Canonical Shape Projection is All You Need for 3D Few-shot Class Incremental Learning

Ali Cheraghian*,1,2, Zeeshan Hayder*,1,2, Sameera Ramasinghe[3], Shafin Rahman[4], Javad Jafaryahya[5], Lars Petersson[2], and Mehrtash Harandi[6]

[1] Data61, CSIRO, Australia, [2] Australian National University, [3] Amazon, [4] North South University, [5] University of Technology Sydney, [6] Monash University
{ali.cheraghian, zeeshan.hayder, lars.petersson}@data61.csiro.au, ramasisa@amazon.com, shafin.rahman@northsouth.edu, javad.jafaryahya@student.uts.edu.au, mehrtash.harandi@monash.edu

Abstract. In recent years, robust pre-trained foundation models have been successfully used in many downstream tasks. Here, we would like to use such powerful models to address the problem of few-shot class incremental learning (FSCIL) tasks on 3D point cloud objects. Our approach is to reprogram the well-known CLIP-based foundation model (trained on 2D images and text pairs) for this purpose. The CLIP model works by ingesting 2D images, so to leverage it in our context, we project the 3D object point cloud onto 2D image space to create proper depth maps. For this, prior works consider a fixed and non-trainable set of camera poses. In contrast, we propose to train the network to find a projection that best describes the object and is appropriate for extracting 2D image features from the CLIP vision encoder. Directly using the generated depth map is not suitable for the CLIP model, so we apply the model reprogramming paradigm to the depth map to augment the foreground and background to adapt it. This removes the need for modification or fine-tuning of the foundation model. In the setting we have investigated, we have limited access to data from novel classes, resulting in a problem with overfitting. Here, we address this problem via the use of a prompt engineering approach using multiple GPT-generated text descriptions. Our method, C3PR, successfully outperforms existing FSCIL methods on ModelNet, ShapeNet, ScanObjectNN, and CO3D datasets. The code is available at https://github.com/alichr/C3PR.

Keywords: 3D shape projection · Model reprogramming · Few-shot class incremental learning

1 Introduction

Maintaining a balance between acquiring new concepts and retaining existing knowledge presents a significant challenge for machine learning algorithms, a

* denotes equal contribution.

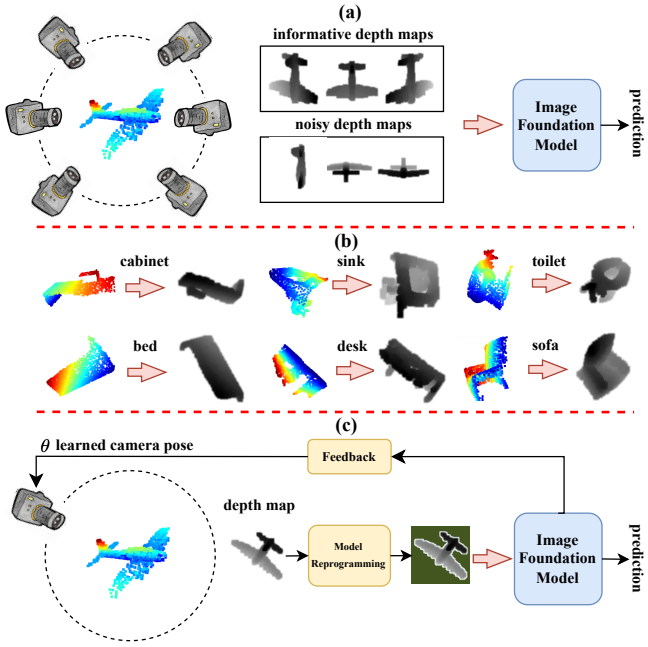

Fig. 1: **(a)** Traditional methods using image-based models for point cloud processing often generate multiple depth maps from fixed camera poses, potentially causing feature alignment issues and difficulties in recognizing 3D objects from unsuitable camera angles. **(b)** In contrast, our proposed approach (C3PR) determines the most suitable camera pose for projecting a single depth map by leveraging feedback from the foundation model. **(c)** To improve the foundation model's performance in handling 3D data for the FSCIL task, we introduce a novel model reprogramming technique (without modifying or fine-tuning the foundation model) to further refine the depth map.

task that comes naturally to humans. This task becomes more difficult when only a few samples are available for new concepts. In the literature, this problem is referred to as few-shot class incremental learning (FSCIL) [7,9,10,24,37]. Substantial effort on this topic has been invested, particularly on image data [7,9,10, 24,37], whereas relatively few studies are tailored specifically towards point cloud data. For example, in the 2D domain, there are vision-language models such as CLIP [29], ALIGN [17], and FLAVA [33] models with an inherent capacity to generate highly versatile representations and demonstrate zero-shot capabilities, all without parameter fine-tuning [20, 25, 42]. Gathering training data for 3D point clouds is substantially more challenging than for image data, and here we explore whether the CLIP model can help alleviate this issue. In this paper, we attempt to leverage CLIP for FSCIL on 3D Point Cloud Objects.

Popular approaches capture 2D images from a fixed set of camera poses and then pass the images to 2D foundation models for feature extraction [48, 51]. The main drawback of such a strategy is that camera poses are not adaptable

to the target dataset, hindering capturing the objects in poses ideal for feature extraction (see Fig. 1 (a)). To address this problem, we propose to learn the camera projection pose based on the dataset that best describes the 3D objects (see Fig. 1 (b)). Our proposed model reprogramming approach is advantageous for representing real-world scanned 3D objects where point cloud data contains sensor noise (see Fig. 1 (c)). The unique camera pose can still capture the global shape of noisy 3D objects, even though the object belongs to novel classes. For language/text modality, many existing works [9–11] leverages vision-language learning to address the overfitting issue associated with few-shot novel classes. We argue that vital knowledge lies within the language domain, which may aid in the training process of the vision domain, particularly for few-shot classes, to alleviate overfitting. However, thus far, existing methods have only utilized class names from the language domain [9, 10, 12]. In contrast, we propose a method based on prompt engineering to better leverage the language domain.

This paper presents a novel approach for extracting valuable insights from foundational models like CLIP, focusing on FSCIL on 3D point cloud data. As CLIP's training data comprises millions of image-text pairs, our target is to transfer this learned knowledge as a complete black box (without knowing or modifying pre-trained parameters) for 3D data. To this end, we propose reprogramming [38] the CLIP model, specifically for point cloud objects. The proposed approach is different from prior attempts [48, 51] where they used a fixed set of camera poses for 3D to 2D image projection and did not augment the input data for better usage of the CLIP encoder. Our C3PR method emphasizes generating depth map images with consistent, informative, background pixels, diverging from conventional model reprogramming techniques that frequently produce random pixel information for the background. We observed that maintaining consistent background pixels contributes to improved generalization, particularly for few-shot novel classes. These steps indirectly add essential perturbation to the input depth map images so that the black box CLIP vision encoder can operate without fine-tuning. Furthermore, we propose a novel approach that leverages information from the language domain using a Large Language Model, GPT, to combat overfitting issues in few-shot novel classes. Specifically, we generate multiple descriptive prompts (instead of single class name-based descriptions as used in [11]) for base and novel classes. It also compensates for the issue with limited data during novel class learning. The generated prompts are then employed in the CLIP text encoder, reducing overfitting during incremental training sessions. Both the vision and text pipelines of our method can be trained end-to-end. We report state-of-the-art performances when evaluated on 3D synthetic datasets, ModelNet [45] and ShapeNet [6], and two 3D real world-scanned datasets, ScanObjectNN [40] and Common Objects in 3D (CO3D) [31].

