PointTFA: Training-Free Clustering Adaption for Large 3D Point Cloud Models

Jinmeng Wu[1] , **Chong Cao**[1] , **Hao Zhang**[3*] , **Basura Fernando**[3] , **Yanbin Hao**[2] and **Hanyu Hong**[1]
[1]School of Electrical and Information Engineering, Wuhan Institute of Technology, Wuhan, China
[2]University of Science and Technology of China
[3]Center for Frontier AI Research (CFAR), Agency for Science, Technology and Research (A*STAR)
{shu2004910, caochong0617}@gmail.com, zhang_hao@ihpc.a-star.edu.sq,
fernando_basura@cfar.a-star.edu.sg, haoyanbin@hotmail.com, hhyhong@163.com

Abstract

The success of contrastive learning models like CLIP, known for aligning 2D image-text pairs, has inspired the development of triplet alignment for Large 3D Point Cloud Models (3D-PCM). Examples like ULIP integrate images, text, and point clouds into a unified semantic space. However, despite showing impressive zero-shot capabilities, frozen 3D-PCM still falls short compared to fine-tuned methods, especially when downstream 3D datasets are significantly different from upstream data. Addressing this, we propose a *Data-Efficient, Training-Free 3D Adaptation method named PointTFA* that adjusts ULIP outputs with representative samples. PointTFA comprises the Representative Memory Cache (RMC) for selecting a representative support set, Cloud Query Refactor (CQR) for reconstructing a query cloud using the support set, and Training-Free 3D Adapter (3D-TFA) for inferring query categories from the support set. A key advantage of PointTFA is that it introduces no extra training parameters, yet outperforms vanilla frozen ULIP, closely approaching few-shot fine-tuning training methods in downstream cloud classification tasks like ModelNet10 & 40 and ScanObjectNN. The code is available at: *https://github.com/CaoChong-git/PointTFA*.

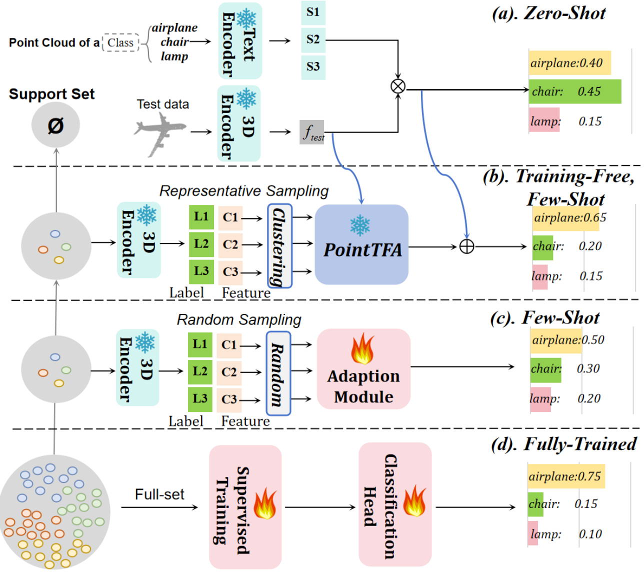


Figure 1: **Comparison of Different Conditions: Zero-shot, Training-Free and Fully-Trained.** (a). Zero-Shot doesn't utilize downstream data; (b). Training-Free (PointTFA), select representative samples for post-processing; (c). Few-Shot utilizes random samples to tune an Adaption Module; (d). Fully-trained use of all training samples to update the model's parameters. Fire/Ice denotes tuned or frozen.

1 Introduction

"*Is it always advisable to fine-tune models with high-dim point cloud inputs in the same manner as for low-dim ones (text)?*" Probably Not, as this question is particularly relevant in light of "**Curse of Dimensionality**" (COD) [Bellman, 1966]. According to COD, as dimensions increase, an exponential growth of samples' amount is required to maintain average distances between them. Otherwise, sparse distributions due to limited samples in high-dim spaces can easily lead to overfitting when training large models.

Recall in the convolution era, 3D cloud models were custom-designed for a variety of tasks, such as point cloud classification [Qiu *et al.*, 2021], object detection [Li *et al.*, 2021; Jiao *et al.*, 2024], and segmentation [Hu *et al.*, 2020; Li *et al.*, 2024]. These models, due to their limited parameters, required individual fine-tuning on different downstream datasets and often paid less attention to the COD issue. However, the sweep of foundational models to point cloud areas has significantly enlarged the scale of model parameters, resulting in an unavoidable COD challenge of data scarcity.

Foundational Model (FM) originated from Natural Language Processing (NLP) [Song *et al.*, 2023; Wu *et al.*, 2023] and became well-known with GPTs [Brown *et al.*, 2020]. They then moved into 2D vision-language areas (CLIP [Radford *et al.*, 2021],[Zhou *et al.*, 2024]) and are now also used in 3D understanding fields (ULIP [Xue *et al.*, 2023a]). Its success comes from two main factors: (1) it uses ***hyper-scale training data***, and (2) it relies on Transformer architecture with ***hyper-scale parameters***. This helps them work well

[*] Corresponding Author

across different tasks, as shown by their excellent zero-shot results [Xue *et al.*, 2023a].

But, when it comes to learning from a small amount of specific data, FMs are less economical than traditional models since large-scale FM would easily suffer from overfitting and incur its original generalizability [Wortsman *et al.*, 2022] with few training samples. Moreover, simply increasing the number of samples for downstream tasks is also not an optimal solution. High-dim samples demand substantially more computational resources than lower-dim ones. Thus, we must consider whether collecting and processing more high-dim data points is more efficient than before, just for one specific downstream 3D cloud task. It would be more practical to explore alternative methods beyond the current practice.

As in Figure 1, transferring a 3D point cloud model (3D-PCM) to downstream tasks typically operates under three conditions: zero-shot, few-shot, and fully fine-tuning. Zero-shot learning is training-free, whereas both few-shot learning and fine-tuning require updating partial or all parameters of the backbone/adaption module. This situation changes when a large-scale foundational model emerges with strong in-context learning ability by presenting a few relevant examples without training. It gains success on 1D text [Brown *et al.*, 2020] and 2D image tasks [Alayrac *et al.*, 2022]. This leads to a new, stricter condition of ***Training-Free, Few-Shot Learning*** for Large 3D-PCM than before. Training-free condition is particularly useful for adapting downstream 3D tasks, as it saves computations while approximating the performance of few-shot/fully tuned, achieving high cost-effectiveness.

