次搜索，筛选过程中需要经历多次遍历以达到收敛。在类别较多的情况下，这个搜索过程会变得异常缓慢。因此，我们希望能够减少提示（Prompt）的数量，同时保持或提升效果。

改进方案

使用GPT-4用于text input

Classification

最初采用GPT-4的想法是增加一个Few-Shot学习步骤。同时，我们对比较GPT-4在Zero-Shot情况下生成的文本与深度图，以及GPT-3的相应输出的对齐程度，感到非常感兴趣。

Few-Shot

PointCLIP V2采用Few-Shot方法，通过可学习的平滑技术进行改进。而我们希望利用GPT-4强大的图文生成能力进行Few-Shot学习。

One text-input

最初的流程是给GPT-4提供四张深度图，然后根据它的回复生成每个类别的描述（每个类别对应一个描述）。我注意到这些描述中包含了许多关于各部分颜色深浅、形状位置等具体信息，这与PointCLIP的文本设计以及PointCLIP V2搜索出的文本输入有很大差异。具体内容见附件`./text/gpt4_few_one_V1.md`。按照这种方法得到的分类准确率只有大约6%，远低于PointCLIP的23.78%。参考PointCLIP和PointCLIP V2搜索出的最佳文本输入，我认为文本输入应该包含“Depth map”并明确提出类别名称。因此，我要求GPT-4在生成描述时以`"A depth map of [CLASS] would show"` 开头。具体内容见附件`./text/gpt4_few_one_V2.md`。这样做之后，准确率有了显著提升。

Four text-input

之后，我尝试使用PointCLIP V2的四种提示，即先使用四张图，然后更换提示词，以Few-Shot例子生成所有类别的描述。使用PointCLIP V2的搜索方法得到最终对应的文本输入。在这种情况下，每个类别对应四个文本输入。生成的文本位于`./text/gpt4_few_4.json`。同时我们尝试生成了zero shot以及GPT3版本，位于`./text/gpt4_zero_4.json`和`./text/gpt3_zero.json`，并进行了实验。具体的准确率和搜索时间对比将在实验结果部分展示。

采用不同核进行Zero-shot深度图映射

Classification

采用核的目的在于使得映射后的结果更加接近原始的点分布，且一定程度上增强了数据的特征，尝试不同的核旨在更好地增强数据特征，提升训练效果。
采用了Gaussian kernel, Exponential kernel, Mish kernel, Sigmoid kernel，在Modelnet40上的实验结果如下：
表格中前一个值为accuracy before search，后一个值为accuracy after search

Visual Encoder	Gaussian	Exponential	Mish	Sigmoid
rn50	38.90, 39.26	35.37, 38.74	36.87, 38.57	36.91, 38.57
rn101	44.00, 46.15	48.95, 51.22	46.15, 48.42	47.85, 49.55
vit_b16	63.82, 64.22	60.33, 63.29	60.45, 64.18	61.26, 64.34
vit_b32	54.78, 54.90	51.90, 52.47	53.77, 55.59	53.44, 55.27

观察可以得到，Gaussian核在除rn101的Encoder上表现均好于其它核，说明Encoder对数据特征提取后学习的效果也存在差异，整体上Gaussian核和Sigmoid核更能提取点的特征，但差别并不大，说明这个数据集在核变换的提取特征方法上效果并没有那么显著。

实验结果

Classification

在这个实验中，初始文本输入被设定为各个类别的名称，且全部使用了以 `vit_b16` 为骨架的CLIP模型。用户可以选择通过 `--post_search` 参数来决定是否进行搜索。所有的text input都被初始化为空。

为了修改文本输入的少量样本（few-shot）方法，实验中只使用了4张图片（分别是四个类别）作为输入，我们发现这样并不能带来准确率的提升。

在平滑（Smooth）方法的少量样本处理中，PointCLIP V2每个类别使用了4张图片。这种做法在实验中被采用以改善和细化模型的性能。

复现实验4 text-inputs搜索只需修改`./zero_shot/trainers/post_search.py`或者`./few_shot/trainers/search_prompts.py`中`modelnet40_1000.json`文件为`text`文件夹中三个`json`文件即可。

复现1 text-input只需修改`best_params.py`中对应backbone模型/数据集的prompts即可。

Methods	**Accuracy**	**Search time(s)**
Zero shot(four gpt4 text-inputs/without images)	44.08	3.157
Few shot(four gpt4 text-inputs/using images)	44.08	3.377
Zero shot(four gpt3 text-inputs)	44.08	3.240
Few shot(four text-inputs/without images)	79.05	3.433
Few shot(one text-input V1/using images)	6.81	0
Few shot(one text-input V2/using images)	25.16	0
Zero shot(PointCLIP V2)	64.50	101.708
Few shot(PointCLIP V2)	83.10	75.96

实验结果分析：

由于财力和时间到限制，我们无法确认GPT4生成PointCLIP V2代码中需要的巨量的text-input候选时的影响效果，但是从前三行以及分析搜索出的最佳text-inputs看出，GPT的版本以及是否使用GPT4的图生文能力对准确率的影响十分小（实际准确率精确到小数点18位，这几个方法的数值依然是相等的），第四行中使用Few shot训练Smooth以及Adapter函数优化Visual feature效果显著，我们认为限制Zero/few shot上限的因素之一是深度图与CLIP的训练集的颜色的差异，这个差异同样也造成了Visual encoder不能很好的得到深度图中物体的结构，而GPT4生成的对于物体形状颜色的信息并不能被对齐到Visual features。

第五第六行的对比则是体现了text-input一般来说需要一些关键词，不过从PointCLIP V2搜索出的最佳text-inputs我们可以分析出不同类别的物体深度图最对应的text-input差异也非常大，我认为是CLIP训练集不同类别的bias限制了text-input通过prompt的优化，而只能使用搜索获得改善。

第七、八行体现了这样的搜索时间的不同，可以分析出在优化训练Smooth以及Adapter后对齐效率更高，此时全局的搜索优化text-input影响不大。

实验的最终结论是得到限制CLIP效果上限的是深度度与训练集的差异以及不同类别在训练集中存在bias，虽然我们尝试优化text失败了，但是也可以得出每个类别text优化方向是不能通过简单prompt实现的，以及对齐的主要优化空间在于Visual部分。我们的实验也说明了可以在优化Visual德条件下进行text input数量的减少，并不会过于影响准确率。

小组成员分工

Presentation

- **Presenters**: 王昱琛, 刘智瀚

Report

- **Written by** 王昱琛:
  - 论文简介
  - 改进原因
  - 改进方案Classification部分
  - 实验结果以及分析
- **Written by** 刘智瀚:
  - Replication of the paper's result
  - Kernel Improment result

Task Allocation

- **Text Input** 改进: 王昱琛
- **Kernel** 改进: 刘智瀚