The main contributions of our proposed method are:

- Learning the preferred object-based camera pose for 3D to 2D projections;
- A novel model reprogramming approach for 2D CLIP encoders targeting 3D point cloud data;

– Prompt engineering based on a GPT model for novel few-shot classes to mitigate overfitting.

2 Related work

3D point cloud processing: In recent years, there has been a surge in research focused on leveraging deep learning for the direct processing of 3D point cloud objects. Notable contributions include PointNet [27] by Qi *et al.*, which introduced the use of multi-layer perceptron (MLP) networks for 3D point cloud processing. However, this approach overlooked the inherent local structures within the input data. Subsequent techniques were developed to address this limitation, including PointNet++ [28], which hierarchically extracts features to take advantage of local information. Building upon this, several works [21, 23, 26, 30, 44, 47] have proposed convolution strategies for extracting local information. PointConv [44] introduced a novel convolution operation that locates a Monte Carlo estimation of the hidden continuous 3D convolution based on a powerful sampling technique. SFCNN [30] proposed graph convolution on spherical points. Other papers [41, 43, 49] treat each point cloud as a graph vertex to learn features in either spatial or spectral domains. DGCNN [43] constructs a graph in feature space and dynamically updates it using MLP for each edge, while PointGCN [49] suggests an approach that generates the graph using k-nearest neighbors from a point cloud to capture local information.

Few-shot class-incremental learning: Tao *et al.* [37] were the pioneers in introducing the FSCIL setting for image data. They proposed the use of a neural gas network to address the issue of forgetting by preserving the topology of classes based on their feature vectors. Following this, in [10], a mixture of the subspaces are generated based on the training data distribution. In a related vein, Mazumder et al [24] devised a method that selectively adapts a subset of model parameters for learning novel classes, thereby curbing overfitting. They also combat forgetting by freezing critical parameters in the model. Cheraghian *et al.* [9] proposed a novel vision-language strategy, incorporating class semantic information from language space using a distillation technique to alleviate the impact of catastrophic forgetting. They also integrated an attention mechanism to counter overfitting on few-shot novel tasks. Introduced in [12], the Specific FSCIL tailored for 3D point cloud data presents the innovative concept of Microshape, which captures essential details from the primary task, effectively addressing both forgetting and overfitting challenges. Additionally, the work by [35] delves into cross-domain FSCIL applied to point-cloud recognition. This study dissects catastrophic forgetting into two distinct components: base class forgetting and incremental class forgetting, offering separate mitigation strategies.

Model reprogramming: Foundation model reprogramming has recently gained attention [8], primarily due to its reduced computational and time requirements compared to the fine-tuning approach. Prior to this, Elsayed et al. [15] introduced adversarial reprogramming. This technique involves mapping ImageNet labels to task labels, aiming to perplex the ImageNet classifier and induce be-

havior akin to a CIFAR-10 or MNIST classifier. Dinh *et al.* [13] proposed that input reprogramming is more effective in situations with limited labeled data than training from scratch and fine-tuning. Tsai et al. [39] introduced a technique called black-box adversarial reprogramming. This method involves transforming the input and integrating label mapping mechanisms into the output. In this paper, we present a novel foundation model reprogramming approach tailored for the CLIP model.

3 Method

In this section, we first introduce the FSCIL problem settings. Let's consider a sequence of T tasks denoted by $\mathbf{Q} = \{\mathcal{Q}^1, \mathcal{Q}^2, \cdots, \mathcal{Q}^T\}$, where $\mathcal{Y}^t$ represents the set of classes in task $\mathcal{Q}^t$, and $\mathcal{Y}^i \cap \mathcal{Y}^j = \emptyset$. Additionally, each class in all tasks is associated with a set of prompt class descriptions generated by an LLM, *e.g.*, GPT [4], denoted as $\mathcal{P}^t$. Thus, we can represent each task as a tuple $\mathcal{Q}^t = \{\mathcal{X}_i^t, \mathbf{y}_i^t, \mathbf{p}_i^t\}_{i=1}^{n_t}$, where $\mathcal{X}_i^t = \{\mathbf{x}_{i,j}^t\}_{j=1}^{l}$ refers to a 3D point cloud object with coordinates $\mathbf{x}_{i,j}^t \in \mathbb{R}^3$. Moreover, $\mathbf{y}_i^t \in \mathcal{Y}^t$ and $\mathbf{p}_i^t \in \mathcal{P}^t$ correspond to the label of the point cloud and its corresponding class prompt description, respectively. In the proposed FSCIL framework, $\mathcal{Q}^1$ is the base task ($t = 1$), where the model undergoes training on a large-scale synthetic 3D dataset. For $t > 1$, the training data are drawn from real-world 3D point clouds consisting of only a few instances. The model is trained sequentially across the tasks $t = 1, \ldots, T$. However, during the t-th task, the model is exposed to $\mathcal{X}^t$, $\mathbf{y}^t$, and $\{\mathcal{P}^1, \mathcal{P}^2, ..., \mathcal{P}^t\}$. During inference, the model trained on the current task $\mathcal{Q}^t$ is expected to classify test samples from both the current and previous tasks, *i.e.*, $\{\mathcal{Q}^1, \mathcal{Q}^2, \cdots, \mathcal{Q}^t\}$.