In this paper, we introduce **PointTFA**, a **T**raining-**F**ree Clustering **A**daptation of large 3D models, aimed at enhancing point cloud classification. Our PointTFA is built on top of the frozen ULIP [Xue *et al.*, 2023a], with three new *parameter-free* modules: Representative Memory Cache (RMC), Cloud Query Refactor (CQR) and Training-Free 3D Adapter (3D-TFA). Specifically, the RMC selects the most representative samples into the support memory set from the training data for each new cloud category via unsupervised clustering algorithms. We validate that this data selection process is necessary and outperforms the randomly sampled support set (i.e., vanilla few-shot) by a large margin. The CQR serves to rebuild a test query with support set via a simplified attention module by eliminating all projection layers in [Vaswani *et al.*, 2017]. By treating the support set directly as “key” and “value”, the query feature is shifted in the identical distributions as the support set. Our 3D-TFA is inspired by 2D TIP-Adapter [Zhang *et al.*, 2022b], designed initially to adapt the CLIP for image few-shot learning. However, 3D-TFA is different because we customize the adaption for 3D point cloud inputs and integrate it with the 3D foundational model, extra data selection (RMC), and feature reconstruction modules (CQR).

We are the first to apply training-free conditions from zero-shot to few-shot for 3D point cloud understanding. By proposing a training-free, few-shot learning method, we omit the training process as zero-shot while approaching the performance of supervised few-shot learning. Experiments on ModelNet10, ModelNet40, and ScanObjectNN show the high cost-efficacy of PointTFA. We briefly summarize our contributions as follows.

- **PointTFA**: We introduce PointTFA, a new way to classify 3D point clouds without training. It has high training efficiency as a zero-shot method, saving on training computing, but still performs close to those finely tuned methods with a few-shot examples.
- **Representative Memory Cache** (RMC): Our research reveals that using clustering algorithms to create a support set for few-shot tasks is more effective than a randomly gathered set.
- **Cloud Query Refactor** (CQR): We demonstrate that a cloud query can be efficiently adapted to the distribution of the support set using a zero-parameter, simplified attention module.
- **Training-Free 3D Adapter** (3D-TFA): We’ve replicated the success of the Training-Free Adapter used in 2D vision models for 3D cloud models. This achieves high efficiency and effectiveness for 3D tasks.

2 Related Work

Our PointTFA is related to the following areas: training-free zero-shot, fine-tuned few-shot 3D cloud classification, and efficient cache models. We delve into each area below.

2.1 Training-Free, Zero-Shot 3D Model

Zero-Shot 3D cloud classification is naturally a *training-free task*, as no downstream training samples are available. Most current approaches use knowledge from similar fields, such as CLIP and ULIP, which are already trained on 2D/3D vision-language tasks. They then apply this pre-trained knowledge to new tasks they haven’t seen before. Notable techniques using 2D-CLIP include PointCLIP-V1/2, CLIP2Point, and ViT-Lens. For example, PointCLIP [Zhang *et al.*, 2022a] converts 3D clouds into several 2D depth images and then makes average predictions by processing these depth images through CLIP. PointCLIP-V2 [Zhu *et al.*, 2023] further expands from its V1 predecessor by using descriptive sentences from GPT-3 [Brown *et al.*, 2020] for category words. Similar in spirit, CLIP2Point [Goyal *et al.*, 2021] pre-trains adapters for CLIP to merge depth and rendered images using upstream datasets, then transfer learned weights to zero-shot downstream tasks. ViT-Lens [Lei *et al.*, 2023] integrates multiple modalities, including 3D clouds, into pre-trained vision transformers. ULIP [Xue *et al.*, 2023a] learns 3D encoders given 2D projected image and text features obtained by frozen CLIP. These training-free methods largely leverage pre-trained knowledge abstracted from large-scale upstream multimodal data.

2.2 Fine-Tuned, Few-Shot 3D Model

Few-shot learning typically includes a *learnable adaptation module* to bridge the distribution gap between upstream and downstream tasks. This module is plugged into a pre-trained base network and is then fine-tuned with a few training examples. For example, with few samples, the PointCLIPs [Zhang *et al.*, 2022a; Zhu *et al.*, 2023] incorporates a learnable adapter. This adapter adjusts individual 2D depth features with global information to better suit downstream tasks.

Moreover, CLIP2Point [Goyal *et al.*, 2021] fine-tunes the gate unit in feature fusion given downstream samples.

2.3 Efficient Cache Model

Efficient cache models, also a training-free paradigm, store training samples and features in a key-value database for adapting to downstream tasks without training. Models like [Khandelwal *et al.*, 2019], TIP-Adapter [Zhang *et al.*, 2022b], and Point-NN [Zhang *et al.*, 2023] use test features as queries for similarity-based retrieval from this database. Specifically, Point-NN pre-extracted hand-crafted query cloud feature, then matched it with training features stored in the memory bank. The TIP shares a similar strategy by replacing features extracted from the frozen CLIP model. Similar to TIP-Adapter, our method distinguishes itself by adding strategies for selecting support samples and aligning query features with the support set's distribution. We extend these training-free benefits to 3D cloud tasks.

3 Method

We first revisit ULIP for 3D recognition in §3.1. Next, we prepare the support memory for the whole training set in §3.2. Finally, we introduce PointTFA, which combines ULIP's pre-trained knowledge with cache information in a training-free manner in §3.3.

3.1 A Revisit of ULIP

ULIP aligns the semantical meanings of three modalities: text, 2D image, and 3D point cloud. Only the 3D encoder is tuned during pre-training on large-scale point cloud datasets (i.e., ShapeNet [Chang *et al.*, 2015]), while CLIP's text and image encoders remain frozen. This ensures that CLIP's knowledge is broadcasted intact to 3D understanding.

For a downstream zero-shot 3D cloud classification task with N new categories, we start by generating the category-specific classifier $\boldsymbol{W}_U \in \mathbb{R}^{N\times D}$, where D is the dimension of the textual feature. We form a sentence using the template "*point cloud of* `[category]`" and input it into the textual encoder to generate textual feature $\boldsymbol{s}_i$. The $\boldsymbol{s}_i$ is further normalized by the L2 norm. Repeating this for all N categories, we obtain the classifier $\boldsymbol{W}_U$, storing N categorical features. This procedure is detailed in Functions (1)-(2).

$$\begin{aligned} \boldsymbol{s}_i &= \mathrm{TextEncoder}\left(\text{“point cloud of [\texttt{category}]”}\right), \quad (1)\\ \boldsymbol{W}_U &= [\boldsymbol{s}_0, \boldsymbol{s}_1, \cdots, \boldsymbol{s}_N], \quad (2)\\ \boldsymbol{s}_i &\in \mathbb{R}^{1\times D}, \quad \boldsymbol{W}_U \in \mathbb{R}^{N\times D} \end{aligned}$$

Given an unseen test point cloud $\boldsymbol{T}$ with Z number of 3D coordinates. We feed $\boldsymbol{T}$ into pre-trained ULIP 3D cloud encoder to get cloud feature $\boldsymbol{f}_{\text{test}} \in \mathbb{R}^{1\times D}$. We also post-process $\boldsymbol{f}_{\text{test}}$ with the L2 norm. By multiplying cloud feature $\boldsymbol{f}_{\text{test}}$ with textual classifier $\boldsymbol{W}_U$, we could get the probability for N categories. Softmax function serves to normalize probability $\boldsymbol{y} \in \mathbb{R}^N$.

$$\boldsymbol{f}_{\text{test}} = \mathrm{3DEncoder}\left(\boldsymbol{T}\right) \quad (3)$$

$$\mathrm{logit} = \boldsymbol{f}_{\text{test}} \times \boldsymbol{W}_U^{\top} \quad (4)$$

$$\boldsymbol{y} = \mathrm{SoftMax}\left(\mathrm{logit}\right) \quad (5)$$

```
Algorithm 1: Data-Efficient RMC
  Input: point cloud tensor $\mathbf{F}_{\text{train}} \in \mathbb{R}^{\sum_{i=1}^{N} K_i \times D}$
  Output: point cloud tensor $\boldsymbol{C} \in \mathbb{R}^{MN\times D}$
          representative memory from K-means
          centroid
Let clusters $M$
for $i$ = 1 to N do
    Select the $i$-th $\mathbf{F}_i \in \mathbb{R}^{K_i\times D}$ from $\mathbf{F_{train}}$.
    Randomly initialize $M$ centroids:
      $\boldsymbol{C}_i = \{c_1, c_2, \ldots, c_M\}$;
    while Not Converged do
        Assign each point cloud feature to the nearest
          centroid;
        Update centroids based on the assigned point
          cloud features;
    end
    return Cluster centroids $\boldsymbol{C}_i \in \mathbb{R}^{M\times D}$
end
Concatenate $\boldsymbol{C}_i$ outputs $\boldsymbol{C} \in \mathbb{R}^{MN\times D}$.
return $\boldsymbol{C}$
```

3.2 Support Memory Preparation

Even though we don't update the model's parameters with training data under training-free conditions, we can still store annotated data in a database for nearest-neighbor searches. To do this, we'll prepare a large support memory to store the features of all training samples using a frozen ULIP model.

Given a training set containing N new classes, where each i-th class has K_i samples (for $i = 1, 2, \cdots, N$), we extract 3D cloud features for each class. The labels are also converted into one-hot vectors and stored in the support memory. For example, the j-th data sample in i-th class is denoted as $\boldsymbol{P}_{i,j}$, a bag of 3D coordinates. We extract its cloud feature $\boldsymbol{p}_{i,j}$ using ULIP 3D encoder and collect its labels $\boldsymbol{L}_{i,j}$ using the following functions. All feature $\boldsymbol{p}_{i,j}$ are post-processed with L2 norm.

$$\begin{aligned} \boldsymbol{p}_{\text{i,j}} &= \mathrm{3DEncoder}(\boldsymbol{P}_{i,j}), \quad (6)\\ \boldsymbol{L}_{\text{i,j}} &= \mathrm{OneHot}([\texttt{category}]), \quad (7)\\ \mathbf{F}_{\text{train}} &= \{\boldsymbol{p}_{i,j}\}, \quad (8)\\ \mathbf{L}_{\text{train}} &= \{\boldsymbol{L}_{i,j}\}, \quad (9)\\ \text{where,} \quad & i \in \{0, 1, 2, \cdots N\}, \quad j \in \{0, 1, 2, \cdots, K_i\}\\ & \boldsymbol{p}_{i,j} \in \mathbb{R}^{1\times D}, \quad \boldsymbol{L}_{i,j} \in \mathbb{R}^{1\times N} \end{aligned}$$

We collect features from all categories into $\mathbf{F}_{\text{train}}$ and corresponding labels in $\mathbf{L}_{\text{train}}$, where $\mathbf{F}_{\text{train}} \in \mathbb{R}^{\sum_{i=1}^{N} K_i \times D}$ and $\mathbf{L}_{\text{train}} \in \mathbb{R}^{\sum_{i=1}^{N} K_i \times N}$. By doing this, we convert the training split into a large support memory storing knowledge of a downstream task.