3.1 Model overview

In this section, we present an overview of our proposed methodology, illustrated in Fig. 2. We begin with a point cloud sample denoted as $\mathcal{X}_i^t$, which is passed through the projection module F. This module utilizes the rotation matrix $\mathbf{R}_i^t$ derived from the transformation module T_r to generate a depth map labelled as $\mathbf{I}_{D_i}^t$. For the projection module F, we use the method proposed by [51], which is fixed during the training stage. The transformation module T_r is crucial for determining the appropriate angle θ for projection. Additionally, a soft constraint loss function $\mathcal{L}_c$ is applied to the rotation matrix to ensure that the transformation module generates a valid rotation matrix. Following this, we extract a binary mask $\mathbf{I}_{M_i}^t$ from the depth map. Within this mask M, background pixels are assigned a value of zero, while non-zero depth map values are set to one. Subsequently, this mask is fed into a model reprogramming module C to add pixel information to the background of the depth map. To ensure that the generated embedding does not converge to a noisy image and follows the pattern of the mask input, an image gradient consistency loss function $\mathcal{L}_g$ is used between the input and output of the module C. The resulting image embedding $\mathbf{I}_{H_i}^t$ is then merged with the original depth map in the subsequent stage, resulting in

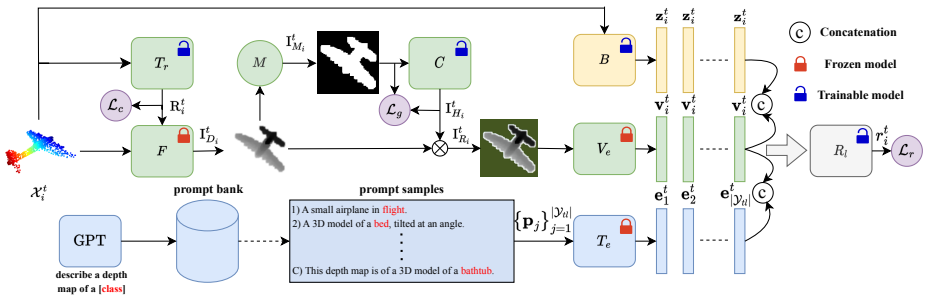

Fig. 2: Beginning with a point cloud sample $\mathcal{X}_i^t$, we use the projection module F to generate a depth map $\mathbf{I}_{D_i}^t$ via the rotation matrix $\mathbf{R}_i^t$ from the transformation module T_r. A mask $\mathbf{I}_{M_i}^t$ is then extracted from the depth map. This mask is input into module C and the resulting image is merged with the depth map, yielding the image $\mathbf{I}_{R_i}^t$. Next, $\mathbf{I}_{R_i}^t$ is input to the CLIP vision encoder model V_e to generate a vision embedding $\mathbf{v}_i^t \in \mathbb{R}^m$. Simultaneously, $\mathcal{X}_i^t$ is fed into B to create a point cloud representation $\mathbf{z}_i^t \in \mathbb{R}^m$. Moreover, the GPT model generates multiple class name descriptions, which are then stored in a prompt bank. To this end, prompts $\left\{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_{|\mathcal{Y}_{tl}|}\right\}$ for all old classes and the current task t are forwarded into the text encoder T_e, producing corresponding text embeddings $\left\{\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_{|\mathcal{Y}_{tl}|}\right\}$, where $\mathbf{e}_i^t \in \mathbb{R}^m$. Finally, $\mathbf{v}_i^t$, $\mathbf{z}_i^t$, and $\mathbf{e}_j^t$ are concatenated and passed through the relation module R_l to generate a scalar value r_i^t, serving as a similarity score.

the production of a reprogrammed depth map image denoted as $\mathbf{I}_{R_i}^t$. After that, $\mathbf{I}_{R_i}^t$ is input into the vision encoder of the CLIP model V_e to generate a vision embedding represented by $\mathbf{v}_i^t \in \mathbb{R}^m$. Simultaneously, the point cloud input $\mathcal{X}_i^t$ is forwarded to a 3D point cloud backbone, for instance, Curvenet [46], to generate a feature description $\mathbf{z}_i^t \in \mathbb{R}^m$. At the same time, GPT is employed to generate multiple descriptions for each class using the command: *describe a depth map of a [class]*, thereby creating several prompts, which are stored in the prompt bank module. Then, the prompts of all classes $\{\mathbf{p}_j\}_{j=1}^{|\mathcal{Y}_{tl}|}$, where $\mathcal{Y}_{tl} = \bigcup_{i=1}^{t} \mathcal{Y}^i$, observed up to task t are fed into the text encoder T_e to generate the corresponding text embeddings denoted as $\{\mathbf{e}_j\}_{j=1}^{|\mathcal{Y}_{tl}|}$, where each $\mathbf{e}_j^t \in \mathbb{R}^m$. The prompts are generated using LLMs, such as GPT [4], given a command: *Describe a depth map of a [class]*. In the subsequent stage, all $\mathbf{v}_i^t$, $\mathbf{z}_i^t$, and $\mathbf{e}_i^t$ are concatenated and passed through the relation module [34] R_l to generate a scalar value r_i^t ranging from 0 to 1. This value serves as a measure of the similarity between the visual and prompt feature embeddings. Finally, a binary cross-entropy loss function $\mathcal{L}_r$ is used to train the entire proposed pipeline.

3.2 Canonical 3D shape projection

A 3D-to-2D projection matrix can be designed to project any 3D point cloud on a 2D plane considering any camera angle. Nevertheless, the feature embedding

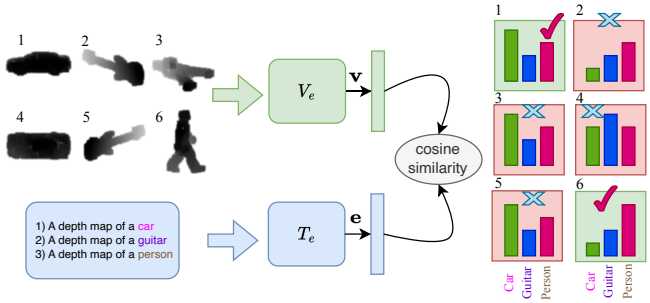

Fig. 3: We demonstrate the sensitivity of the CLIP model to the angle of depth map projection. Here, we consider two random projections to generate depth maps. Additionally, we assume that there are three classes: car, guitar, and person. Even though the number of classes is limited to only three, the CLIP model can only correctly classify two of these depth maps. This highlights the importance of finding the optimum projection angle to fully leverage the CLIP model for 3D point cloud processing.

produced by the visual encoder of large image-based foundation models (*e.g.*, CLIP) can be affected by the content that can be controlled by the projection angle. Figure 3 shows that an adjustment is pivotal due to the CLIP model's sensitivity to the projection angle, especially during depth map processing. We are interested in finding the optimal projection angle to enhance the performance of the CLIP model. When projecting from a 3D point cloud to 2D images, we leverage the mapping function introduced by [51]. We provide optimum projection angle θ via a rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$, where $\mathbf{R} = T_r(\mathcal{X})$, generated by the transformation module T_r to ascertain the most effective projection angle for converting a 3D point cloud into a depth map. The generated $\mathbf{R}$ is then applied to the given 3D point to derive the appropriate angle for the projection module θ. To ensure that $\mathbf{R}$ is a valid rotation matrix, we apply the following loss function at the output of the transformation module T_r,

$$\mathcal{L}_c = \frac{1}{n}\sum_{i=1}^{n}\left(\left\|\mathbf{R}_i^T\mathbf{R}_i - \mathbf{I}\right\|_{\mathcal{F}}^2 + \left|\det(\mathbf{R}_i) - 1\right|\right) \tag{1}$$

where $\mathbf{R}_i = T_r(\mathcal{X}_i)$, n is the number of samples in the training set, $\|\cdot\|_{\mathcal{F}}^2$ denotes the Frobenius norm, $\mathbf{I}$ is the identity matrix. The orthogonality condition $\left\|\mathbf{R}_i^T\mathbf{R}_i - \mathbf{I}\right\|_F^2$ guarantee that the columns of $\mathbf{R}_i$ form an orthonormal basis to preserve distances and angles for rotations. $\left|\det(\mathbf{R}_i) - 1\right|$ affirms that $\mathbf{R}_i$ is an orientation-preserving transformation matrix and doesn't involve any reflection.

In practice, the T_r matrix may not be guaranteed to be a rotation matrix on a per-batch basis as the above condition is only a soft constraint. This is overcome by continuous re-orthogonalization during the training process. Moreover, alternative methods like euler-angle or quaternion representations could be used. Nevertheless, the proposed approach of enforcing orthogonality through the L_c is a known method in the literature [1]. In the end, when provided with a 3D

point cloud sample $\mathcal{X}_i$, the transformation module T_r computes the desired rotation matrix $\mathbf{R}_i$ while taking into account all the aforementioned constraints to ensure its validity. This process results in the generation of a learned depth map image, $\mathbf{I}_{D_i} = F(\mathcal{X}_i, \mathbf{R}_i)$

3.3 Model reprogramming

Model reprogramming of computer vision models, especially foundational ones, is a crucial step in their effective deployment, enabling a seamless transition to new tasks and domains with minimal adjustments to existing architectures [38]. We present a novel model reprogramming approach specifically tailored for the CLIP model applied to the FSCIL task on 3D point cloud objects. It is worth noting that the CLIP model is primarily trained on RGB images rather than depth maps. Therefore, to enhance the CLIP model's effectiveness with depth maps, we incorporate pixel information into the background of the depth map images. More specifically, this modification assists the CLIP model in accurately discerning the boundary of the depth map shape information through a C module, which takes the form of a U-Net [32]. To accomplish this, we initially compute the mask map $\mathbf{I}_{M_i}$ corresponding to a given depth map $\mathbf{I}_{D_i}$, which isolates the shape boundary from the background. Next, we input $\mathbf{I}_{M_i}$ into the C module to extract the pixel information for the background, denoted as $\mathbf{I}_{H_i}$. To ensure the consistency of the generated pixels in $\mathbf{I}_{M_i}$ from the C module and to prevent convergence to arbitrary values, we employ the following loss function,

$$\mathcal{L}_g = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{\partial \mathbf{I}_{M_i}}{\partial x} - \frac{\partial \mathbf{I}_{H_i}}{\partial x} \right)^2 + \left(\frac{\partial \mathbf{I}_{M_i}}{\partial y} - \frac{\partial \mathbf{I}_{H_i}}{\partial y} \right)^2 \tag{2}$$

where $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ denote the derivatives of the images in the x and y directions, respectively. This loss enforces a smoothness constraint on the gradient to separate it from foreground objects explicitly.

It helps to learn novel classes with few examples. Subsequently, we combine the depth map $\mathbf{I}_{D_i}$ from the projection module F with the extracted values $\mathbf{I}_{H_i}$ from C to generate the final image, $\mathbf{I}_{R_i} = \mathbf{I}_{D_i} \odot C(M(\mathbf{I}_{D_i}))$, where $\odot$ represents the element-wise multiplication.

3.4 Prompt engineering

Advancements in prompt engineering have unveiled significant potential, primarily driven by LLMs [18, 19]. LLMs can extract richer information from appropriately tailored prompts. With this motivation, we employ GPT to obtain more insights from plain class names and enhance the FSCIL task on 3D data. In contrast to prevailing methods [9, 10, 12] that rely solely on the class name for the FSCIL task, we extend our method by generating additional descriptions to extract more semantic information from the CLIP text encoder (denoted as T_e). This augmentation ensures a more comprehensive utilization of the LLM's

capabilities, thereby enhancing the overall efficacy of our approach. In the FSCIL, a significant challenge arises due to the scarcity of training samples for novel classes, potentially leading to overfitting. To address this issue, we propose an innovative strategy leveraging the capabilities of LLM, specifically the GPT model, and utilizing this as novel prompts for the text encoder of the CLIP model. More precisely, we utilize the GPT model to generate multiple descriptions for each class by providing it with a specific command. These prompts are formally defined for each sample as follows:

$$\mathbf{p}_i = \text{GPT}(\text{Commands}). \tag{3}$$

Here, the Commands are structured as *Describe a depth of a [class].* For every class in the dataset, we substitute the respective class names in place of '[class]' in the command and input them into GPT-4. This results in the generation of depth map-specific descriptions with comprehensive category-wise semantics. Subsequently, we feed the descriptions of each category through CLIP's textual encoder $\mathcal{T}_e$. Finally, the generated prompts $\mathbf{p}_i$ are used in the training of the few-shot classes.

3.5 Class incremental learning

Training. For the base task, denoted as $\mathcal{Q}^1$, we conduct training for the transformation model T_r, the 3D backbone B, and the relation module R_l. In the case of subsequent tasks, denoted as $\mathcal{Q}^t$, where $t > 1$, only the relation module undergoes fine-tuning. Throughout both base and subsequent novel tasks, all other components in our proposed architecture (refer to Figure 2) remain frozen. Additionally, for each previously learned class, we randomly select a sample to be stored in a compact memory module $\mathcal{M}$. Given the i^{th} point cloud $\mathcal{X}_i^t$ of the t^{th} task, the B module generates the features $\mathbf{z}_i^t$ and the V_e module produces $\mathbf{v}_i^t$. Simultaneously, the prompts of all classes for tasks $\mathcal{Q}^1, \ldots, \mathcal{Q}^t$ are fed into the text encoder T_e to generate the corresponding text embeddings denoted as $\{\mathbf{e}_j\}_{j=1}^{|\mathcal{Y}_{tl}|}$. Then, we concatenate $\mathbf{z}_i^t$, $\mathbf{v}_i^t$, and $\mathbf{e}_j^t$ and forward them into the relation module R_l, which provides a score between $[0, 1]$, representing the similarity between them. In other words, for each training sample, we generate a score against each of the classes in both the subsequent and previous tasks as $r_{ij}^t = \gamma \circ R \circ (\mathbf{z}_i^t \oplus \mathbf{v}_i \oplus \mathbf{e}_j), j \in \mathcal{Y}_{tl}$, where $\mathcal{Y}_{tl} = \bigcup_{i=1}^{t} \mathcal{Y}^i$, $\oplus$ is the concatenation operator, R is the relation module, and γ is the sigmoid function. Finally, for each feature r_{ij} and the corresponding ground truth $\mathbf{y}_i$, we employ a binary cross-entropy cost function to train the model as follows,

$$L_r = -\frac{1}{|\mathcal{Y}_{tl}|\,|S|} \sum_{k \in \mathcal{Y}_{tl}} \sum_{\mathbf{y}_i \in \mathcal{S}} \Big(\mathbf{1}(y_i^t == k) \log(r_{ik}) + \big(1 - \mathbf{1}(y_i^t == k)\big) \log(1 - r_{ik}) \Big),$$