3.3 PointTFA

We present an overview of PointTFA, a training-free adaptation created to customize ULIP for the downstream 3D cloud

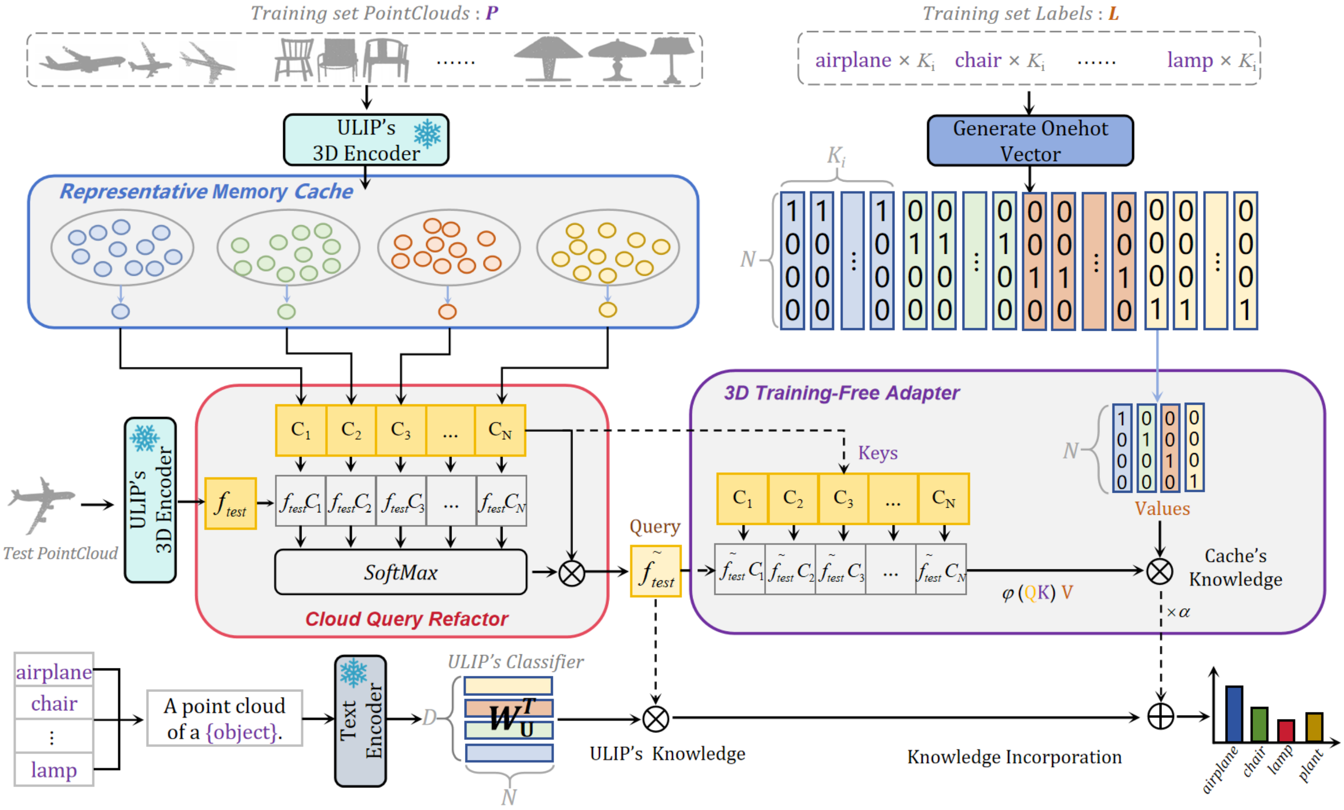


Figure 2: **PointTFA Framework Overview.** The PointTFA framework comprises three modules: Representative Memory Cache (RMC), Cloud Query Refactor (CQR), and Training-Free 3D Adapter (3D-TFA). RMC transforms the N-class training set into an efficient key-value cache, CQR reshapes the test feature using cache keys to obtain a query, and 3D-TFA predicts the query category using the key-value cache. The final prediction is obtained by a residual connection with ULIP priors.

classification in Figure 2. PointTFA consists of three components: the Representative Memory Cache (RMC), Cache Query Refactor (CQR), and 3D Training-Free Adapter (3D-TFA). The RMC is used to gather knowledge in pre-memory efficiently. The CQR transfers cache knowledge to the test features. Lastly, the 3D-TFA is responsible for predicting the final category. Let's explore each of these three modules in more detail.

Representative Memory Cache (RMC). In §3.2, we have stored all training samples in support memory to keep knowledge of downstream tasks. However, since samples in the same category are similar, keeping them all can waste computing resources because of redundancy.

To solve the issue, we suggest compressing the support memory based on the "Data-Efficiency" principle. This principle focuses on retaining maximum information relevant to the downstream task while reducing the number of samples stored in the support memory. One effective way to do this is by selecting representative samples from each category through an unsupervised clustering algorithm.

Specifically, we individually select a few samples from each category to form a smaller Representative Memory Cache than before. This is done by applying the *K-Means clustering* algorithm category-by-category. Denoting data features in the i-th category as $\boldsymbol{p}_i = [\boldsymbol{p}_{i,0}; \boldsymbol{p}_{i,1}; \boldsymbol{p}_{i,2}; \cdots; \boldsymbol{p}_{i,K_i}]$, we cluster them into M centers. This process reduces the number of data samples in a specific category from K_i to M, with M being set significantly smaller than K_i. We use matrix $\boldsymbol{C}_i$ to store the features of M centers, and collect all M one-hot vectors $\boldsymbol{L}_{i,0} \sim \boldsymbol{L}_{i,M}$ into $\boldsymbol{L}_i$ as labels for this category. Below Functions (10-11) show this process.

$$\boldsymbol{C}_i = \text{K-Means}\left(\boldsymbol{p}_i, M\right), \; \boldsymbol{C}_i \in \mathbb{R}^{M\times D}, \boldsymbol{p_i} \in \mathbb{R}^{K_i\times D} \quad (10)$$

$$\boldsymbol{L}_i = [\boldsymbol{L}_{i,0}; \boldsymbol{L}_{i,1}; \boldsymbol{L}_{i,2}; \cdots; \boldsymbol{L}_{i,M}], \quad \boldsymbol{L}_i \in \mathbb{R}^{M\times N} \quad (11)$$

After getting representative data for a category, we repeat this process for all categories, collecting features and labels separately into $\boldsymbol{C}_{\text{RMC}} \in \mathbb{R}^{(M\cdot N)\times D}$ and $\boldsymbol{L}_{\text{RMC}} \in \mathbb{R}^{(M\cdot N)\times N}$. Algorithm 1 shows the complete process of obtaining the efficient RMC.

$$\boldsymbol{C}_{\text{RMC}} = [\boldsymbol{C}_0, \boldsymbol{C}_1, \cdot\cdot, \boldsymbol{C}_i, \cdot\cdot, \boldsymbol{C}_N] \quad i \in \{1, 2, \cdots N\} \quad (12)$$

$$\boldsymbol{L}_{\text{RMC}} = [\boldsymbol{L}_0, \boldsymbol{L}_1, \cdots, \boldsymbol{L}_N], \quad (13)$$

After being processed by Data-Efficient strategy, the support memory, which previously contained full-set, is upgraded to a condensed memory containing MN samples, only covering important knowledge about each category.

Specifically, the cloud features $\mathbf{F}_{\text{train}} \in \mathbb{R}^{\sum_{i=1}^{N} K_i \times D}$ are simplified to $\boldsymbol{C}_{\text{RMC}} \in \mathbb{R}^{MN \times D}$, and the label features $\mathbf{L}_{\text{train}} \in \mathbb{R}^{\sum_{i=1}^{N} K_i \times N}$ are simplified to $\boldsymbol{L}_{\text{RMC}} \in \mathbb{R}^{MN \times N}$.

We further utilize the <feature, label> from the condensed RMC as "*key-value*" caches for retrieving downstream point cloud information. We verified in §4 that the RMC approaches the accuracy of using the full training set as support memory while with fewer query-key computations.

Cloud Query Refactor (CQR). To mitigate the distribution gap between the query point cloud and support point clouds in the RMC. We project the test query into the support-set space. Then, we reconstruct the query using the support set. We adopt the attention mechanism to achieve this target but remove all learnable linear projections.

Given a query $\boldsymbol{f}_{\text{test}} \in \mathbb{R}^{1 \times D}$ and a support set of cache $\boldsymbol{C}_{\text{RMC}}$, our goal is to generate a new $\tilde{\boldsymbol{f}}_{test}$ using $\boldsymbol{C}_{\text{RMC}} \in \mathbb{R}^{MN \times D}$. Specifically, the new of $\tilde{\boldsymbol{f}}_{test}$ projected into the cache space is obtained by a weighted sum of each element in the cache as Function (14). Hereby, $\boldsymbol{c}_i \in \mathbb{R}^{1 \times D}$ denotes the i-th row of the $\boldsymbol{C}_{\text{RMC}}$, τ is a constant temperature value for adjust density in weighting and set to 100.

$$\tilde{\boldsymbol{f}}_{\text{test}} = \sum_{i=1}^{MN} w_i \cdot \boldsymbol{c}_i \tag{14}$$

$$w_i = \frac{e^{\boldsymbol{f}_{\text{test}} \cdot \boldsymbol{c}_i^{\top} \cdot \tau}}{\sum_{i=1}^{MN} e^{\boldsymbol{f}_{\text{test}} \cdot \boldsymbol{c}_i^{\top} \cdot \tau}} \tag{15}$$

After reconstructing the query using samples from the RMC support set, the query features effectively shift to match the distribution of the downstream training set. We also observe that samples belong the same category are grouped denser after re-construction, with t-SNE visualization in §5. We experimentally verify the Cloud Query Refactor's (CQR) effects in §4.3 ablations. All these together demonstrate that the CQR is both simple and effective.

Training-Free 3D Adapter (3D-TFA). Our 3D-TFA, functioned by $\varphi(\cdot)$, is designed to generate categorical predictions for the query cloud $\tilde{\boldsymbol{f}}_{\text{test}}$, using a support memory that contains "*key-value*" pairs. To achieve this, we initially match the query feature with the "keys" in the support set to find the most relevant samples. This requires a similarity function $\theta(\cdot)$ to measure the "*query-key*" distance, as outlined in Function (16).

$$\begin{aligned} \boldsymbol{y}_{\text{TFA}} &= \varphi\left(\tilde{\boldsymbol{f}}_{\text{test}}, \boldsymbol{C}_{\text{RMC}}, \boldsymbol{L}_{\text{RMC}}\right) \\ &= \theta\left(\tilde{\boldsymbol{f}}_{\text{test}}, \boldsymbol{C}_{\text{RMC}}\right) \times \boldsymbol{L}_{\text{RMC}} \end{aligned} \tag{16}$$

We use the similarity function from 2D-TIP, as in Function (17), to measure the distances of 3D cloud features. This function adapts well to 3D features, despite the switch the feature extraction encoder from 2D-CLIP to 3D-ULIP.

$$\begin{aligned} \tilde{\boldsymbol{w}} &= \theta\left(\tilde{\boldsymbol{f}}_{\text{test}}, \boldsymbol{C}_{\text{RMC}}\right) \\ &= e^{-\beta\left(1 - \tilde{\boldsymbol{f}}_{\text{test}} \times \boldsymbol{C}_{\text{RMC}}^{\top}\right)} \end{aligned} \tag{17}$$

We define $\tilde{\boldsymbol{w}} \in \mathbb{R}^{1 \times MN}$ as the affinity vector of the query to the keys, where β is a positive constant. Thus, when $\tilde{\boldsymbol{f}}_{\text{test}}$ is close to a key in $\boldsymbol{C}_{\text{RMC}}$, the corresponding element in $\tilde{\boldsymbol{w}}$ will be larger. By combining Functions (16) and (17), we can obtain the prediction $\boldsymbol{y}$ which consists of N categorical probabilities.

$$\boldsymbol{y}_{\text{TFA}} = \tilde{\boldsymbol{w}} \times \boldsymbol{L}_{\text{RMC}} \tag{18}$$

Finally, we integrate the predictions from the 3D-TFA (Function (18)) with the zero-shot predictions (Function (5)) using a weighted sum.

$$\boldsymbol{y}_{\text{fuse}} = \alpha \cdot \boldsymbol{y}_{\text{TFA}} + \boldsymbol{y} \tag{19}$$

The constant α determines the amount of information from the support cache that influences the final query cloud predictions. Specifically, a large α is appropriate when there's a significant distribution shift between downstream and upstream data. Conversely, a small α suggests retaining more information from the upstream data. We set α in around 20 and β in around 10.

4 Experiments

We evaluate our plug-and-play PointTFA on five major Large 3D Point Cloud Models: PointNet2 (ssg) [Qi *et al.*, 2017], PointMLP [Ma *et al.*, 2022], PointBERT [Yu *et al.*, 2022], PointNEXT [Qian *et al.*, 2022], and PointBERT-ULIP-2 [Xue *et al.*, 2023b]. We assess the performance in a Training-Free Few-Shot scenario across three downstream datasets. Details of datasets are below.