where $\mathcal{S}$ denotes the set of true labels in the current task, along with the memory $\mathcal{M}$. Finally, the total loss to train the entire pipeline is as follows,

$$\mathcal{L}_t = \lambda_1 \mathcal{L}_c + \lambda_2 \mathcal{L}_g + \lambda_3 \mathcal{L}_r \tag{4}$$

where λ_1, λ_2, and λ_3 are the weighting factors for the constraint loss ($\mathcal{L}c$), gradient loss ($\mathcal{L}g$), and the relation loss ($\mathcal{L}_r$) respectively.
Inference. During the inference process, when given an unlabeled sample $\mathcal{X}^c$, where $c \in \mathcal{Y}_{tl}$, the label prediction y^* is computed using the following equation:

$$y^* = \underset{j \in \mathcal{Y}_{tl}}{\arg\max} \big(R \circ (\mathbf{z}_c \oplus \mathbf{v}_c \oplus \mathbf{s}_j) \big), \tag{5}$$

where $\mathbf{z}_c$ and $\mathbf{v}_c$ represent the calculated features for the sample $\mathcal{X}^c$. We use a standardized prompt for all classes, specifically *This is a depth map of a [class].*

4 Experiments

Datasets. We employed two synthetic 3D datasets, Modelnet40 [45] and Shapenet [6], in addition to two real scanned 3D datasets, ScanobjectNN [40] and CO3D [31]. We follow the setup proposed by [11] for Few-Shot Classification with Incremental Learning (FSCIL) on point cloud objects. Our proposed method is assessed in two distinct scenarios. Firstly, we evaluate its performance on the FSCIL task within the same dataset, where both the base and novel tasks originate from identical datasets. Secondly, we subject our method to more demanding and realistic challenges by testing it in cross-dataset scenarios. In this setting, the base task involves synthetic datasets, while the novel tasks comprise classes from real scanned few-shot datasets.
Prompt generation. GPT-4 was employed to produce diverse class descriptions for each class. The following command was utilized: *Describe a depth of a [class].* Approximately 100 class descriptions were generated for each class and utilized during the training phase.
Implementation details. In our experimentation, we utilize Curvenet as the backbone for 3D point clouds (B) [46]. The CLIP vision (V_e) and text encoder (T_e) are instantiated using the ViT-B-16 architecture [14]. The C model is constructed based on the U-Net architecture [32]. The projection module (F) follows the methodology proposed by Zhu et al. [51], and for the transformation module (T_r), we employ the STN model [16]. Additionally, the relation module comprises several fully connected layers with sizes (1024, 2048, 1). We use the farthest 1024 points from 3D point cloud objects as input for all samples. For the base task, we train our proposed method (Figure 2) in an end-to-end manner for 250 epochs using the Adam optimizer with a learning rate of 0.001. For the novel task, we fine-tune our model for 200 epochs using the Adam optimizer with a learning rate of 0.001. It is important to note that we randomly shift and scale points in the input point cloud during both base and novel class training, with the additional step of randomly dropping out points. The feature vector dimensions for B, T_e, and V_e are set to 512. Additionally, the weights of the loss functions λ_1, λ_2, and λ_3 are configured as 0.01, 1, and 1 across the entire datasets. Our experiments are conducted using the *PyTorch* framework.
Evaluation metrics. Following the training of each task, we compute the accuracy by consolidating both base and novel classes. Subsequently, following the

recommendation in [36], we determine the relative accuracy dropping rate, denoted as Δ, using the formula $\Delta = \frac{|acc_T - acc_0|}{acc_0} \times 100$, where acc_T and acc_0 denote the accuracy of the last and first tasks, respectively. The value of Δ serves as a comprehensive metric for method evaluation. A lower relative accuracy signifies superior performance.

4.1 Main results

We conducted a comparative analysis of our approach against established methods in the field, including IL2M [2], ScaIL [3], EEIL [5], LwF [22], FACT [50], Sem-aware [9], and Microshape [12]. Notably, all these methods are tailored for image data, with the exception of Microshape [12], which was originally designed for point cloud data.

Within dataset: In this scenario, both the base and novel tasks are derived from the same dataset. Roughly half of the classes in the dataset are assigned to the base task, while the remaining half are incrementally introduced as multiple novel tasks. It is presumed that these novel tasks encompass only a limited number of samples for training. Our proposed method has been evaluated on three datasets—ModelNet, CO3D, and Shapenet. As depicted in Table 1, our approach demonstrates superior performance across all methods in terms of the relative accuracy dropping rate, denoted as Δ, indicating the effectiveness of our proposed method.

Cross dataset: In this arrangement, the base task centers around synthetic point cloud objects, while the novel few-shot tasks incorporate real-scanned 3D objects. Notably, the latter often exhibit incompleteness or noise in contrast to the synthetic data, which typically boast cleanliness and completeness. Consequently, this configuration introduces a more realistic and challenging scenario compared to within-dataset experiments. Three datasets are considered in this setting: *Shapenet* $\rightarrow$ *CO3D*, *ModelNet40* $\rightarrow$ *ScanObjectNN*, and *Shapenet* $\rightarrow$ *ScanObjectNN*. As illustrated in Table 2, our method outperforms other approaches in all three setups, underscoring its efficacy in scenarios involving a domain gap between base and novel tasks.

Table 1: Summary of FSCIL results for within-dataset experiments.

	ModelNet						CO3D							ShapeNet							
Method	20	25	30	35	40	$\Delta\downarrow$	25	30	35	40	45	50	$\Delta\downarrow$	25	30	35	40	45	50	55	$\Delta\downarrow$
FT	89.8	9.7	4.3	3.3	3.0	96.7	76.7	11.2	3.6	3.2	1.8	0.8	99.0	87.0	25.7	6.8	1.3	0.9	0.6	0.4	99.5
Joint	89.8	88.2	87.0	83.5	80.5	10.4	76.7	69.4	64.8	62.7	60.7	59.8	22.0	87.0	85.2	84.3	83.0	82.5	82.2	81.3	6.6
LwF [22]	89.8	36.0	9.1	3.6	3.1	96.0	76.7	14.7	4.7	3.5	2.3	1.0	98.7	87.0	60.8	33.5	15.9	3.8	3.1	1.8	97.9
IL2M [2]	89.8	65.5	58.4	52.3	53.6	40.3	76.7	31.5	27.7	18.1	27.1	21.9	71.4	87.0	58.6	45.7	40.7	50.1	49.4	49.3	43.3
ScaIL [3]	89.8	66.8	64.5	58.7	56.5	37.1	76.7	39.5	34.1	24.1	30.1	27.5	64.1	87.0	56.6	51.8	44.3	50.3	46.3	45.4	47.8
EEIL [5]	89.8	75.4	67.2	60.1	55.6	38.1	76.7	61.4	52.4	42.8	39.5	32.8	57.2	87.0	77.7	73.2	69.3	66.4	65.9	65.8	22.4
FACT [50]	90.4	81.3	77.1	73.5	65.0	28.1	77.9	67.1	59.7	54.8	50.2	46.7	40.0	87.5	75.3	71.4	69.9	67.5	65.7	62.5	28.6
Sem-aware [9]	91.3	82.2	74.3	70.0	64.7	29.1	76.8	66.9	59.2	53.6	49.1	42.9	44.1	87.2	74.9	68.1	69.0	68.1	66.9	63.8	26.8
Microshape [11]	**93.6**	**83.1**	**78.2**	**75.8**	67.1	28.3	78.5	67.3	60.1	56.1	51.4	47.2	39.9	87.6	**83.2**	**81.5**	**79.0**	76.8	73.5	72.6	17.1
C3PR (ours)	91.6	82.3	75.8	72.2	**70.9**	**22.5**	**81.5**	**69.4**	**66.5**	**63.0**	**54.2**	**53.8**	**34.0**	**88.0**	81.6	77.8	76.7	**76.9**	**76.2**	**74.7**	**15.1**