Datasets. We tested on ModelNet10 & 40 [Wu *et al.*, 2015], [Wu *et al.*, 2015] and ScanObjectNN datasets [Uy *et al.*, 2019]. Specifically, ModelNet10 covers ten categories, with 3991 training samples and 908 test samples. ModelNet40 extended categories to forty, with 9843 training and 2648 testing samples. ScanObjectNN contains 2902 object scans across fifty categories. Moreover, this dataset provides three variants, namely OBJ_ONLY, OBJ_BG, and OBJ_T50RS, representing raw data, data with added background, and data with different noises, respectively.

4.1 Comparison With Train-Free, K-Shot Methods

Training-free few-shot learning is a relatively new area, emerging with the development of large foundational models. Currently, only a few pioneering works exist in 3D point cloud understanding. To provide a broader context, we extend our comparison to include both Training-Free Zero-Shot Learning and Fine-Tuned Few-Shot Learning.

We compare PointTFA with strong transfer learning SOTAs, including PointCLIP, CLIP2Point, RECON, OpenShape, ViT-Lens, ULIP, Point-NN, and TIP-3D in Table 1. Our PointTFA, built on top of ULIP-2 (`PointBERT`), shows competitive performance under strict conditions of Training-Free, few-shot settings for downstream tasks. Our PointTFA, with 16 shots, even surpasses fine-tuned few-shot methods PointCLIP and CLIP2Point.

To test PointTFA's upbound, we adopt the full set as a support set. This setting further improves on 16-shot settings. However, we could still balance performance and computations during testing with few-shot settings.

Method	Conditions (K-shot)	2D-data	3D-data	ModelNet40	ModelNet10	OBJ_ONLY	OBJ_BG	OBJ_T50RS
PointCLIP [Zhang *et al.*, 2022a]	***Train-Free*** (0)	✓	✓	20.18	30.23	19.28	21.34	15.38
CLIP2Point [Huang *et al.*, 2023]	***Train-Free*** (0)	✓	✓	49.38	66.63	30.46	35.46	23.32
PointCLIP-V2 [Zhu *et al.*, 2023]	***Train-Free*** (0)	✓	✓	64.22	73.13	50.09	41.22	35.36
RECON [Qi *et al.*, 2023]	***Train-Free*** (0)	✓	✓	61.70	75.60	43.70	40.40	30.50
OpenShape [Liu *et al.*, 2023]	***Train-Free*** (0)	✓	✓	85.30	-	-	56.70	-
VIT-Lens [Lei *et al.*, 2023]	***Train-Free*** (0)	✓	✓	87.60	-	-	60.10	-
ULIP-1 (`PointBERT`) [Xue *et al.*, 2023a]	***Train-Free*** (0)	-	✓	60.40	-	-	48.50	-
ULIP-2 (`PointBERT`) [Xue *et al.*, 2023b]	***Train-Free*** (0)	-	✓	75.60	-	-	-	-
Point-NN [Zhang *et al.*, 2023]	***Train-Free*** (Full)	-	✓	81.80	-	71.10	74.90	64.90
TIP-3D (our impl) [Zhang *et al.*, 2022b]	***Train-Free*** (16)	-	✓	86.06	89.76	73.49	75.56	59.61
CLIP2Point [Huang *et al.*, 2023]	Fine-Tune (16)	✓	✓	87.46	-	-	-	-
PointCLIP [Zhang *et al.*, 2022a]	Fine-Tune (16)	✓	✓	87.20	-	-	-	-
PointCLIP-V2 [Zhu *et al.*, 2023]	Fine-Tune (16)	✓	✓	89.55	-	-	-	-
Our PointTFA	***Train-Free*** (16)	-	✓	**89.79**	**92.62**	**80.90**	**82.10**	**67.18**
Our PointTFA	***Train-Free*** (Full)	-	✓	**90.88**	**93.17**	**83.48**	**84.85**	**68.22**

Table 1: Our approach significantly outperforms the original SOTA, exceeding the most advanced performance by more than 10% on ModelNet10 and ScanObjectNN. "2D-data" indicates the use of images in the inference process, and "3D-data" indicates the use of point clouds in the inference process.

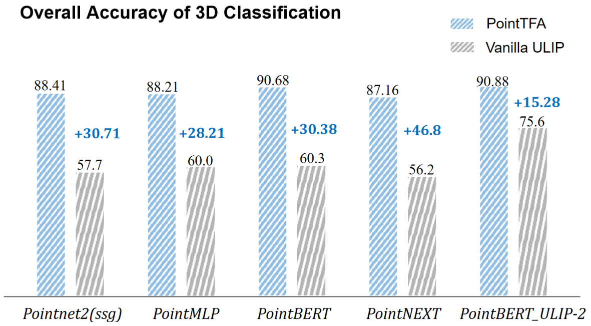


Figure 3: **Overall Accuracy of 3D Classification(%).** Our method demonstrates a substantial improvement over vanilla frozen ULIP 3D backbones in classifiying ModelNet40 dowstream data.

4.2 Comparison to Vanilla ULIP

We plug PointTFA into five vanilla frozen ULIP backbones without training for downstream tasks. We apply PointTFA with full set as support memory and present comparison in Figure 3. We observe that the Training-Free PointTFA significantly introduces improvement for all backbones. The most noteworthy improvement is observed for PointNEXT, where PointTFA boosts the final performance of the pre-trained PointNEXT model by 46.8%. This substantiates the effectiveness of our approach in bridging the domain gap between ULIP pre-training and unknown point clouds.

4.3 Ablations

We conducted an ablation study in three areas: the proportion of training samples, the validity of three modules, and the cache-building strategy. This section's 3D model adapted to PointTFA is PointBERT_ULIP-2.

Proportion of Training Set. We randomly select different percentages of samples for each category in the training set and utilize these samples as support memory to construct cache models. We gradually increased the sample proportion, i.e., from 10% to 100% on the three datasets.