Table 2: Summary of FSCIL results for within-dataset and cross-dataset experiments.

	ShapeNet → CO3D												ModelNet → ScanObjectNN					ShapeNet → ScanObjectNN				
Method	39	44	49	54	59	64	69	74	79	84	89	$\Delta\downarrow$	26	30	34	37	$\Delta\downarrow$	44	49	54	59	$\Delta\downarrow$
FT	81.0	20.2	2.3	1.7	0.8	1.0	1.0	1.3	0.9	0.5	1.6	98.0	88.4	6.4	6.0	1.9	97.9	81.4	38.7	4.0	0.9	98.9
Joint	81.0	79.5	78.3	75.2	75.1	74.8	72.3	71.3	70.0	68.8	67.3	16.9	88.4	79.7	74.0	71.2	19.5	81.4	82.5	79.8	78.7	3.3
LwF [22]	81.0	57.4	19.3	2.3	1.0	0.9	0.8	1.3	1.1	0.8	1.9	97.7	88.4	35.8	5.8	2.5	97.2	81.4	47.9	14.0	5.9	92.8
IL2M [2]	81.0	45.6	36.8	35.1	31.8	33.3	34.0	31.5	30.6	32.3	30.0	63.0	88.4	58.2	52.9	52.0	41.2	81.4	53.2	43.9	45.8	43.7
ScaIL [3]	81.0	50.1	45.7	39.1	39.0	37.9	38.0	36.0	33.7	33.0	35.2	56.5	88.4	56.5	55.9	52.9	40.2	81.4	49.0	46.7	40.0	50.9
EEIL [5]	81.0	75.2	69.3	63.2	60.5	57.9	53.0	51.9	51.3	47.8	47.6	41.2	88.4	70.2	61.0	56.8	35.7	81.4	74.5	69.8	63.4	22.1
FACT [50]	81.4	76.0	70.3	68.1	65.8	63.5	63.0	60.1	58.2	57.5	55.9	31.3	89.1	72.5	68.3	63.5	28.7	82.3	74.6	69.9	66.8	18.8
Sem-aware [9]	80.6	69.5	66.5	62.9	63.2	63.0	61.2	58.3	58.1	57.2	55.2	31.6	88.5	73.9	67.7	64.2	27.5	81.3	70.6	65.2	62.9	22.6
Microshape [11]	82.6	77.9	73.9	72.7	67.7	66.2	65.4	63.4	60.6	58.1	57.1	30.9	**89.3**	73.2	68.4	65.1	27.1	82.5	74.8	71.2	67.1	18.7
C3PR (ours)	**83.6**	**80.0**	**77.8**	**75.4**	**72.8**	**72.3**	**70.3**	**67.9**	**64.9**	**64.1**	**63.2**	**24.4**	88.3	**75.7**	**70.6**	**67.8**	**23.2**	**84.5**	**77.8**	**75.5**	**71.9**	**14.9**

4.2 Ablation Studies

Evaluating model reprogramming: In the conducted experiment (refer to Figure 4 **(a)**), we systematically omitted the model reprogramming module C and directly fed the depth map image $\mathbf{I}_D$ into the vision encoder V_e of the CLIP model. The results clearly illustrate the significance of the model reprogramming in our proposed approach, as evidenced by notable differences between the outcomes with and without this module.

Impact of the prompt engineering: In our ablation study (see Figure 4 **(b)**), we investigate the influence of prompt engineering in our methodology. Specifically, we streamline the training stage by substituting the original multiple prompts per class with a single fixed prompt. The outcomes underscore the effectiveness of our GPT-based prompt engineering approach.

Impact of canonical shape projection: Here, we evaluate our proposed canonical shape projection methodology in comparison to PointCLIPv2 [51], which generates multiple fixed-depth map projections for each input point cloud. Specifically, we eliminate the transformation module T_r Subsequently, we adopt the depth map generation method employed in PointCLIP v2, involving the use of ten fixed camera pose angles for computing ten depth map images. These images are then fed into the vision encoder V_e of the CLIP model, and the relation module is applied to merge the feature vector, mirroring our proposed pipeline. Notably, our method surpasses PointCLIPv2 by a significant margin (see Figure 4 **(c)**), highlighting the efficacy of our learned camera pose angle.

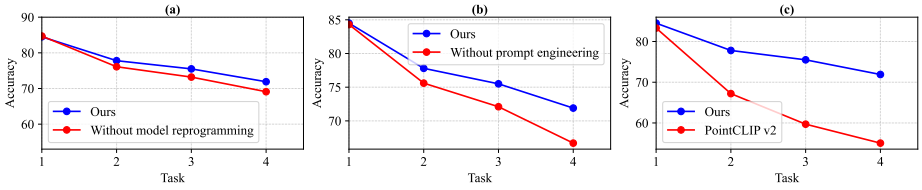

Fig. 4: This experiment is performed within the Shapenet to ScanobjectNN setting. In **(a)**, we present the influence of our novel CLIP model reprogramming strategy. **(b)** Highlights the impact of prompt engineering as proposed in our method. **(c)** Our approach is compared with PointCLIP v2 [51].

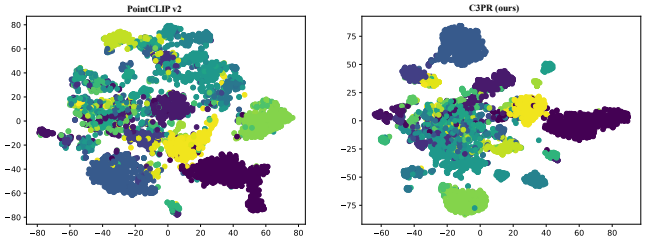

Fig. 5: We visualize the quality of features (from 44 base classes of ShapeNet) generated by the CLIP vision encoder using **(Right)** our proposed method, C3PR, and **(Left)** PointCLIPv2 [51]. Our method produced a denser cluster than PointCLIP.