Proportion	10%	20%	40%	60%	80%	100%
ModelNet10	88.44	90.86	91.19	92.84	92.29	93.17
ModelNet40	87.60	88.21	89.30	89.42	90.50	90.88
OBJ_ONLY	72.98	78.83	81.21	82.44	82.79	83.48
OBJ_BG	76.25	78.14	79.52	82.79	83.13	84.85
OBJ_T50RS	64.43	66.38	67.73	67.80	68.04	68.22

Table 2: **PointTFA (%) with different proportions of training set.** The accuracy gradually converges as the proportion of training samples increases.

Shots/Clusters	1	2	4	8	16
3D-TFA	74.80	77.92	83.10	83.79	86.06
+ RMC	83.14	85.41	87.44	88.13	88.49
+ CQR (PointTFA)	87.72	88.05	88.74	88.94	89.79

Table 3: **Validity of three modules.** We first perform a 3D implementation of the TIP-Adapter, then add the RMC and CQR modules in turn and analyze the potency of each module on ModelNet40.

As depicted in Table 2, the accuracy gradually converges as the percentage of samples increases. Notably, when the training samples of ScanObjectNN(OBJ_ONLY) grow from 10% to full-set, the performance fluctuates, hovering close to 10%. This observation underscores the impact of adopting different cache sizes or structures on the final performance. This analysis motivates our ongoing research to identify a balance between cache size and final performance—(RMC), leveraging a small amount of data to achieve excellent performance for enhanced data efficiency.

Validity of Three Modules. We validated the effectiveness of RMC, CQR, and 3D-TFA by adding modules incrementally. In Table 3, with the sequential addition of the RMC and CQR modules, the complete entity PointTFA leads the TIP-Adapter's [Zhang *et al.*, 2022b] 3D version (only-3D-TFA) by 12.92% at 1-shot on ModelNet40. We can conclude that the three modules of PoinTFA each synergize with each other.

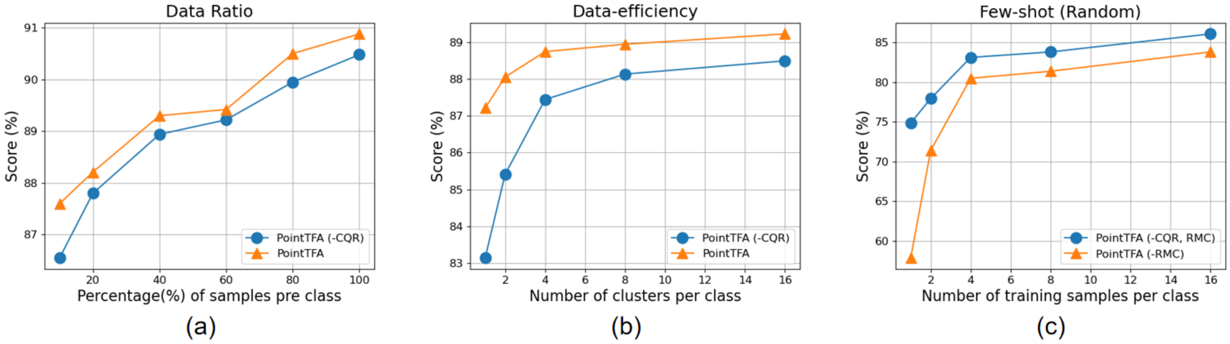


Figure 4: Performance of PointTFA and PointTFA (-CQR) with different cache construction strategies on ModelNet40 dataset.

we separately test $\{y_{\text{fuse}}, y_{\text{TFA}}, y\}$ of Function 19 in Table 4 to verify 3D-TFA adaptability.

	y	y_{TFA}	y_{fuse}
Accuracy	75.60	88.09	**89.79**

Table 4: Analysis results of 3D-TFA module.

Cache Construction Strategies. We test three different cache settings on ModelNet40: a cache constructed with different percentages of the training set, a random cache, and a catch with RMC. To show the effects of CQR under different cache settings, We denote the performance w/ and w/o CQR as PointTFA and PointTFA (-CQR).

In Figure 4 (a), PointTFA consistently outperforms PointTFA (-CQR) across all ratios of training data. This indicates that the Cloud Query Refactor (CQR) is effective when the support memory reaches a certain size.

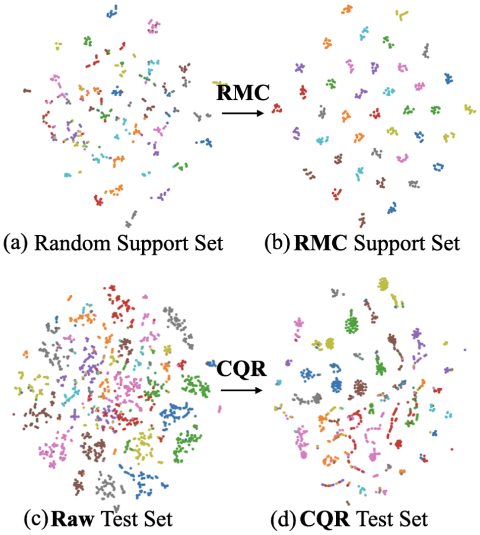
Figure 4 (b) shows that PointTFA consistently outperforms PointTFA (-CQR) under a different number of K-means cluster settings. Notably, when number of clusters is set to 1, PointTFA improves by 4.58%. This result highlights the RMC could effectively select representative few-shot samples while balancing cache size.

In Figure 4 (c), we analyze results for PointTFA (-CQR, RMC), showing that reshaping test features with random few-shot cache keys initially leads to decreased performance. The shows that CQR needs to be used with RMC.

In short, our ablation study shows that RMC can effectively select representative samples into a small cache size. Additionally, the CQR enhances the generalizability of PointTFA, even with limited cache availability.

5 Visualization

To show the effect of the CQR and RMC, we used t-SNE to visually compare the support sets (Random *vs* RMC) and test queries (Raw *vs* CQR). Figure 5 (a-b) show that the RMC support set selects more representative categorical samples than a random cache. Figure 5 (c-d) show that The test features after CQR become ***denser*** within the ***same*** class and more ***separable*** across ***different*** classes.



(a) Random Support Set (b) **RMC** Support Set

(c) **Raw** Test Set (d) **CQR** Test Set

Figure 5: t-SNE of Random&RMC support set, Raw&CQR test set.