Feature quality: In Figure 5, we assess the feature quality produced by the CLIP vision encoder using our proposed method, C3PR, and compare it with PointCLIPv2 [51]. As illustrated, our approach yields more discriminative clusters, enhancing overall generalization capabilities.

Influence of $\mathcal{L}_c$ and $\mathcal{L}_g$: In Table 3a, we highlight the impact of both $\mathcal{L}_c$ and $\mathcal{L}_g$ within our proposed pipeline. It is evident that both loss functions contribute significantly, but $\mathcal{L}_c$ holds greater importance, underscoring the significance of identifying the learned projection angle for depth map images on image foundation models.

Model reprogramming: Conventional vs. our: In this section, we contrast our innovative model reprogramming approach with the conventional method introduced by [15]. Table 3b illustrates the superior effectiveness of our proposed model reprogramming strategy.

Table 3: Ablation study on the influence of loss (a) and model reprogramming (b).

(a) Influence of $\mathcal{L}_c$ and $\mathcal{L}_g$ in our approach.

ShapeNet → ScanObjectNN					
Method	44	49	54	59	$\Delta\downarrow$
$\mathcal{L}_r$	84.4	76.2	70.1	65	22.9
$\mathcal{L}_r + \mathcal{L}_g$	84.3	77.3	70.8	66.9	20.6
$\mathcal{L}_r + \mathcal{L}_c$	**85.7**	**80.2**	74.5	70.8	17.3
$\mathcal{L}_r + \mathcal{L}_g + \mathcal{L}_c$	84.5	77.8	**75.5**	**71.9**	**14.9**

(b) Comparing our proposed model reprogramming strategy, C3PR, with the conventional model reprogramming technique employed in [15].

ShapeNet → ScanObjectNN					
Method	44	49	54	59	$\Delta\downarrow$
[15]	**85.7**	**80.2**	74.5	70.8	17.3
C3PR (ours)	84.5	77.8	**75.5**	**71.9**	**14.9**

4.3 Discussion

Point-cloud classification: Although our learning projection module Tr and model reprogramming module C have demonstrated effectiveness in the FSCIL context, they can also be applied in the point-cloud classification setting (shown in the table 4a. However, it is worth to note that our proposed prompt engineering strategy is specifically tailored to improve performance in FSCIL problems.

Module C role:

The role of Module C is twofold: it learns background information for the depth maps and smoothens the foreground. Specifically, we combine the depth

map $\mathbf{I}_{D_i}$ from the projection module F with the extracted values $\mathbf{I}_{T_i}$ from Module C to generate the final image, $\mathbf{I}_{R_i} = \mathbf{I}_{D_i} \odot C(M(\mathbf{I}_{D_i}))$, where $\odot$ represents element-wise multiplication. To this end, we conducted experiments to demonstrate the impact of Module C in our proposed method (see table 4b).

Table 4: Ablation on point-cloud classification (a) and impact of C module (b).

(a) The performance of our proposed method on point-cloud classification in ModelNet40.

Method	Accuracy
DGCNN [43]	92.9
CurveNet [46]	93.8
Ours	**94.6**

(b) The impact of module C in our proposed method.

ShapeNet → ScanObjectNN					
Method	44	49	54	59	$\Delta\downarrow$
Without C	84.5	76.3	73.2	68.9	18.4
With C	**84.5**	**77.8**	**75.5**	**71.9**	**14.9**

Limitation: The proposed method exhibits several limitations that warrant discussion. Firstly, its reliance on storing exemplars in memory, with one sample per class, and addressing the forgetting issue raises the question of exploring exemplar-free alternatives, which could offer intriguing avenues for further research. Secondly, our method would perform worse on new concepts from significantly different domains, synthetic to real-scan datasets, as can be seen in Table 2. However, it works well when the new concepts come from similar domains, synthetic to synthetic datasets, as can be seen in Table 1.

5 Conclusion

In this paper, we leverage the CLIP architecture of image-to-text matching to address the FSCIL problem on 3D point cloud objects. Traditional approaches usually project a 3D object on a 2D plane to produce images from different camera angles. Among those 2D images, some have bad/poor camera angles, making them difficult to recognize. Therefore, we propose a learning mechanism to find one camera projection that best describes the 3D objects. In doing so, we adopt a model reprogramming approach that perturbs the input 2D images (from the best projection angle) with background-foreground colorization (segmentation) without modifying or fine-tuning the original CLIP architectures. Our reprogramming approach is different from conventional approaches because our custom loss produces a uniform background color instead of the noisy background of previous works. In the text pipeline, we use the CLIP text encoder and train the whole architecture end-to-end. Instead of using a single class name-based text input, we suggest using prompts based on GPT-based description. We propose a prompt engineering approach to train the networks with 3D object and text description pairs. Finally, we evaluate our proposed method using 3D synthetic datasets, ModelNet and ShapeNet, and two 3D real-world-scanned datasets, ScanObjectNN and CO3D, and report state-of-the-art performances. In future, we will investigate model reprogramming for other vision problems beyond FSCIL, like segmentation and domain adaptation on 3D point clouds.

Acknowledgment. Mehrtash Harandi thanks the Australian Research Council for support through the Discovery Program (DP230101176).

References

1. Bansal, N., Chen, X., Wang, Z.: Can we gain more from orthogonality regularizations in training deep networks? In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018), https://proceedings.neurips.cc/paper_files/paper/2018/file/bf424cb7b0dea050a42b9739eb261a3a-Paper.pdf
2. Belouadah, E., Popescu, A.: Il2m: Class incremental learning with dual memory. In: CVPR (2019)
3. Belouadah, E., Popescu, A.: Scail: Classifier weights scaling for class incremental learning. In: WACV (2020)
4. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 1877–1901. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf
5. Castro, F.M., Marín-Jiménez, M.J., Guil, N., Schmid, C., Alahari, K.: End-to-end incremental learning. In: ECCV (2018)
6. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
7. Chen, K., Lee, C.G.: Incremental few-shot learning via vector quantization in deep embedded space. In: ICLR (2021)
8. Chen, P.Y.: Model reprogramming: Resource-efficient cross-domain machine learning (2023)
9. Cheraghian, A., Rahman, S., Fang, P., Roy, S.K., Petersson, L., Harandi, M.: Semantic-aware knowledge distillation for few-shot class-incremental learning. In: CVPR (2021)
10. Cheraghian, A., Rahman, S., Ramasinghe, S., Fang, P., Simon, C., Petersson, L., Harandi, M.: Synthesized feature based few-shot class-incremental learning on a mixture of subspaces. In: ICCV (2021)
11. Chowdhury, T., Cheraghian, A., Ramasinghe, S., Ahmadi, S., Saberi, M., Rahman, S.: Few-shot class-incremental learning for 3d point cloud objects. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) Computer Vision – ECCV 2022. pp. 204–220. Springer Nature Switzerland, Cham (2022)
12. Chowdhury, T., Cheraghian, A., Ramasinghe, S., Ahmadi, S., Saberi, M., Rahman, S.: Few-shot class-incremental learning for 3d point cloud objects. In: ECCV (2022)
13. Dinh, T., Seo, D., Du, Z., Shang, L., Lee, K.: Improved input reprogramming for gan conditioning (2022)
14. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. ICLR (2021)
15. Elsayed, G.F., Goodfellow, I., Sohl-Dickstein, J.: Adversarial reprogramming of neural networks. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=Syx_Ss05tm

16. Jaderberg, M., Simonyan, K., Zisserman, A., kavukcuoglu, k.: Spatial transformer networks. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 28. Curran Associates, Inc. (2015), https://proceedings.neurips.cc/paper_files/paper/2015/file/33ceb07bf4eeb3da587e268d663aba1a-Paper.pdf
17. Jia, C., Yang, Y., Xia, Y., Chen, Y.T., Parekh, Z., Pham, H., Le, Q., Sung, Y.H., Li, Z., Duerig, T.: Scaling up visual and vision-language representation learning with noisy text supervision. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 139, pp. 4904–4916. PMLR (18–24 Jul 2021), https://proceedings.mlr.press/v139/jia21b.html
18. Jia, M., Tang, L., Chen, B.C., Cardie, C., Belongie, S., Hariharan, B., Lim, S.N.: Visual prompt tuning. In: European Conference on Computer Vision (ECCV) (2022)
19. Lee, D.H., Pujara, J., Sewak, M., White, R.W., Jauhar, S.K.: Making large language models better data creators. In: The 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2023), https://openreview.net/forum?id=2Rdfdri2oT
20. Lee, K.Y., Zhong, Y., Wang, Y.X.: Do pre-trained models benefit equally in continual learning? In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 6485–6493 (January 2023)
21. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. In: NeurIPS (2018)
22. Li, Z., Hoiem, D.: Learning without forgetting. IEEE Transactions on Pattern Analysis and Machine Intelligence (2018)
23. Liu, Y., Fan, B., Xiang, S., Pan, C.: Relation-shape convolutional neural network for point cloud analysis. In: CVPR (2019)
24. Mazumder, P., Singh, P., Rai, P.: Few-shot lifelong learning. In: AAAI (2021)
25. Pei, Y., Qing, Z., CEN, J., Wang, X., Zhang, S., Wang, Y., Tang, M., Sang, N., Qian, X.: Learning a condensed frame for memory-efficient video class-incremental learning. In: Oh, A.H., Agarwal, A., Belgrave, D., Cho, K. (eds.) Advances in Neural Information Processing Systems (2022), https://openreview.net/forum?id=lCGYC7pXWNQ
26. Poulenard, A., Rakotosaona, M.J., Ponty, Y., Ovsjanikov, M.: Effective rotation-invariant point cnn with spherical harmonics kernels. In: 3DV (2019)
27. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: CVPR (2017)
28. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: NeurIPS (2017)
29. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 139, pp. 8748–8763. PMLR (18–24 Jul 2021), https://proceedings.mlr.press/v139/radford21a.html
30. Rao, Y., Lu, J., Zhou, J.: Spherical fractal convolutional neural networks for point cloud recognition. In: CVPR (2019)
31. Reizenstein, J., Shapovalov, R., Henzler, P., Sbordone, L., Labatut, P., Novotny, D.: Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In: ICCV (2021)

32. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. pp. 234–241. Springer International Publishing, Cham (2015)
33. Singh, A., Hu, R., Goswami, V., Couairon, G., Galuba, W., Rohrbach, M., Kiela, D.: FLAVA: A foundational language and vision alignment model. In: CVPR (2022)
34. Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P.H., Hospedales, T.M.: Learning to compare: Relation network for few-shot learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2018)
35. Tan, Y., Xiang, X.: Cross-domain few-shot incremental learning for point-cloud recognition. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 2307–2316 (January 2024)
36. Tan, Z., Ding, K., Guo, R., Liu, H.: Graph few-shot class-incremental learning. In: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (2022)
37. Tao, X., Hong, X., Chang, X., Dong, S., Wei, X., Gong, Y.: Few-shot class-incremental learning. In: CVPR (2020)
38. Tsai, Y.Y., Chen, P.Y., Ho, T.Y.: Transfer learning without knowing: Reprogramming black-box machine learning models with scarce data and limited resources. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 9614–9624. PMLR (13–18 Jul 2020), https://proceedings.mlr.press/v119/tsai20a.html
39. Tsai, Y.Y., Chen, P.Y., Ho, T.Y.: Transfer learning without knowing: Reprogramming black-box machine learning models with scarce data and limited resources. In: Proceedings of the 37th International Conference on Machine Learning. ICML'20, JMLR.org (2020)
40. Uy, M.A., Pham, Q.H., Hua, B.S., Nguyen, D.T., Yeung, S.K.: Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In: ICCV (2019)
41. Wang, C., Samari, B., Siddiqi, K.: Local spectral graph convolution for point set feature learning. In: ECCV (2018)
42. Wang, R., Duan, X., Kang, G., Liu, J., Lin, S., Xu, S., Lü, J., Zhang, B.: Attriclip: A non-incremental learner for incremental knowledge learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3654–3663 (June 2023)
43. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog) (2019)
44. Wu, W., Qi, Z., Fuxin, L.: PointCONV: Deep convolutional networks on 3D point clouds. In: CVPR (2019)
45. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D ShapeNets: A deep representation for volumetric shapes. In: CVPR (2015)
46. Xiang, T., Zhang, C., Song, Y., Yu, J., Cai, W.: Walk in the cloud: Learning curves for point clouds shape analysis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 915–924 (October 2021)
47. Xu, Y., Fan, T., Xu, M., Zeng, L., Qiao, Y.: Spidercnn: Deep learning on point sets with parameterized convolutional filters. In: ECCV (2018)
48. Zhang, R., Guo, Z., Zhang, W., Li, K., Miao, X., Cui, B., Qiao, Y., Gao, P., Li, H.: Pointclip: Point cloud understanding by clip. arXiv preprint arXiv:2112.02413 (2021)

49. Zhang, Y., Rabbat, M.: A graph-cnn for 3d point cloud classification. In: ICASSP (2018)
50. Zhou, D.W., Wang, F.Y., Ye, H.J., Ma, L., Pu, S., Zhan, D.C.: Forward compatible few-shot class-incremental learning. In: CVPR (2022)
51. Zhu, X., Zhang, R., He, B., Guo, Z., Zeng, Z., Qin, Z., Zhang, S., Gao, P.: Pointclip v2: Prompting clip and gpt for powerful 3d open-world learning. arXiv preprint arXiv:2211.11682 (2022)