6 Conclusions

We introduce PointTFA, a training-free adaption of large 3D point cloud models that constructs a cache model using few-shot knowledge to create an adapter. This approach effectively bridges the domain gap between ULIP pre-training and downstream point clouds. We propose a "Data-efficiency" (RMC) to capture representative information in support memory, forming a cache model that balances data dimension and baseline accuracy. Additionally, we present a reshaping technique (CQR) for projecting cache knowledge onto test features, enhancing the comprehensive understanding of cache information by test point clouds. Furthermore, PointTFA significantly improves the original capabilities of the ULIP pretrained 3D backbone, achieving state-of-the-art performance in training-free few-shot 3D Classification.

Acknowledgements

This research/project is supported by the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-016).

References

[Alayrac *et al.*, 2022] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

[Bellman, 1966] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[Brown *et al.*, 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[Chang *et al.*, 2015] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[Goyal *et al.*, 2021] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. In *International Conference on Machine Learning*, pages 3809–3820. PMLR, 2021.

[Hu *et al.*, 2020] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11108–11117, 2020.

[Huang *et al.*, 2023] Tianyu Huang, Bowen Dong, Yunhan Yang, Xiaoshui Huang, Rynson WH Lau, Wanli Ouyang, and Wangmeng Zuo. Clip2point: Transfer clip to point cloud classification with image-depth pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22157–22167, 2023.

[Jiao *et al.*, 2024] Yang Jiao, Zequn Jie, Shaoxiang Chen, Lechao Cheng, Jingjing Chen, Lin Ma, and Yu-Gang Jiang. Instance-aware multi-camera 3d object detection with structural priors mining and self-boosting learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 2598–2606, 2024.

[Khandelwal *et al.*, 2019] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

[Lei *et al.*, 2023] Weixian Lei, Yixiao Ge, Jianfeng Zhang, Dylan Sun, Kun Yi, Ying Shan, and Mike Zheng Shou. Vit-lens: Towards omni-modal representations. *arXiv preprint arXiv:2308.10185*, 2023.

[Li *et al.*, 2021] Zhichao Li, Feng Wang, and Naiyan Wang. Lidar r-cnn: An efficient and universal 3d object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7546–7555, 2021.

[Li *et al.*, 2024] Hao Li, Dingwen Zhang, Yalun Dai, Nian Liu, Lechao Cheng, Jingfeng Li, Jingdong Wang, and Junwei Han. Gp-nerf: Generalized perception nerf for context-aware 3d scene understanding. *CVPR*, 2024.

[Liu *et al.*, 2023] Minghua Liu, Ruoxi Shi, Kaiming Kuang, Yinhao Zhu, Xuanlin Li, Shizhong Han, Hong Cai, Fatih Porikli, and Hao Su. Openshape: Scaling up 3d shape representation towards open-world understanding, 2023.

[Ma *et al.*, 2022] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. *arXiv preprint arXiv:2202.07123*, 2022.

[Qi *et al.*, 2017] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[Qi *et al.*, 2023] Zekun Qi, Runpei Dong, Guofan Fan, Zheng Ge, Xiangyu Zhang, Kaisheng Ma, and Li Yi. Contrast with reconstruct: Contrastive 3d representation learning guided by generative pretraining. *arXiv preprint arXiv:2302.02318*, 2023.

[Qian *et al.*, 2022] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in Neural Information Processing Systems*, 35:23192–23204, 2022.

[Qiu *et al.*, 2021] Shi Qiu, Saeed Anwar, and Nick Barnes. Geometric back-projection network for point cloud classification. *IEEE Transactions on Multimedia*, 24:1943–1955, 2021.

[Radford *et al.*, 2021] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[Song *et al.*, 2023] Linxin Song, Jieyu Zhang, Lechao Cheng, Pengyuan Zhou, Tianyi Zhou, and Irene Li. Nlpbench: Evaluating large language models on solving nlp problems. *arXiv preprint arXiv:2309.15630*, 2023.

[Uy *et al.*, 2019] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1588–1597, 2019.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Wortsman *et al.*, 2022] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali

Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7959–7971, 2022.

[Wu *et al.*, 2015] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[Wu *et al.*, 2023] Jinmeng Wu, Tingting Mu, Jeyan Thiyagalingam, and John Y Goulermas. Memory-aware attentive control for community question answering with knowledge-based dual refinement. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2023.

[Xue *et al.*, 2023a] Le Xue, Mingfei Gao, Chen Xing, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. Ulip: Learning a unified representation of language, images, and point clouds for 3d understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1179–1189, 2023.

[Xue *et al.*, 2023b] Le Xue, Ning Yu, Shu Zhang, Junnan Li, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. Ulip-2: Towards scalable multimodal pre-training for 3d understanding. *arXiv preprint arXiv:2305.08275*, 2023.

[Yu *et al.*, 2022] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19313–19322, 2022.

[Zhang *et al.*, 2022a] Renrui Zhang, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Qiao, Peng Gao, and Hongsheng Li. Pointclip: Point cloud understanding by clip. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8552–8562, 2022.

[Zhang *et al.*, 2022b] Renrui Zhang, Wei Zhang, Rongyao Fang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-adapter: Training-free adaption of clip for few-shot classification. In *European Conference on Computer Vision*, pages 493–510. Springer, 2022.

[Zhang *et al.*, 2023] Renrui Zhang, Liuhui Wang, Yali Wang, Peng Gao, Hongsheng Li, and Jianbo Shi. Parameter is not all you need: Starting from non-parametric networks for 3d point cloud analysis. *arXiv preprint arXiv:2303.08134*, 2023.

[Zhou *et al.*, 2024] Pengyuan Zhou, Lin Wang, Zhi Liu, Yanbin Hao, Pan Hui, Sasu Tarkoma, and Jussi Kangasharju. A survey on generative ai and llm for video generation, understanding, and streaming. *arXiv preprint arXiv:2404.16038*, 2024.

[Zhu *et al.*, 2023] Xiangyang Zhu, Renrui Zhang, Bowei He, Ziyu Guo, Ziyao Zeng, Zipeng Qin, Shanghang Zhang, and Peng Gao. Pointclip v2: Prompting clip and gpt for powerful 3d open-world learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2639–2650, 2023.